AZURE PROJECT

JANUSHIYA.R
rajakumarjanu@gami.com

Table of Contents

Business Request

In this project, the company has identified a gap in understanding its customer demographics, specifically the gender distribution within the customer base and how it influences product purchases.

Since a significant amount of customer data is stored in an on-premises SQL database, key stakeholders have requested a comprehensive KPI dashboard.

The dashboard should provide insights at both high-level and detailed views, including:

- Total products sold
- Total sales revenue
- Clear gender-based distribution of customers

Additionally, users should be able to filter the data by product category and gender through a user-friendly interface, enabling efficient and fast data-driven analysis.

Our Solution Overview

To address this business request, we designed a robust end-to-end data pipeline that extracts data from an on-premises SQL database, ingests it into Azure, and applies the necessary data transformations to make the data analytics-ready and query-friendly.

The transformed data is then loaded into a custom-built reporting layer, which feeds a business intelligence dashboard aligned with the specified KPI requirements.

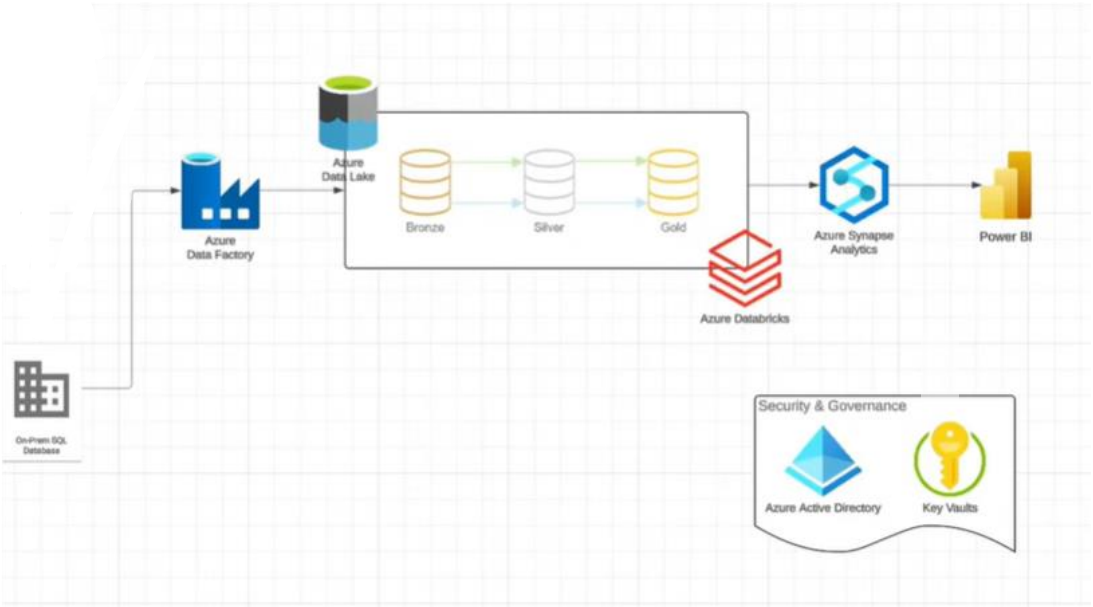
The pipeline is automated to run daily, ensuring that stakeholders always have access to accurate, consistent, and up-to-date data for informed decision-making.

AdventureWorksLT2022 Dataset

AdventureWorksLT2022 is a lightweight sample SQL Server database provided by Microsoft. It represents a simplified sales and customer management system for a fictional company that sells products online.

The dataset includes core tables such as customers, products, sales orders, and addresses, making it suitable for learning and practicing SQL queries, data modeling, and analytics without requiring large storage or high compute resources.

Flow of the project



Data is extracted from the on-premises SQL database using Azure Data Factory, which runs on a schedule. The data is stored and processed in Azure Data Lake using the bronze, silver, and gold layers. The Gold layer contains business-ready data, which is queried through Azure Synapse Analytics and finally visualized using Power BI dashboards.

SQL Database → Azure Data Factory (Scheduled data ingestion from SQL Server) → Azure Data Lake (Bronze → Silver → Gold) → Azure Synapse Analytics (Querying Gold layer data from Data Lake) → Power BI Dashboard

Security is applied
Azure Active Directory controls who can access what
Azure Key Vault securely stores passwords and keys

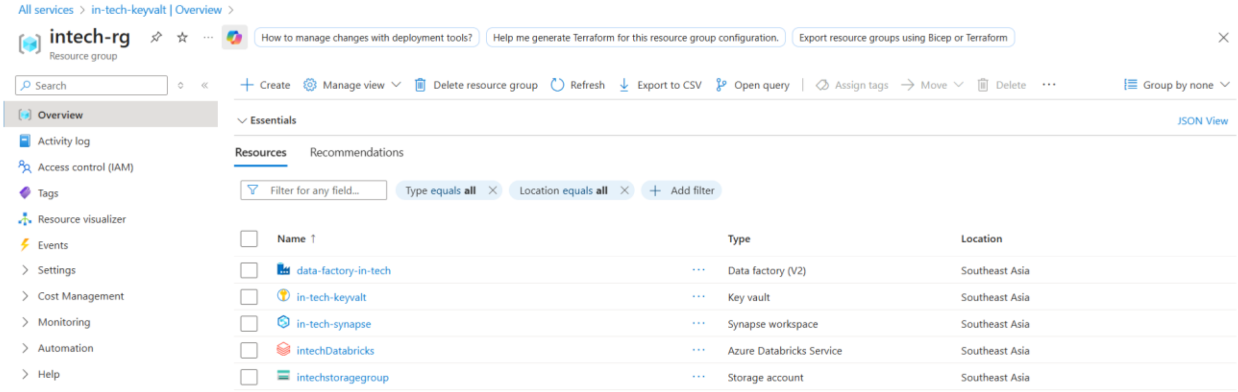
Steps for Data Ingestion- Data Factory

Note: All resources were created in the Southeast Asia region, while the resource group is located in the US region.

Step 1: Resource Creation

All required Azure services were created under the same resource group, including:

- Azure Data Factory
- Azure Databricks
- Azure Synapse Analytics
- Azure Storage Account (Data Lake), with a file system created inside it



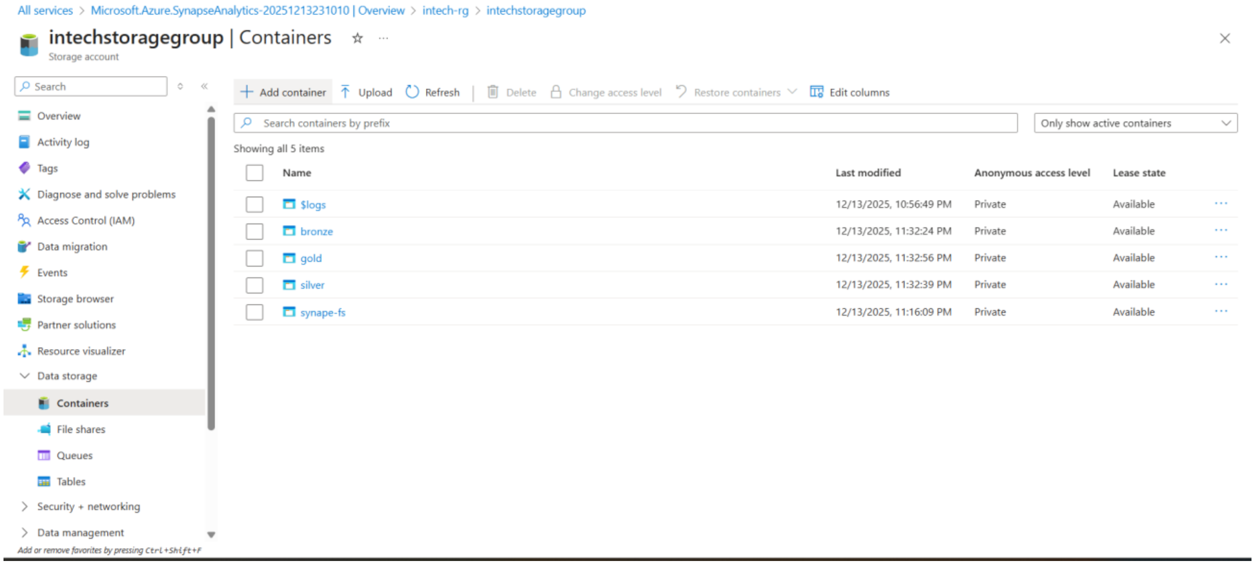
Name ↑	Type	Location
data-factory-in-tech	Data factory (V2)	Southeast Asia
in-tech-keyvalt	Key vault	Southeast Asia
in-tech-synapse	Synapse workspace	Southeast Asia
intechDatabricks	Azure Databricks Service	Southeast Asia
intechstoragegroup	Storage account	Southeast Asia

Step 2: Create Data Lake Containers

Within the Azure Storage Account (Data Lake) created in Step 1, create containers named:

- bronze
- silver
- gold

These containers represent the Bronze, Silver, and Gold layers used for storing raw, cleaned, and business-ready data.



Showing all 5 items

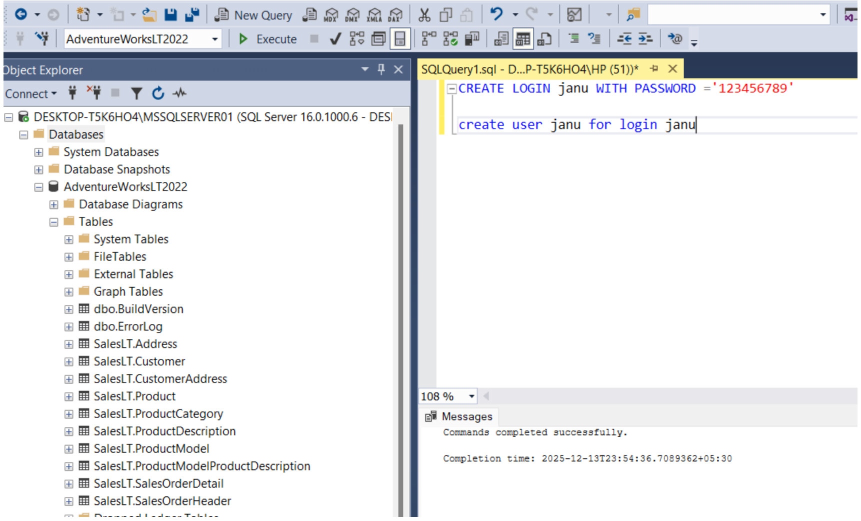
Name	Last modified	Anonymous access level	Lease state
$logs	12/13/2025, 10:56:49 PM	Private	Available
bronze	12/13/2025, 11:32:24 PM	Private	Available
gold	12/13/2025, 11:32:56 PM	Private	Available
silver	12/13/2025, 11:32:39 PM	Private	Available
synape-fs	12/13/2025, 11:16:09 PM	Private	Available

Step 3: Create Azure Key Vault

Create an Azure Key Vault within the same resource group.
The Key Vault is used to securely store sensitive information, such as credentials required for accessing the SQL Server. Power BI and other services use these stored secrets to securely connect to the Data Lake and other resources.

