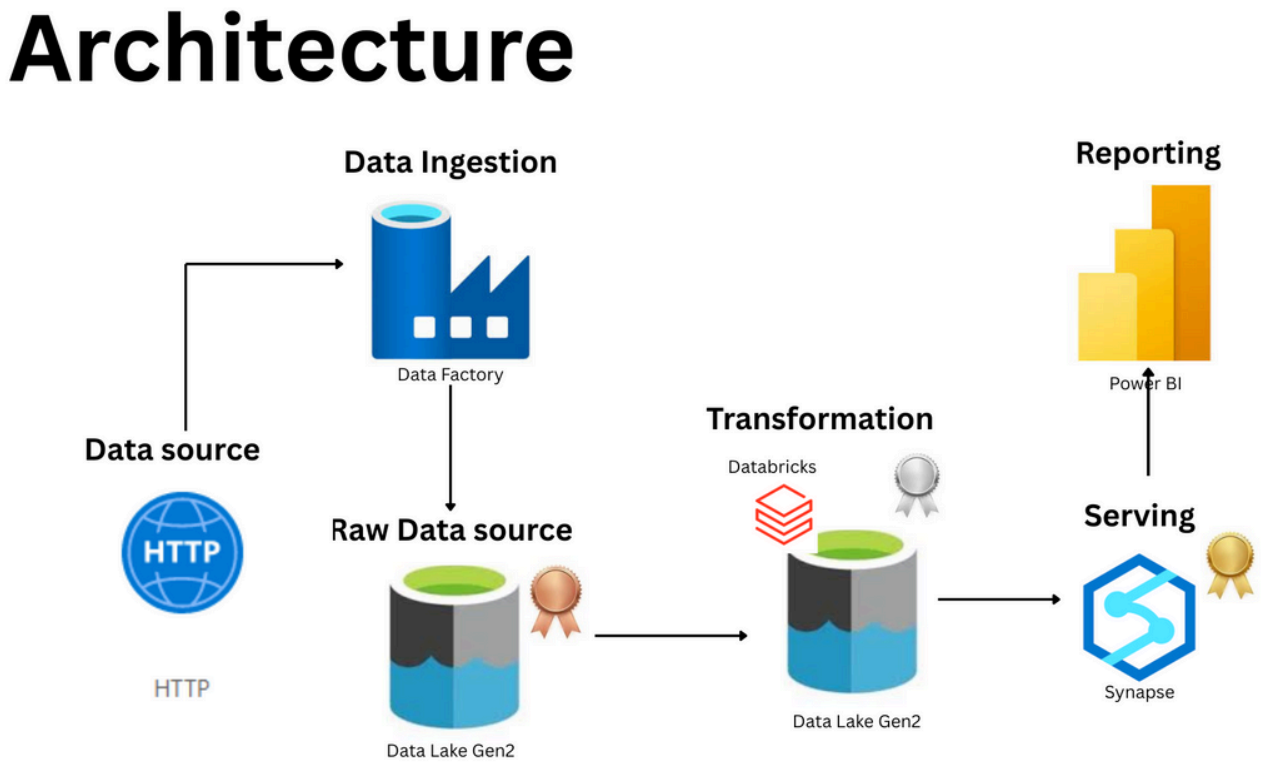
End To End Data Pipeline on Microsoft Azure Cloud



workflow of medallion architecture...

Data Source

Represents raw data (e.g., HTTP endpoint or APIs).

Data Ingestion (Azure Data Factory)

Azure Data Factory (ADF) is used to **extract and load the data** from the source into the data lake.

🥉Raw Data Store (Bronze Layer - Data Lake Gen2)

Stores **raw, unprocessed data** just as ingested.

🥈 Transformation (Silver Layer - Databricks + Data Lake Gen2)

Databricks performs data cleansing, parsing, joining, and validation.

🥇 Serving (Gold Layer - Synapse)

Final transformed and business-level aggregated data is stored in **Gold Layer**.

Reporting (Power BI)

- Connects to the **Gold layer (Synapse)** for visual analytics.
- Provides insights and dashboards to end users and stakeholders.

SERVICES AND TOOLS USED

→ **Azure Data Factory**

→ **Data Lake Gen2**

→ **Databricks**

→ **Synapse**

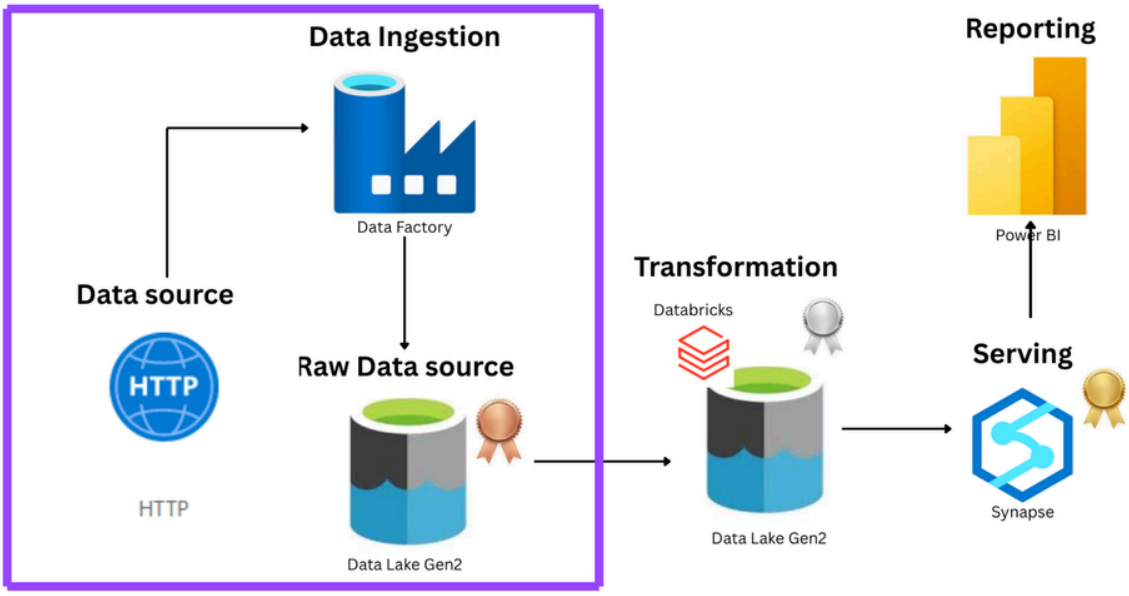
→ **Power BI**

→ **Visual studio**

→ **Github**

Raw / Bronze Layer

Architecture



ABOUT Dataset

The **Adventure Works** dataset contains multiple CSV files like Customers, Products, Sales (from 2015–2017), Returns, Territories, and more. I used this dataset because it provides a great opportunity to work with real-time business data. It includes multiple related tables, helping to understand how data is managed and analyzed in real-world scenarios.

Original dataset link → Adventure Works

Github dataset link →
Adventure-Works-Data-Engineering-Project/Data at main · anshlambaoldgit/Adventure-...

upload dataset to your github as well..

STEP 1 : go to AZURE PORTAL

search portal.azure.com

sign up using **gmail** and **VISA/MASTER** card

STEP 2 : Create Resource Group

Search Resource Group → Click **Create**

Basics:

→ Provide **resource name** and **region**.

Tags:

→ **name, value**

Step 3 : Inside Resource Group Create Resource

STORAGE ACCOUNT

Click Create →

Basics: fill

- **Storage account name**
- **Primary service** – Choose Azure Blob Storage or Azure Data Lake Storage Gen2.
- **Performance** – Standard
- **Redundancy** – LRS

Advanced:

- By default, it creates blob storage account to create data lake.
  → Enable **Hierarchical Namespace**
- **Access tier** – hot (If we frequently access data)

Networking:

- Network access – enable public access

reference...

