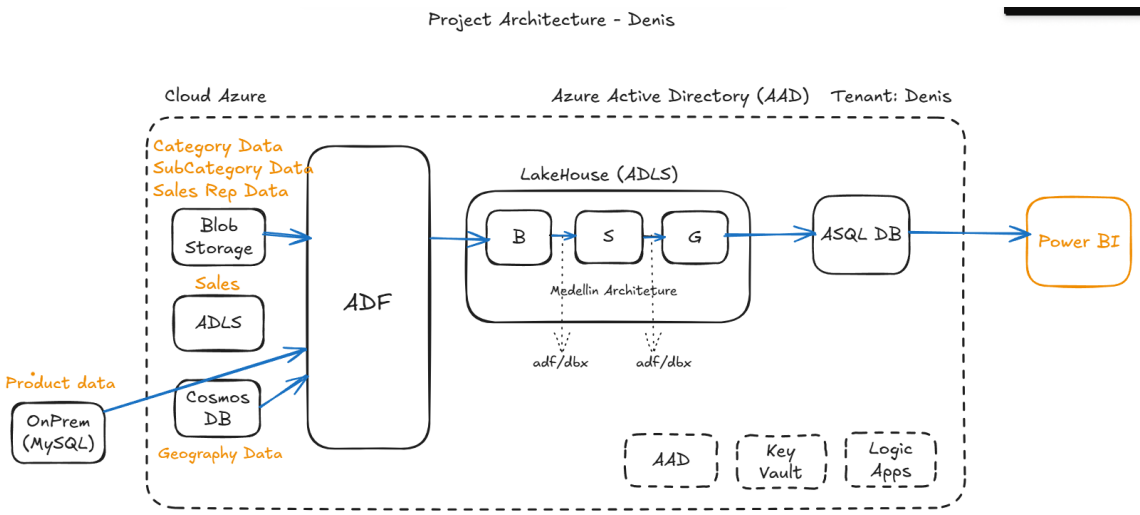
Project Objective: To gather data from various suppliers (On-Prem MySQL, Blob/ADLS, Cosmos DB), process it through the factory manager (Azure Data Factory) and assembly line (Lakehouse – Bronze, Silver, Gold), and store the finished products in the warehouse (Azure SQL DB) for easy access and further use.

Ingest Data from Multiple Sources → Transform with ADF Data Flows → Store in Gold Layer & Azure SQL Database

Project Architecture:



Requried Steps :-

Step 1 :- Sign in to Azure Portal

1.1-Go to https://portal.azure.com

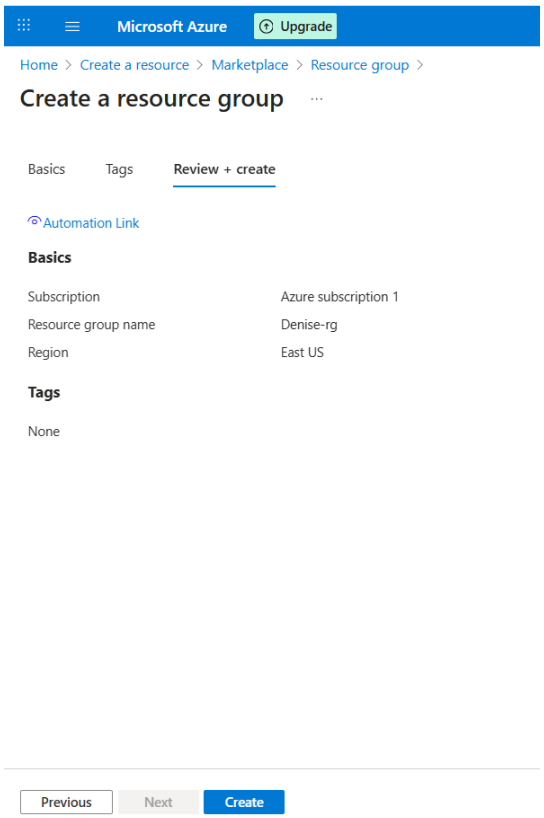
1.2-Sign in with your Microsoft/Azure account