Step 4: Create SQL Server Credentials and Store in Key Vault

Create a SQL Server username and password for database access.
Store the SQL Server password securely in Azure Key Vault.
Azure services retrieve the password from the Key Vault to authenticate and access the SQL database when ingesting data.

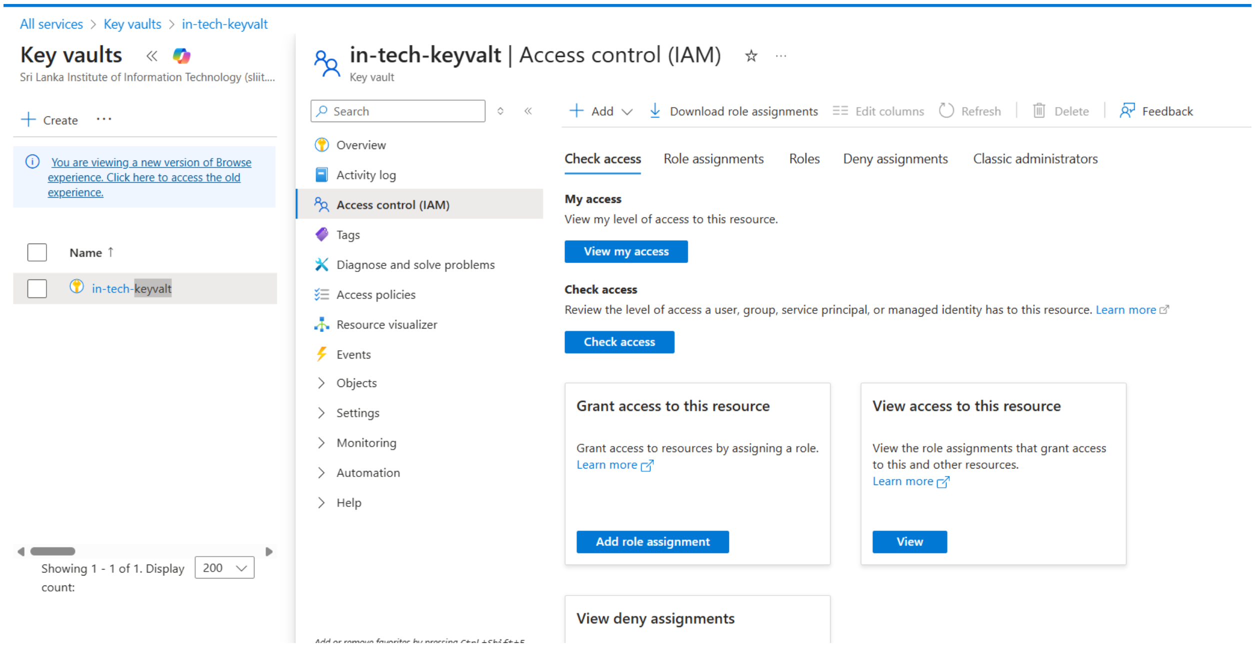


```sql
CREATE LOGIN janu WITH PASSWORD ='123456789'

create user janu for login janu
```

```
Commands completed successfully.

Completion time: 2025-12-13T23:54:36.7089362+05:30
```

Assign Key Vault Access

Under Access Control (IAM) of the Azure Key Vault, assign the Key Vault Administrator role to yourself.
If you are using a group, you can also assign this role to the group to manage access collectively.



You are viewing a new version of Browse experience. Click here to access the old experience.

Showing 1 - 1 of 1. Display count: 200

My access
View my level of access to this resource.

Check access
Review the level of access a user, group, service principal, or managed identity has to this resource. Learn more

Grant access to this resource
Grant access to resources by assigning a role. Learn more

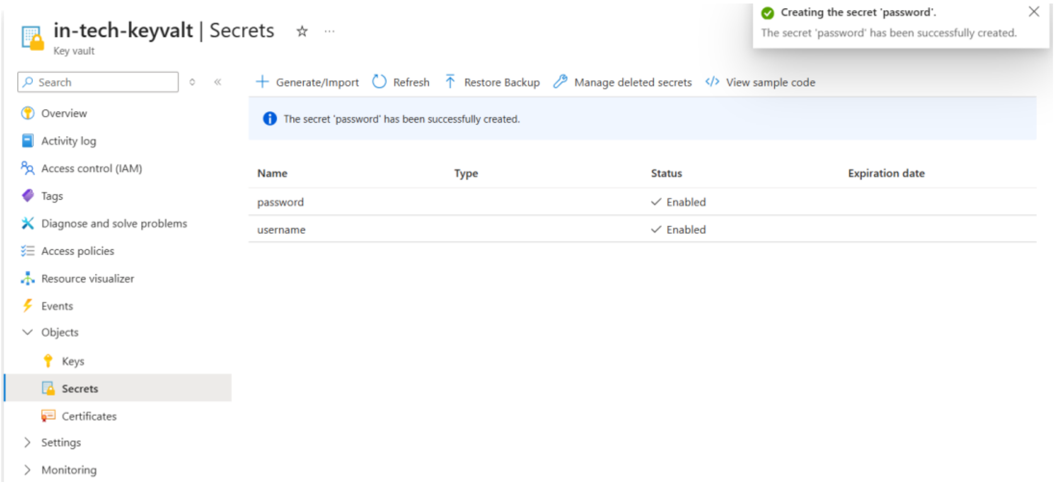
View access to this resource
View the role assignments that grant access to this and other resources. Learn more

Create Secrets in Azure Key Vault

On the same Key Vault page, under Objects, create secrets to store the SQL Server credentials.
Create two secrets:

- One for the SQL Server username
- One for the SQL Server password

Set the secret values to the corresponding SQL Server username and password (for example, *janu* and its password- 123456789).



Creating the secret 'password'.

The secret 'password' has been successfully created.

The secret 'password' has been successfully created.

Name	Type	Status	Expiration date
password		✓ Enabled	
username		✓ Enabled	

Step 5: Azure Data Factory – Copy Data from SQL Server (Single Table)

To ingest data from SQL Server into Azure Data Lake, perform the following steps:

1. Open Azure Data Factory and navigate to the Author tab.
2. Create a new pipeline.
3. From Move & Transform, drag and drop the Copy Data activity into the pipeline.

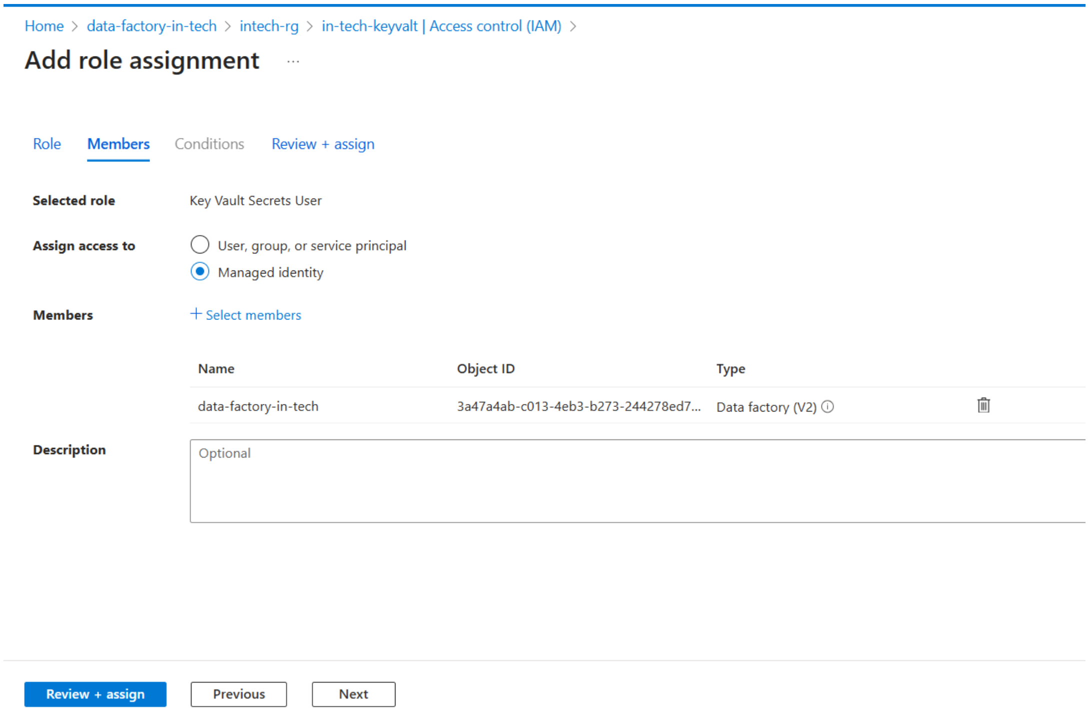
Source Configuration (SQL Server)

4. Set the Source as SQL Server and the Sink as Azure Data Lake.
5. While configuring the source, create a Linked Service for SQL Server.
6. To connect to the on-premises SQL Server, choose Manual Setup for the Integration Runtime (since Express setup did not work).
   - Download Microsoft Self-Hosted Integration Runtime.
   - Generate an Authentication Key from Azure Data Factory.
   - Paste the key into the Integration Runtime installer to register it successfully.
7. After successful registration, provide the following details:
   - Server name
   - Database name
   - Username
8. For the password, select Azure Key Vault and create an Azure Key Vault linked service.

Key Vault Configuration for Password Access

9. Ensure that the secret value is retrieved from Azure Key Vault (Active Directory authentication works for this setup).

10. In Azure Key Vault → Access Control (IAM), assign the Key Vault Secrets User role to the Azure Data Factory managed identity.
11. Once permissions are granted, return to Azure Data Factory and select the password using the secret name stored in Key Vault.


Home > data-factory-in-tech > intech-rg > in-tech-keyvalt | Access control (IAM) >

Add role assignment ...