View automation template

Basics

Subscription	Azure subscription 1
Resource group	DATAENGINEERPROJECT1
Location	East US
Storage account name	dpstoragedatalake
Primary service	Azure Blob Storage or Azure Data Lake Storage Gen 2
Performance	Standard
Replication	Locally-redundant storage (LRS)

Advanced

Enable hierarchical namespace	Enabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

Networking

Network connectivity	Public endpoint (all networks)
Default routing tier	Microsoft network routing

Data protection

Point-in-time restore	Disabled
Blob soft delete	Enabled
Blob retainment period in days	7
Container soft delete	Enabled
Container retainment period in days	7
File share soft delete	Enabled
File share retainment period in days	7
Versioning	Disabled
Blob change feed	Disabled
Version-level immutability support	Disabled

Encryption

Encryption type	Microsoft-managed keys (MMK)
Enable support for customer-managed keys	Blobs and files only

---

create Containers → For all three layers: **bronze**, **silver**, and **gold** separately. 3️⃣ 2️⃣ 1️⃣

Go to Resource Group → Click Data Lake that created → Data Storage account-> options available

🔹 **Containers**

- **File Shares**
- **Queues**
- **Tables**

DATA FACTORIES

Click Create → Choose **resource group** and provide **unique name**

Go to Resource Group → Click Data Factory that created →options available

- **Home**
- **Author**
- **Monitor**
- **Manage**
- **Learning Center**

Step 3: Create Pipeline

To create a pipeline we need **source** and **destination**.

→ Go to **ADF in DATA FACTORY** → Click **Pipeline** → 3 dots → Name the pipeline in **Properties**

In Activity:

→ Choose **Move and Transform** → Drag and drop **Copy Data(**Copy Data is a Copy Activity**)** on canvas

||---------- STATIC PIPELINE ----------||

======> **GITHUB data** -
🌐 Adventure-Works-Data-Engineering-Project/Data at main · anshlambaoldgit/Adventure-...

→ Choose any file → Click **Raw** → Copy **BASE URL**

Example:
https://raw.githubusercontent.com/anshlambaoldgit/Adventure-Works-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Products.csv
Base URL – https://raw.githubusercontent.com

Relative URL – anshlambaoldgit/Adventure-Works-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Products.csv

→ **Copy base URL alone**

NOTE: ALWAYS CREATE LINKED SERVICE FIRST

======> **Create Two Linked Services (**one for GitHub and one for Storage**)**

→ Go to **Manage** in ADF → **Linked Services**

- **Link Service 1 (GitHub)** → Click **New** → HTTP → Name the linked service → Paste base URL → Authentication type: **Anonymous** → Click **Test Connection** → Create
- **Link Service 2 (Storage)** → Click **New** → Azure Data Lake Storage Gen2 → Name it → Choose storage account created in step 2
  → Click **Test Connection** → Create

======> **Create Dataset**

→ Go to **Author** in ADF → Click **Copy Data** on canvas

- To create **dataset for linked service 1** → Choose **source** → New → HTTP → Select format (CSV)
  → Name the dataset → Choose created linked service 1 → Copy and paste **relative URL** → Click OK → **Preview Data** to see the data
- To create **dataset for linked service 2** → Choose **sink** → Click New → Azure Data Lake Storage Gen2
  → Select format (CSV) → Name the dataset → Choose created linked service 2
  → Choose file path by clicking **Browse** and **bronze** → Directory name as **products** → File name as **products.csv**
  → Import schema: **None** → Click OK

→ Right now, we don't have any file on storage data lake so we don't have import schema.

======> **Run Pipeline**

→ Click **Debug** on author ADF canvas → Once succeed.

======> **view the data**

→ Go to **Resource Group** → **Data Lake** → **Container** → Inside **bronze** view data → Edit → Preview.

======> **save Pipeline**

→ Publish All

NOTE : To build **STATIC PIPELINE** create copy activity again and again for every file → which is not recomended way.

||---------- DYNAMIC PIPELINE ----------||

In our scenario, three elements are changing in the **Copy Activity**:

- **Relative URL**
- **Folder**
- **File**

We will create **three parameters** and dynamically change these values using the **ForEach Activity**.

======> **Create a New Pipeline**

→ Go to **Author** in ADF.

→ Click on the **three dots** next to Pipelines → Select **New pipeline**.

→ Create **one dataset** for all the data:

- In the **Source** → Click **New** → Choose **HTTP** → Select **CSV**.
- Name it.
- Select **Linked Service 1** (already created).
- In the **Relative URL**, instead of hardcoding a single URL, **use a parameter**.

======> **Create Parameters for Source and Destination**

→ Open the dataset → Go to **Advanced** → Click on **Parameters** → Click **+** to add a new parameter.

→ Name the parameter (e.g., p_rel_url) → Click **OK**.

→ In **Relative URL**, reference the parameter using dynamic content.

→ Go to **Copy Activity → Source** → Provide the value for the dataset parameter.

→ Repeat the same steps for the **Sink**:

- Create a dataset → Choose **Data Lake** → Select **CSV** → Name it.
- Choose **Linked Service 2** (already created).
- Select the **File System** → Define the **Folder** using dynamic content.

→ Create parameters for:

- **Folder**
- **File name**

You now have **three parameters**:

- One for the **source**
- Two for the **sink** (folder and file name)

======> Create ForEach Activity

→ Go to the **dynamic pipeline**.

→ Add a **ForEach** activity under **Iteration & Conditionals**.

→ In **Settings**, check the box for **Sequential** execution.

→ For the **Items** property, provide an **array** in JSON format.

Example JSON:

```
[
 {
  "p_rel_url": "anshlambaoldgit/Adventure-Works-Data-Engineering-Project/refs/heads/main/Data/AdventureWorks_Calendar.csv",
  "p_sink_folder": "AdventureWorks_Calendar",
  "p_sink_file": "AdventureWorks_Calendar.csv"
 }
]
```

======> Use Lookup Activity to Pull Data

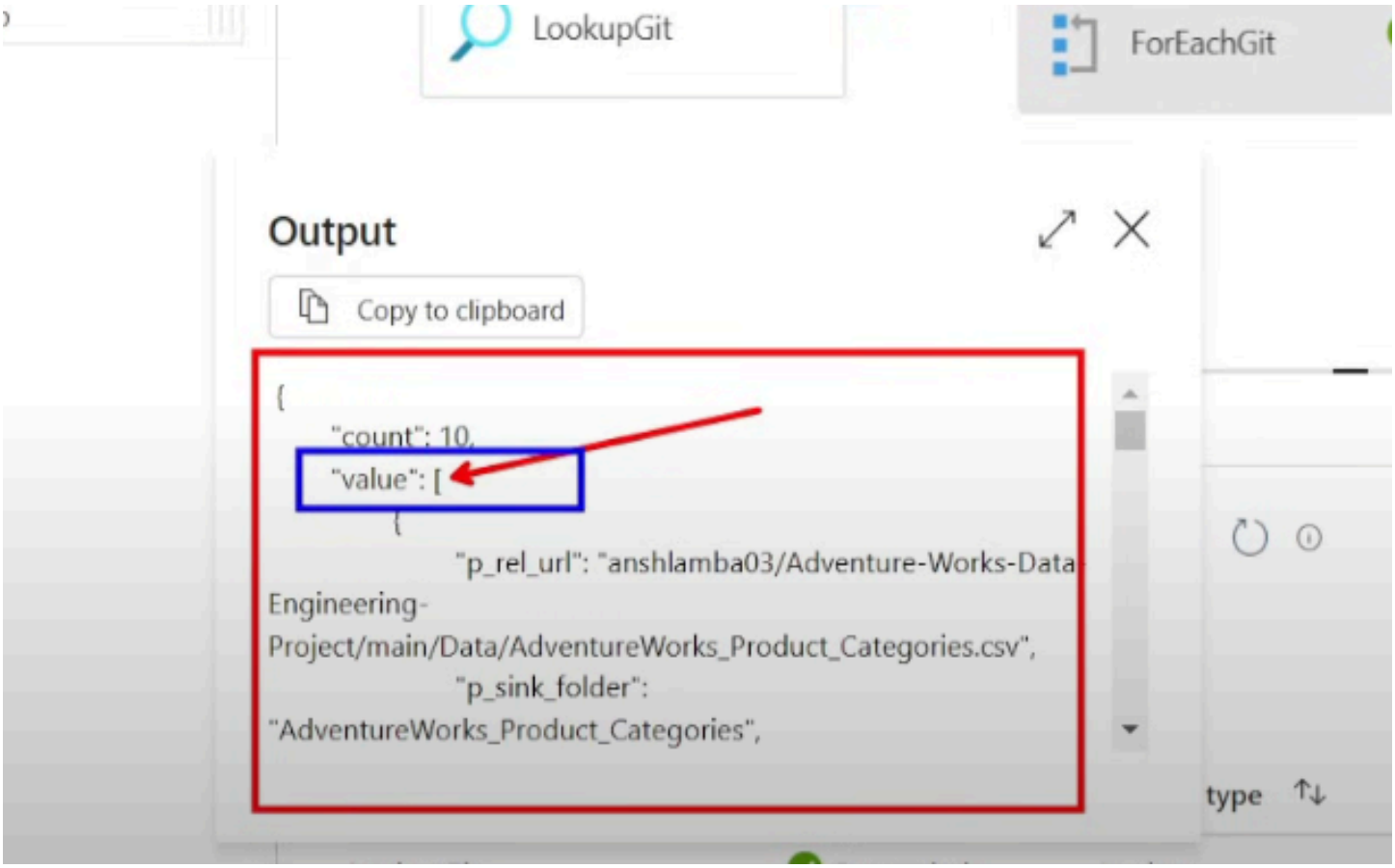
→ In the **Activities** pane, drag and drop **Lookup** under **General**.

