

# AZURE DATA ENGINEERING REAL-TIME END-TO-END PROJECT

NGUYEN CONG LUC  
[nguyencongluc.82@gmail.com](mailto:nguyencongluc.82@gmail.com)  
[Luc Nguyen | LinkedIn](#)  
0329206845

## **INDEX**

<i>CHAPTER 1. Project Overview</i> .....	4
<b>1.1 Overview</b> .....	4
<b>1.2 The technologies involved</b> .....	4
<i>CHAPTER 2. On-Prem &amp; Cloud Setup</i> .....	6
<b>2.1 Setup sql server</b> .....	6
<b>2.2 Setup Key vault</b> .....	8
<i>CHAPTER 3. Data Ingestion (E)</i> .....	9
<b>3.1 Set up Microsoft Integration Runtime</b> .....	9
<b>3.2 Create Azure Data Factory pipeline</b> .....	10
<b>3.2.1 Create lookup in pipeline</b> .....	11
<b>3.2.2 Create forEach</b> .....	13
<b>3.2.3 Run pipeline</b> .....	19
<i>CHAPTER 4. Data Transformation(T)</i> .....	22
<b>4.1 Create notebook to transformation data in databricks</b> .....	22
<b>4.2 Link Azure databricks to pipeline in Azure data factory</b> .....	24
<i>CHAPTER 5. Data loading (L)</i> .....	30
<b>5.1 Create Synapse Workspace resource</b> .....	30
<b>5.2 Create View from data of ADL Gen 2(Layer 2)</b> .....	31
<i>CHAPTER 6. Data Reporting</i> .....	39
<b>6.1 Connect data from Azure synapse analyst</b> .....	39
<b>6.2 Design dashboard</b> .....	41
<i>CHAPTER 7. Pipelines Testing/Automation</i> .....	43
<i>CHAPTER 8. SUMMARY</i> .....	44

# **Introduction**

As a passionate and dedicated individual eager to start my journey in Data Engineering, I am excited to present a detailed Azure Data Engineering project that I have worked on to build foundational skills in this field.

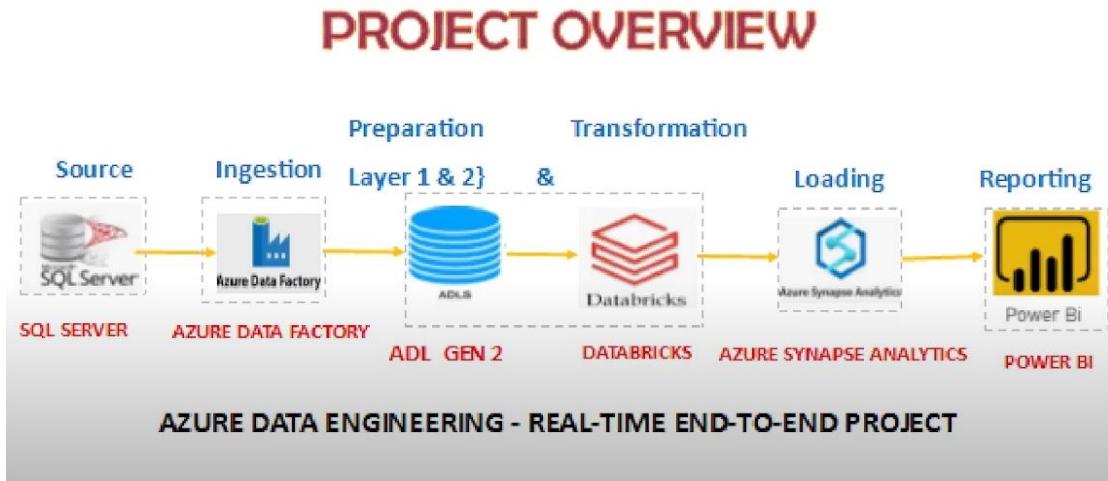
This project showcases my ability to learn and apply various Azure services, including SQL Server, Azure Data Factory (ADF), Azure Data Lake Storage (ADLS Gen 2), Databricks, Azure Synapse Analytics, and Power BI. Through this hands-on experience, I gained insights into data pipeline orchestration, data transformation, storage, and visualization processes.

While I may be at the beginning of my professional journey, I am confident that my enthusiasm for learning and my commitment to building scalable data solutions will make me a valuable addition to any team.

# CHAPTER 1. Project Overview

## 1.1 Overview

A comprehensive real-time, end-to-end Azure Data Engineering project. It showcases how to build a complete data pipeline using various Azure services.



Data used is sales data of Adventure Works Cycles

Adventure Works Cycles is a fictitious company that manufactures and sells bicycles, bicycle accessories, and related products, used as the setting in AdventureWorks, a sample database developed by Microsoft.

## 1.2 The technologies involved

Here's a breakdown of the technologies involved:

1. SQL Server: Acts as the source database for the initial data.
2. Azure Data Factory (ADF): Used for orchestrating data movement and transformations.
3. ADLS Gen 2 (Azure Data Lake Storage): Stores data in two layers. Layer 1 for raw data from SQL Server via Azure Data Factory and Layer 2 for processed data from Databricks.

4. Databricks: Implements advanced data processing and transformation, using Spark for big data workloads.

5. Azure Synapse Analytics: Powers data analysis and serves as the data warehouse for reporting and insights.

6. Power BI: Provides the visualization and reporting layer to deliver business insights

Resources		Recommendations (1)
<input type="button" value="Filter for any field..."/> <input type="button" value="Type equals all"/> <input type="button" value="Location equals all"/> <input type="button" value="Add filter"/>		
Showing 1 to 6 of 6 records. <input type="checkbox"/> Show hidden types <input type="radio"/>		
<input type="checkbox"/>	Name ↑↓	Type ↑↓
<input type="checkbox"/>	Azure-data-factory-2024-demo	Data factory (V2) Location ↑↓
<input type="checkbox"/>	Azure-Databricks-workspace-demo	Azure Databricks Service East Asia
<input type="checkbox"/>	azure-synapse-workspace-demo	Synapse workspace East Asia
<input type="checkbox"/>	azuredatfactorygen2	Storage account East Asia
<input type="checkbox"/>	azuresynapsewksgen2	Storage account East Asia
<input type="checkbox"/>	de-demo-project-key	Key vault East US

Microsoft Azure

Home > auto-data-engr-project Resource group

Overview    Activity log    Access control (IAM)    Tags    Resource visualizer    Events    Settings    Monitoring    Automation    Help

Subscription (auto) : Azure subscription 2 Subscription ID : b387ff99-74df-4f6a-b9aa-c50d4796bf2 Tags (edit) : Add tags

Deployments : 2 Deploying 1 Failed 11 Succeeded Location : East Asia

Resources    Recommendations (1)

Showing 1 to 6 of 6 records.  Show hidden types

<input type="checkbox"/>	Name ↑↓	Type ↑↓	Location ↑↓
<input type="checkbox"/>	Azure-data-factory-2024-demo	Data factory (V2)	East US
<input type="checkbox"/>	Azure-Databricks-workspace-demo	Azure Databricks Service	East Asia
<input type="checkbox"/>	azure-synapse-workspace-demo	Synapse workspace	East Asia
<input type="checkbox"/>	azuredatfactorygen2	Storage account	East Asia
<input type="checkbox"/>	azuresynapsewksgen2	Storage account	East Asia
<input type="checkbox"/>	de-demo-project-key	Key vault	East US

Microsoft Azure

Home > azuredatfactorygen2 Storage account

azuredatfactorygen2 | Containers

Containers    File shares    Queues    Tables

Overview    Activity log    Tags    Diagnose and solve problems    Access Control (IAM)    Data migration    Events    Storage browser    Partner solutions    Data storage    Containers    File shares    Queues    Tables

Search containers by prefix:

Name	Last modified	Anonymous access level	Lease state
Logs	12/27/2024, 12:47:09 AM	Private	Available
layer-1	12/29/2024, 6:22:22 AM	Private	Available
layer-2	12/29/2024, 6:22:34 AM	Private	Available

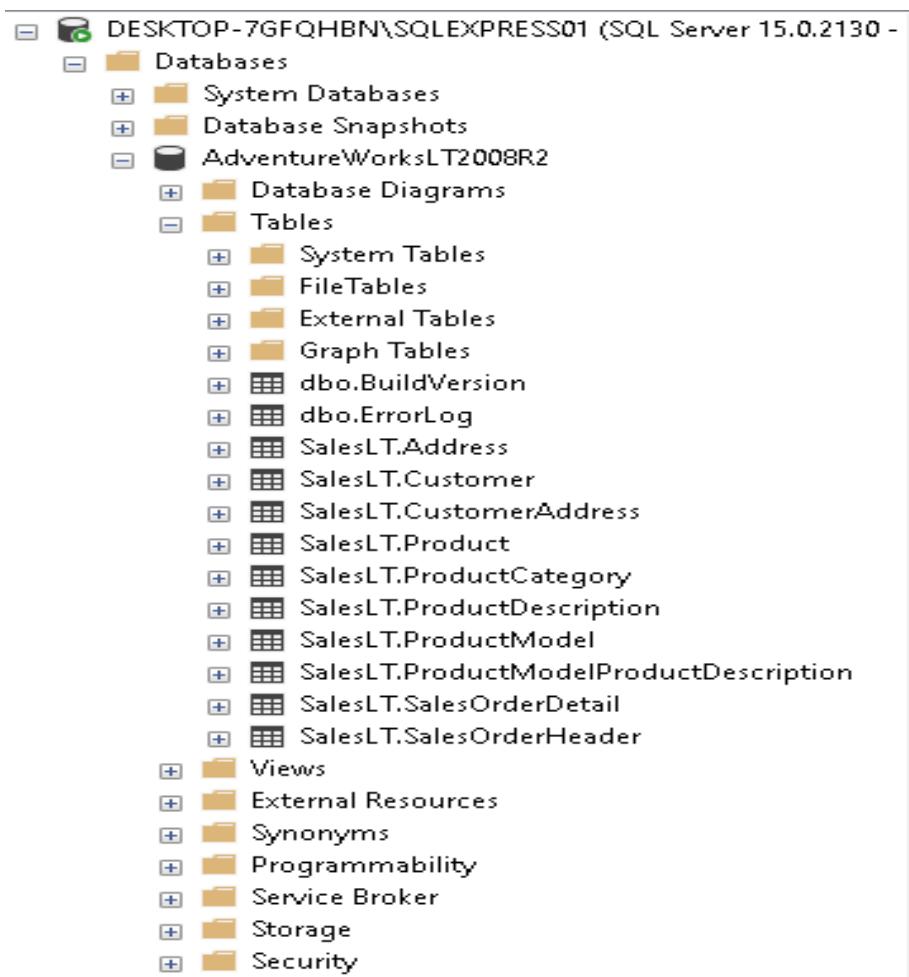
# CHAPTER 2. On-Prem & Cloud Setup

## 2.1 Setup sql server

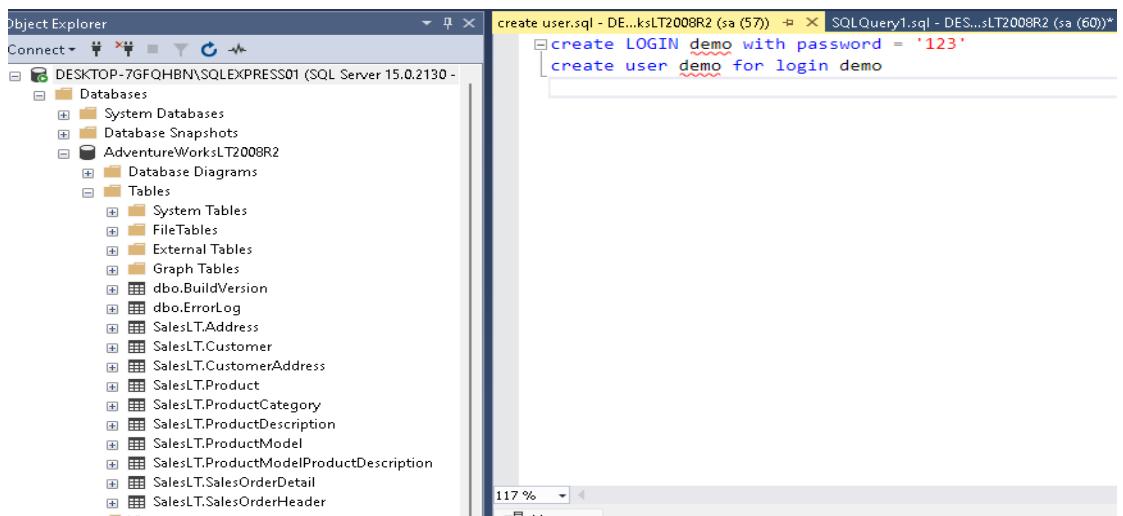
In this project, SQL Server acts as the source database, storing the initial raw data that is extracted, processed, and transformed through the Azure data pipeline.

Create Database SQL Server from A full database backup of AdventureWorksLT2008R2.