Role
Members
Conditions
Review + assign

Selected role: Key Vault Secrets User

Assign access to: ○ User, group, or service principal ● Managed identity

Members: + Select members

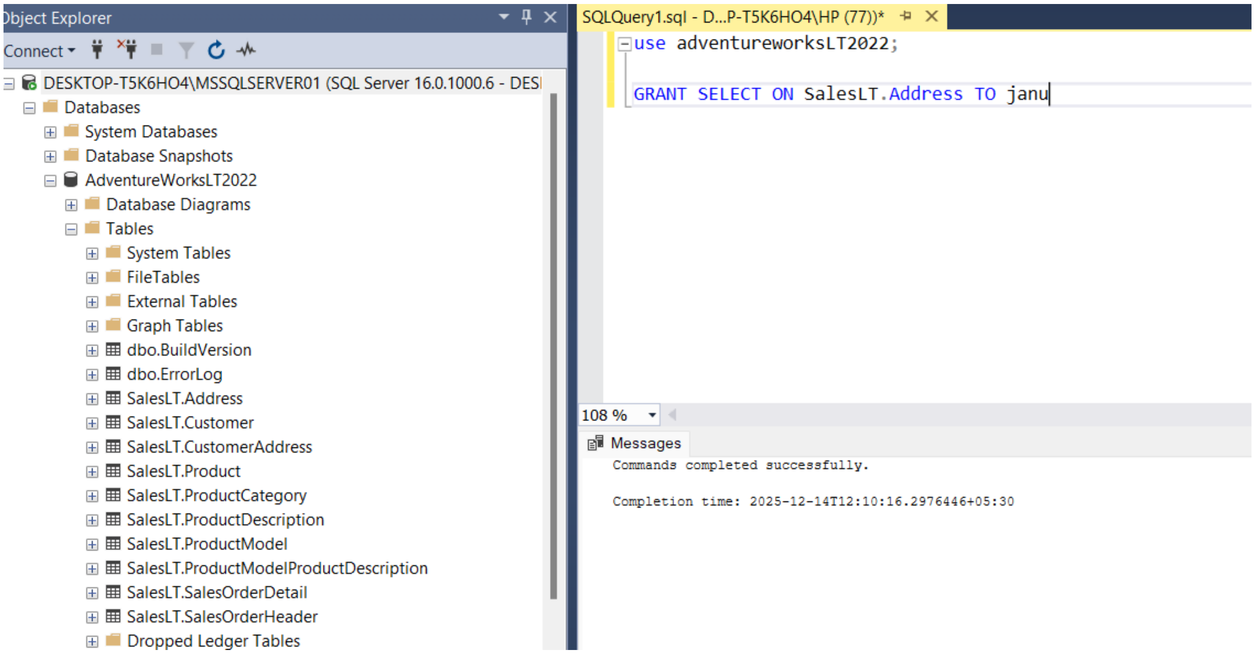
Name	Object ID	Type
data-factory-in-tech	3a47a4ab-c013-4eb3-b273-244278ed7...	Data factory (V2)

Description: Optional

Review + assign
Previous
Next

Note: The **AutoResolveIntegrationRuntime**, which appears when creating a new SQL Server linked service, can be used only for Azure-based services. It does not support connections to on-premises SQL Server.

To view and access tables from SQL Server, appropriate permissions must be granted to the user on the required tables in SQL Server.



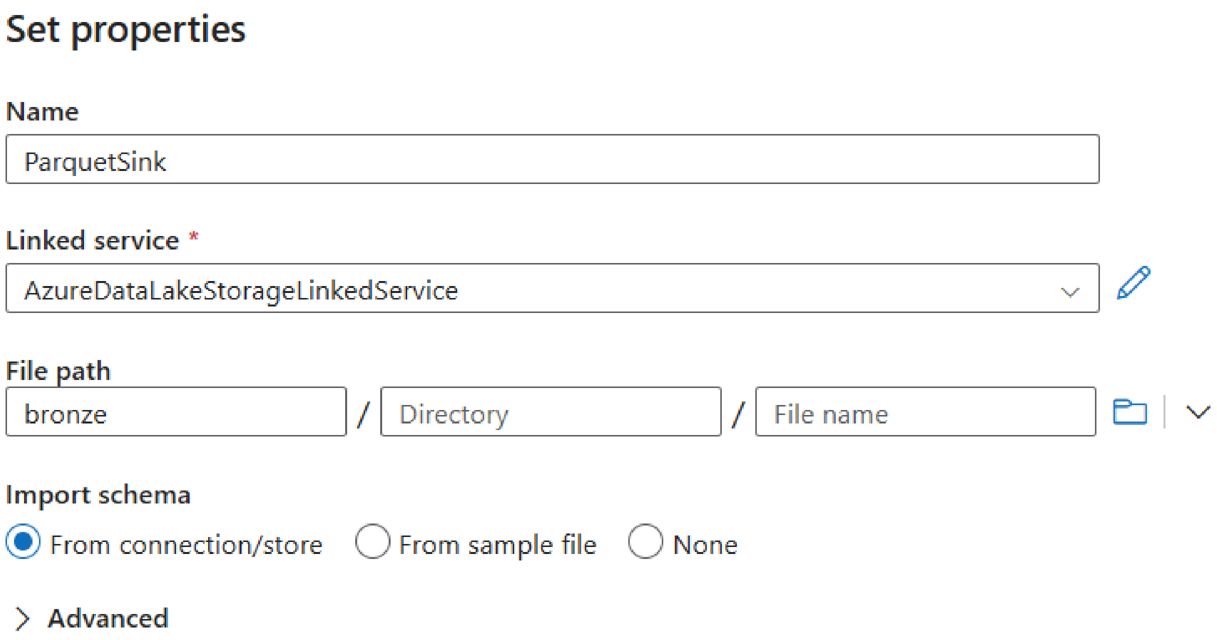
```
use adventureworksLT2022;

GRANT SELECT ON SalesLT.Address TO janu
```

```
Commands completed successfully.

Completion time: 2025-12-14T12:10:16.2976446+05:30
```

Sink Configuration (Data Lake)

Configure the Sink as Azure Data Lake Storage and select Parquet as the file format. Parquet is a column-based storage format that is optimized for efficient querying and analytics.

The hierarchical namespace is enabled on the storage account, which allows the Data Lake to organize data using folders and directories. This hierarchical structure is used to store data in the Bronze, Silver, and Gold layers.



Set properties

Name: ParquetSink

Linked service *: AzureDataLakeStorageLinkedService

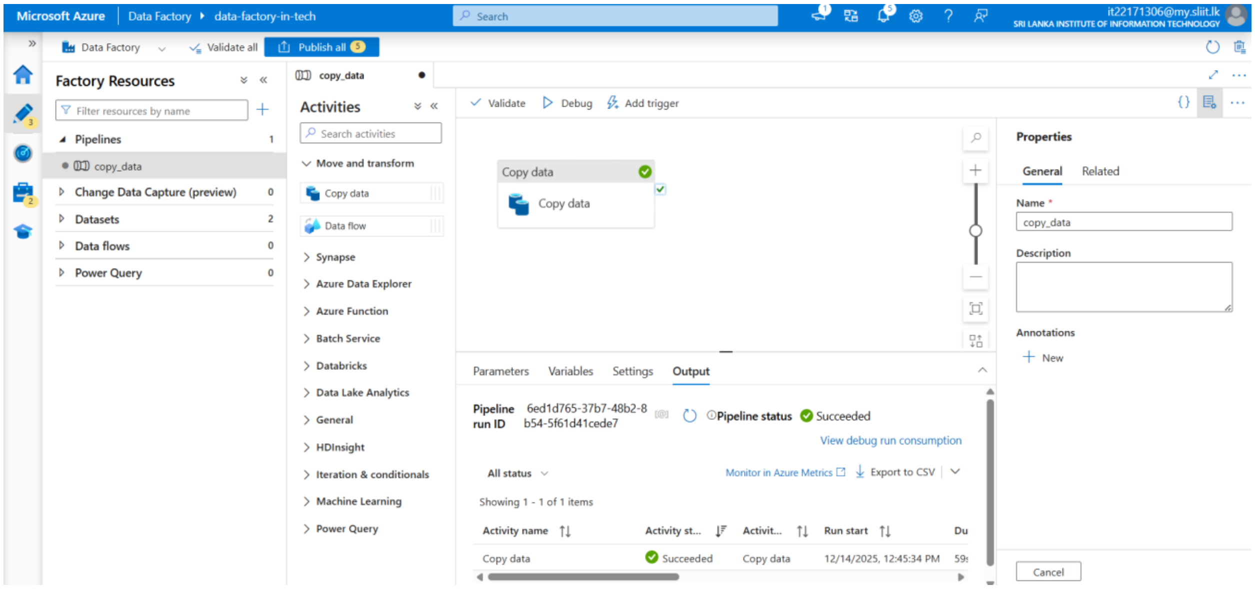
File path: bronze / Directory / File name

Import schema: ● From connection/store ○ From sample file ○ None

> Advanced

The pipeline was debugged and an error was encountered indicating that Java was not added to the system environment variables.

To resolve this issue, Java was added to the system variables, and the Self-Hosted Integration Runtime service was restarted.



Pipeline run ID 6ed1d765-37b7-48b2-8b54-5f61d41cede7   Pipeline status ✓ Succeeded

View debug run consumption

All status   Monitor in Azure Metrics   Export to CSV

Showing 1 - 1 of 1 items

Activity name	Activity st...	Activit...	Run start	Du
Copy data	✓ Succeeded	Copy data	12/14/2025, 12:45:34 PM	59s

Step 6: Insert Multiple Tables Using Lookup Activity

To ingest multiple tables from SQL Server, a new pipeline was created using the Lookup activity.

1. In Azure Data Factory, create a new pipeline.
2. Add a Lookup activity and configure it with a SQL query to retrieve the list of tables from the required schema.

The following query was used to fetch all tables from the SalesLT schema:

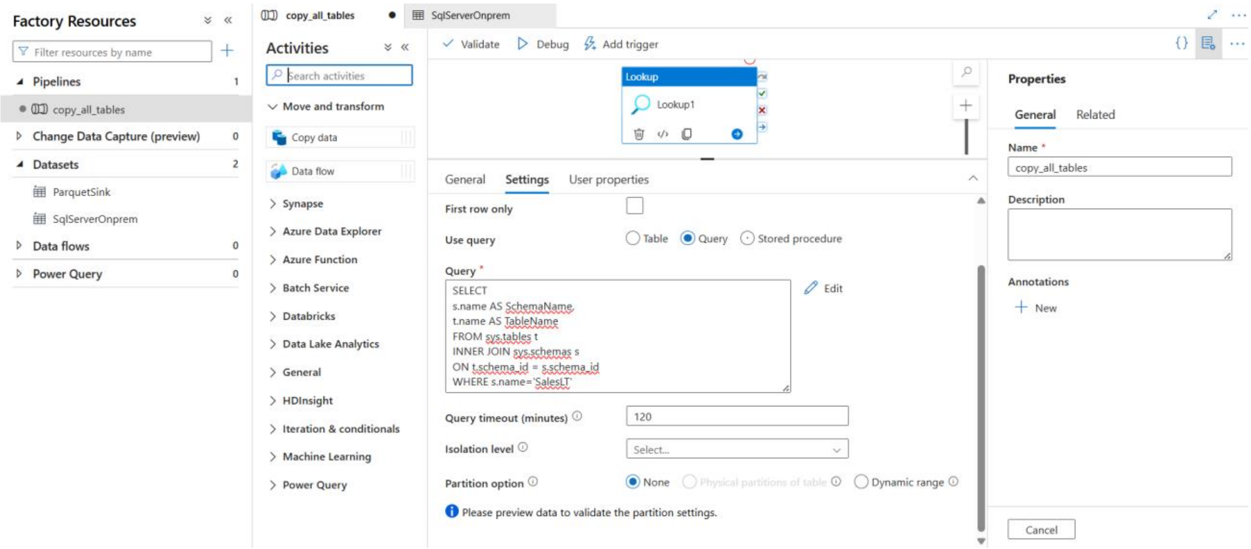
```
SELECT
s.name AS SchemaName,
t.name AS TableName
FROM sys.tables t
INNER JOIN sys.schemas s
ON t.schema_id = s.schema_id
WHERE s.name='SalesLT'
```

3. In SQL Server, permissions were granted to allow access to all tables in the schema:

```
use adventureworksLT2022;
GRANT SELECT ON SCHEMA::SalesLT TO janu
```

4. The output of the Lookup activity was used to control the pipeline logic for loading multiple tables.

Lookup Activity:
The Lookup activity reads a small amount of data (such as table names or control values) and passes it to other activities in the pipeline to drive dynamic processing.