→ Name the activity.

→ In **Settings**, create a new dataset:

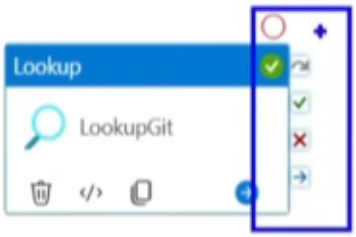
- Choose **Azure Data Lake** → Select **JSON** → Name it.
- Choose **Linked Service 2** (for storage).
- Browse to select the **parameter container** → Choose the uploaded **JSON file**.
- Uncheck **"First row only"** to iterate through all rows.

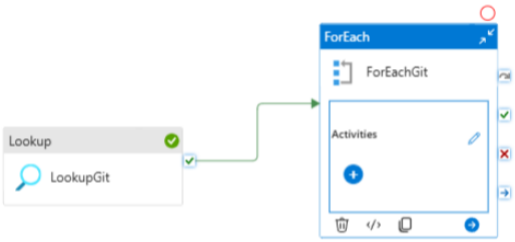
⚠️ If there is any activity already placed on the canvas, go to **General → Activity Status**, and **deactivate** it if needed.



In Azure Data Factory (ADF), when connecting one activity to another, you create **nodes (linked activities)** that define the **flow of execution** in a pipeline.

You can **visually connect** these activities using the **canvas** in ADF.





======> Configure Activities Inside ForEach

→ Go to **ForEach → Settings**.

→ In the **Items** section, **add .value** from the **Lookup** activity output.

- Example: @activity('Lookup1').output.value
- This is necessary because .value represents the **array** to iterate over.

→ Now, place your **Dynamic Copy Activity inside** the ForEach:

- Cut the dynamic copy activity from the main canvas.
- Click the **pencil icon** inside the **ForEach** activity → it opens a **nested canvas**.
- Paste the copy activity here.

→ Now, **set dynamic content** for source and sink parameters:

- For **Source** → rel_url parameter → Click on **Dynamic Content** → Choose item().p_rel_url
- For **Sink Folder** → Choose item().p_sink_folder
- For **Sink File Name** → Choose item().p_sink_file

Each parameter value is dynamically read from the current **JSON object** in the array during iteration.

======> Debug the Pipeline

- After completing the setup, click on **Debug** in the top menu to run and test your pipeline.
- This will simulate a pipeline execution and verify whether each file is processed correctly based on the parameters.

======> Validate the Output

To ensure your dynamic pipeline worked:

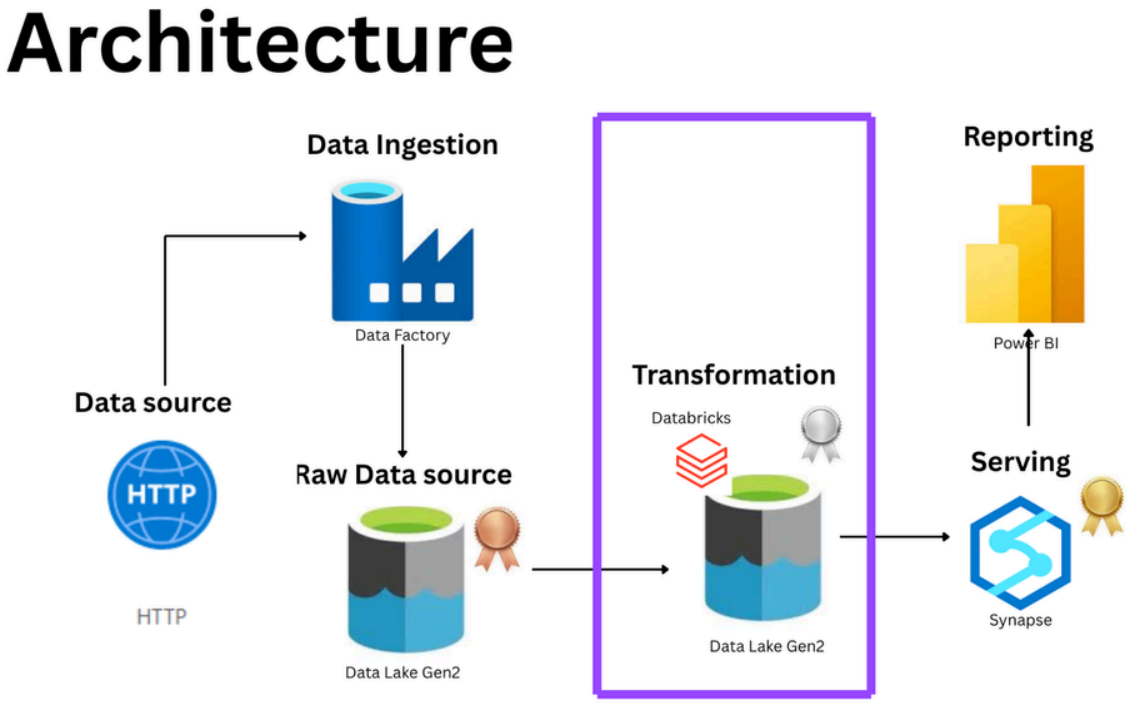
→ Go to **Resource Group** → Open your **Storage Account** (Data Lake).

→ Navigate to the **bronze** container (or the folder where output is written).

→ You should see that all the **target folders** were created and the corresponding **files were copied** as defined in your JSON.

Now the raw layer was all set .....

Transformation Layer / Cleansed Layer / Silver Layer



Step 1: Create Azure data bricks

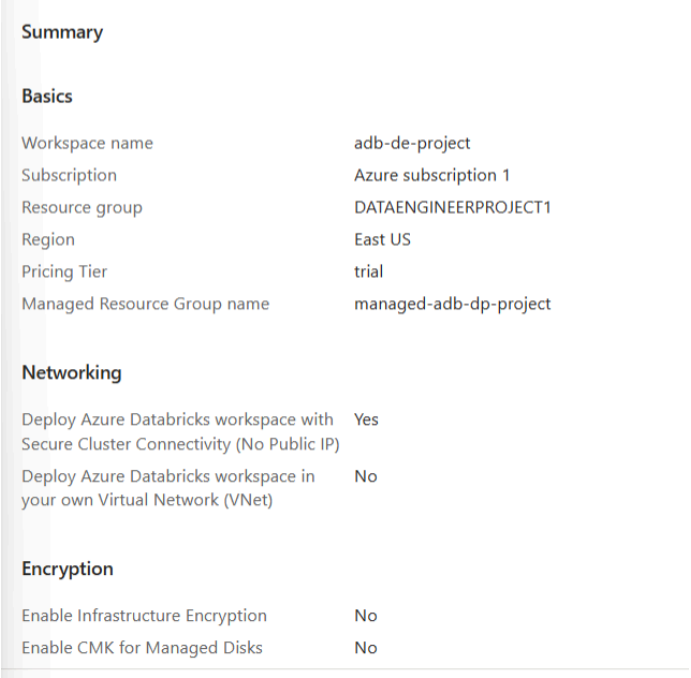
→ Navigate to Azure Portal
Go to:
Home → + Create a resource → Search Databricks → Choose Databricks from Microsoft → Create