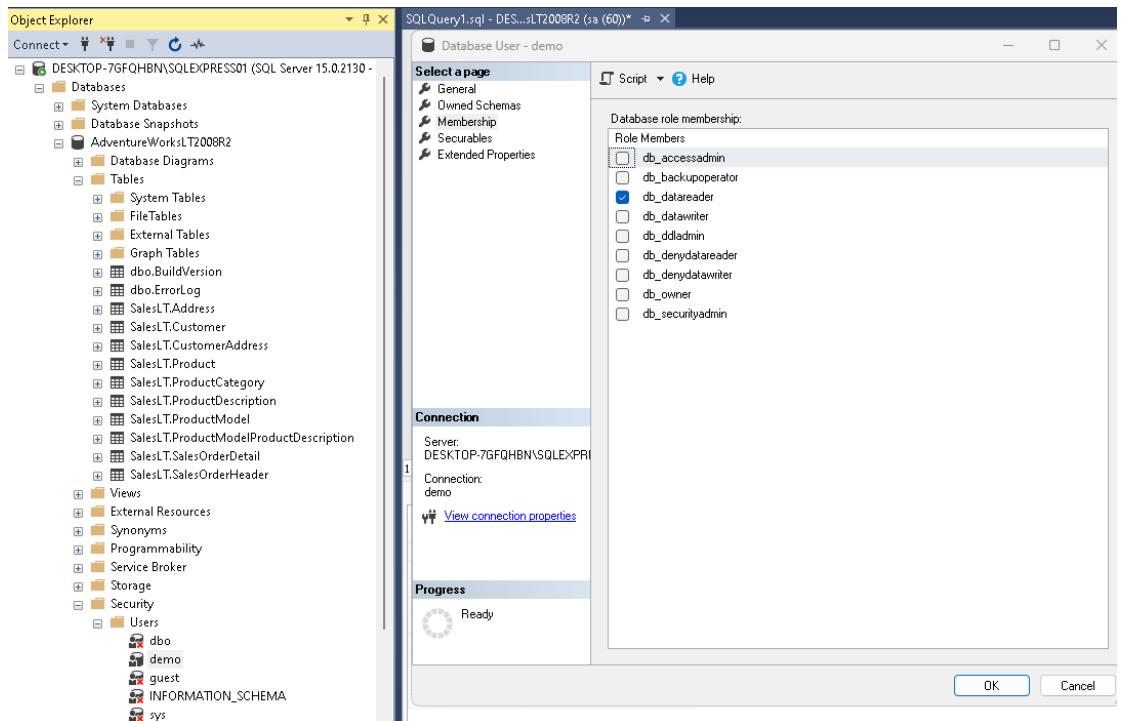
Resource: <https://github.com/Microsoft/sql-server-samples/releases/download/adventureworks2008r2/adventure-works-2008r2-Lt.bak>



## Create user for database:



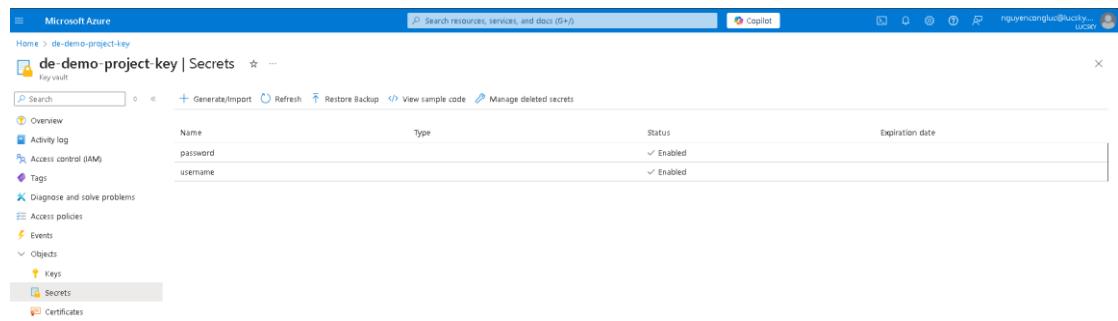
## Set read permissions for user



## 2.2 Setup Key vault

In this project, Azure Key Vault securely stores the SQL Server credentials (username and password) and other sensitive data. It enables authorized services like Azure Data Factory and Databricks to access these secrets securely using Managed Identity, ensuring encryption, controlled access, and seamless integration.

Create secrets with pass and username of account “demo”



The screenshot shows the Azure Key Vault interface for the 'de-demo-project-key' vault. The left sidebar has 'Keys' selected under 'Objects'. The main area displays two secrets in a table:

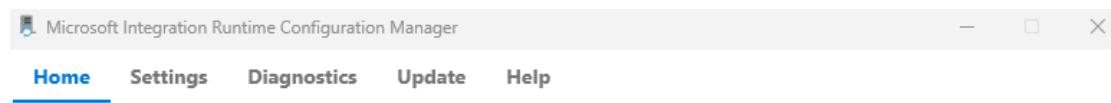
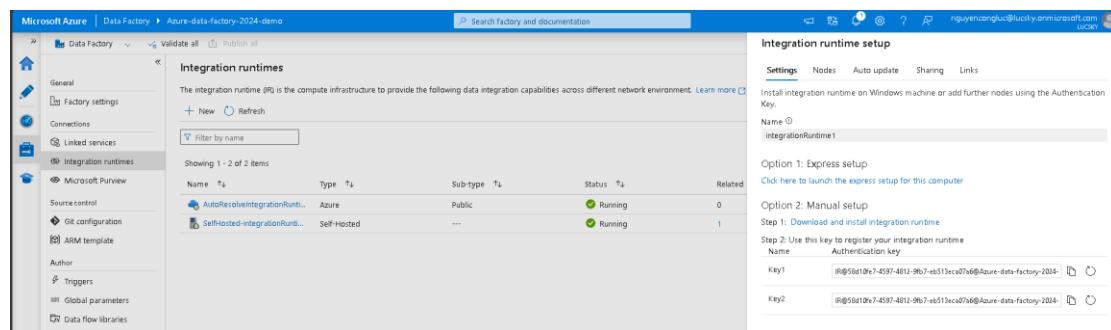
Name	Type	Status	Expiration date
password	password	✓ Enabled	
username	username	✓ Enabled	

# CHAPTER 3. Data Ingestion (E)

In this project, Azure Data Factory (ADF) orchestrates the data pipeline by extracting, transforming, and loading data between services like SQL Server, Azure Data Lake, and Databricks

## 3.1 Set up Microsoft Integration Runtime

In this project, Microsoft Integration Runtime enables secure and efficient data movement between on-premises SQL Server and Azure services like Data Factory.



### ✓ Self-hosted node is connected to the cloud service

Data Factory: Azure-data-factory-2024-demo  
Integration Runtime: SelfHosted-integrationRuntime1  
Node: DESKTOP-7GFQHBN

[Stop Service](#)

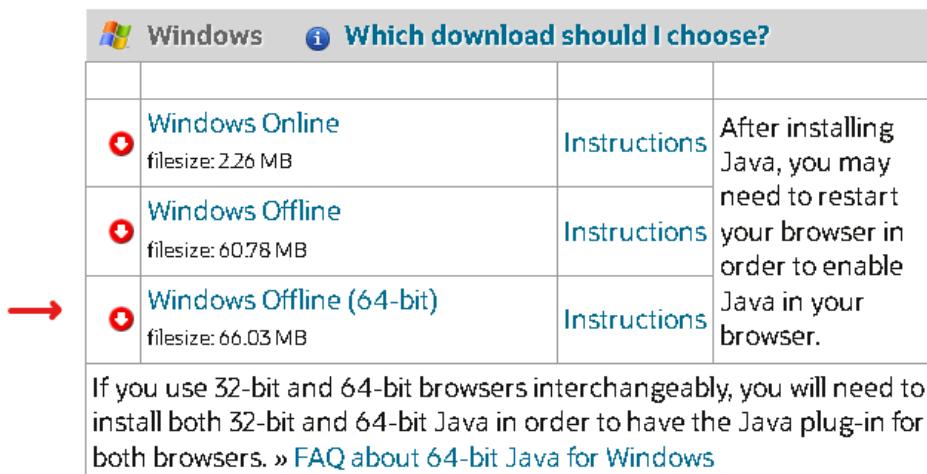
### Data Source Credential ⓘ

Credential store: On-premises  
Credential status: In sync  
Last backup time: N/A  
[Generate Backup](#) [Import Backup](#)

✓ Connected to the cloud service (Data Factory V2) [↻](#)

Setup JRE to running

## [Java Downloads for All Operating Systems](#)



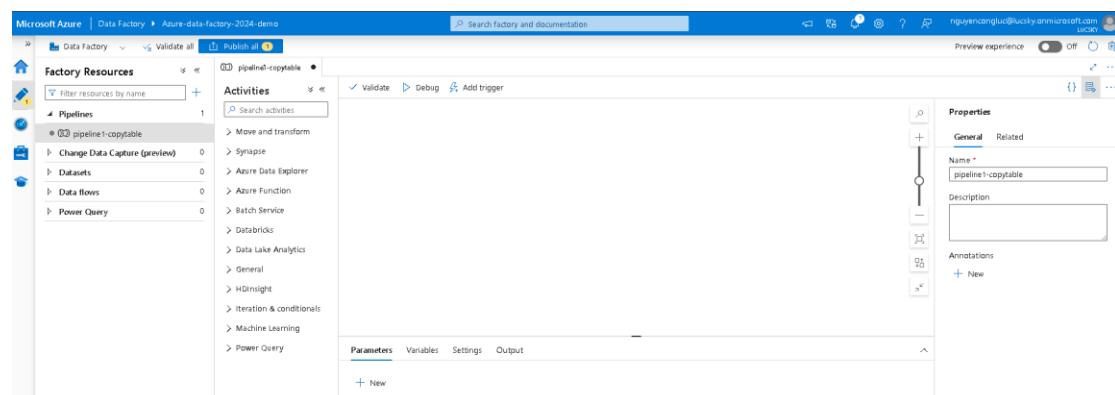
The screenshot shows a Windows operating system interface with a title bar "Windows" and a message "Which download should I choose?". Below is a table with three rows:

	<b>Windows Online</b> filesize: 2.26 MB	<a href="#">Instructions</a>
	<b>Windows Offline</b> filesize: 60.78 MB	<a href="#">Instructions</a>

A red arrow points to the third row. Below the table is a note: "If you use 32-bit and 64-bit browsers interchangeably, you will need to install both 32-bit and 64-bit Java in order to have the Java plug-in for both browsers. » [FAQ about 64-bit Java for Windows](#)".

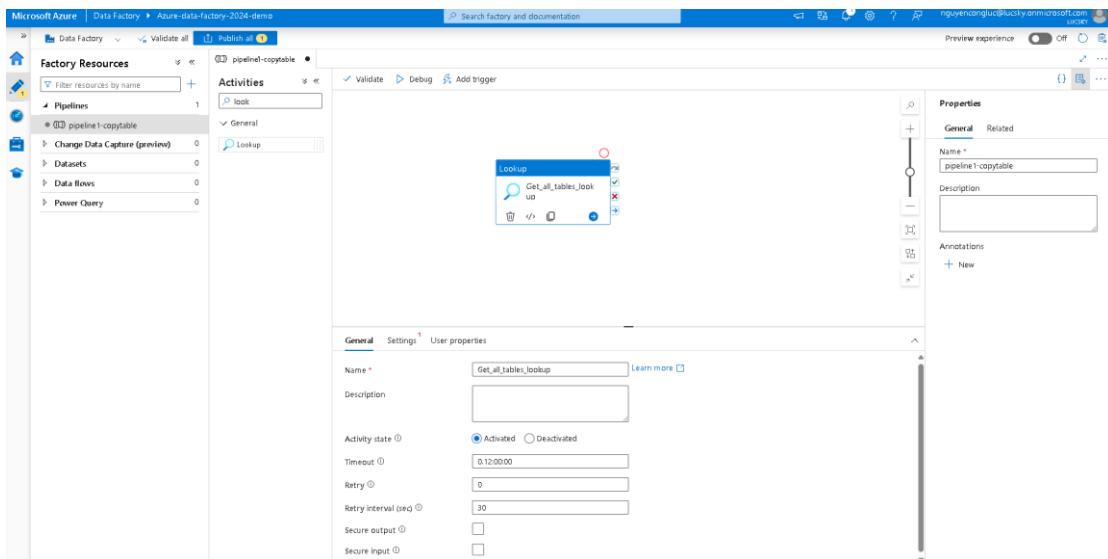
## 3.2 Create Azure Data Factory pipeline

In this project, the Azure Data Factory pipeline automates the movement and transformation of data from SQL Server to Azure Data Lake, Databricks, and other services, ensuring seamless integration and processing within the data pipeline.

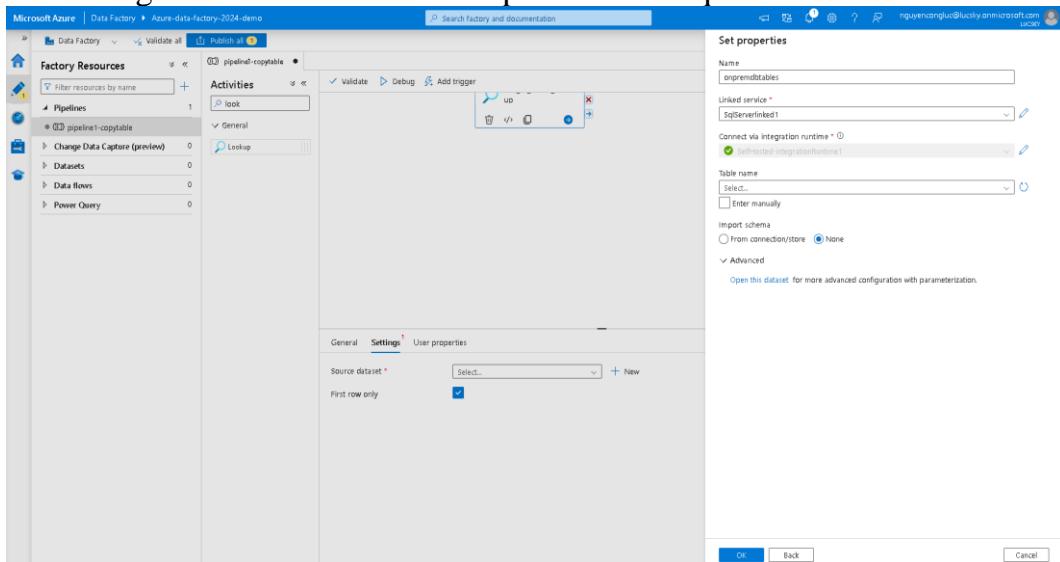


### 3.2.1 Create lookup in pipeline

Lookup activity can retrieve a dataset from any of the data sources supported like sql server



Setting source data set for lookup to connect sql server



## Create query copy all table and schema

SQLQuery1.sql - DE...2008R2 (demo (60))\*

```

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';

```

Results

	SchemaName	TableName
1	SalesLT	Address
2	SalesLT	Customer
3	SalesLT	CustomerAddress
4	SalesLT	Product
5	SalesLT	ProductCategory
6	SalesLT	ProductDescription
7	SalesLT	ProductModel
8	SalesLT	ProductModelProductDescription
9	SalesLT	SalesOrderDetail
10	SalesLT	SalesOrderHeader

## Add query to lookup

The screenshot shows the Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, Data flows, and Power Query. The main area shows a pipeline named 'pipeline1-copytable' with one activity: a 'Lookup' activity. The 'Activities' pane shows the 'Lookup' activity selected. The 'General' tab of the 'Settings' pane is active, showing the 'Source dataset' as 'SqlServerTable1' and the 'Query' as:

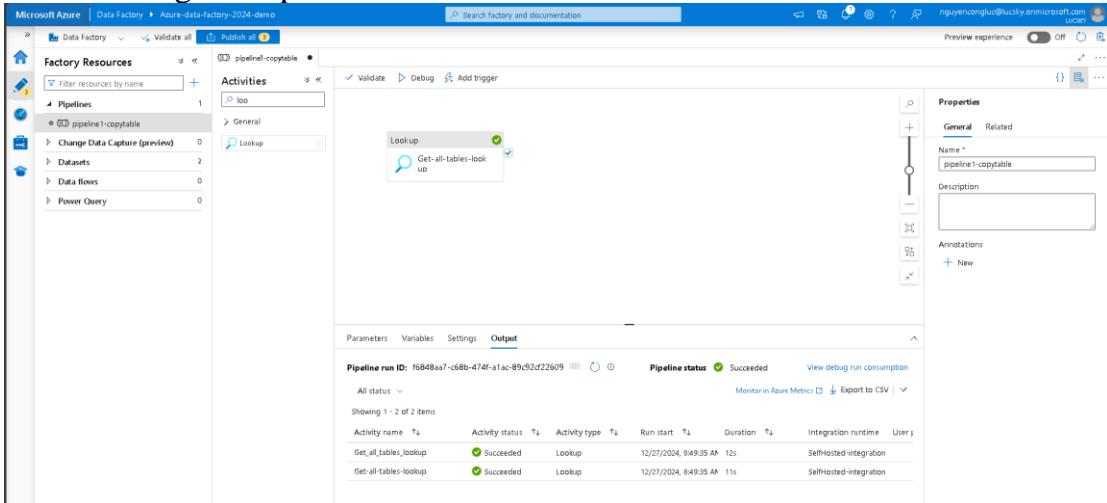
```

SELECT
    s.name AS SchemaName,
    t.name AS TableName

```

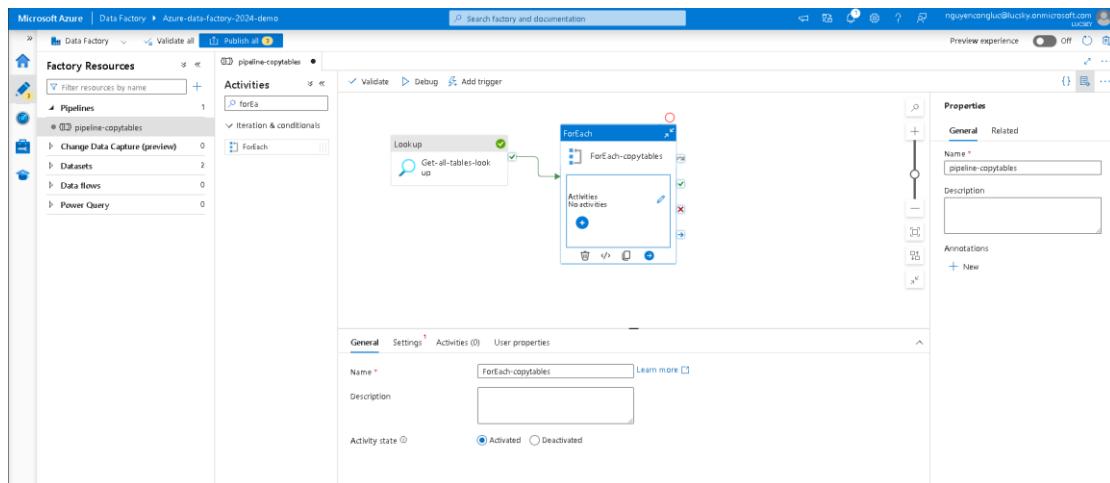
The 'Preview data' pane on the right shows the results of the query, which matches the table list from the previous screenshot.

## Debug lookup



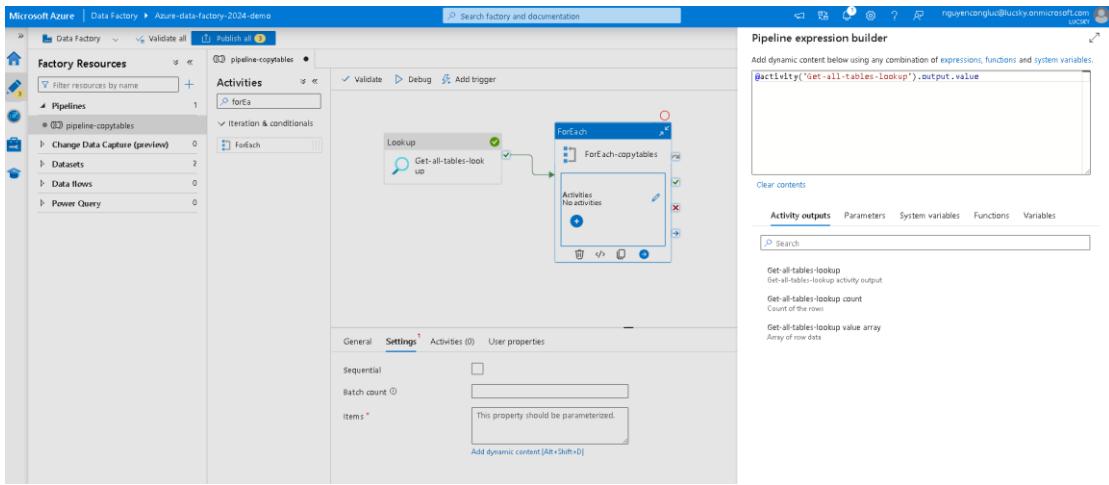
## 3.2.2 Create forEach

In this project, the ForEach activity allows you to iterate over the list of table names from the Lookup and perform actions (like data extraction or transformation) on each table



Add Pipeline expression builder to forEach.

The Pipeline Expression Builder in the ForEach activity's Items setting allows you to create dynamic expressions that reference outputs or parameters, enabling the pipeline to iterate over data (e.g., table names) and perform actions dynamically during the loop.

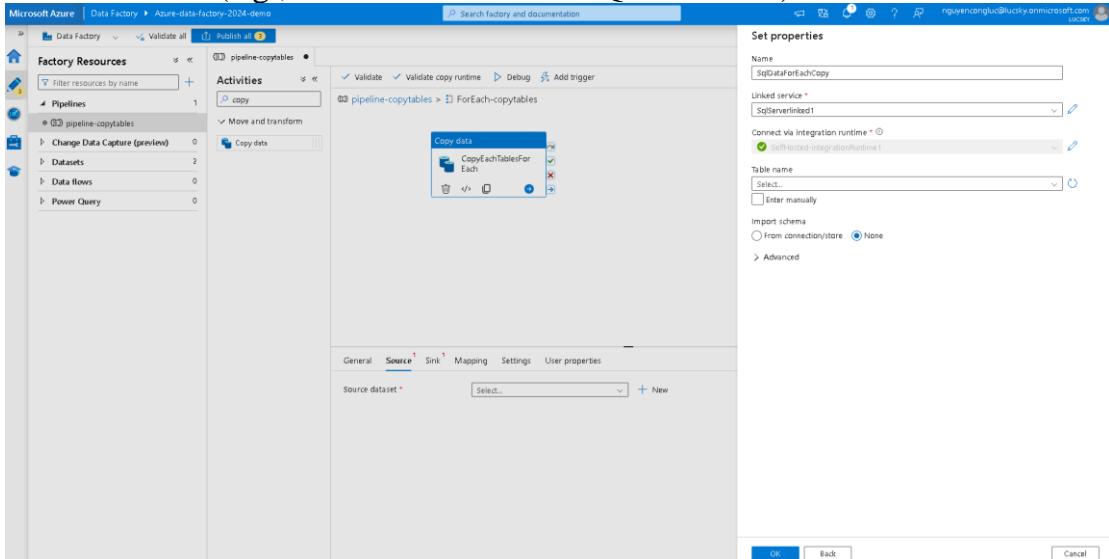


### Add Copy Data Activity inside forEach

In the project, adding a Copy Data activity inside ForEach allows you to dynamically copy data from each table in the schema to a target destination (e.g., Azure Data Lake or SQL Server) during the pipeline execution.

#### Add source data to Copy Data

In the Copy Data activity, the Source dataset defines the structure and location of the data to be copied. In your project, it specifies the source from which data will be extracted, such as SQL Server tables in the SalesLT schema. It tells the pipeline where to pull the data from before it is copied to the destination (e.g., Azure Data Lake or SQL Database).



#### Add query to Copy Data

The query dynamically generates and executes a query to select all data from each table in the SalesLT schema. The Copy Data activity then extracts this data and copies it to a specified destination (e.g., Azure Data Lake or another database).

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

The screenshot shows the Azure Data Factory pipeline editor. A pipeline named 'pipeline-copytables' is selected. Inside, there's a 'ForEach' loop named 'ForEach-copytables'. The 'Copy data' activity is configured with a 'Source dataset' of 'SqlDataForEachCopy' and a 'Query' of @concat('SELECT \* FROM ', item().SchemaName, '.', item().TableName). The 'Sink' tab is currently active.

### Add query to Copy Data

The query dynamically generates and executes a query to select all data from each table in the SalesLT schema. The Copy Data activity then extracts this data and copies it to a specified destination (e.g., Azure Data Lake or another database)

### Adding a Sink dataset

Set up file in Sink dataset: with the Parquet format in Azure Data Lake Gen 2 in the Copy Data activity allows you to store the copied data in a Parquet file format in Azure Data Lake Gen 2.

The screenshot shows the 'Sink' tab for the 'Copy data' activity. The 'Select format' section is displayed, showing various file formats: Avro, Binary, DelimitedText, Iceberg, JSON, ORC, and Parquet. The Parquet format is selected. The 'Continue' button is visible at the bottom right.

Set up linked service in Sink dataset:

It defines how ADF connects to external data stores like Azure Data Lake Storage Gen 2. The linked service provides authentication details, endpoints, and other necessary configurations for connecting to the storage account.

### Create layer-1 containers

## End setup sink

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A pipeline named 'pipeline-copytables' is selected. In the 'Activities' section, a 'Copy data' activity is highlighted. The 'Sink' tab is active, showing configuration for a 'ParquetDatasetLayer1'. The 'File path' is set to 'File system / Directory / File name'. The 'Import schema' section has 'From connection/store' selected. The 'Advanced' section includes a note about opening the dataset for advanced configuration with parameterization.

Set parameters for ParquetDatasetLayer1 in open of sink dataset

The schemaname and tablename parameters allow dynamic handling of data in the Sink dataset. They enable flexibility by letting you pass different schema and table names at runtime, which can be used to create dynamic file paths or names. This helps store data from various tables into different Parquet files without creating separate datasets for each one, making the pipeline reusable and scalable.

The screenshot shows the Microsoft Azure Data Factory dataset editor for 'ParquetDatasetLayer1'. The 'Parameters' tab is active, displaying two parameters: 'schemaname' and 'tablename', both of type string. The 'General' tab shows the dataset name and type. The 'Properties' pane on the right shows the dataset's name and description.

## Setup value for names

In this case, `@item().SchemaName`, `@item().TableName` refers to the schema name and table name that are dynamically passed through the pipeline (likely from a previous activity like Lookup or ForEach) for each table. This allows you to dynamically use the schema name, table name when copying data into the Parquet file in the Sink dataset.

The screenshot shows the Microsoft Azure Data Factory Pipeline expression builder interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, and Data flows. In the center, a pipeline named 'pipeline-copytables' is selected, and its activities are shown: a 'Copy' activity followed by a 'Move and transform' activity. The 'Copy' activity is expanded, showing its 'Sink' tab. The 'Sink dataset' dropdown is set to 'ParquetDatasetLayer1'. Under 'Dataset properties', two variables are defined: 'schemaname' with the value '`@item().SchemaName`' and 'tablename' with the value '`@item().TableName`'. The 'Mapping' tab is also visible, showing a single column mapping. On the right, the 'Pipeline expression builder' pane displays the expression `@item().SchemaName` with a tooltip 'Add dynamic content [Alt+Shift+D]'. Below it, the 'ForEach iterator' tab is selected, showing the current item 'ForEach-copytables'.

Setting the directory in the Parquet Sink dataset

Setting the directory as

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

in the Parquet Sink dataset is used to dynamically create a folder structure in your Azure Data Lake Storage. This concatenates the schema name and table name to define the folder where the Parquet file will be stored.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines', 'Datasets', and 'Data flows'. In the center, a pipeline named 'pipeline-copytables' is selected, showing a single step 'ParquetDatasetLayer1'. The 'File path' field for this step contains the expression `layer-1/{@concat(dataset().schemaname, '/', dataset().tablename)}`. To the right, the 'Pipeline expression builder' pane displays the expression with parameters 'schemaname' and 'tablename' listed below it.

Setting the file name as

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

in the Sink dataset for Parquet means the file will be named dynamically based on the tablename.

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

This screenshot is identical to the one above, showing the same pipeline structure and the expression `layer-1/{@concat(dataset().schemaname, '/', dataset().tablename)}` in the 'File path' field. The 'Pipeline expression builder' pane also shows the expression with parameters 'schemaname' and 'tablename'.

### 3.2.3 Run pipeline

Public all pipelines

The screenshot shows the Microsoft Azure Data Factory interface with the 'Factory Resources' sidebar. The 'Publish all' button at the top is highlighted in blue. The pipeline 'pipeline-copytables' is visible in the center.

## Run pipeline

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. In the center, there's a configuration dialog for a 'CopyEachTable' activity. The dialog has tabs for General, Source, Sink, Mapping, Settings, and User properties. Under the Sink tab, it's set to 'ParquetDatasetLayer1'. Below that, under 'Dataset properties', there are fields for 'Name' (set to '@Item().SchemaName') and 'tablename' (set to '@Item().TableName'). At the bottom right of the dialog, there's a red arrow pointing to the 'OK' button.

## Monitor

The screenshot shows the Microsoft Azure Data Factory Monitor. On the left, the navigation menu includes 'Dashboards', 'Runs', 'Pipeline runs' (which is selected), 'Trigger runs', 'Change Data Capture (preview)', 'Runtimes & sessions', 'Integration runtimes', 'Data flow debug', and 'Notifications'. The main area displays the 'pipeline-copytables - Activity runs' section. It shows a timeline of activities: a 'Lookup' activity followed by an 'ForEach' activity containing a 'CopyEachTable' activity. Below this, a table lists 12 items from the most recent run, showing details like activity name, status, type, start time, duration, runtime, and user properties. At the bottom right, there are links for 'Monitor in Azure Metrics' and 'Export to CSV'.