First row only □

Use query ○ Table ● Query ○ Stored procedure

Query *

```
SELECT
s.name AS SchemaName,
t.name AS TableName
FROM sys.tables t
INNER JOIN sys.schemas s
ON t.schema_id = s.schema_id
WHERE s.name='SalesLT'
```

Edit

Query timeout (minutes) 120

Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.

Name * copy_all_tables

Description

Annotations

New

Load Multiple Tables Using ForEach

1. After configuring the Lookup activity, connect it to a ForEach activity.
2. Inside the ForEach activity, add a Copy Data activity.

Source Configuration (Dynamic Query)

3. In the Source settings of the Copy Data activity, use the following dynamic SQL query to read each table dynamically:

@{concat('SELECT * FROM ', item().schemaName,'.',item().TableName)}

This query dynamically selects data from each table returned by the Lookup activity.

Sink Configuration (Parquet Format)

4. Use the previously created Parquet sink dataset pointing to the Data Lake (Bronze layer).
5. In the sink dataset, create two parameters:

- schemaname
- tablename

6. In the Sink settings, assign values to the parameters:

- schemaname: @item().schemaName
- Tablename: @item().TableName

7. Configure the directory path in the Bronze layer as:

@{concat(dataset().schemaname,'/',dataset().tablename)}

8. Configure the file name as:

@{concat(dataset().tablename,'.parquet')}

This structure stores each table in its own folder under the Bronze layer.

ForEach Settings

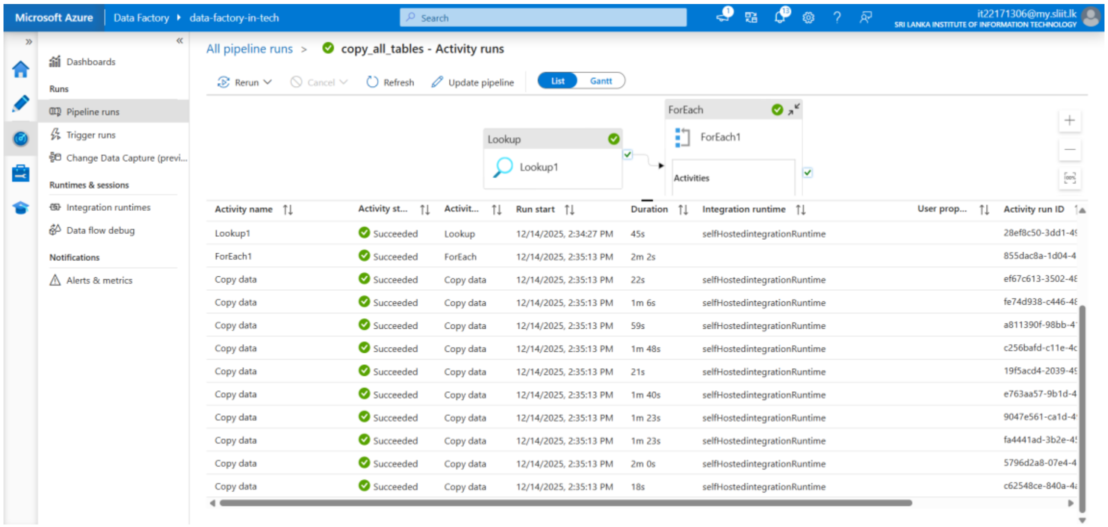
9. In the ForEach activity settings, set Items to: @activity('Lookup1').output.value

This ensures the Copy Data activity runs once for each table returned by the Lookup.

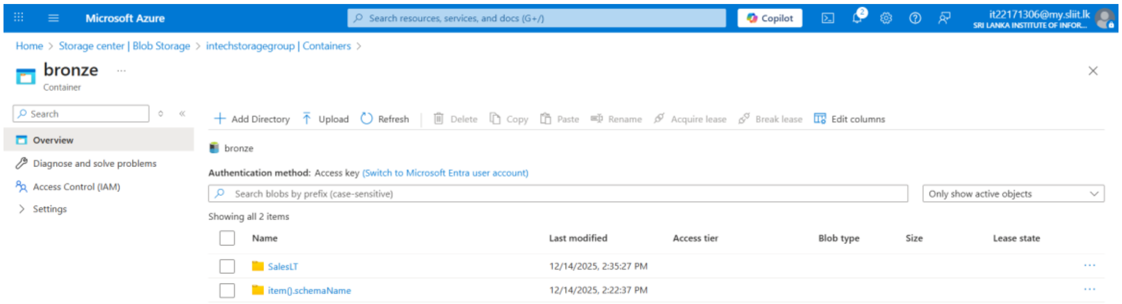
Note: The @ symbol is required to enable dynamic content expressions in Azure Data Factory.

Step 7: Publish and Monitor the Pipeline

1. After completing the pipeline configuration, click Publish to save the changes.
2. The pipeline can be updated and republished later if required.
3. Trigger the pipeline manually using Trigger Now, or schedule it as needed.
4. After triggering, navigate to the Monitor tab to track execution status and review logs.



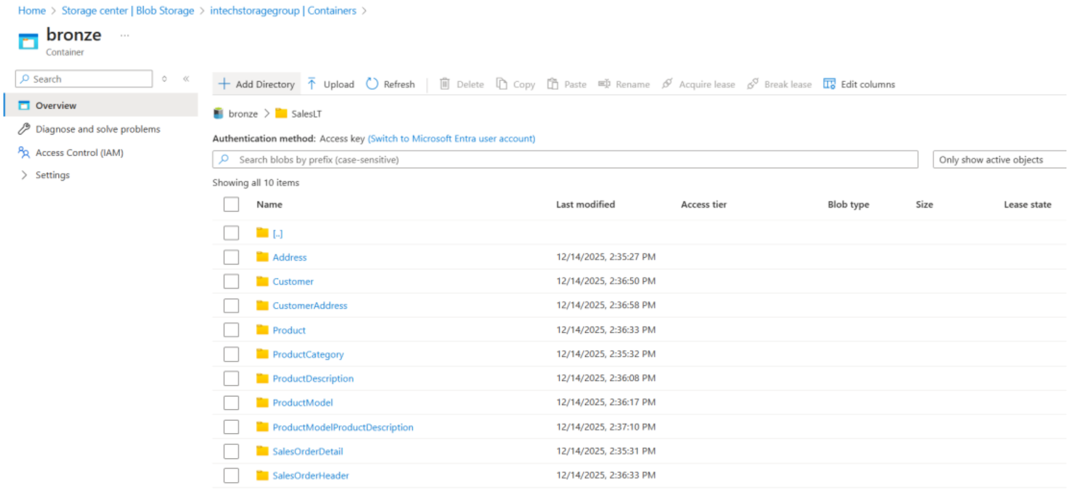
Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID
Lookup1	✓ Succeeded	Lookup	12/14/2025, 2:34:27 PM	45s	selfHostedintegrationRuntime		28ef8c50-3dd1-4
ForEach1	✓ Succeeded	ForEach	12/14/2025, 2:35:13 PM	2m 2s			855dac8a-1d04-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	22s	selfHostedintegrationRuntime		ef67c613-3502-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	1m 6s	selfHostedintegrationRuntime		fe74d938-c446-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	59s	selfHostedintegrationRuntime		a811390f-98bb-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	1m 48s	selfHostedintegrationRuntime		c256bafd-c11e-4c
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	21s	selfHostedintegrationRuntime		19f5acd4-2039-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	1m 40s	selfHostedintegrationRuntime		e763aa57-9b1d-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	1m 23s	selfHostedintegrationRuntime		9047e561-ca1d-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	1m 23s	selfHostedintegrationRuntime		fa4441ad-3b2e-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	2m 0s	selfHostedintegrationRuntime		5796d2a8-07e4-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 2:35:13 PM	18s	selfHostedintegrationRuntime		c62548ce-840a-4



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

Showing all 2 items

□	Name	Last modified	Access tier	Blob type	Size	Lease state
□	SalesLT	12/14/2025, 2:35:27 PM				
□	item().schemaName	12/14/2025, 2:22:37 PM				



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

Showing all 10 items

□	Name	Last modified	Access tier	Blob type	Size	Lease state
□	[..]					
□	Address	12/14/2025, 2:35:27 PM				
□	Customer	12/14/2025, 2:36:50 PM				
□	CustomerAddress	12/14/2025, 2:36:58 PM				
□	Product	12/14/2025, 2:36:33 PM				
□	ProductCategory	12/14/2025, 2:35:32 PM				
□	ProductDescription	12/14/2025, 2:36:08 PM				
□	ProductModel	12/14/2025, 2:36:17 PM				
□	ProductModelProductDescription	12/14/2025, 2:37:10 PM				
□	SalesOrderDetail	12/14/2025, 2:35:31 PM				
□	SalesOrderHeader	12/14/2025, 2:36:33 PM				

Steps for Data Transformation – Azure Databricks

Step 1: Create Compute (Cluster) – single node depend on your subscription

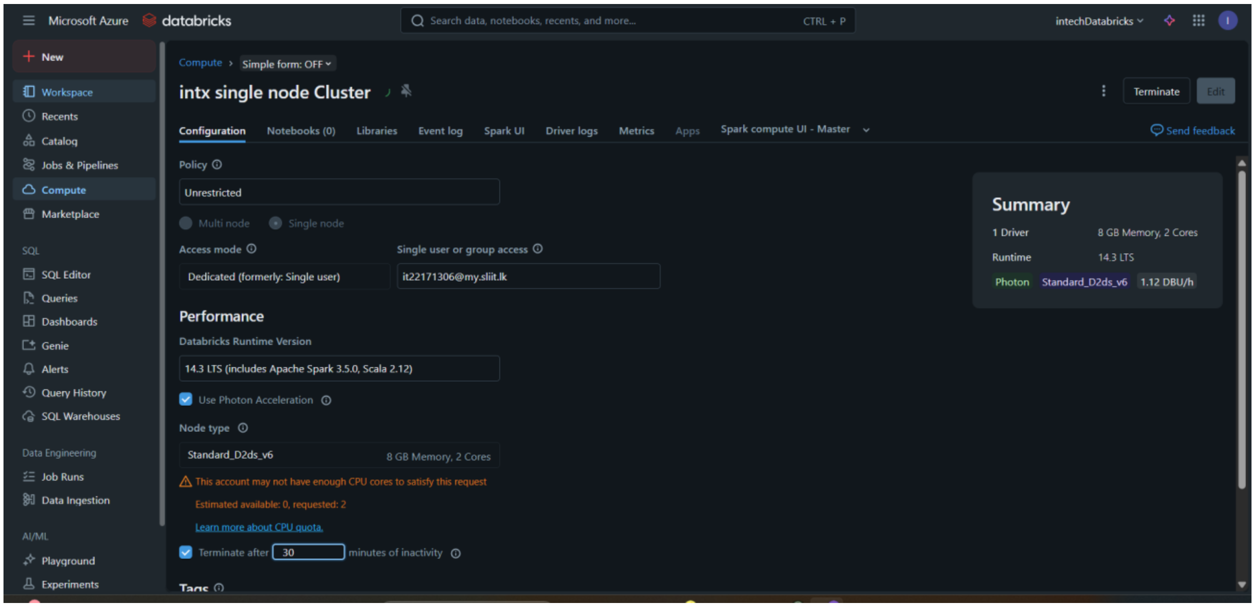
Create a compute cluster in Azure Databricks.
A cluster is a group of machines that provides the resources required to run data processing tasks.