→ Workspace Setup

- Choose **Resource Group**
- Provide **Name to the Workspace**
- Choose **Pricing Tier → Trial**

→ Continue with Setup Managed Resource Group

- **Provide Name to the Managed Resource Group**
- Click **Next → Networking**
- Click **Next → Encryption**
- Click **Next → Security & Compliance**
- Click **Next → Tags**
- Click **Review**



Summary

Basics

Workspace name	adb-de-project
Subscription	Azure subscription 1
Resource group	DATAENGINEERPROJECT1
Region	East US
Pricing Tier	trial
Managed Resource Group name	managed-adb-dp-project

Networking

Deploy Azure Databricks workspace with Secure Cluster Connectivity (No Public IP)	Yes
Deploy Azure Databricks workspace in your own Virtual Network (VNet)	No

Encryption

Enable Infrastructure Encryption	No
Enable CMK for Managed Disks	No

→ After Creating the Azure Databricks Workspace

Launch Databricks

- Click **Go to Resource** button
- Click **Launch Databricks**
- It gets to **Databricks Workspace**

Step 2: Now Let's Create Compute

- Click create
- Modify Name
- policy : Unrestricted
- Cluster Type : Single Node

→ Cluster Configuration

- **Access Mode**: No isolation shared (because single user)

- **Runtime Version**: Anything with **LTS** (Long Term Support)
- **Uncheck**: Photon Acceleration
- **Node Type**: Go with **cheapest**
  - General Purpose (Standard_DS3_v2)
  - In Azure Databricks, **node type** refers to the specific size and configuration of the virtual machines (VMs) used in a cluster.
  - It defines the **CPU, memory, and storage capacity**.
- **Terminate After**: Give smaller minutes like **20 mins**
- **Click Create**

Implement this Storage Access

Step 3 : Create App in Microsoft Entra ID

→ Go to → Home → Microsoft Entra ID → Manage → App registrations
(Because we will be registering an application)

→ Click **New Registration**

- Name it (any name)
- Keep **everything as default**
- Click **Create**

→ Save the following information

Essentials

Display name	: dpproject_app	Client credentials	: Add a certificate or secret
Application (client) ID	: cf54fb70-3e99-417d-a8c9-b41e648fd839	Redirect URIs	: Add a Redirect URI
Object ID	: e8cdcf99-a397-45d4-b91d-31da3b7621a0	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	: 85d13684-9ab2-4f72-8ab2-db356d2d4d59	Managed application in l...	: dpproject_app
Supported account types	: My organization only		

App ID: cf54fb70-3e99-417d-a8c9-b41e648fd839

Object ID: e8cdcf99-a397-45d4-b91d-31da3b7621a0

Directory ID: 85d13684-9ab2-4f72-8ab2-db356d2d4d59

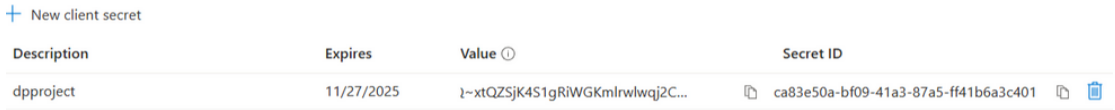
Step 4 : Create a Client Secret

→ Go to Microsoft Entra ID → Manage → Certificates & Secrets

→ Click **New Client Secret**

- Add a **description** (any name)
- Click **Add**

→ Copy the following credentials



+ New client secret

Description	Expires	Value	Secret ID
dpproject	11/27/2025	~xtQZSjK4S1gRiWGKmlrwlwqj2C...	ca83e50a-bf09-41a3-87a5-ff41b6a3c401

Value: vBU8Q~xtQZSjK4S1gRiWGKmlrwlwqj2CNbiFXaF-

Secret ID: ca83e50a-bf09-41a3-87a5-ff41b6a3c401

Step 5 : Assign Role to App (For Accessing Data Lake)

→ Go to Home → Storage Account (created for Data Lake)

→ Click Access Control (IAM)

→ Click Add → Add Role Assignment

→ Select Role: **Storage Blob Data Contributor**

- (It provides both **read and write permissions**)

→ Click Next → Select Members

- **Search** the app created earlier
- **Select** it

→ Click Review + Assign

Step 6 : Create Notebook in Azure Databricks

→ Go to → Azure Databricks → Launch Workspace

→ In **Workspace**:

- Create **Folder**
- Create **Notebook**

→ Start the Cluster

- Click the **3 dots** on top of the notebook
- Choose the **cluster created**
- It will start the cluster

→ Notebook Commands

- Shift + Enter: Run the current code cell
- Alt + Enter: Run and open a new code cell

Step 7 : Now: Pull App Credentials into Notebook

→ We have to write code that pulls the **credentials of the app** and **employs those within our code**.

→ For code, **use this documentation**:
🔗 [Azure Databricks → Connect to Azure Storage]

→ Go to:
Azure service principal in the documentation → Copy this

You can use `spark.conf.set` in notebooks, as shown in the following example:

Python Copy

```python
service_credential = dbutils.secrets.get(scope="<secret-scope>",key="<service-credential-

spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net", "OAut
spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account>.dfs.core.windows.n
spark.conf.set("fs.azure.account.oauth2.client.id.<storage-account>.dfs.core.windows.net"
spark.conf.set("fs.azure.account.oauth2.client.secret.<storage-account>.dfs.core.windows.
spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-account>.dfs.core.window
```

→ Remove service credentials from this

→ Paste your Storage Account in the code

✅ Purpose

I'm configuring Spark to authenticate with Azure Data Lake Storage Gen2 using the OAuth 2.0 Client Credentials flow. This allows my Spark jobs to read from and write to the data lake securely, without relying on shared keys or manually managed credentials.

```
08:47 PM (1s)                    6
df = spark.read.format("csv").option("header", "true").option("inferSchema",
"true").load("abfss://bronze@dpstoragedatalake.dfs.core.windows.net/
AdventureWorks_Calendar")
```

Step 8 : Reading CSV Data with Spark

spark.read.format("csv")

- This tells Spark that the data source is in **CSV format**.
- spark.read returns a **DataFrameReader object**.
- .format("csv") sets the **input format to CSV**.

Step 9 : Set Header Option

.option("header", "true")

- This option specifies that the **first row** in the CSV file contains **column headers**.
- Without this, Spark might treat the **first row as data**.

Step 10 : Enable Schema Inference

.option("inferSchema", "true")

- Spark will **automatically detect the data types** (e.g., Integer, String, Date) of each column.
- If this were **false**, all columns would be read as **strings**, which could affect downstream processing.

Step 11 : Load the Data

.load("abfss://bronze@dpstoragedatalake.dfs.core.windows.net/AdventureWorks_Calendar")

- This is where Spark actually **loads the data** from the given location.