Step 2: Create a Resource Group

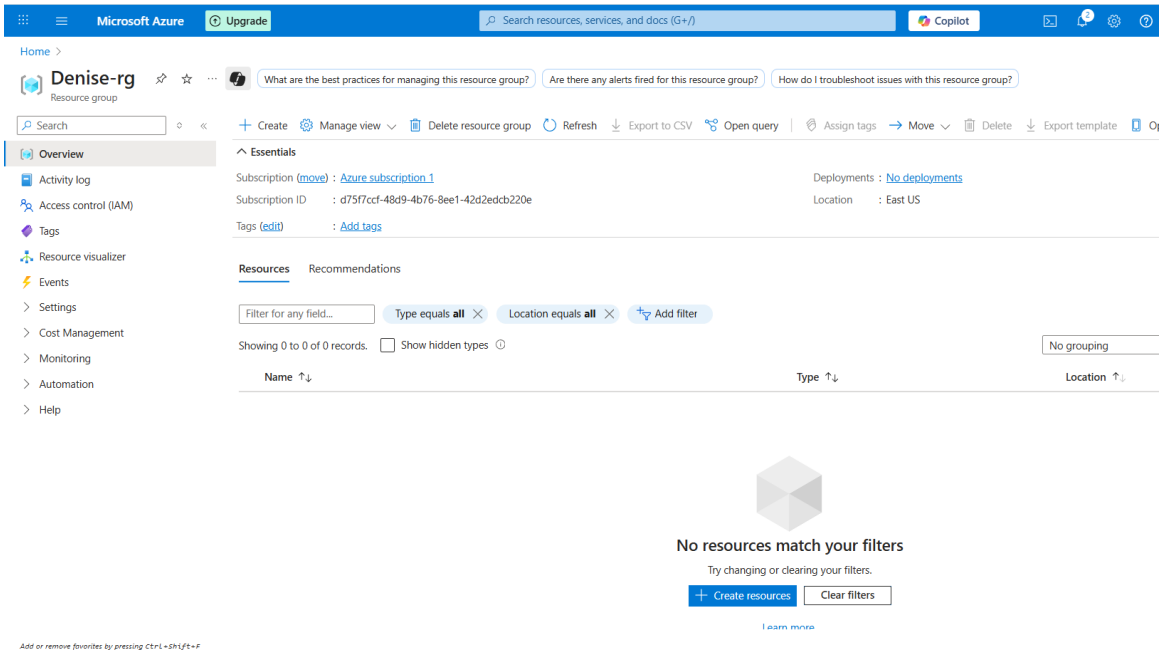
A resource group is like a folder to keep related Azure resources (e.g., storage, databases) organized together.

- **2.1 How to Create:**
  1. In the left-hand menu, click “Resource groups”
  2. Click “+ Create”
- **2.2Fill in Basic:**
  1. **Subscription**: Choose your Azure
  2. **Resource group name**: Denise-RG
  3. **Region**: Select your nearest