A cluster consists of:

- CPU cores
- Memory
- Apache Spark engine

Purpose of the cluster:

- To execute notebooks and code
- To process large volumes of data
- To run Spark-based transformation jobs



intx single node Cluster

Policy: Unrestricted

○ Multi node ● Single node

Access mode: Dedicated (formerly: Single user)

Single user or group access: it22171306@my.sliit.lk

Performance

Databricks Runtime Version: 14.3 LTS (includes Apache Spark 3.5.0, Scala 2.12)

☑ Use Photon Acceleration

Node type: Standard_D2ds_v6 8 GB Memory, 2 Cores

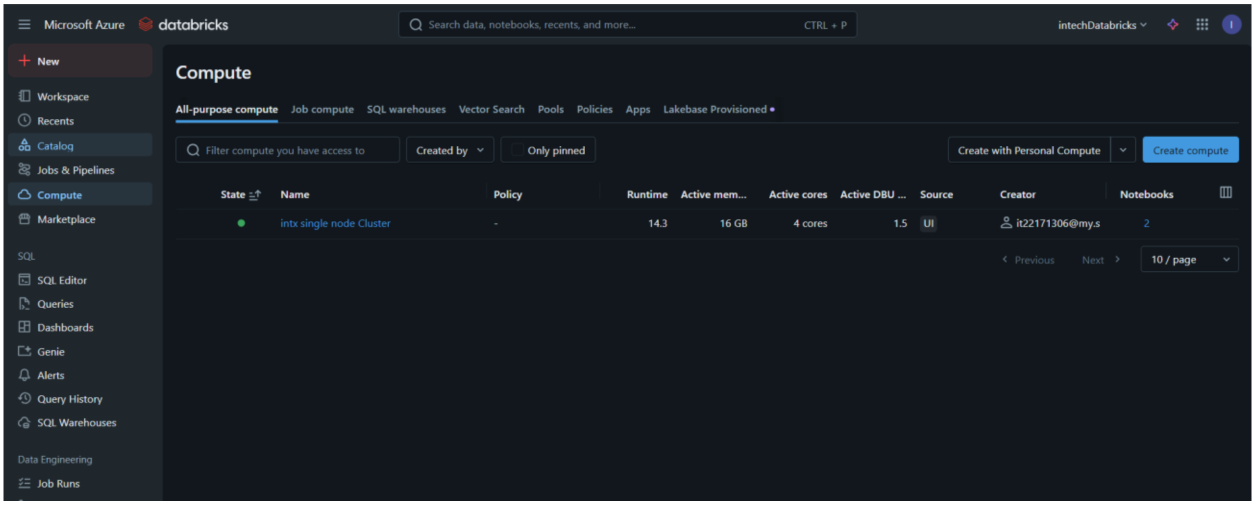
This account may not have enough CPU cores to satisfy this request

Estimated available: 0, requested: 2

☑ Terminate after 30 minutes of inactivity

Summary

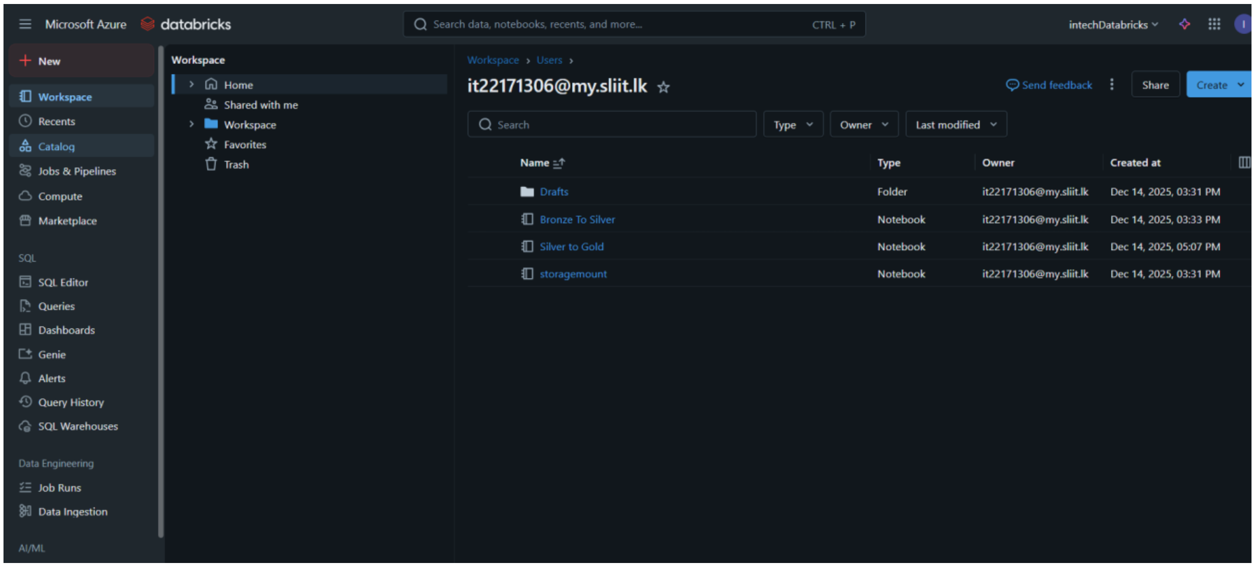
1 Driver	8 GB Memory, 2 Cores
Runtime	14.3 LTS
Photon	Standard_D2ds_v6 1.12 DBU/h



Compute

State	Name	Policy	Runtime	Active mem...	Active cores	Active DBU ...	Source	Creator	Notebooks
●	intx single node Cluster	-	14.3	16 GB	4 cores	1.5	UI	it22171306@my.s	2

Databricks Notebooks Used in the Project



Name	Type	Owner	Created at
Drafts	Folder	it22171306@my.sliit.lk	Dec 14, 2025, 03:31 PM
Bronze To Silver	Notebook	it22171306@my.sliit.lk	Dec 14, 2025, 03:33 PM
Silver to Gold	Notebook	it22171306@my.sliit.lk	Dec 14, 2025, 05:07 PM
storagemount	Notebook	it22171306@my.sliit.lk	Dec 14, 2025, 03:31 PM

Notebook 1: Storage Mount

This notebook establishes a connection between Azure Databricks and Azure Data Lake Storage.

The storage account is mounted to the Databricks file system to provide an easy access point for reading and writing data.

The notebook is executed using the Databricks cluster created earlier.

Notebook 2: Bronze to Silver Transformation

In this notebook, each table from the Bronze layer was read in Parquet format.

Date-related columns were identified and standardized to a consistent yyyy-MM-dd format.

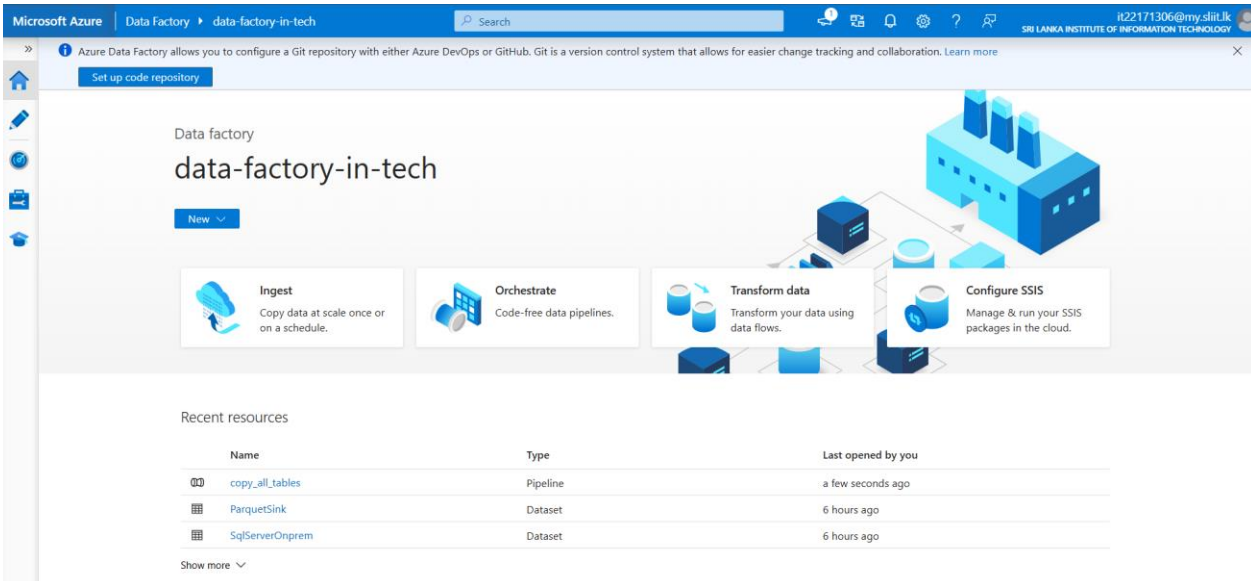
The cleaned and standardized data was then written to the Silver layer, ensuring improved data quality and consistency across all SalesLT (Schema) tables.

Notebook 3: Silver to Gold Transformation

This notebook reads the cleaned Silver-layer data and standardizes column names into snake_case (for example, ModifiedDate → modified_date).

This step prepares the data for the Gold layer, making it easier to query and suitable for analytics and reporting.

Creating a Pipeline to Automate Bronze, Silver, and Gold Processing – Azure Data Factory



Microsoft Azure
Data Factory ▸ data-factory-in-tech
Search
it22171306@my.sliit.lk
SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY

Azure Data Factory allows you to configure a Git repository with either Azure DevOps or GitHub. Git is a version control system that allows for easier change tracking and collaboration. Learn more

Data factory

data-factory-in-tech

Ingest – Copy data at scale once or on a schedule.

Orchestrate – Code-free data pipelines.

Transform data – Transform your data using data flows.

Configure SSIS – Manage & run your SSIS packages in the cloud.

Recent resources

Name	Type	Last opened by you
copy_all_tables	Pipeline	a few seconds ago
ParquetSink	Dataset	6 hours ago
SqlServerOnprem	Dataset	6 hours ago

Attaching Databricks Notebooks to Azure Data Factory Pipeline

The Databricks notebooks were integrated into the previously created Azure Data Factory pipeline to automate the execution of Bronze, Silver, and Gold transformations.

Step 1: Add Databricks Notebook Activity

A Databricks Notebook activity was added to the pipeline.
A Databricks linked service was created in Azure Data Factory using an existing interactive cluster.

Configure Authentication

The authentication type was set to Access Token.