Step 12 : Once Data is Loaded → Perform Transformations

```
from pyspark.sql.functions import *
```

```
from pyspark.sql.types import *
```

Import all the functions and types needed for data transformation

refer notebook..... for complete code

To Visualize

08:33 AM (3s)

```
df_sale.groupBy('OrderDate').agg(count('OrderNum
```

(2) Spark Jobs

Table

	OrderDate	count
1	2017-01-06	151
2	2017-01-27	142
3	2017-02-26	119
4	2017-01-24	173

- Click **+ icon** → **visualize**
- Modify as per your preferred **charts**
- Click **save**

Step 12 : LOAD transformed data to silverlayer

2 minutes ago (5s) 21

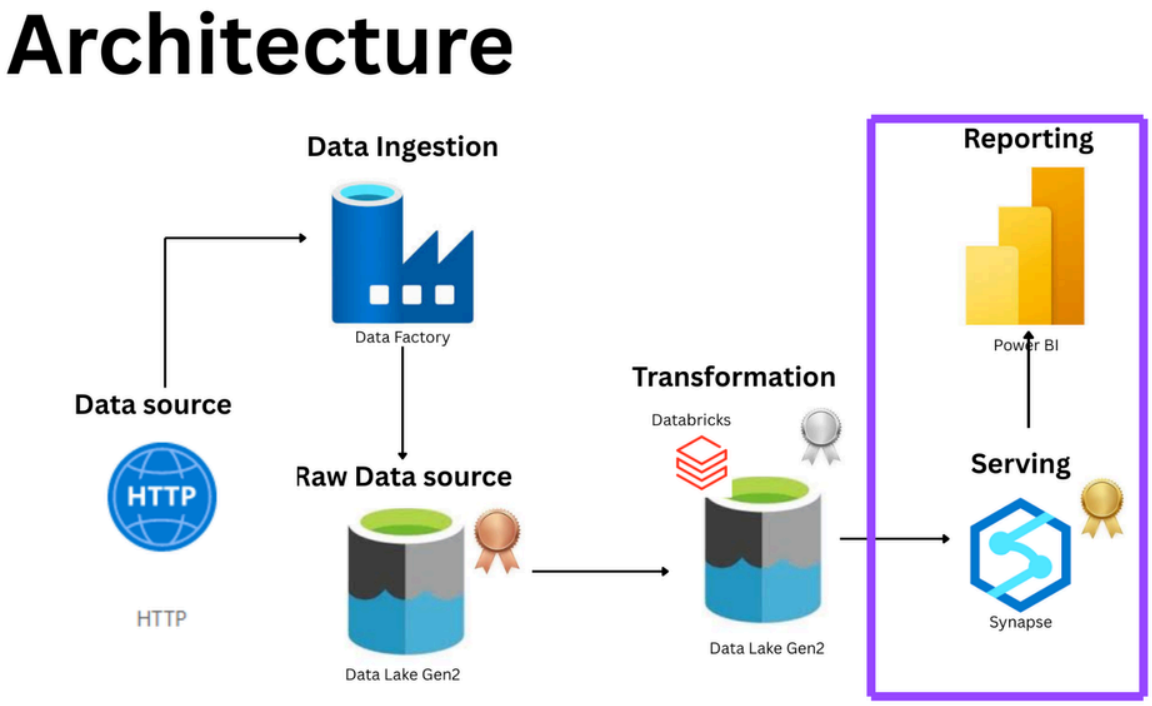
```
df_cal.write.format("parquet")\
    .mode('append')\
    .option("path","abfss://silver@dpstoragedatalake.dfs.core.windows.net/AdventureWorks_Calendar").save()
```

(1) Spark Jobs

Successfully loading the all DataFrame into the Silver layer of your **medallion architecture** in **Parquet format.**

Now we are done with transformation layer ....

Gold Layer / Serving Layer / Presentation Layer



❌ ERROR: Microsoft.Synapse Not Registered

Error Meaning:
You're trying to create an Azure Synapse resource, but the resource provider **Microsoft.Synapse** is not registered yet for your subscription.

✅ Fix It:

1. Go to **Azure Portal**
2. Search for **"Subscriptions"**
3. Click your **Free Trial subscription**
4. Click on your subscription (**Azure subscription 1**) — this will open the subscription's details page.
5. In the left-hand menu, scroll down and click on **"Resource providers"**
6. In the search bar, type:
   Microsoft.Synapse
7. If status is **"Not Registered"**, click on it → click **"Register"** button on top.

🔹 REGION SETUP

- **Region**: West US 2 (has free trial for Synapse)

Step 1 : Creating an Azure Synapse Analytics Workspace:

→ Go to **Create a resource** → search **Azure Synapse Analytics** → click **Create**.

→ Choose a **Resource Group**.

→ Name the **Managed Resource Group**.

→ Name the **Workspace**.

→ In **Data Lake Storage** configuration:

- Do **not** use an existing one.
- **Create a new Storage Account (specifically for Synapse)**.
- Create a **new File System**.

→ Click **Next: Security**.

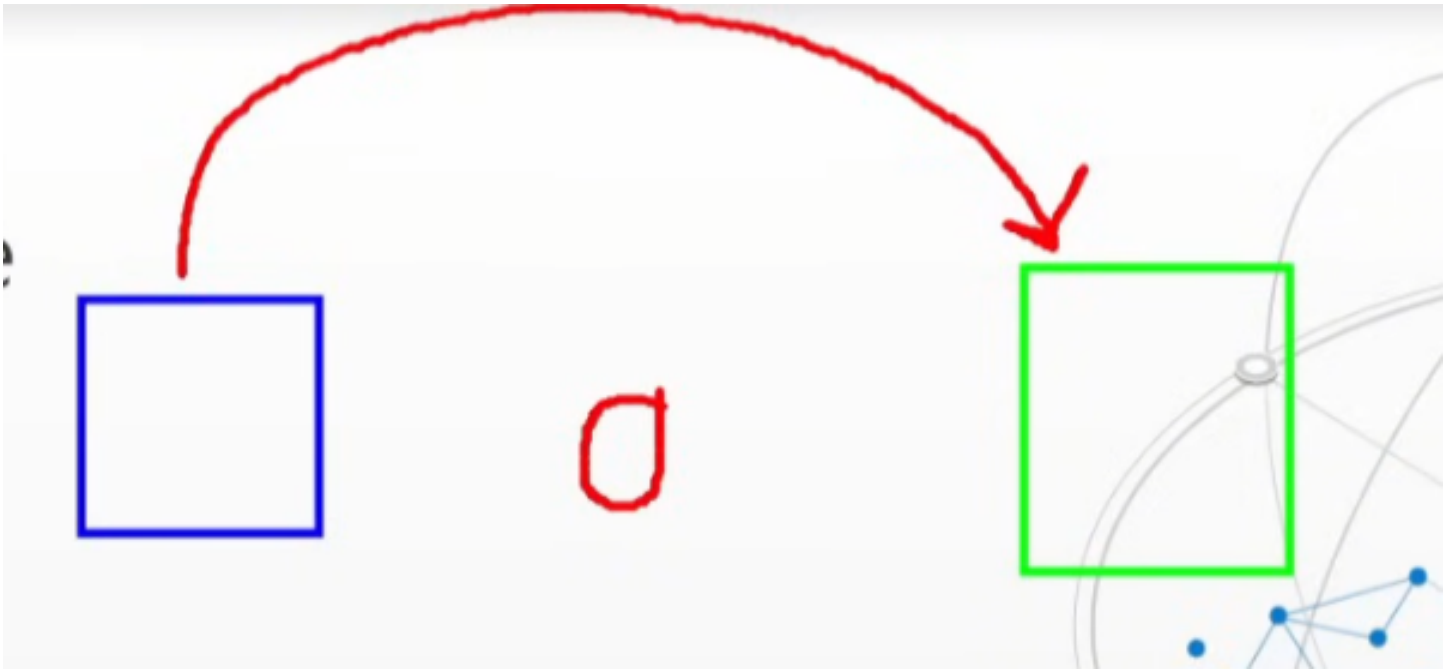
→ In **Dedicated SQL Pool**, provide a **password**.

→ Click **Next: Networking**.

→ Click **Review + Create**.

NOTE : Synapse + Azure Data Factory Integration:

- **No third-party application needed** because both are Azure products.
- By default, Synapse Analytics has a **credential** — you just need to assign a **role** to that credential.