- Click Review + Create, then click Create



- **You've now created a resource group!**

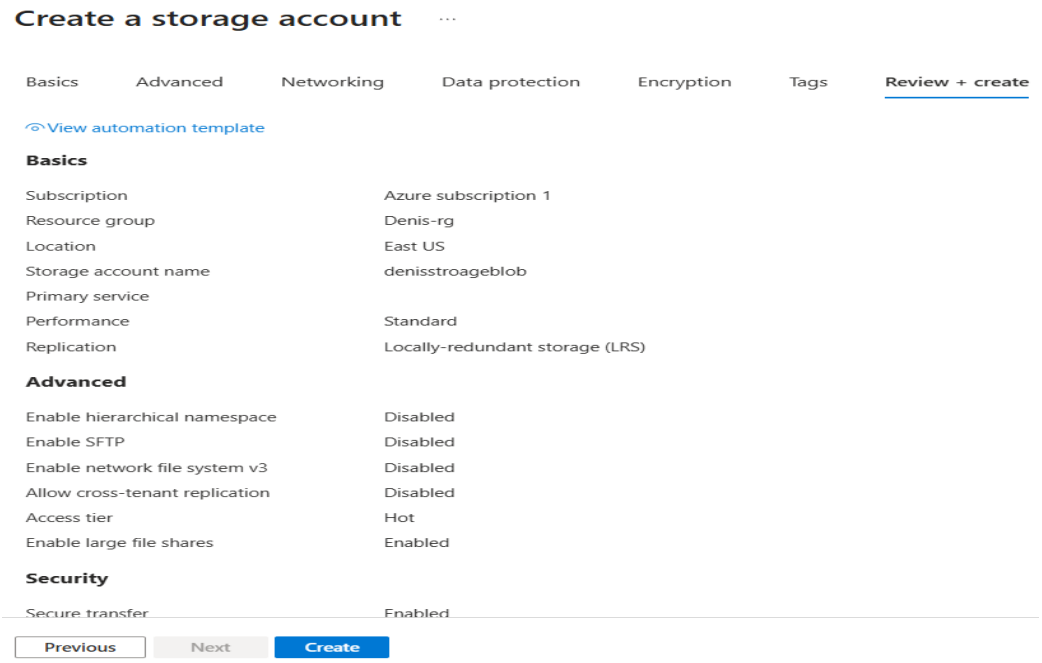


3. Source and Target Setup:-

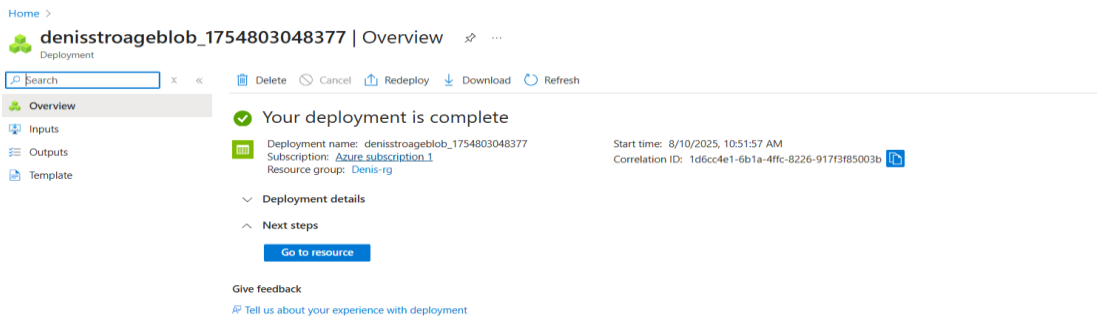
3.1 Azure Blob Storage:-

- Blob storage lives inside something called a storage account — like a big folder in the cloud(without hierarchical)
    1. **Container Name:** e.g., raw-data
    2. **Data Format:** CSV / JSON / Parquet
    3. **Storage Account Name:** denisstroageblob
    4. **Click** "Create a resource"
    5. Search for **"Storage account"** and click Create

Click Review + Create, then Create Reference:



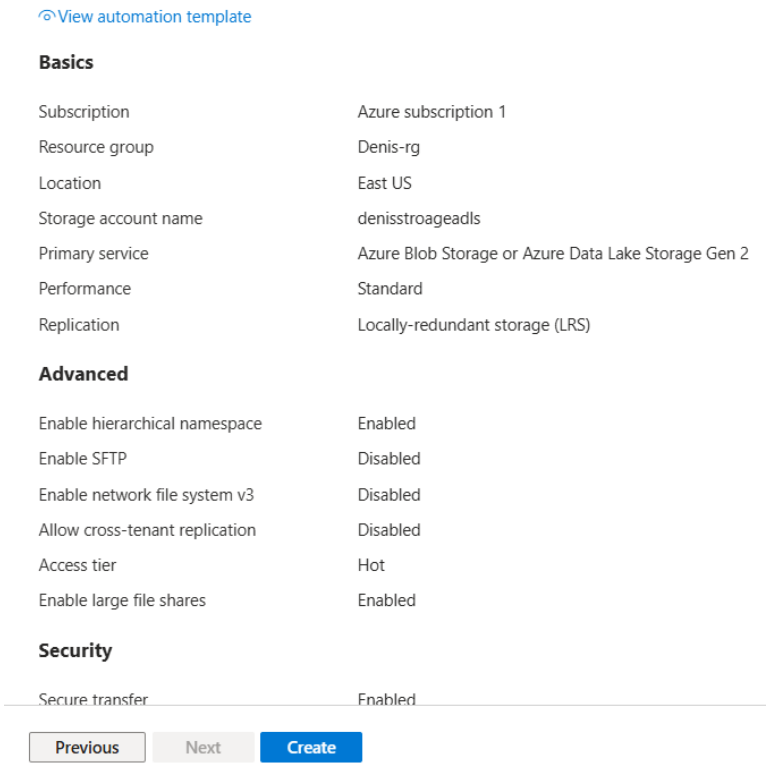
- **Your storage account is now created!**