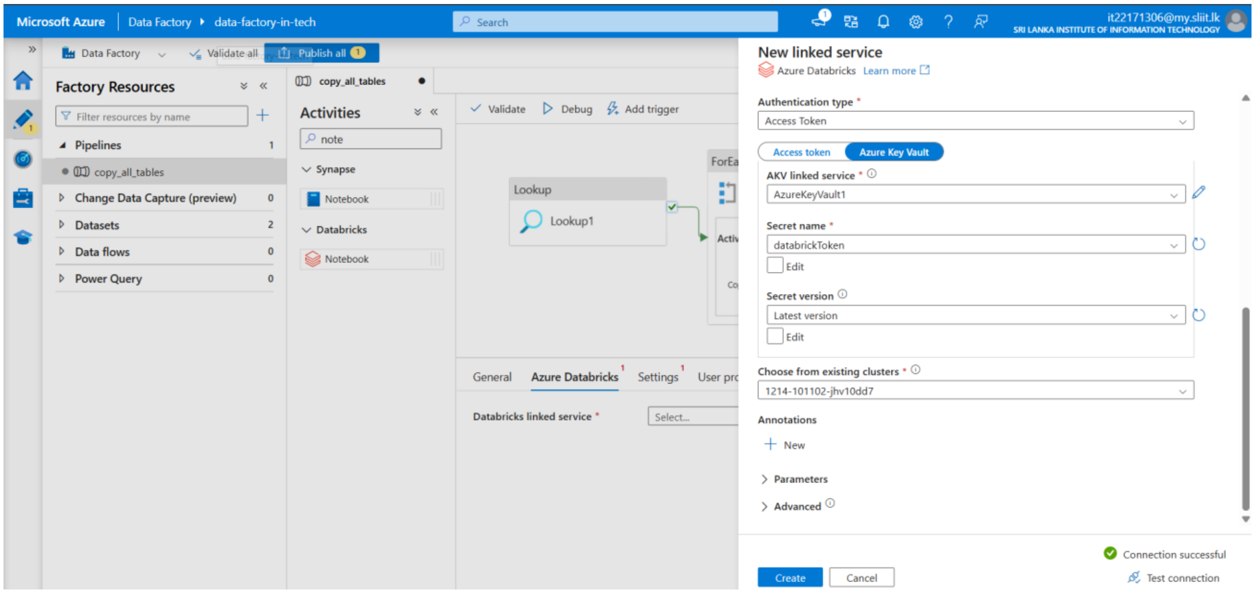
- An access token was generated in Azure Databricks under:
  User Settings → Developer → Access Tokens
- This token allows Azure Data Factory to securely trigger Databricks notebooks.

Secure the Access Token

Since the access token contains sensitive information:

- The token was stored securely in Azure Key Vault as a secret
- Azure Data Factory retrieves the token from Key Vault using a Key Vault linked service

This approach ensures secure and centralized credential management.


New linked service
Azure Databricks Learn more

Authentication type *
Access Token

Access token | Azure Key Vault

AKV linked service *
AzureKeyVault1

Secret name *
databrickToken
□ Edit

Secret version
Latest version
□ Edit

Choose from existing clusters *
1214-101102-jhv10dd7

Annotations
+ New

> Parameters
> Advanced

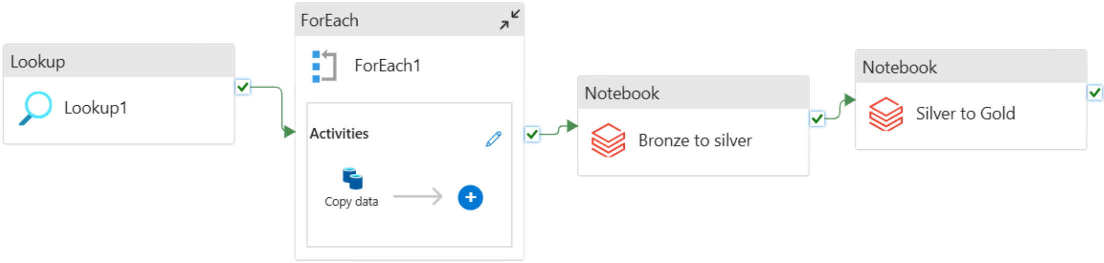
Connection successful
Create | Cancel | Test connection

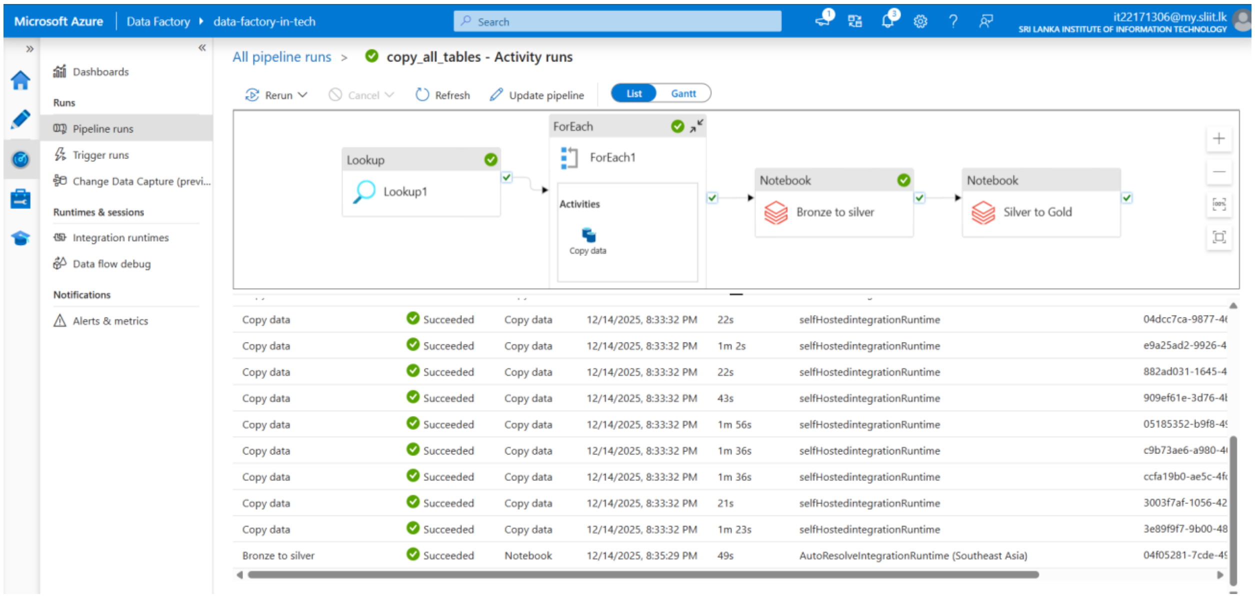
Configure Notebook Path and Publish the Pipeline

In the Settings tab of the Databricks Notebook activity, the Notebook Path was configured by providing the path of the Bronze-to-Silver notebook stored in the Users folder of Azure Databricks.

The same procedure was followed to configure the Silver-to-Gold notebook.

After configuring both notebook activities, the pipeline was published to save and deploy the changes.



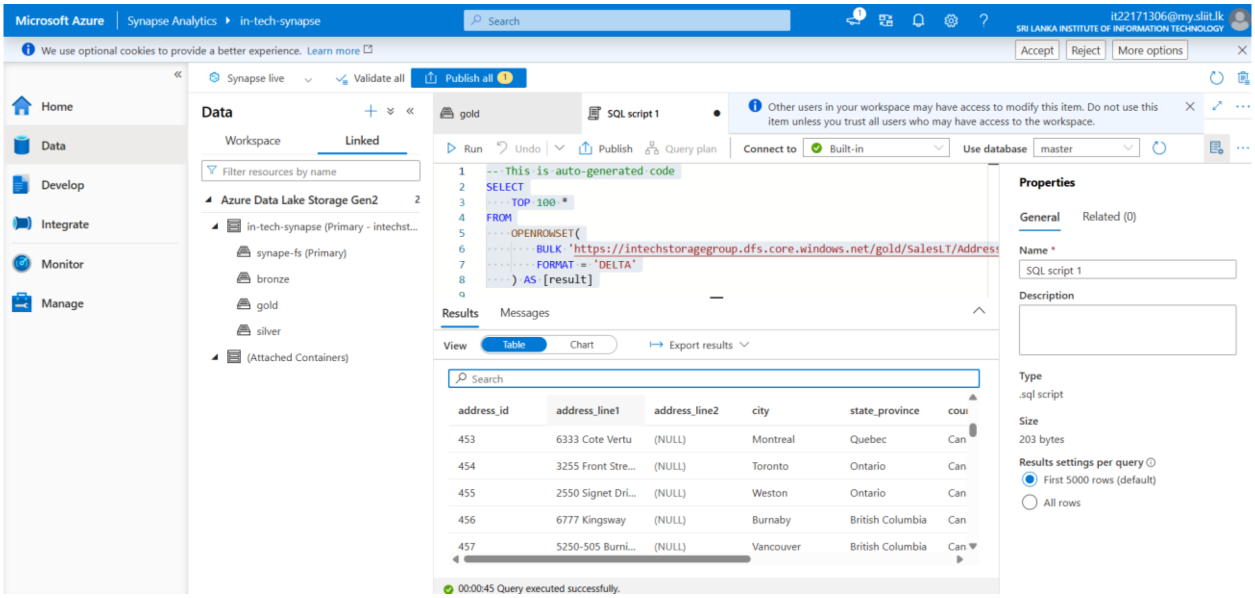


Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	22s	selfHostedintegrationRuntime	04dcc7ca-9877-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	1m 2s	selfHostedintegrationRuntime	e9a25ad2-9926-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	22s	selfHostedintegrationRuntime	882ad031-1645-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	43s	selfHostedintegrationRuntime	909ef61e-3d76-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	1m 56s	selfHostedintegrationRuntime	05185352-b9f8-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	1m 36s	selfHostedintegrationRuntime	c9b73ae6-a980-4
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	1m 36s	selfHostedintegrationRuntime	ccfa19b0-ae5c-4f
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	21s	selfHostedintegrationRuntime	3003f7af-1056-42
Copy data	✓ Succeeded	Copy data	12/14/2025, 8:33:32 PM	1m 23s	selfHostedintegrationRuntime	3e89f9f7-9b00-48
Bronze to silver	✓ Succeeded	Notebook	12/14/2025, 8:35:29 PM	49s	AutoResolveIntegrationRuntime (Southeast Asia)	04f05281-7cde-4

Pipeline Testing and Monitoring – Azure Synapse Analytics

Azure Synapse Analytics was used to query, analyze, and serve data for reporting after the pipeline execution.

The serverless SQL pool was used to query data directly from the Data Lake, as it is cost-effective and suitable for moderate data volumes.
The dedicated SQL pool was not used because it is more expensive and typically designed for very large datasets and enterprise-scale workloads.



We use optional cookies to provide a better experience. Learn more

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

Connect to: Built-in

Use database: master

```
-- This is auto-generated code
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://intechstoragegroup.dfs.core.windows.net/gold/SalesLT/Address
        FORMAT = 'DELTA'
    ) AS [result]
```

address_id	address_line1	address_line2	city	state_province	cou
453	6333 Cote Vertu	(NULL)	Montreal	Quebec	Can
454	3255 Front Stre...	(NULL)	Toronto	Ontario	Can
455	2550 Signet Dri...	(NULL)	Weston	Ontario	Can
456	6777 Kingsway	(NULL)	Burnaby	British Columbia	Can
457	5250-505 Burni...	(NULL)	Vancouver	British Columbia	Can

00:00:45 Query executed successfully.