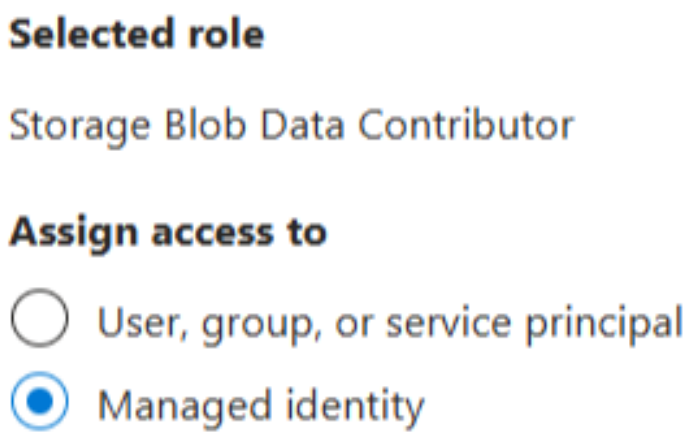


Step 2 : Assigning Role to Synapse Managed Identity (for Data Lake Access)

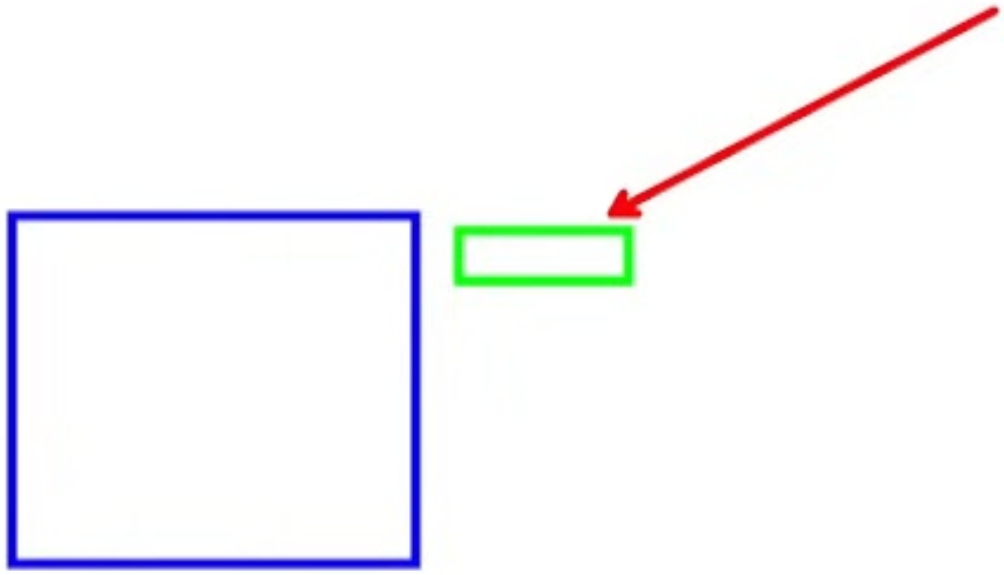
→ Go to **Resource Group** → select the **created Storage Account (data storage)**.

→ Go to **Access Control (IAM)** → click **Add** → click **Role Assignment**.

→ Choose **Role**: Storage Blob Data Contributor → click **Next**.



→ Choose **Managed Identity**.



→ Assigning Role to the Automatically Created Managed Identity:

- Select **Member** → choose **Managed Identity** → select **Synapse Workspace**.
- Select the **Name** of your created Synapse Workspace.

→ Click **Review + Create**.

✅ Now Synapse has access to your Data Lake.

Step 3 : Create a Database in Synapse

→ Go to **Synapse workspace** → **Develop** → **+** → **SQL script**

->Create a database:

- Go to **Data tab** → **+** → **SQL database**
- Name the database → Click **Create**
- Now in **Use Database**, the created database is visible.

Step 5 : Assign Role to Your Own Storage Account

→ Go to **Home** → **Storage account** → your storage account

→ Access control → **Add** → **Add role assignments**

→ User, group → **Select members** → your mail ID → **Create**

→ It will assign the role.

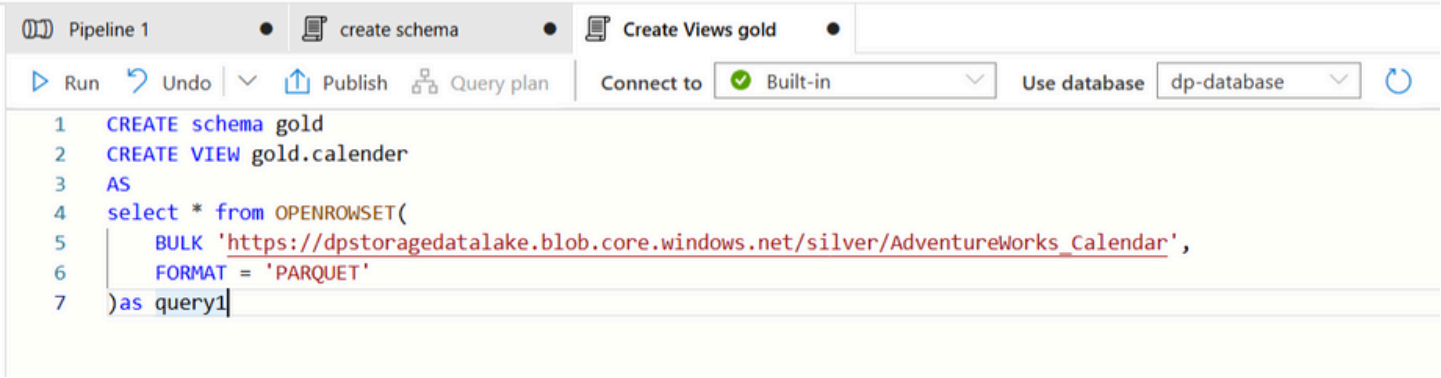
Step 4 : Pull the data in Silver data and create views on top of it.

```
SELECT *
FROM
    OPENROWSET(
        BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Calendar',
        FORMAT = 'PARQUET'
    ) as query1
```

- OPENROWSET lets you **query external data** (like files in cloud storage) as if it were a table, without importing it first.
- BULK means accessing a bulk file (large file like):
  - Parquet

    - CSV
    - JSON
    - ORC
- It tells SQL to **read data in bulk** from external storage like:
    - Azure Data Lake
    - Azure Blob Storage
    - Local file systems (for SQL Server)

Step 5 : Create Schema for Gold Layer



→ Go to **Develop in synapse** → **New SQL Script** → Name: create schema

> Now **create view** in the database we created under the **Gold schema**

Step 6 : Change URL from Blob to DFS

```
BULK 'https://dpstoragedatalake.blob.core.windows.net/silver/AdventureWorks_Calendar',
```

to

```
BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Calendar',
```

- In URL, **storage account by default stores data in Blob Storage**
- **Change to bfs** (used for hierarchical access, i.e., DFS)

Step 7 : Complete Script to Create Views

create schema gold

```
--CREATE VIEW Calender
------------------------

CREATE VIEW gold.calender
AS
select * from OPENROWSET(
   BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Calendar',
   FORMAT = 'PARQUET'
)as query1

------------------------
--CREATE VIEW Customers
------------------------

CREATE VIEW gold.Customers
AS
select * from OPENROWSET(
   BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Customers',
   FORMAT = 'PARQUET'
)as query2

------------------------
--CREATE VIEW Product_Categories
------------------------

CREATE VIEW gold.Product_Categories
AS
select * from OPENROWSET(
   BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Product_Categories',
```

```
    FORMAT = 'PARQUET'
)as query3

------------------------
--CREATE VIEW Product_Subcategories
------------------------

CREATE VIEW gold.Product_Subcategories
AS
select * from OPENROWSET(
    BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Product_Subcategories',
    FORMAT = 'PARQUET'
)as query4

------------------------
--CREATE VIEW Products
------------------------

CREATE VIEW gold.Products
AS
select * from OPENROWSET(
    BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Products',
    FORMAT = 'PARQUET'
)as query5

------------------------
--CREATE VIEW Returns
------------------------

CREATE VIEW gold.Returns
```

```
AS
select * from OPENROWSET(
   BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Returns',
   FORMAT = 'PARQUET'
)as query6

------------------------
--CREATE VIEW Sales
------------------------

CREATE VIEW gold.Sales
AS
select * from OPENROWSET(
   BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Sales',
   FORMAT = 'PARQUET'
)as query7

------------------------
--CREATE VIEW Territories
------------------------

CREATE VIEW gold.Territories
AS
select * from OPENROWSET(
   BULK 'https://dpstoragedatalake.dfs.core.windows.net/silver/AdventureWorks_Territories',
   FORMAT = 'PARQUET'
)as query8
```

After creating views, we need to **create external tables** in Synapse.

Step 8 : Create Master Key for the Database

→ First, we need to create **Master Keys** for this database.

→ Go to the official documentation:
[Create Master Key - Microsoft Docs]

→ Copy this query :

```
CREATE MASTER KEY [ ENCRYPTION BY PASSWORD ='password' ]

[ ; ]
```

remove the square brackets.