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID	Log
Get-all-tables-lookup	Succeeded	Lookup	12/27/2024, 1:06:25 PM	16s	Self-Hosted-integrationRi		72f6942b-2fcf-4b1c-ab49-726319b0371b	<a href="#">View Log</a>
ForEach-copytables	Succeeded	ForEach	12/27/2024, 1:06:42 PM	39s			ab103003-e6d9-4f5f-ba55-993ce1f0f019	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	27s	Self-Hosted-integrationRi		37a3a0e6-d0b6-4734-a317-fe0697eb0dc	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	29s	Self-Hosted-integrationRi		54cb9bf5-4530-48ff-9cd1-2aa99279fc3	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	29s	Self-Hosted-integrationRi		04690bcf-01d4-4b9e-baef-1d053f7bd09e	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	29s	Self-Hosted-integrationRi		8ec9611e-4dc5-4bd5-ab99-2f3d3ab1093	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	31s	Self-Hosted-integrationRi		78771101-31e9-4955-bea2-9fe21523d038	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	37s	Self-Hosted-integrationRi		c366606d-5107-471c-bfbd-bf165cb4fb1	<a href="#">View Log</a>
CopyEachTableForEach	Succeeded	Copy data	12/27/2024, 1:06:42 PM	37s	Self-Hosted-integrationRi		b7ee7ffe-5bf4-456d-a7a7-fd1c79fa999	<a href="#">View Log</a>

## Data stored at Azure data lake storage Gen 2

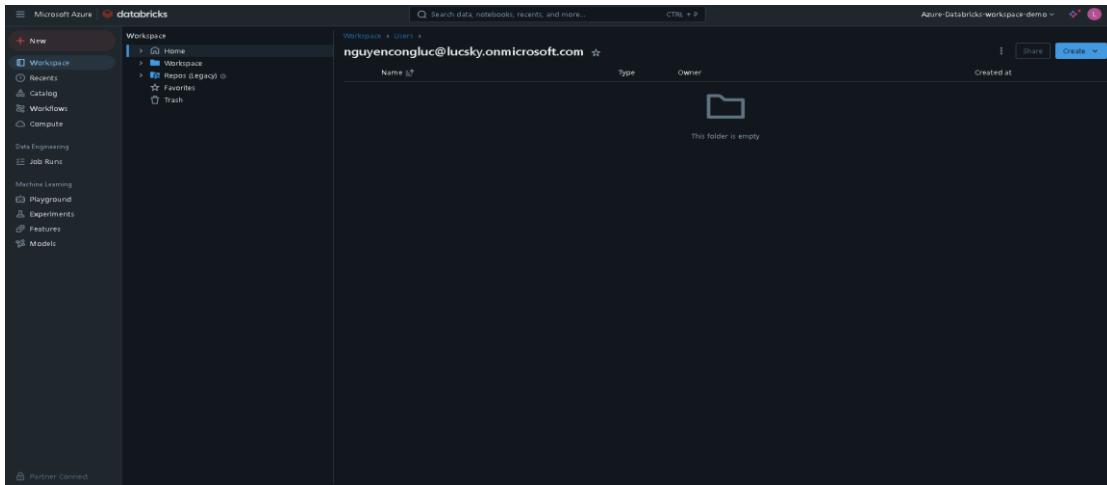
The screenshot shows the Azure Data Lake Storage Gen 2 interface for the 'layer-1' container. The left sidebar includes links for Overview, Diagnose and solve problems, Access Control (IAM), Settings (Shared access tokens, Manage ACL, Access policy, Properties, Metadata), and a search bar. The main area displays a table of stored blobs with columns: Name, Modified, Access tier, Archive status, Blob type, Size, and Lease state. The table lists 12 blobs, all of which are 'Standard' access tier, 'Standard' archive status, and blob type, with sizes ranging from 1.07 KB to 1.07 MB and modified dates from 12/27/2024, 1:07:04 PM to 12/27/2024, 1:07:05 PM.

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[3]	12/27/2024, 1:07:06 PM				-	***
Address	12/27/2024, 1:07:04 PM				-	***
Customer	12/27/2024, 1:07:01 PM				-	***
CustomerAddress	12/27/2024, 1:07:07 PM				-	***
Product	12/27/2024, 1:07:05 PM				-	***
ProductCategory	12/27/2024, 1:07:05 PM				-	***
ProductDescription	12/27/2024, 1:07:16 PM				-	***
ProductModel	12/27/2024, 1:07:07 PM				-	***
ProductModelProductDescription	12/27/2024, 1:07:08 PM				-	***
SalesOrderDetail	12/27/2024, 1:07:09 PM				-	***
SalesOrderHeader	12/27/2024, 1:07:05 PM				-	***

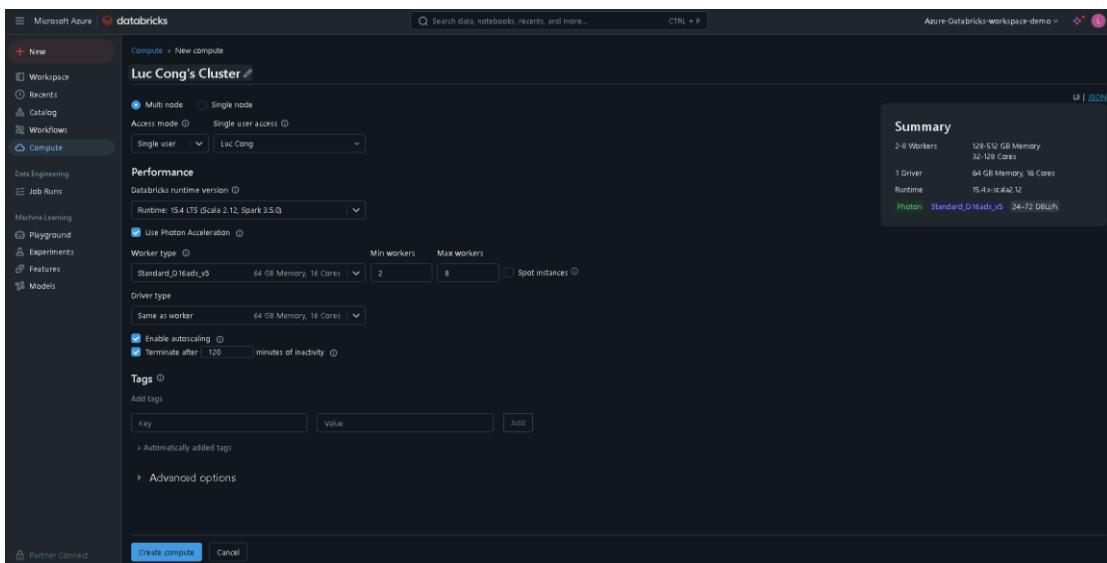
# CHAPTER 4. Data Transformation(T)

## 4.1 Create notebook to transformation data in databricks

Using Databricks for transformation data

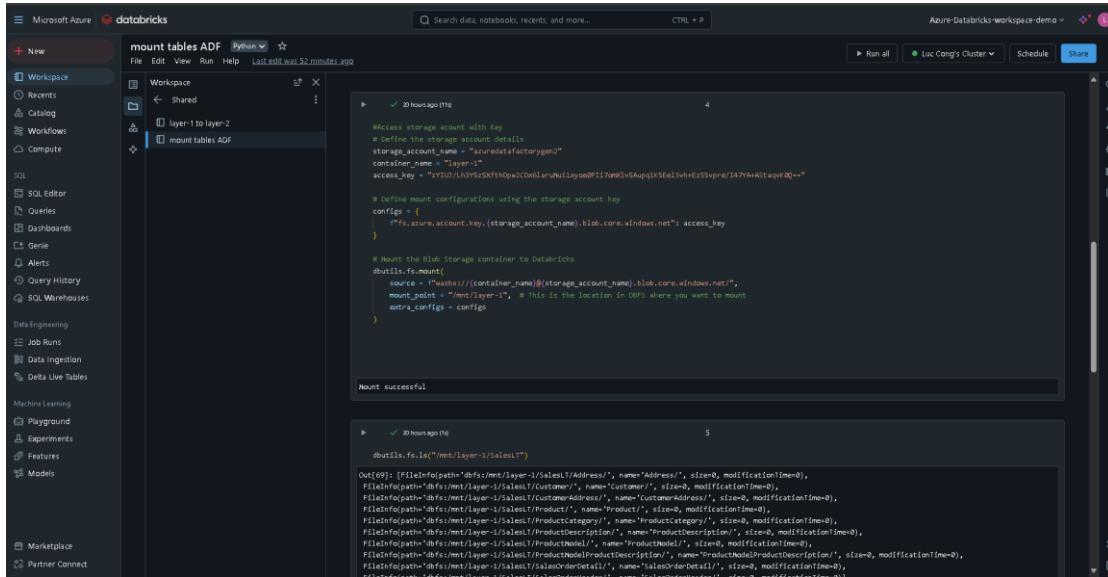


Create cluster



## Connect Databricks Azure data lake storage gen 2

This code allows Databricks to interact with Azure data lake storage gen 2 as if it's part of the local file system by mounting the specified container. Once mounted, you can access the files stored in that container directly within Databricks



```
# Mount the Blob Storage container to Databricks
dbutils.fs.mount(
    source = "wasbs://layer-1@storageaccountname.blob.core.windows.net",
    mount_point = "/mnt/layer-1", # This is the location in DBFS where you want to mount
    extra_configs = configs
)
```

Mount successful

```
[1]: 20 hours ago (8)
#dbutils.fs.ls("/mnt/layer-1/SalesLT")
Out[1]: [FileInode(path='dbfs:/mnt/layer-1/SalesLT/Address', name='Address', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/Customer', name='Customer', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/CustomerDemographic', name='CustomerDemographic', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/ProductCategory', name='ProductCategory', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/ProductDescription', name='ProductDescription', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/ProductModel', name='ProductModel', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/ProductModelProductDescription', name='ProductModelProductDescription', size=0, modificationTime=0),
FileInode(path='dbfs:/mnt/layer-1/SalesLT/SalesOrderDetail', name='SalesOrderDetail', size=0, modificationTime=0),
```

## Data transformation to all the SalesLT tables

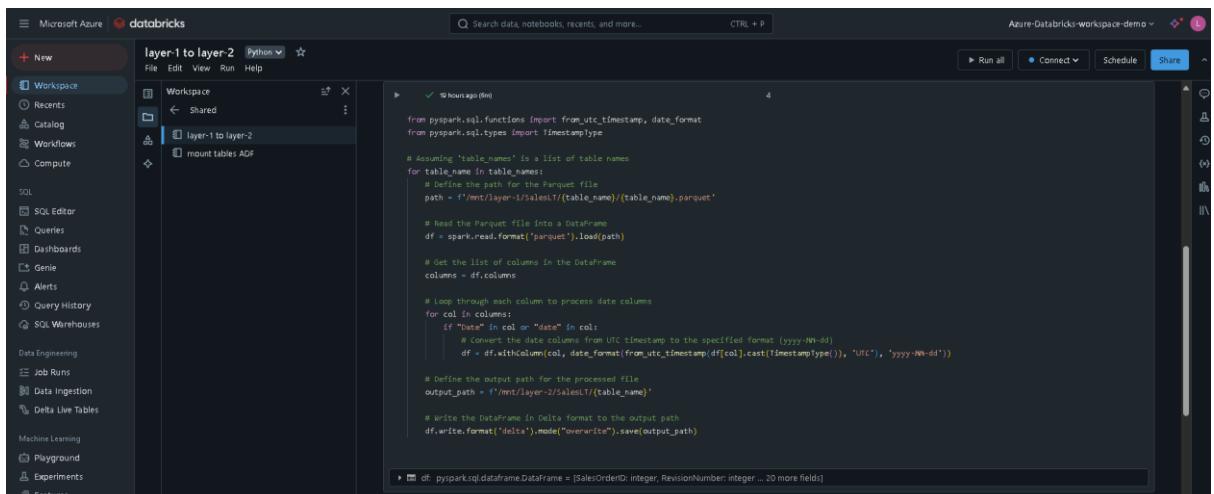


```
table_names = []

# Assuming dbutils.fs.ls('mnt/layer-1/SalesLT') returns a list of objects with 'name' attributes
for i in dbutils.fs.ls('mnt/layer-1/SalesLT'):
    table_names.append(i.name.split('/')[0])
print(table_names)
```

```
[1]: ['Address', 'Customer', 'CustomerAddress', 'Product', 'ProductCategory', 'ProductDescription', 'ProductModel', 'ProductModelProductDescription', 'SalesOrderDetail', 'SalesOrderHeader']
```

## Address column need convert to other data type, from date time to timestamp



```
from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

# Assuming 'table_names' is a list of table names
for table_name in table_names:
    # Define the path for the Parquet file
    path = f'mnt/layer-1/SalesLT/{table_name}/{table_name}.parquet'

    # Read the Parquet file into a DataFrame
    df = spark.read.format('parquet').load(path)

    # Get the list of columns in the DataFrame
    columns = df.columns

    # Loop through each column to process date columns
    for col in columns:
        if 'Date' in col or 'Date' in col:
            # Convert the date columns from UTC timestamp to the specified format (yyyy-MM-dd)
            df = df.withColumn(col, date_format(from_utc_timestamp(df[col].cast(TimestampType())), 'yyyy-MM-dd'))

    # Define the output path for the processed file
    output_path = f'/mnt/layer-2/SalesLT/{table_name}'

    # Write the DataFrame in Delta format to the output path
    df.write.format('delta').mode("overwrite").save(output_path)
```

## Result

	SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber
1	71774	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071774
2	71776	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071776
3	71780	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071780
4	71782	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071782
5	71783	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071783
6	71784	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071784
7	71796	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071796
8	71797	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071797
9	71815	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071815
10	71816	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071816
11	71831	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071831
12	71832	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071832
13	71845	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071845
14	71846	1	2004-06-01	2004-06-13	2004-06-08	5	false	S071846

Layer 2- Clean data created by Databricks is created at Azure data lake gen

2

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
\$_\$azurertmpfolder\$	12/28/2024, 10:09:55 AM				-	...
SalesLT	12/28/2024, 10:09:43 AM				-	...