Name *: SQL script 1

Description:

Type: .sql script

Size: 203 bytes

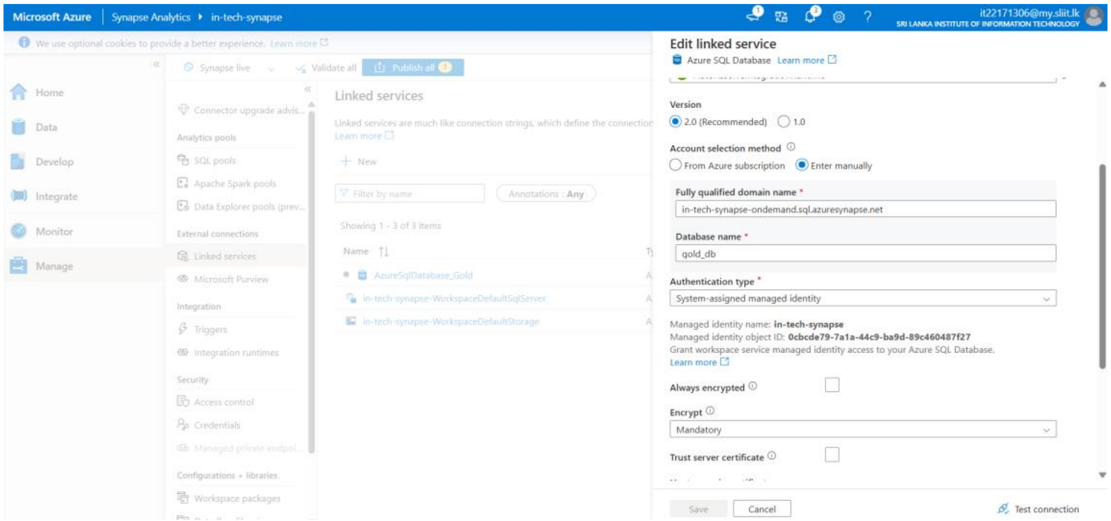
Results settings per query
● First 5000 rows (default)
○ All rows

Creating a Linked Service to Azure SQL Database

A Linked Service was created in Azure Synapse Analytics under the Manage tab to connect to the Azure SQL Database.

The Fully Qualified Domain Name (FQDN) was obtained from the Properties section of Synapse Analytics under Settings, specifically from the Serverless SQL endpoint.

This FQDN was used while configuring the linked service.
After completing the configuration, the linked service was published to apply the changes.



Edit linked service

Azure SQL Database Learn more

Version

● 2.0 (Recommended) ○ 1.0

Account selection method

○ From Azure subscription ● Enter manually

Fully qualified domain name *

in-tech-synapse-ondemand.sql.azuresynapse.net

Database name *

gold_db

Authentication type *

System-assigned managed identity

Managed identity name: **in-tech-synapse**
Managed identity object ID: **0cbcde79-7a1a-44c9-ba9d-89c460487f27**
Grant workspace service managed identity access to your Azure SQL Database.
Learn more

Always encrypted □

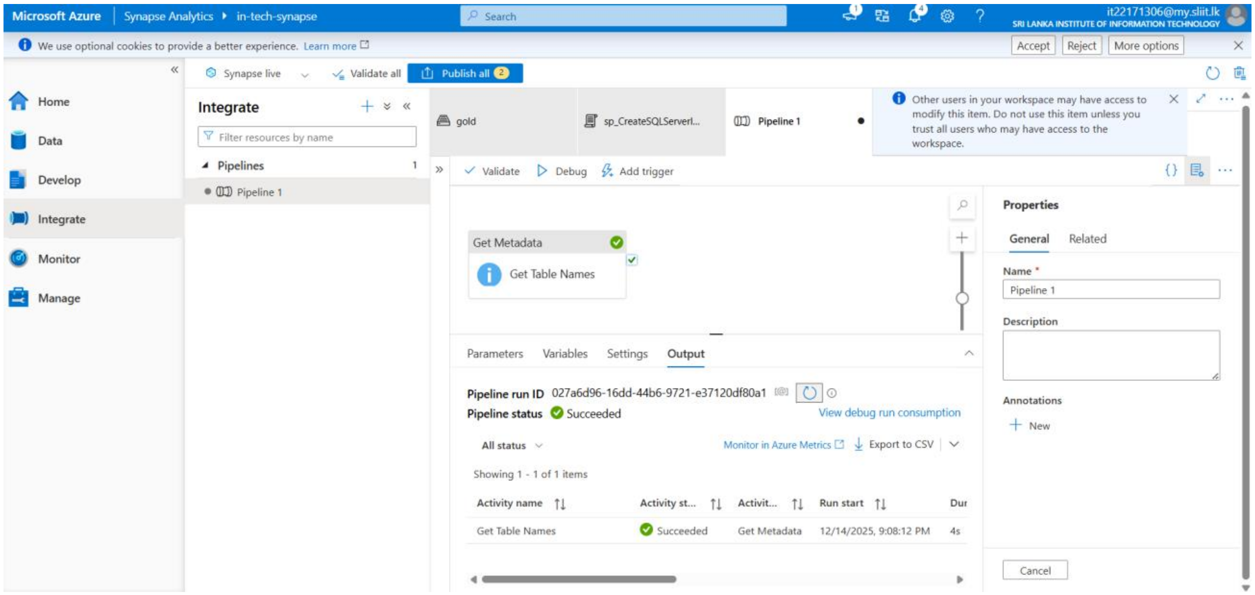
Encrypt

Mandatory

Trust server certificate □

Creating a Pipeline in Synapse to Retrieve Table Names

A pipeline was created in Azure Synapse Analytics to retrieve table names from the database.
This pipeline helps dynamically identify available tables and can be used to control downstream processing or validation steps.



We use optional cookies to provide a better experience. Learn more

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

Pipeline run ID 027a6d96-16dd-44b6-9721-e37120df80a1

Pipeline status ✓ Succeeded

View debug run consumption

All status

Monitor in Azure Metrics

Export to CSV

Showing 1 - 1 of 1 items

Activity name	Activity st...	Activit...	Run start	Dur
Get Table Names	✓ Succeeded	Get Metadata	12/14/2025, 9:08:12 PM	4s

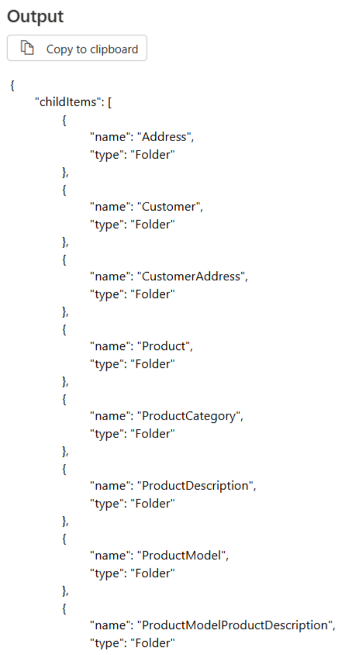
Name * Pipeline 1

Description

Annotations

+ New

Output



Output

```
{
    "childItems": [
        {
            "name": "Address",
            "type": "Folder"
        },
        {
            "name": "Customer",
            "type": "Folder"
        },
        {
            "name": "CustomerAddress",
            "type": "Folder"
        },
        {
            "name": "Product",
            "type": "Folder"
        },
        {
            "name": "ProductCategory",
            "type": "Folder"
        },
        {
            "name": "ProductDescription",
            "type": "Folder"
        },
        {
            "name": "ProductModel",
            "type": "Folder"
        },
        {
            "name": "ProductModelProductDescription",
            "type": "Folder"
```

Creating Views Dynamically Using ForEach and Stored Procedure – Synapse

A ForEach activity was added to the Synapse pipeline to iterate through the list of table names.
Inside the ForEach, a stored procedure activity was used to dynamically create views for each table in the Gold layer.

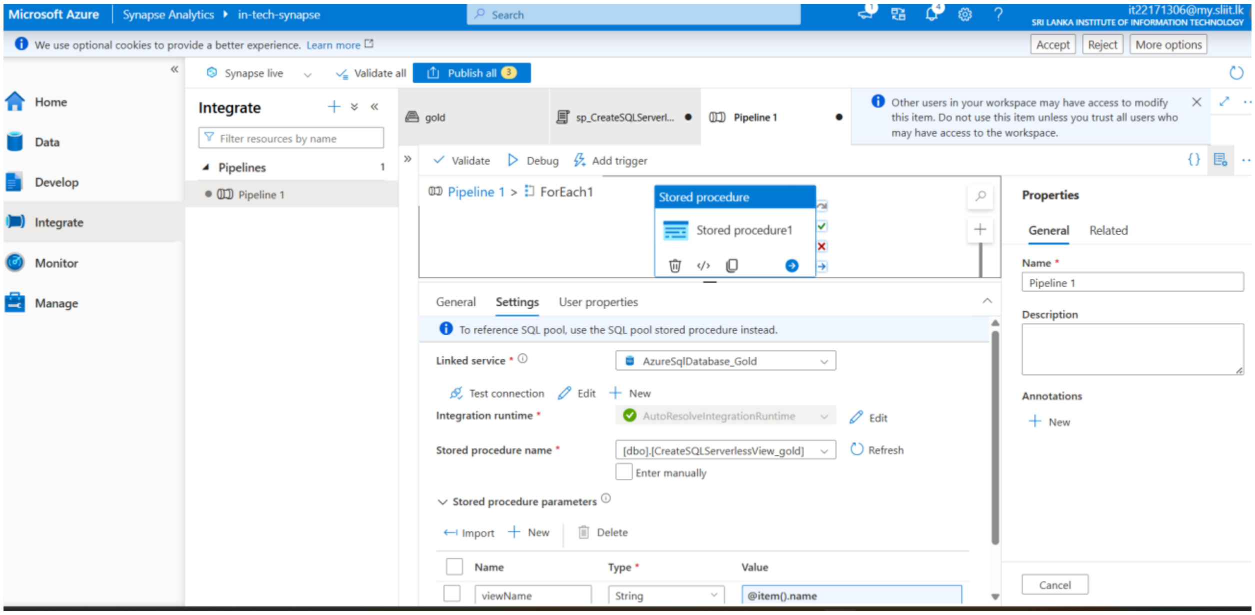
Stored Procedure Purpose

The stored procedure creates or updates a serverless SQL view for each Gold-layer table.
Each view reads data directly from the Gold layer (Delta format) stored in Azure Data Lake.

What this stored procedure does:

- Accepts the table name as a parameter (@ViewName)
- Dynamically builds a SQL statement to:
  - Create or alter a view with the same name as the table
  - Read data from the Gold layer using OPENROWSET
  - Access Delta format files stored in Azure Data Lake
- Executes the generated SQL dynamically

```
USE gold_db
GO

CREATE OR ALTER PROC CreateSQLServerlessView_gold @ViewName nvarchar(100)
AS
BEGIN
  DECLARE @statement VARCHAR(MAX)
  SET @statement = N'CREATE OR ALTER VIEW ' + @ViewName + ' AS
    SELECT
      *
    FROM
      OPENROWSET(
        BULK ''https://intechstoragegroup.dfs.core.windows.net/gold/SalesLT/' + @ViewName + '/'',
        FORMAT = ''DELTA''
      ) AS [result]'
    EXEC (@statement)
END
GO
```



Pipeline 1 > ForEach1 — Stored procedure: Stored procedure1

Settings:

To reference SQL pool, use the SQL pool stored procedure instead.