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD ='Qwerty@012'
```

Step 9 : Create an External Table in Synapse

To create an external table in Synapse, we need to follow **three main steps**:

1. credential
2. external data source
3. external file format

-> **Credential**

- Used to authenticate access to the external storage (like Azure Data Lake or Blob Storage).
- We need a credential to make Synapse **pull, read, or write** in Data Lake.
- There are many credential types, but we used **Managed Identity**, so we must inform that we used Managed Identity.

use this CREATE DATABASE SCOPED CREDENTIAL (Transact-SQL) - SQL Server ->copy code

```
CREATE DATABASE SCOPED CREDENTIAL credential_name

WITH IDENTITY = 'identity_name'

  [ , SECRET = 'secret' ]

[ ; ]
```

remove brackets

```sql
CREATE DATABASE SCOPED CREDENTIAL cred_prabha
with
    IDENTITY = 'Managed Identity'
```

→ External Data Source

- Defines the location (URL) of your external storage and links it with the credential.
- We need to create **two external data sources** – one for **Silver** and another for **Gold**.

```sql
CREATE EXTERNAL DATA SOURCE source_silver

WITH
(
    LOCATION = 'https://dpstoragedatalake.dfs.core.windows.net/silver',
    CREDENTIAL = cred_prabha
)
CREATE EXTERNAL DATA SOURCE source_gold

WITH
(
    LOCATION = 'https://dpstoragedatalake.dfs.core.windows.net/gold',
    CREDENTIAL = cred_prabha
)
```

The external data source (e.g., source_silver) points to the storage URL. Since access requires authentication, we created a credential (cred_prabha) using Managed Identity. This credential is linked to the data source, allowing Synapse to securely access the data whenever it's queried.

→ External File Format

- Specifies the format of the files (e.g., **Parquet**, **CSV**) so Synapse knows how to read the data.

use → CREATE EXTERNAL FILE FORMAT (Transact-SQL) - SQL Server

```sql
CREATE EXTERNAL FILE FORMAT format_parquet
WITH
(
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
)
```

🔹 **Purpose of Parquet File Format**

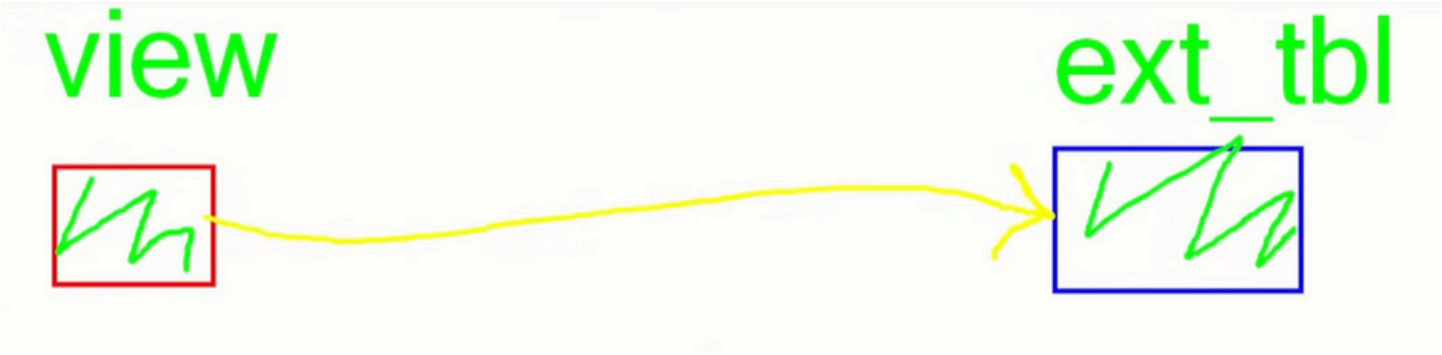
- It is a **columnar storage format** ideal for analytics.
- Enables **faster query performance** and **lower I/O** by reading only the required columns.

🔹 **Purpose of Data Compression (Snappy)**

- **Reduces file size**, saving storage and speeding up data transfers.
- **Snappy** offers a good balance between **compression speed and efficiency**.

Step 10 : Create External Table Using CETAS (Create External Table As Select)

We already have a **view** created on top of our data in the **silver layer**.



Our goal:
Move this processed data to the **gold layer** in a structured form using an **external table**.

What happens step-by-step:

- We **use the view as a source** – since the view already filters or processes the data.
- Using **CETAS**, we:

```
CREATE EXTERNAL TABLE gold_table

WITH (...)

AS

SELECT * FROM silver_view;
```

  - **Push the data** from the view to the Gold layer (in storage).
  - **Create an external table** on top of that gold-layer data.

🔐 **Note:** Make sure you **created the Managed Identity access control** in the **Storage Account** where the **Gold layer** is present.