## 4.2 Link Azure databricks to pipeline in Azure data factory

Adding a Databricks linked service in Azure Data Factory (ADF) allows ADF to connect and interact with Databricks. This enables ADF to trigger Databricks notebooks or jobs within its pipelines for data transformation, automate workflows, and monitor execution. It integrates Databricks into the ETL process, ensuring seamless collaboration between ingestion (ADF) and transformation (Databricks).

The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a navigation sidebar with various options like General, Connections, Linked services, Integration runtimes, Microsoft Purview, Source control, Author, Triggers, Global parameters, Data flow libraries, Security, Credentials, Customer managed key, Outbound rules, Managed private endpoints, Workflow orchestration manager, and Apache Airflow. The 'Linked services' option is selected. In the main area, under 'Linked services', it says 'Linked service defines the connection information to a data store or compute.' There's a 'New' button and a search bar. Below that, there's a table showing two existing linked services: 'AzureDataLakeStorageLinkedLayer1' (Type: Azure Data Lake Storage Gen2) and 'SqlServerLinked1' (Type: SQL server). A 'Create' button is at the bottom.

## Create access token to access databricks on azure data factory

dapi632317582e85def72eee6294afff37fb-2

The screenshot shows the Azure Databricks workspace settings page. The left sidebar has options like New, Workspace, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. Under 'Settings', 'Access tokens' is selected. It says 'Personal access tokens can be used for secure authentication to the [Databricks API] instead of passwords.' There's a 'Generate new token' button. A table lists one token: 'Comment': '2024-12-29 04:54:00 +07', 'Creation': '2024-08-29 04:54:00 +07', and 'Expiration': '2025-08-29 04:54:00 +07'. A 'Delete' icon is next to the expiration date.

## Save access token at secret of Key Vault to security

The screenshot shows the Microsoft Azure Key Vault 'Create a secret' form. At the top, it says 'Subscription 3' and 'Copilot'. The form has fields for 'Name' (set to 'AzureDatabricksAccessToken'), 'Secret value' (containing 'dapi632317582e85def72eee6294afff37fb-2'), 'Content type (optional)', 'Set activation date' (unchecked), 'Set expiration date' (unchecked), 'Enabled' (set to 'Yes'), and 'Tags' (0 tags).

## Create databricks linked service successfull

Microsoft Azure | Data Factory | Azure-data-factory-2024-demo | Search factory and documentation

General Factory settings Linked services Integration runtimes Microsoft Purview Source control Git configuration ARM template Author Triggers Global parameters Data flow libraries Security Credentials Customer managed key Outbound rules Managed private endpoints Workflow orchestration manager Apache Airflow

Linked services

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

+ New

Annotations : Any

Showing 1 - 2 of 2 items

Name	Type	Related
AzureDataLakeStorageLinkedLayer1	Azure Data Lake Storage Gen2	1
SqlServerlinked1	SQL server	3

New linked service

Connect via integration runtime \*  AutoResolveIntegrationRuntime

Auto selection method \*  From Azure subscription  Enter manually

Azure subscription \*  Azure subscription 3 (03b7f999-74df-406a-b9aa-c5c0d4796825)

Databricks workspace \*  Azure-Databricks-workspace-demo

Select cluster  Existing interactive cluster  Existing instance pool

Databrick Workspace URL \*  https://adb-11501579347397717.azure.databricks.net

Authentication type \*  Access Token  Azure Key Vault

Access token \*

Choose from existing clusters \*  Luc Cong's Cluster

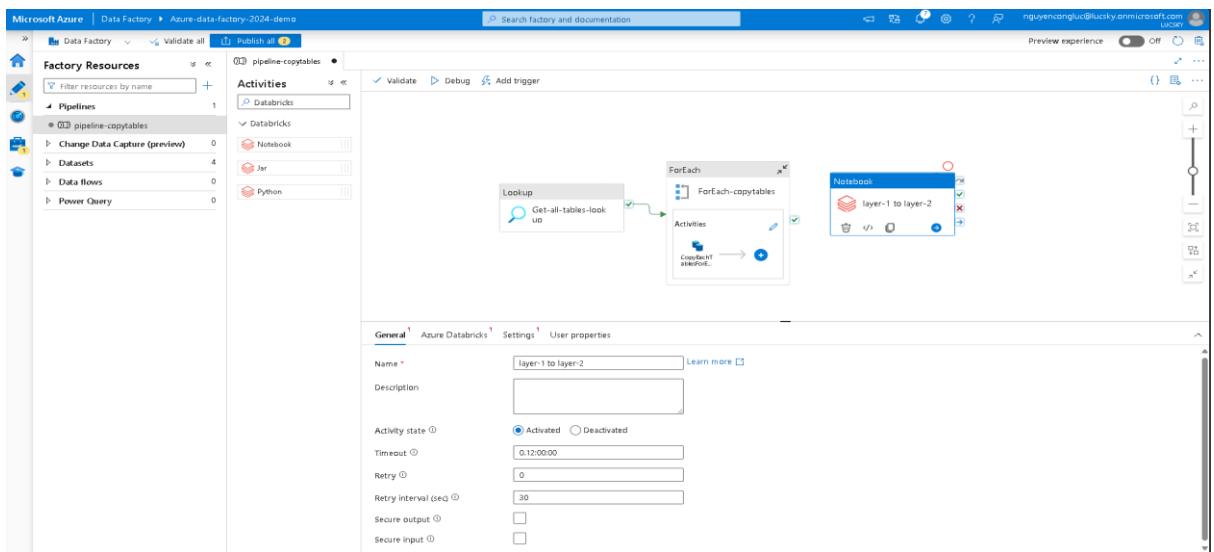
Annotations

> Parameters

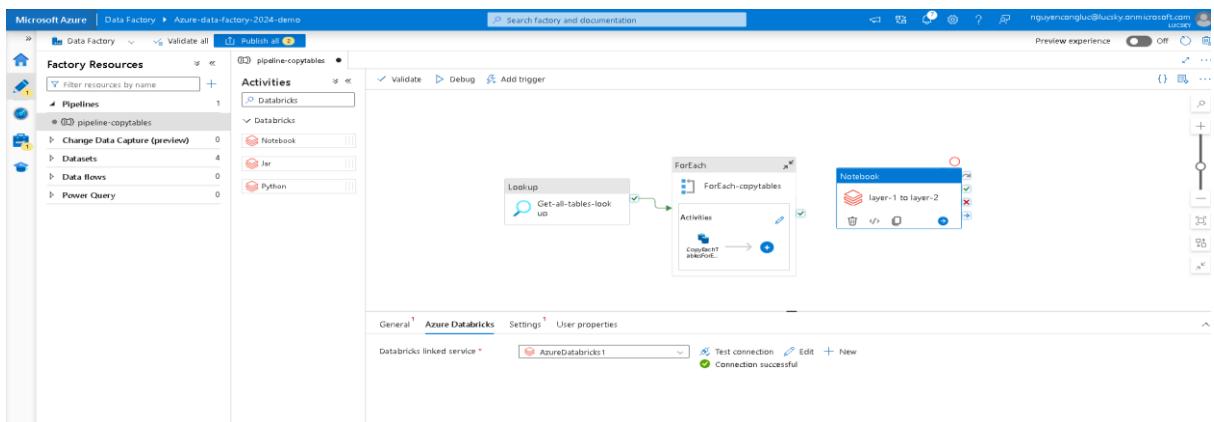
The screenshot shows the 'Linked services' section of the Azure Data Factory blade. It lists two existing linked services: 'AzureDataLakeStorageLinkedLayer1' (Type: Azure Data Lake Storage Gen2) and 'SqlServerlinked1' (Type: SQL server). Below this, there is a form for creating a new linked service. The 'Connect via integration runtime' field has 'AutoResolveIntegrationRuntime' selected. The 'Auto selection method' field has 'From Azure subscription' selected. The 'Azure subscription' dropdown shows 'Azure subscription 3 (03b7f999-74df-406a-b9aa-c5c0d4796825)' with a checked checkbox. The 'Databricks workspace' dropdown shows 'Azure-Databricks-workspace-demo' with a checked checkbox. Under 'Select cluster', 'Existing interactive cluster' is selected. The 'Databrick Workspace URL' field contains the URL 'https://adb-11501579347397717.azure.databricks.net'. The 'Authentication type' field has 'Access Token' selected. The 'Access token' field contains a redacted token. The 'Choose from existing clusters' dropdown shows 'Luc Cong's Cluster' with a checked checkbox. At the bottom, there are buttons for 'Create', 'Back', 'Test connection', and 'Cancel'.

## Adding a Databricks Notebook Activity in pipeline

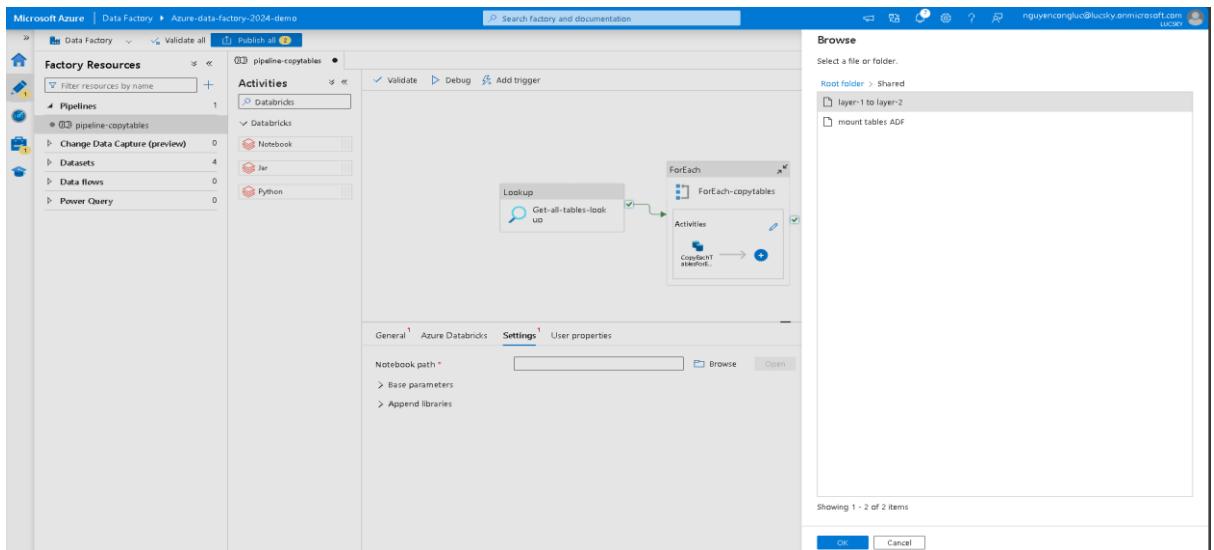
Adding a **Databricks Notebook Activity** named "**layer-1 to layer-2**" into an Azure Data Factory (ADF) pipeline is used for **data transformation**. Specifically, this activity processes and refines data from **Layer-1 (raw or ingested data)** to **Layer-2 (cleaned or structured data)** in the data lake.