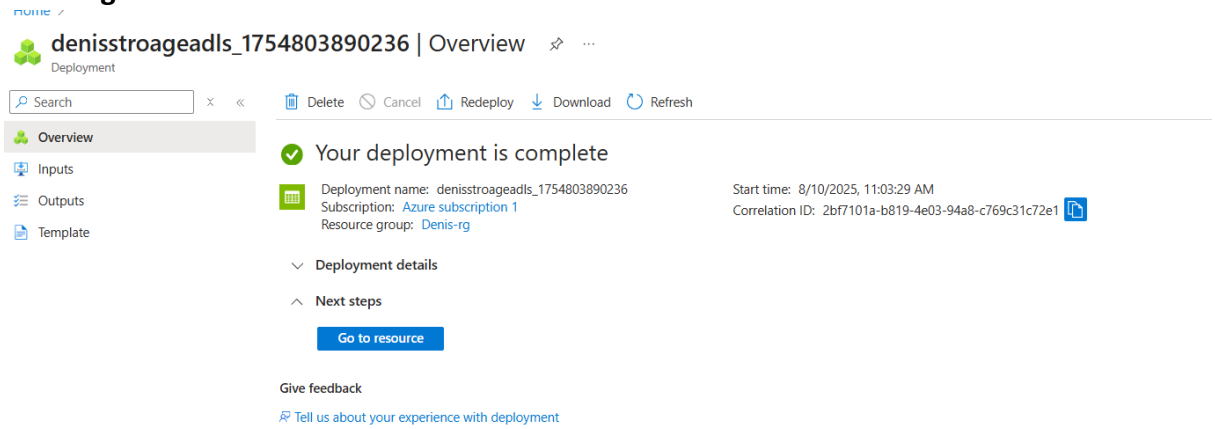
3.2 Azure Blob Storage:-

- Blob storage lives inside something called a storage account — like a big folder in the cloud(with hierarchical)
    1. **Container Name:** e.g., raw-data
    2. **Data Format:** CSV / JSON / Parquet
    3. **Storage Account Name:** denisstroageadls
    4. **Click** "Create a resource"
    5. Search for **"Storage account"** and click Create

Click Review + Create, then Create Reference:



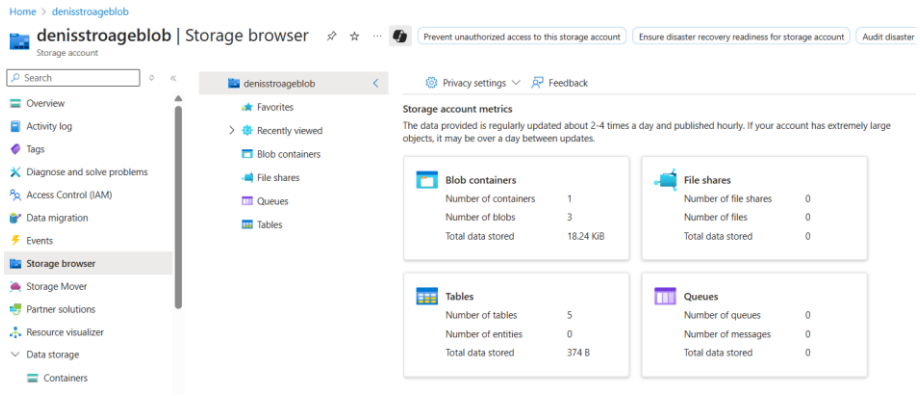
- **Your storage account is now created!**