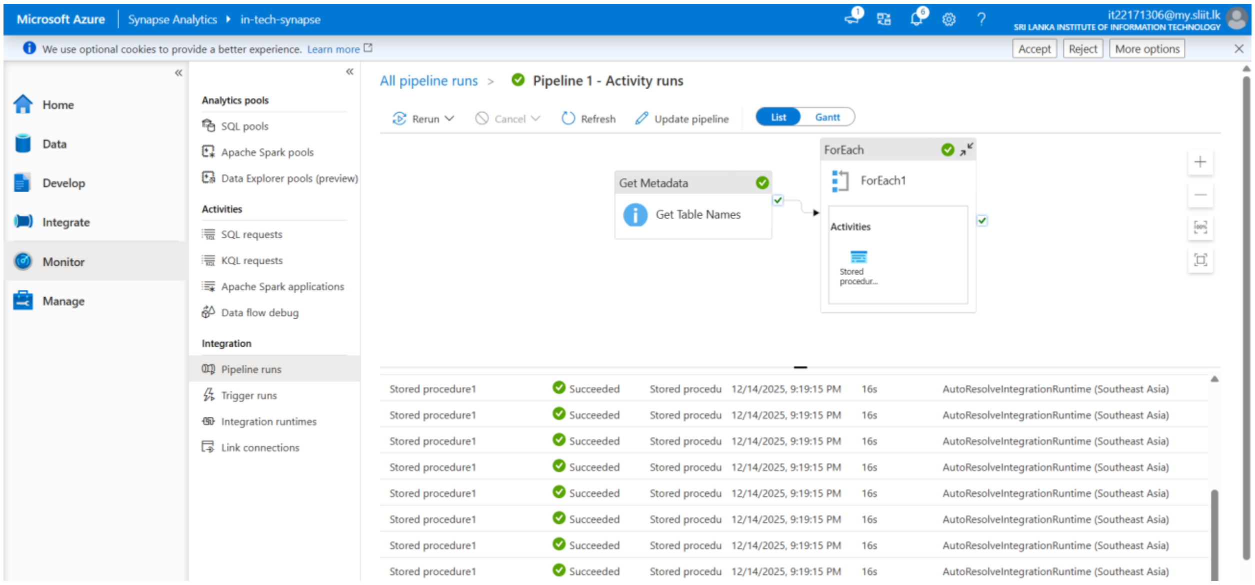
- Linked service *: AzureSqlDatabase_Gold
- Integration runtime *: AutoResolveIntegrationRuntime
- Stored procedure name *: [dbo].[CreateSQLServerlessView_gold]
- □ Enter manually

Stored procedure parameters

Name	Type *	Value
viewName	String	@item().name

Properties — General: Name *: Pipeline 1

Publish and trigger it



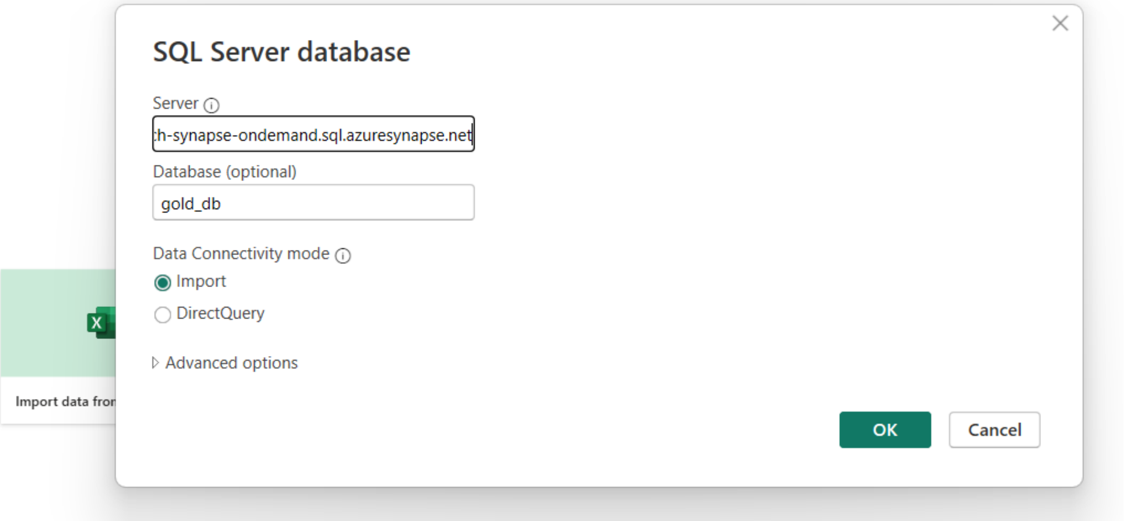
All pipeline runs > Pipeline 1 - Activity runs

Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)
Stored procedure1	✓ Succeeded	Stored procedu	12/14/2025, 9:19:15 PM	16s	AutoResolveIntegrationRuntime (Southeast Asia)

Note: This process must be re-executed whenever there are schema changes in the Gold-layer tables to ensure that the serverless SQL views reflect the updated structure.

Power BI Report

Server: Serverless SQL endpoint



SQL Server database

Server

:h-synapse-ondemand.sql.azuresynapse.net

Database (optional)

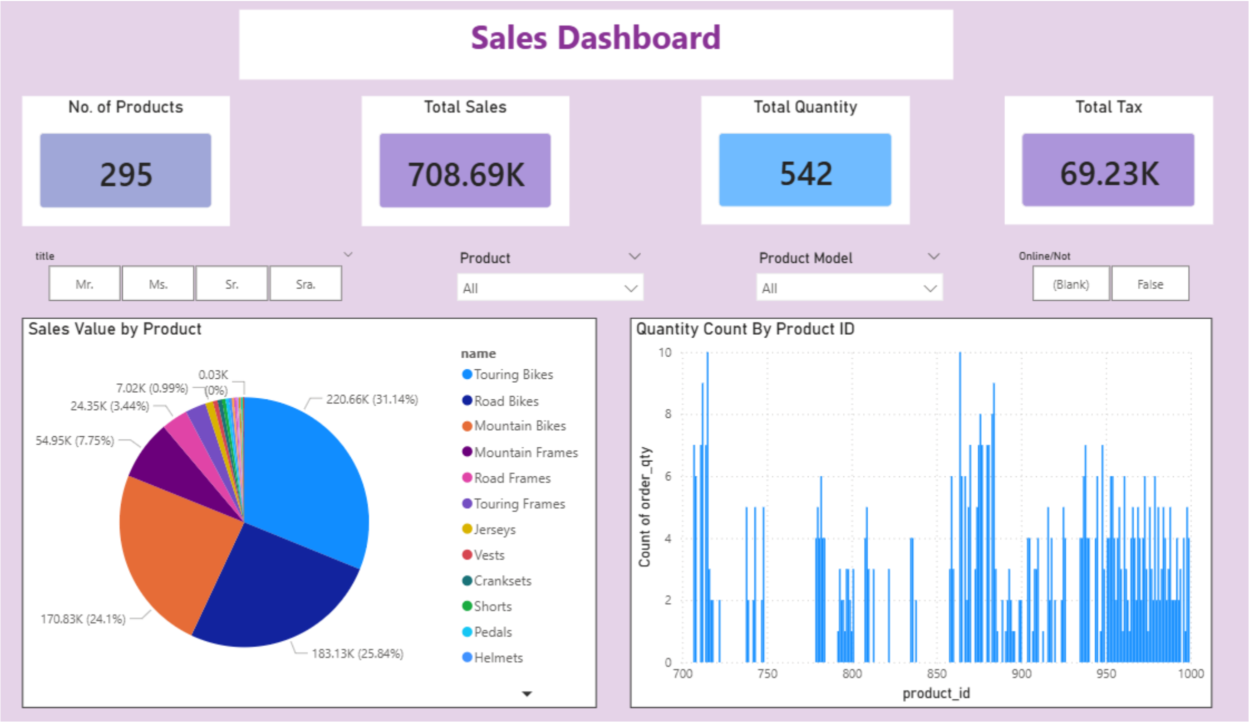
gold_db

Data Connectivity mode

● Import

○ DirectQuery

▷ Advanced options



Sales Dashboard

No. of Products: 295

Total Sales: 708.69K

Total Quantity: 542

Total Tax: 69.23K

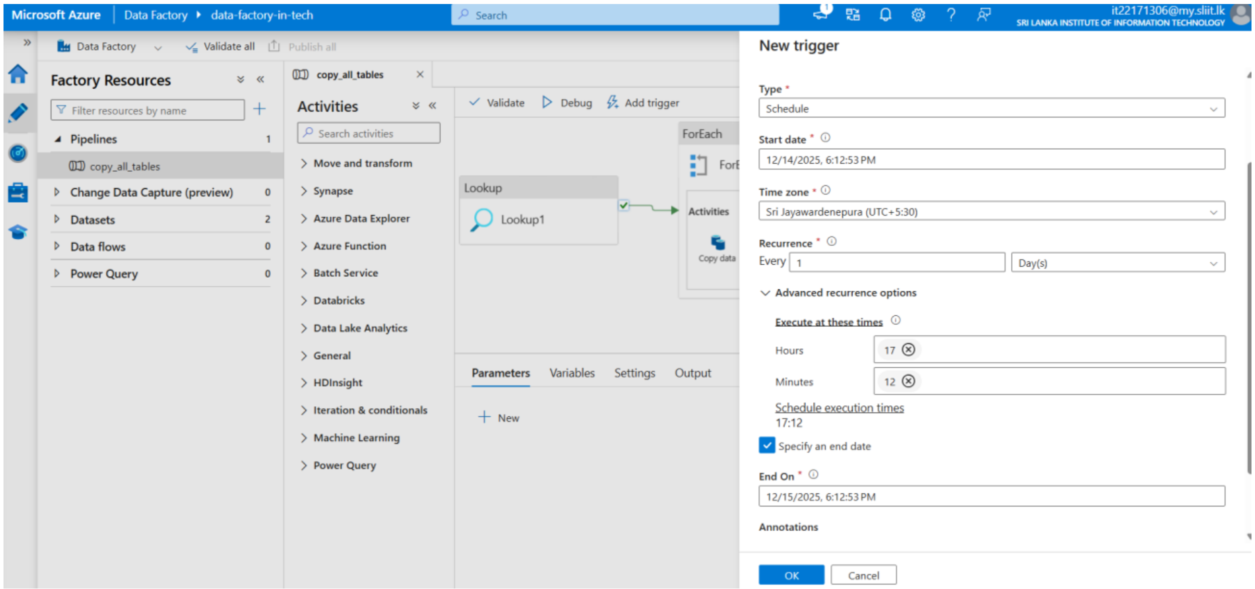
title: Mr. | Ms. | Sr. | Sra.

Product: All

Product Model: All

Online/Not: (Blank) | False

Adding Trigger in the Data Factory to run the pipeline



New trigger

Type *: Schedule

Start date *: 12/14/2025, 6:12:53 PM

Time zone *: Sri Jayawardenepura (UTC+5:30)

Recurrence *: Every 1 Day(s)

Advanced recurrence options

Execute at these times

Hours: 17

Minutes: 12

Schedule execution times
17:12

☑ Specify an end date

End On *: 12/15/2025, 6:12:53 PM

Annotations

OK | Cancel