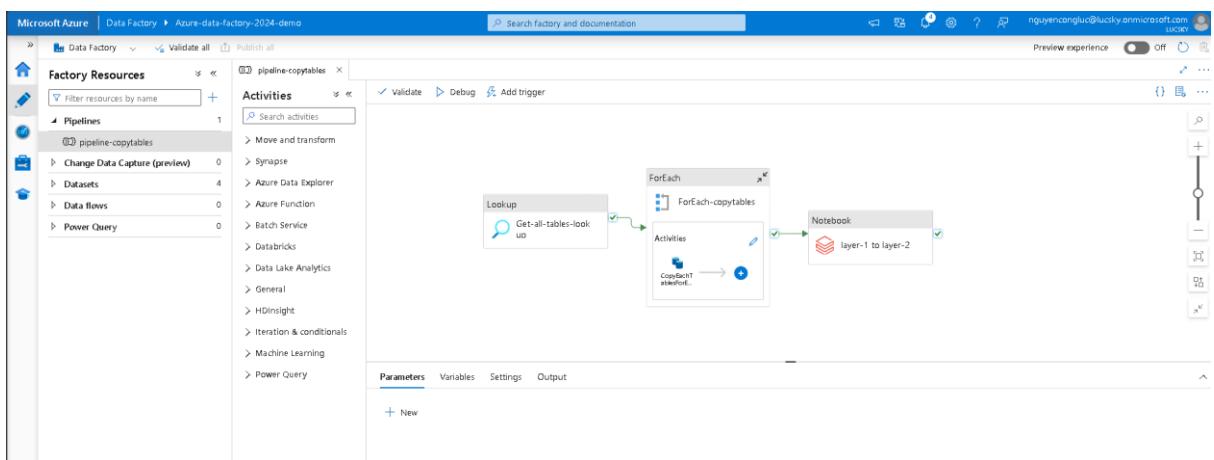
## Set Databricks linked service



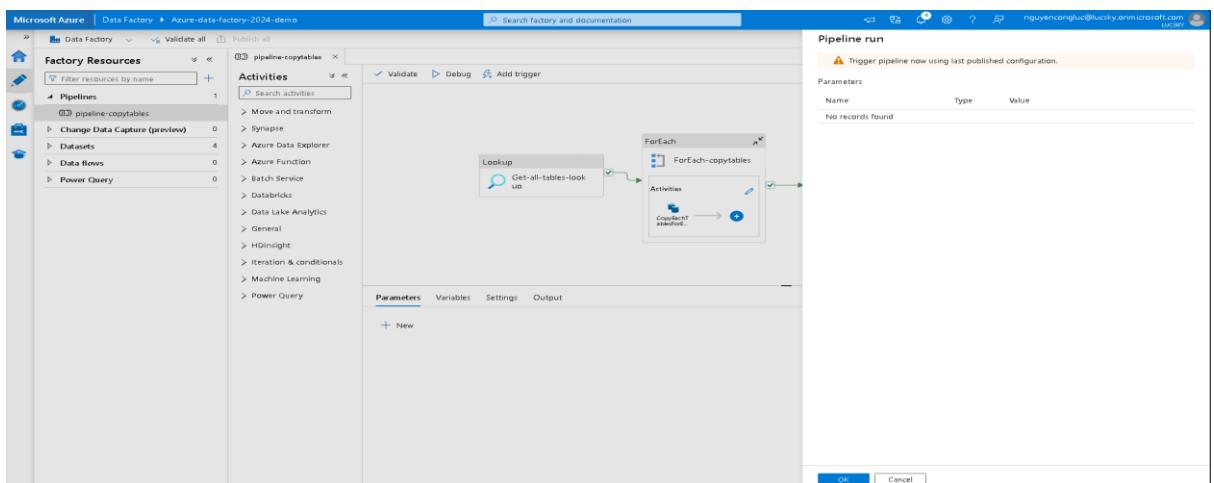
## Set notebook path



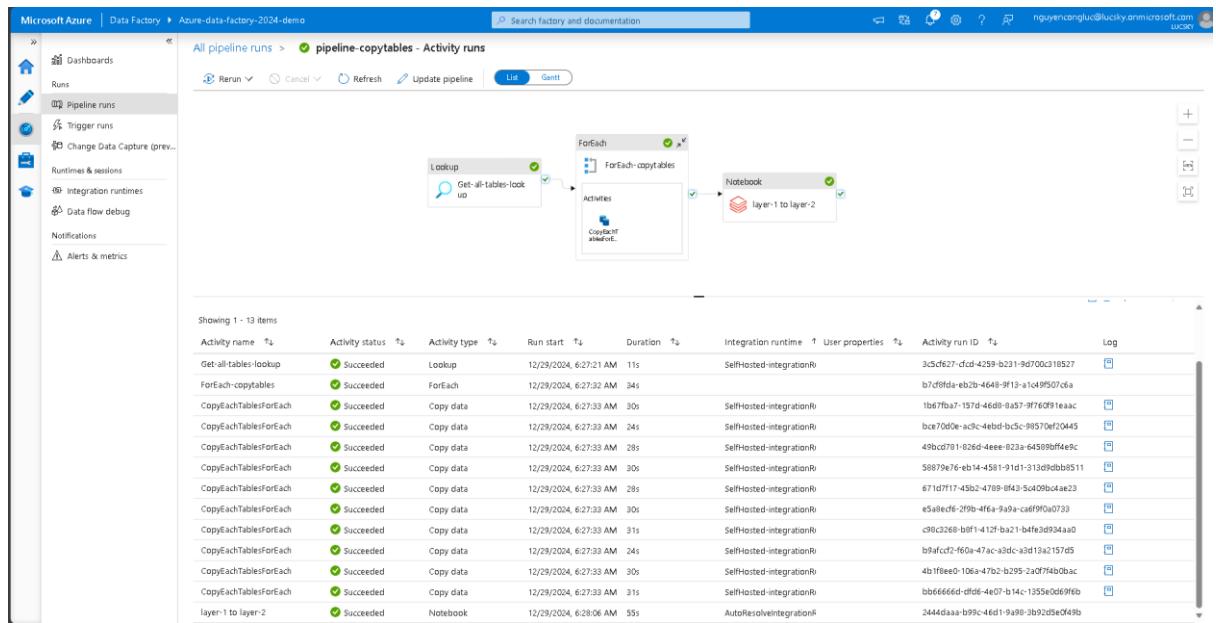
## Complete pipeline for data



## Running pipeline



## successful result



## Processed data

The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar navigation includes 'Home', 'azuredatfactorygen2', 'Containers' (selected), 'blob', 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', 'Settings' (expanded to show 'Shared access tokens', 'Manage ACL', 'Access policy', 'Properties', and 'Metadata'), and 'Metadata'. The main area displays the 'layer-2' container details. It shows the authentication method as 'Access key' (Switch to Microsoft Entra user account) and the location as 'layer-2 / SalesLT'. A search bar at the top right allows searching by blob prefix (case-sensitive). The table below lists the blobs in the container:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[...]	12/29/2024, 6:28:19 AM				-	---
Address	12/29/2024, 6:28:22 AM				-	---
Customer	12/29/2024, 6:28:25 AM				-	---
CustomerAddress	12/29/2024, 6:28:28 AM				-	---
Product	12/29/2024, 6:28:31 AM				-	---
ProductCategory	12/29/2024, 6:28:34 AM				-	---
ProductDescription	12/29/2024, 6:28:37 AM				-	---
ProductModel	12/29/2024, 6:28:40 AM				-	---
ProductModelProductDescription	12/29/2024, 6:28:43 AM				-	---
SalesOrderDetail	12/29/2024, 6:28:46 AM				-	---
SalesOrderHeader	12/29/2024, 6:28:48 AM				-	---

# CHAPTER 5. Data loading (L)

## 5.1 Create Synapse Workspace resource

Synapse Workspace is used to load and prepare data for enterprise-level analytics and reporting..

The screenshot shows the 'Create Synapse workspace' wizard in the Microsoft Azure portal. The 'Basics' step is active. The workspace name is 'azure-synapse-workspace-demo', located in 'UK South' from a subscription account named 'azuredataproxygen2'. A note at the bottom states: 'We will automatically grant the workspace identity data access to the specified Data Lake Storage Gen2 account, using the Storage Blob Data Contributor role. To manage access to the storage account after you create your workspace, perform these tasks: Action other users to the Contributor role on workspace.'

Account:

sqladminuser

abcd@1234

## Dashboard of Azure Synapse

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. At the top, there's a navigation bar with 'Microsoft Azure' and 'Synapse Analytics' followed by the workspace name 'azure-synapse-workspace-demo'. A cookie consent message is present. On the left, a sidebar has icons for Home, Data, Workspaces, and More. The main area is titled 'Synapse Analytics workspace' and 'azure-synapse-workspace-demo'. It features three main buttons: 'Ingest' (blue), 'Explore and analyze' (light blue), and 'Visualize' (yellow). Below these are sections for 'Discover more' (Knowledge center, Browse partners) and 'Recent resources'. A large, stylized 3D bar chart visualization is prominently displayed.

## 5.2 Create View from data of ADL Gen 2(Layer 2)

### Dashboard of Azure Synapse

Dedicated SQL pool: Create a real database in Azure Synapse with distributed tables and fixed resource requirements.

Serverless SQL pool: Does not create a real database but only queries data from external sources (like Azure Data Lake Gen 2), does not require fixed resources.

The screenshot shows the 'Create SQL database' dialog box. It includes fields for 'Select SQL pool type' (radio buttons for 'Serverless' and 'Dedicated', with 'Serverless' selected), and a 'Database' input field containing 'layer-2\_db'. The background shows the Azure Synapse Analytics workspace interface with a sidebar and a central workspace area.

We can query at all table in ADL Gen 2 when linked

The screenshot shows the Azure Synapse Analytics workspace interface with a 'Data' view. The sidebar shows 'Azure Data Lake Storage Gen2' and 'layer-2'. The main area displays a list of tables and folders under 'layer-2'. A context menu is open over the 'Address' table, showing options like 'New SQL script', 'New data flow', 'New integration dataset', 'Copy link', 'Manage access...', 'Rename...', 'Download', 'Delete', and 'Properties...'. The table list includes 'Address', 'Customer', 'CustomerAddress', 'Product', 'ProductCategory', 'ProductDescription', 'ProductModel', 'ProductModelProductDescription', 'SalesOrderDetail', and 'SalesOrderHeader'. Each item has a 'Last Modified' timestamp and a 'Content Type' column.

## Select TOP 100 rows

 Address

### Source folder format settings

Specify the format and layout of your data.

Folder path

<https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/>

File type

Delta format



## Result of query

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays 'Data' resources, including 'Workspace' and 'Linked' resources like 'Azure Data Lake Storage Gen2' and 'azure-synapse-workspace-demo'. The main area shows a query editor titled 'layer-2' with the following SQL script:

```
-- This is auto-generated code
2 SELECT
3   TOP 100 *
4 FROM
5   OPENROWSET(
6     BULK 'https://azuredatalfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/',
7     FORMAT = 'DELTA'
8   ) AS [result]
9
```

The results pane shows a table with columns: AddressID, AddressLine1, AddressLine2, City, StateProvince, CountryRegion, PostalCode, rowguid, and ModifiedDate. The data includes several rows of address information. The status bar at the bottom indicates '00:00:22 Query executed successfully.'

## Create a view at temp Database in Synapse Analytics layer-2\_db

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays 'Data' resources, including 'Workspace' and 'Linked' resources like 'Azure Data Lake Storage Gen2' and 'azure-synapse-workspace-demo'. The main area shows a query editor titled 'layer-2' with the following SQL script:

```
-- This is auto-generated code
1 CREATE VIEW address
2 AS
3 SELECT *
4 FROM
5   OPENROWSET(
6     BULK 'https://azuredatalfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/',
7     FORMAT = 'DELTA'
8   ) AS [result]
9
```

The results pane shows a message indicating the start of the query execution: '10:28:49AM Started executing query at Line 1 (0 record affected)'. The status bar at the bottom indicates 'Total execution time: 00:00:08.406'.

Microsoft Azure | Synapse Analytics | azure-synapse-workspace-demo

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

Data Workspace Linked

SQL database layer-2-db (SQL)

Views dbo.address

Columns AddressID (int, null), AddressLine1 (varchar), AddressLine2 (varchar), City (varchar(8000, null)), StateProvince (varchar), CountryRegion (varchar), PostalCode (varchar), rowguid (varchar(8000, null)), ModifiedDate (varchar)

System views

Schemas

Security

layer-2 layer-1 SQL script 1

Run Undo Publish Query plan Connect to Built-in Use database layer-2\_db

1 -- This is auto-generated code  
2 CREATE VIEW address  
3 AS  
4 SELECT \*  
5 FROM OPENROWSET(  
6 Bulk "https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/",  
7 FORMAT = 'DELTA'  
8 ) AS [result]

Properties General Related (0)  
Name \* SQL script 1  
Description

Type .sql script  
Size 200 bytes  
Results settings per query  
 First 5000 rows (default)  
 All rows

Results Messages

10:28:49 AM Started executing query at Line 1  
(0 record affected)

Total execution time: 00:00:08.406

Select top 100 at view

Microsoft Azure | Synapse Analytics | azure-synapse-workspace-demo

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

Data Workspace Linked

SQL database layer-2-db (SQL)

Views dbo.address

Columns AddressID, AddressLine1, AddressLine2, City, StateProvince, CountryRegion, PostalCode, rowguid, ModifiedDate

System views

Schemas

Security

layer-2 layer-1 SQL script 1 SQL script 2

Run Undo Publish Query plan Connect to Built-in Use database layer-2\_db

1 SELECT TOP (100) [AddressID]  
2 ,[AddressLine1]  
3 ,[AddressLine2]  
4 ,[City]  
5 ,[StateProvince]  
6 ,[CountryRegion]  
7 ,[PostalCode]  
8 ,[rowguid]  
9 ,[ModifiedDate]  
10 FROM [dbo].[address]

Properties General Related (0)  
Name \* SQL script 2  
Description

Type .sql script  
Size 165 bytes  
Results settings per query  
 First 5000 rows (default)  
 All rows

Results Messages

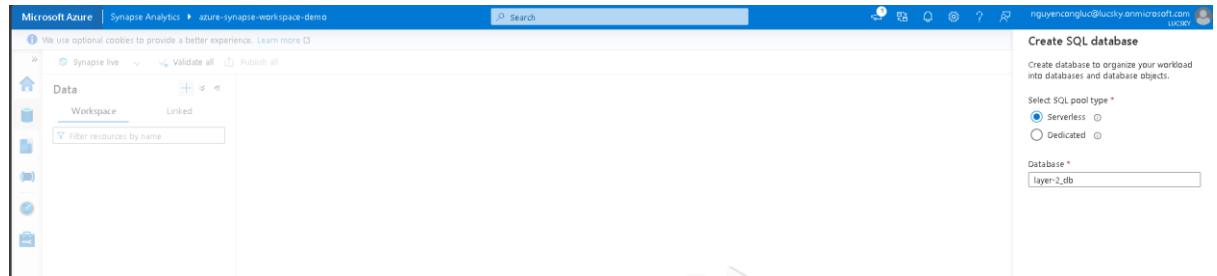
View Table Chart Export results ▾

00:00:07 Query executed successfully.

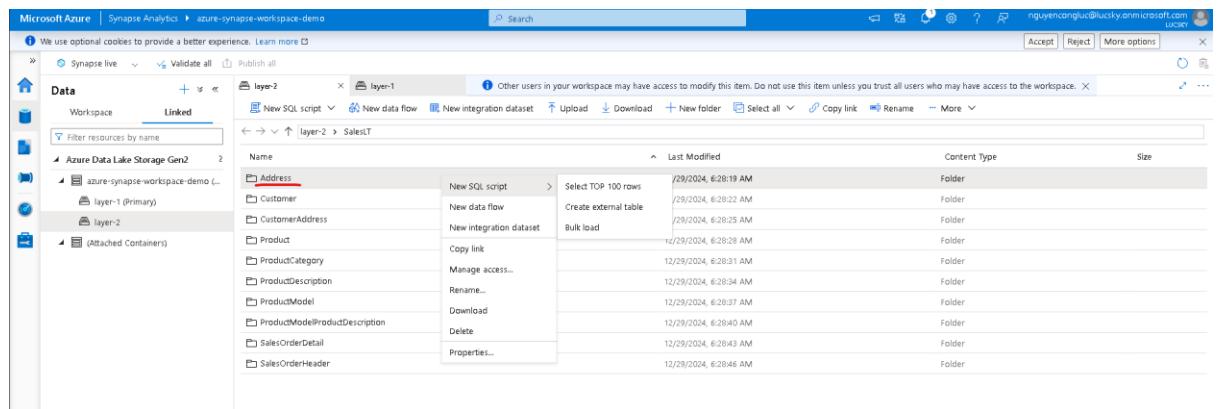
AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
9	8713 Yosemite ... (NULL)		Bothell	Washington	United States	98011	268af621-7ed7... (NULL)	2002-07-01
11	1310 Lasalle Str... (NULL)		Bothell	Washington	United States	98011	981b3303-ac2... (NULL)	2003-04-01
25	9179 Jumping St. (NULL)		Dallas	Texas	United States	75201	c9df3bb9-4ff0... (NULL)	2002-09-01
28	9228 Via Del Sol (NULL)		Phoenix	Arizona	United States	85004	12ae5e1-fc2e... (NULL)	2001-09-01
32	26910 Indela R... (NULL)		Montreal	Quebec	Canada	H1Y 2H5	84a95f62-3ae9... (NULL)	2002-08-01

Dedicated SQL pool: Create a real database in Azure Synapse with distributed tables and fixed resource requirements.

Serverless SQL pool: Does not create a real database but only queries data from external sources (like Azure Data Lake Gen 2), does not require fixed resources.



We can query at all table in ADL Gen 2 when linked



## Select TOP 100 rows

Address

### Source folder format settings

Specify the format and layout of your data.