```sql
-----------------------------------
--CREATE EXTERNAL TABLE EXTSALES
-----------------------------------

CREATE EXTERNAL TABLE gold.extsales
WITH
(
    LOCATION = 'extsales',
    DATA_SOURCE = source_gold,
    FILE_FORMAT = format_parquet
) AS
SELECT * FROM gold.Sales
```

CREATE EXTERNAL TABLE gold.extsales

- This creates an **external table** called extsales under the **gold schema**.
- External tables do **not store data in Synapse SQL**, they point to files in **external storage** (e.g., ADLS Gen2).

📌 WITH (...) - Table Properties

- **LOCATION** = 'extsales'
  → This is a **folder** inside your external storage (like ADLS Gen2).
  → It is **relative to the base path** defined in your DATA_SOURCE.
- Example:
  If your DATA_SOURCE points to:
  https://dpstoragedatalake.dfs.core.windows.net/gold/
  then the full path becomes:
  https://dpstoragedatalake.dfs.core.windows.net/gold/extsales/
- **DATA_SOURCE** = source_gold
  → Refers to a previously created **external data source** that defines:
  - The **base URL** of your storage.
  - The **credential** (like cred_prabha) to access it.
- **FILE_FORMAT** = format_parquet
  → Refers to a defined **external file format** — in this case, **Parquet**.

📌 AS SELECT * FROM gold.Sales

- This is a **CTAS** (Create Table As Select) pattern.
- It reads all data from the **internal table** gold.Sales and writes it to the **external storage** in **Parquet** format under the path extsales.

Step 11 : view Final Output

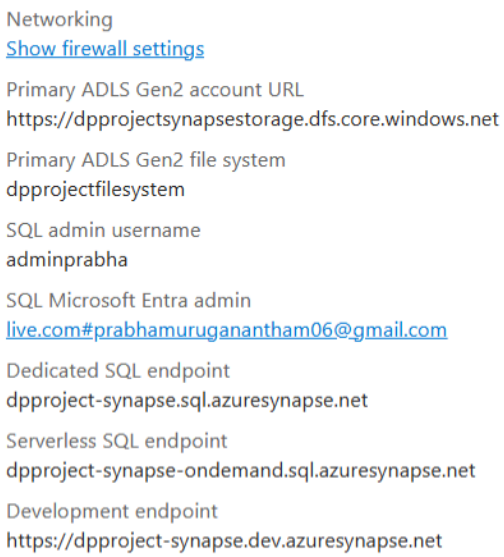
Once all steps are done, you can **see data migrated to the Gold layer** in your **storage account**.

Step 12 : Download Power BI

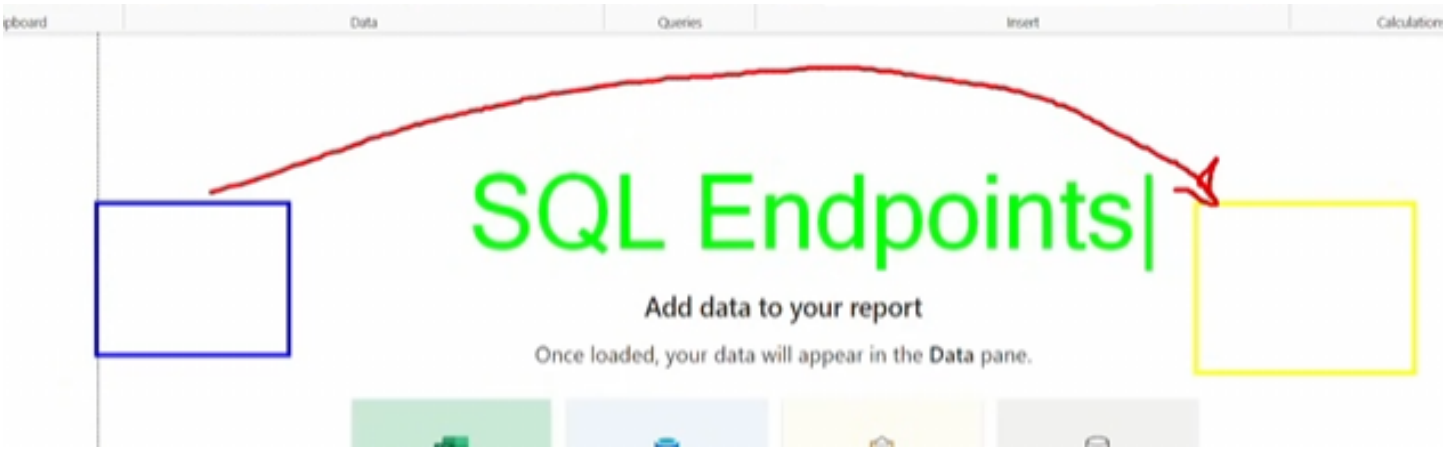
Download Power BI Desktop from the official site:
🔗 [Power BI Download Link]

Step 13 : Get Your Synapse SQL Endpoint

- Go to **Home** → Open your **Synapse Workspace**.



- Copy the **Serverless SQL Endpoint**.



To connect **Power BI** with **Azure Synapse Analytics**, you use the **SQL endpoint** of the Synapse workspace, also called the **Dedicated SQL pool endpoint** or **Serverless SQL endpoint**, depending on your configuration.

Step 14 : Connect Power BI to Synapse

→ Open **Power BI Desktop**.

→ Click on **Get Data**.

→ Select **Azure** → Choose **Azure Synapse Analytics**.

→ Click **Connect**.

→ Paste the **SQL endpoint** you copied from Synapse into the **Server** field.

Step 15 : Authenticate

You can authenticate in two ways:

🔹 Option 1: Microsoft Account

- Choose **Microsoft Account** and **sign in** to connect.

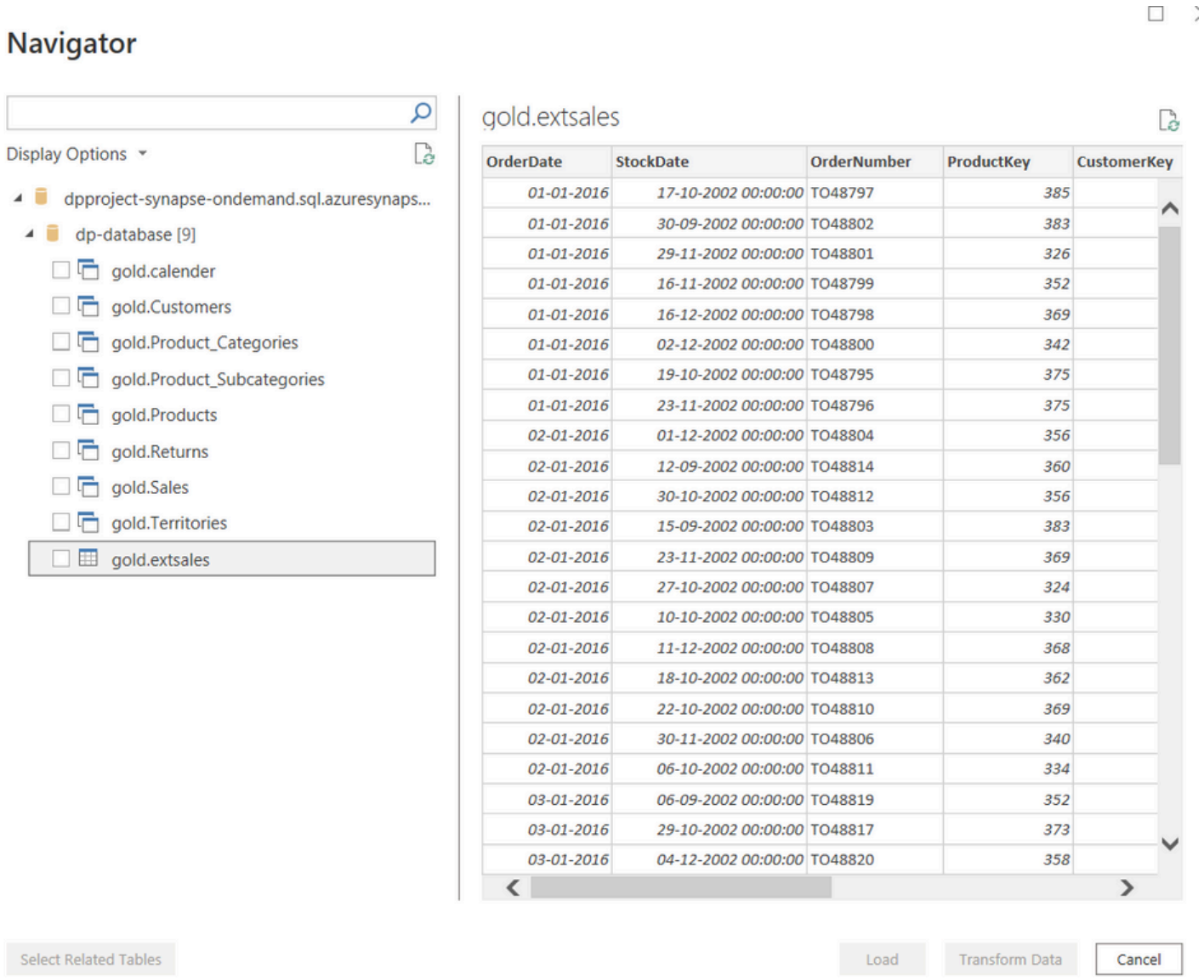
🔹 Option 2: Database Credential

- Use the Synapse admin credentials:
  - **Username**: adminprabha (as you created in Synapse workspace)
  - **Password**: (enter the associated password)

Step 16 : Load Data

In Power BI, **navigate to your database**.

Locate and load the **gold.extsales** table that you created and stored in the **Gold layer**.



OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey
01-01-2016	17-10-2002 00:00:00	TO48797	385	
01-01-2016	30-09-2002 00:00:00	TO48802	383	
01-01-2016	29-11-2002 00:00:00	TO48801	326	
01-01-2016	16-11-2002 00:00:00	TO48799	352	
01-01-2016	16-12-2002 00:00:00	TO48798	369	
01-01-2016	02-12-2002 00:00:00	TO48800	342	
01-01-2016	19-10-2002 00:00:00	TO48795	375	
01-01-2016	23-11-2002 00:00:00	TO48796	375	
02-01-2016	01-12-2002 00:00:00	TO48804	356	
02-01-2016	12-09-2002 00:00:00	TO48814	360	
02-01-2016	30-10-2002 00:00:00	TO48812	356	
02-01-2016	15-09-2002 00:00:00	TO48803	383	
02-01-2016	23-11-2002 00:00:00	TO48809	369	
02-01-2016	27-10-2002 00:00:00	TO48807	324	
02-01-2016	10-10-2002 00:00:00	TO48805	330	
02-01-2016	11-12-2002 00:00:00	TO48808	368	
02-01-2016	18-10-2002 00:00:00	TO48813	362	
02-01-2016	22-10-2002 00:00:00	TO48810	369	
02-01-2016	30-11-2002 00:00:00	TO48806	340	
02-01-2016	06-10-2002 00:00:00	TO48811	334	
03-01-2016	06-09-2002 00:00:00	TO48819	352	
03-01-2016	29-10-2002 00:00:00	TO48817	373	
03-01-2016	04-12-2002 00:00:00	TO48820	358	

Step 17 : Perform Visualization

Once the data is loaded into Power BI, you can now **create reports, dashboards, and visualizations** on top of the **gold.extsales** data.

Finally END to END ETL pipeline is created.....