Folder path

<https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/>

File type

Delta format

## Result of query

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays a file structure under 'Data' with 'Linked' selected, showing 'Azure Data Lake Storage Gen2' and 'azure-synapse-workspace-demo' (Primary, layer-1, layer-2). The main area contains a SQL script editor with the following code:

```

1 -- This is auto-generated code
2 SELECT *
3 TOP (100)
4 FROM
5 OPENROWSET(
6      BULK 'https://azuredatalfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/',
7      FORMAT = 'DELTA'
8 ) AS [result]
9

```

The 'Properties' pane on the right shows the script is named 'SQL script 1'. The results pane shows a table with 10 rows of address data, and a message at the bottom states '0:00:00 Query executed successfully.'

## Create a view at temp Database in Synapse Analytics layer-2\_db

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays a file structure under 'Data' with 'Linked' selected, showing 'Azure Data Lake Storage Gen2' and 'azure-synapse-workspace-demo' (Primary, layer-1, layer-2). The main area contains a SQL script editor with the following code:

```

1 -- This is auto-generated code
2 CREATE VIEW address
3 AS
4 SELECT *
5 FROM
6 OPENROWSET(
7      BULK 'https://azuredatalfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/',
8      FORMAT = 'DELTA'
9 ) AS [result]
10

```

The 'Properties' pane on the right shows the script is named 'SQL script 1'. The results pane shows a message indicating the start of execution and the total execution time of 0:00:00.046.

## Views created

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays a file structure under 'Data' with 'Workspace' selected, showing 'SQL database' (layer-2\_db (SQL)) and 'Views' (dbo.address). The main area contains a SQL script editor with the same code as the previous screenshot:

```

1 -- This is auto-generated code
2 CREATE VIEW address
3 AS
4 SELECT *
5 FROM
6 OPENROWSET(
7      BULK 'https://azuredatalfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/',
8      FORMAT = 'DELTA'
9 ) AS [result]
10

```

The 'Properties' pane on the right shows the script is named 'SQL script 1'. The results pane shows a message indicating the start of execution and the total execution time of 0:00:00.046. The 'Views' section in the left sidebar shows the newly created 'dbo.address' view, which has a single column 'AddressID' of type int.

## Select top 100 at view to test

Microsoft Azure | Synapse Analytics | izuru-synapse-workspace-demo

We use optional cookies to provide a better experience. Learn more [Accept](#) [Reject](#) [More options](#)

Data [+ <](#) [layer-2](#) [layer-1](#) [SQL script 1](#) [SQL script 2](#) [... 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.](#)

Workspace [Linked](#)

Filter resources by name

SQL database [layer-2\\_db \(SQL\)](#)

External tables

External resources

Views [dbo.address](#)

Columns

System views

Schemas

security

Run [Undo](#) [Publish](#) [Query plan](#) [Connect to](#) [Built-in](#) Use database [layer-2\\_db](#)

1 `SELECT TOP (100) [AddressID]`  
2 , [AddressLine1]  
3 , [AddressLine2]  
4 , [City]  
5 , [StateProvince]  
6 , [CountryRegion]  
7 , [PostalCode]  
8 , [rowguid]  
9 , [ModifiedDate]  
10 `FROM [dbo].[address]`

Properties

General Related (0)

Name

Description

Type  SQL script

Size 165 bytes

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

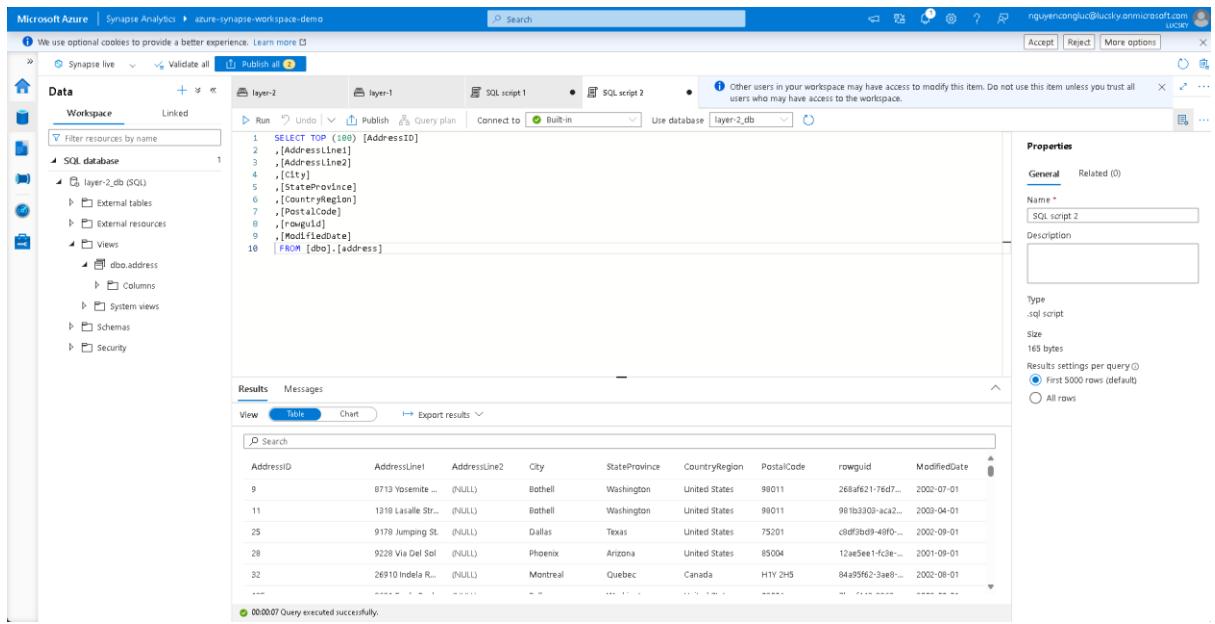
Results Message

View [Table](#) [Chart](#) [Export results](#)

Search

AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
9	8713 Yosemite ...	(NULL)	Bethell	Washington	United States	98011	268af621-7ed7...	2002-07-01
11	1318 Lasalle Str...	(NULL)	Bethell	Washington	United States	98011	98103300-ac2a...	2003-04-01
25	9178 Jumping St.	(NULL)	Dallas	Texas	United States	75201	c0df2bd9-49f0...	2002-09-01
28	9228 Via Del Sol	(NULL)	Phoenix	Arizona	United States	85004	12aeSee1-fc3e...	2001-09-01
32	26910 Indels R...	(NULL)	Montreal	Quebec	Canada	H1Y 2H5	84af9f62-3ae8...	2002-08-01
...	...	...	...	...	...	...	...	...

00:00:07 Query executed successfully.



Continue create other views from all tables

```
CREATE VIEW Address
AS
SELECT *
FROM
    OPENROWSET(
        BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-
2/SalesLT/Address/',
        FORMAT = 'DELTA'
    ) AS [result];
GO
CREATE VIEW Product
AS
SELECT *
FROM
    OPENROWSET(
        BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-
2/SalesLT/Product/',
        FORMAT = 'DELTA'
    ) AS [result];
GO
CREATE VIEW ProductCategory
AS
SELECT *
FROM
    OPENROWSET(
        BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-
2/SalesLT/ProductCategory/',
        FORMAT = 'DELTA'
    ) AS [result];
GO
```

-- Continue with other views...

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays the 'Data' blade, which includes sections for Workspace, Data, External tables, External resources, Views, and System views. Under 'Views', there are entries for 'dbo.Address', 'dbo.Customer', 'dbo.CustomerAddress', 'dbo.Product', 'dbo.ProductCategory', 'dbo.ProductDescription', 'dbo.ProductModel', 'dbo.ProductModelProduct...', 'dbo.SaleOrderDetail', 'dbo.SaleOrderHeader', and 'System views'. The main area shows a query editor window with the following SQL script:

```
1 CREATE VIEW Address
2 AS
3 SELECT *
4 FROM
5     OPENROWSET(
6         BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Address/',
7         FORMAT = 'DELTA'
8     ) AS [result];
9 GO
10
11 CREATE VIEW Customer
12 AS
13 SELECT *
14 FROM
15     OPENROWSET(
16         BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Customer/',
17         FORMAT = 'DELTA'
18     ) AS [result];
19 GO
20
21 CREATE VIEW CustomerAddress
22 AS
23 SELECT *
24 FROM
25     OPENROWSET(
26         BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/CustomerAddress/',
27         FORMAT = 'DELTA'
28     ) AS [result];
29 GO
30
31 CREATE VIEW Product
32 AS
33 SELECT *
34 FROM
35     OPENROWSET(
36         BULK 'https://azuredatfactorygen2.dfs.core.windows.net/layer-2/SalesLT/Product/',
37         FORMAT = 'DELTA'
38     ) AS [result];
39 GO
```

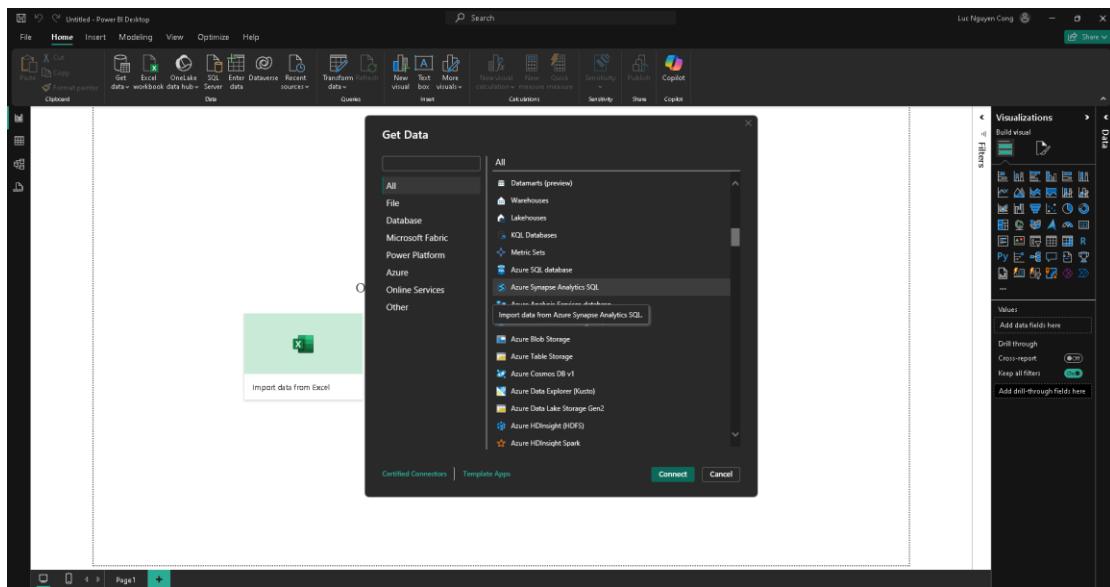
The right side of the screen shows the 'Properties' panel, which is currently set to the 'General' tab. It displays the following information:

- Name: SQL script 1
- Description: (empty)
- Type: sql script
- Size: 208 bytes
- Results setting per query:
  - First 3000 rows (default)
  - All rows

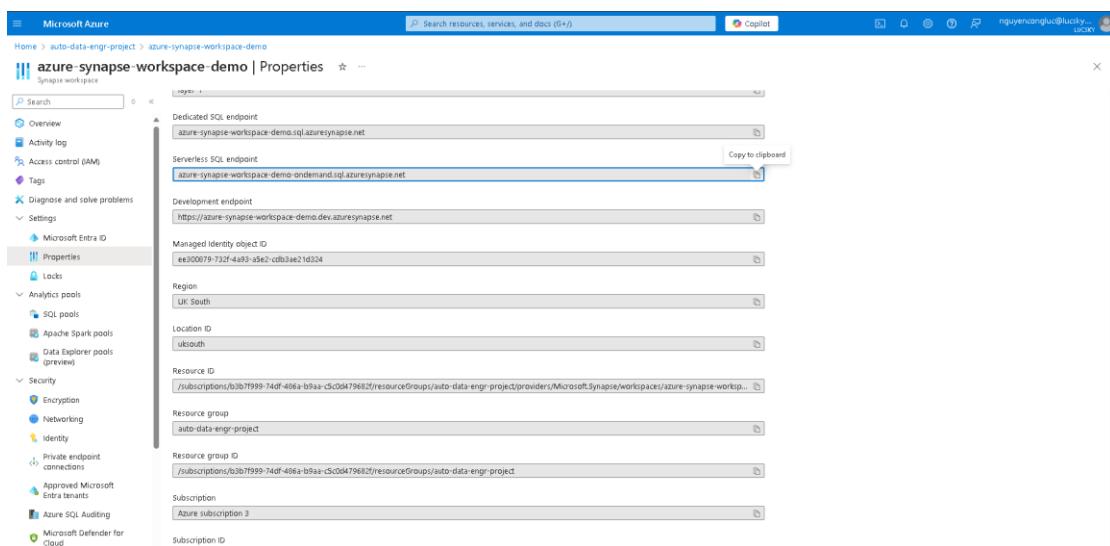
# CHAPTER 6. Data Reporting

## 6.1 Connect data from Azure synapse analyst

Select resource

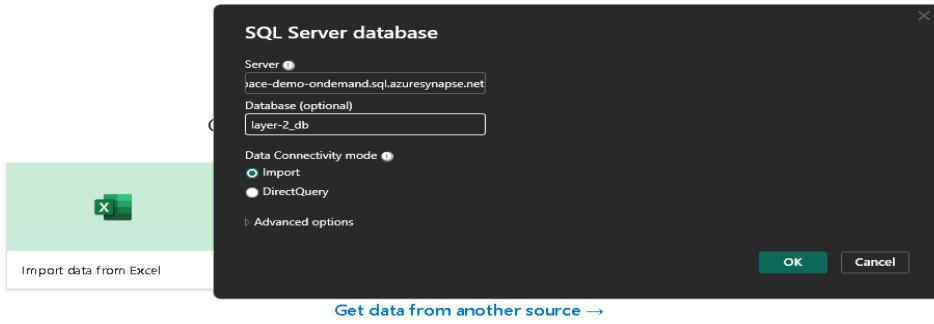


Get server name from Azure synapse analyst

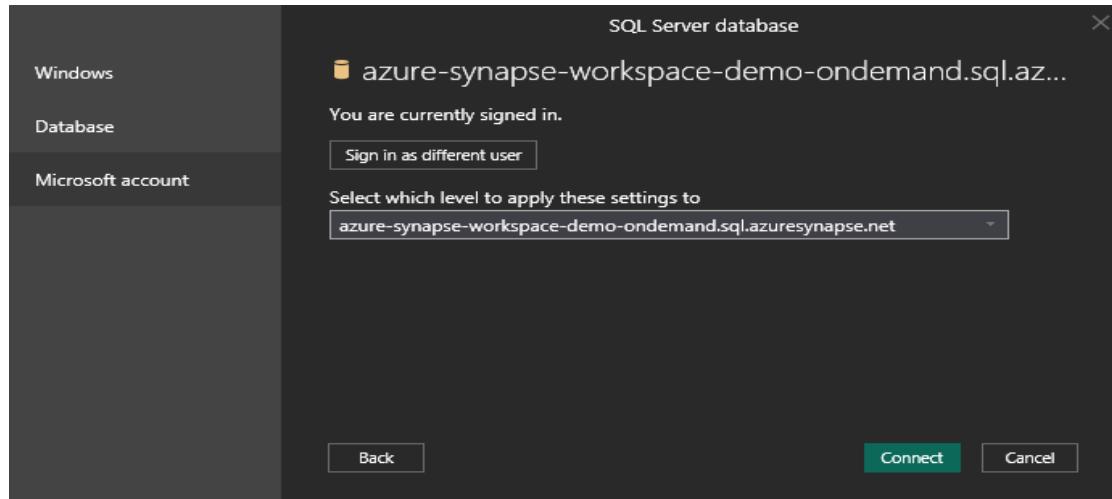


The screenshot shows the Azure portal's Properties blade for the 'azure-synapse-workspace-demo' workspace. The 'Serverless SQL endpoint' field contains the value 'azure-synapse-workspace-demo-on-demand.sql.azuresynapse.net'. A 'Copy to clipboard' button is visible next to the input field. The left sidebar shows the workspace's settings and other configurations.

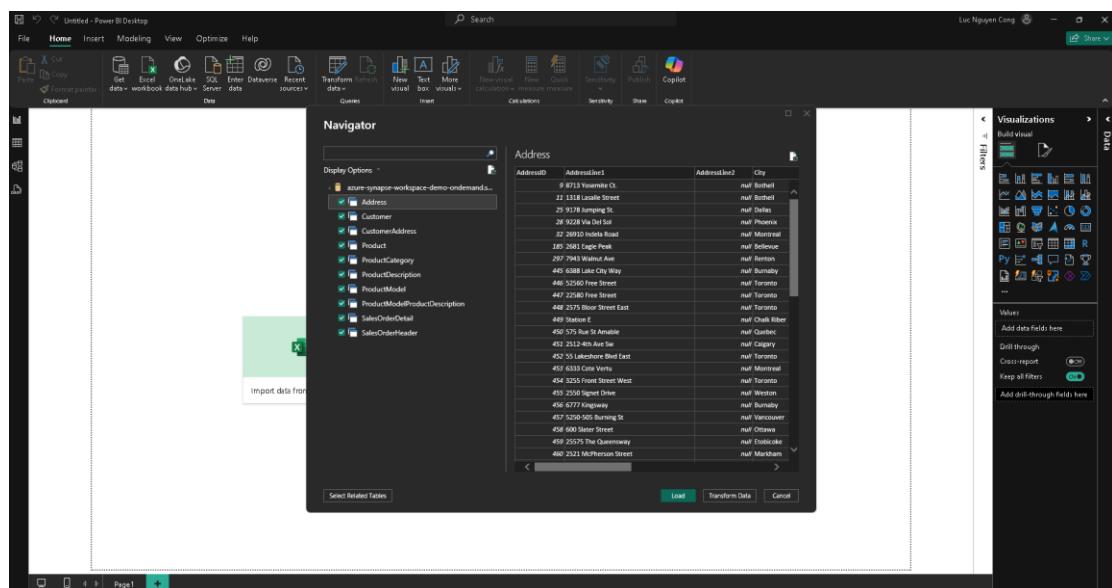
## Setup connection



## Action



## Load data to Power BI successfully



## 6.2 Design dashboard

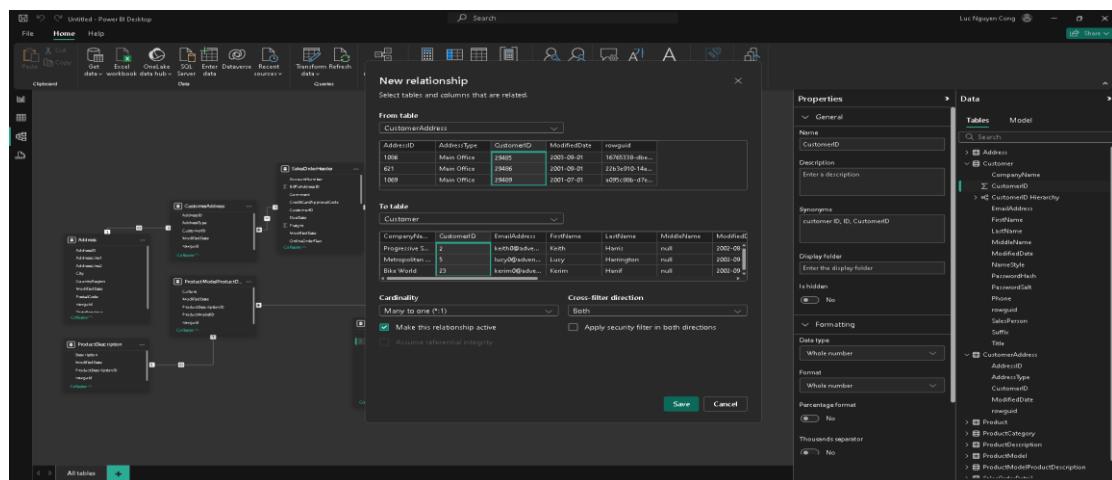
Report View: Used for designing and visualizing reports with charts, tables, and images.

Data View: Displays raw data from tables, allowing users to inspect and work with the data.

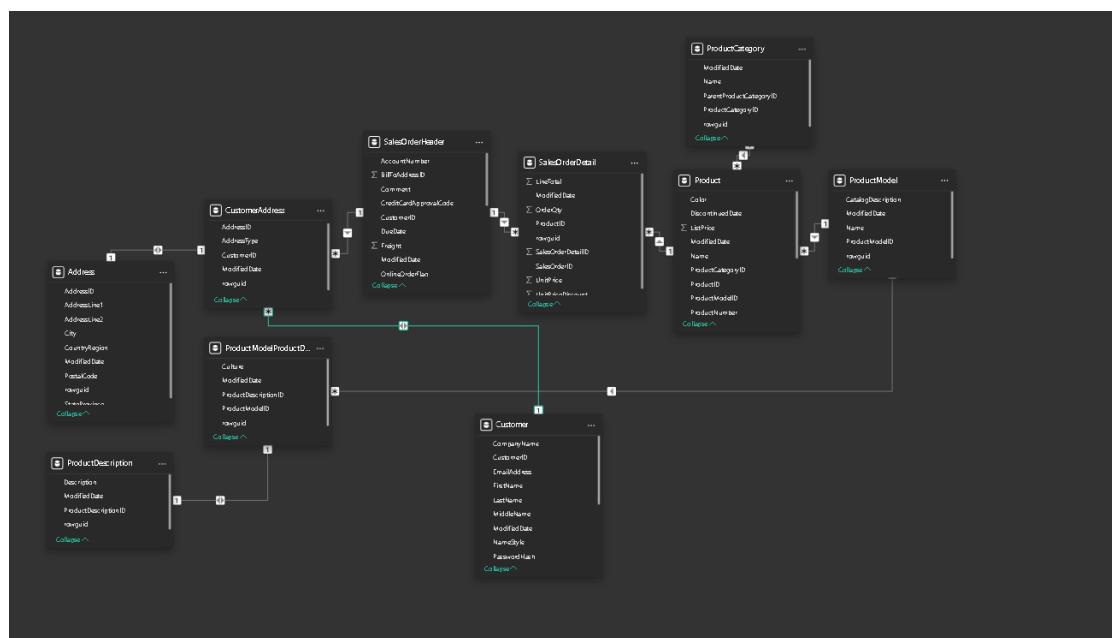
Model View: Manages relationships between tables in the data model.

DAX Query View: Analyzes and optimizes DAX queries, helping to understand and improve report performance.

Create relationships for a table CustomerAddress and Customer

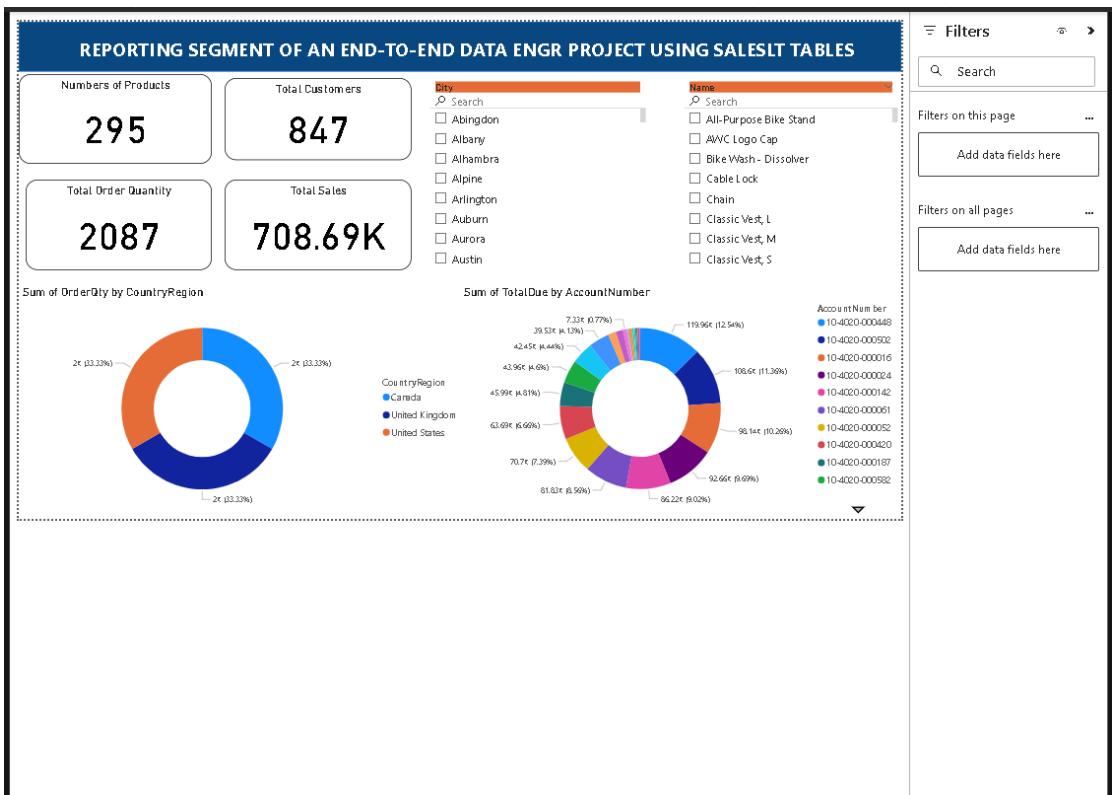


Model View



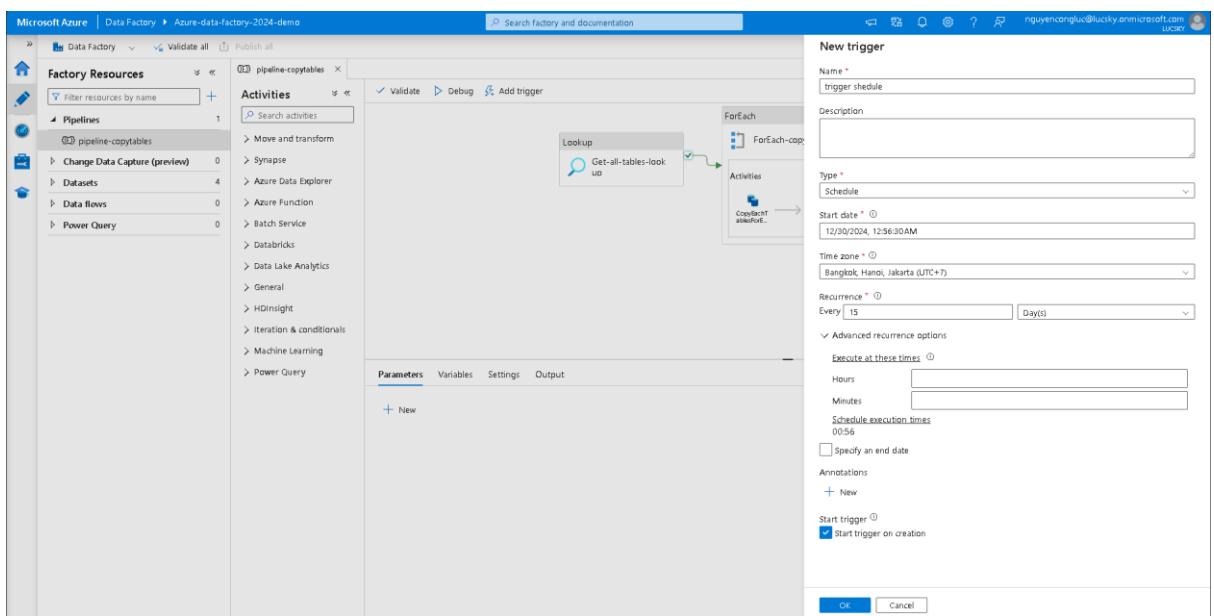
## Design with Card and measure

## Result



# CHAPTER 7. Pipelines Testing/Automation

Create trigger to automation update data. Any changes in data (sql server) are also reflected on Power BI



# CHAPTER 8. SUMMARY

In this end-to-end Data Engineering project, I successfully built and tested the entire pipeline, starting from collecting source data, performing ETL (Extract, Transform, Load), and finally Report data via Power BI. The process includes setting up and activating Azure Data Factory, using Azure Data Bricks to transform data, and checking integrity through automatic updates.

The final result demonstrated that the pipeline worked correctly, meeting the requirements for real-time data processing and updating. This is a solid foundation for more complex data engineering projects in the future.

This project not only provides a comprehensive view of the data processing process but also illustrates how to apply modern tools to achieve the highest efficiency.