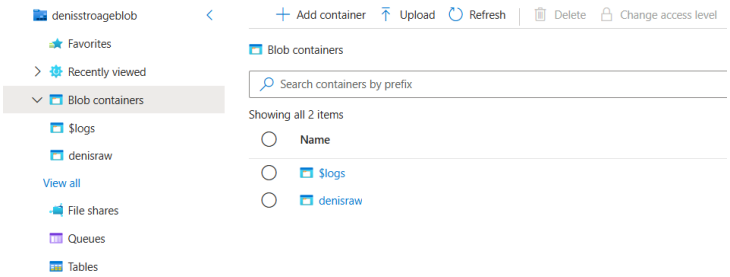
4.Create a Blob Container:-

4.1 Denisstroageblob:-

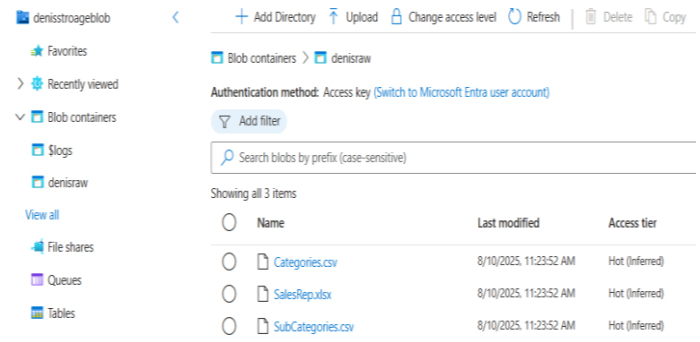
- Containers are like folders inside the storage account to organize files.



- Click "+ Stroage browser" and to the Blob container
- Give it a name : Denisraw

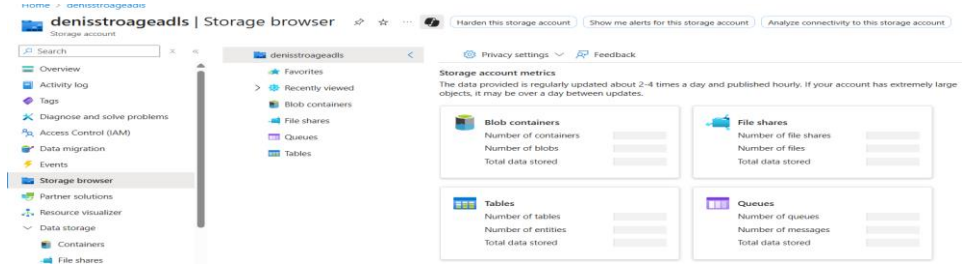


- Click Create
- Now upload data into the denisraw (**Categories.csv, SalesRep.xlsx, SubCategories.csv**)

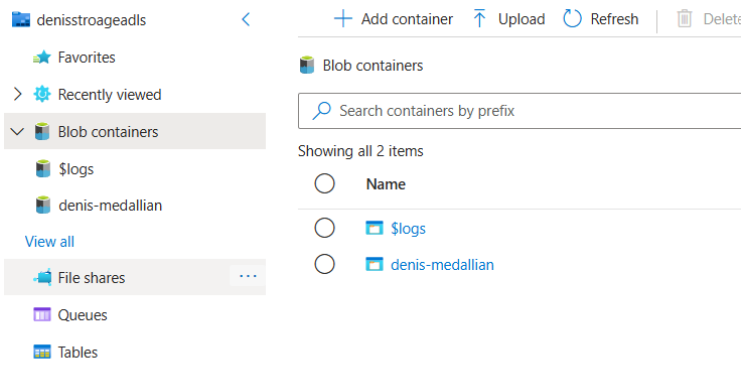


4.2 denisstroageadls:-

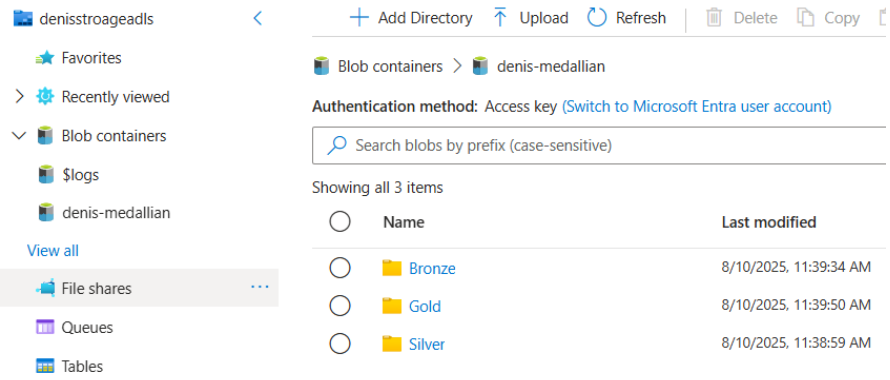
- Containers are like folders inside the storage account to organize files.



- Click "+ Stroage browser" and to the Blob container
- Give it a name : Denis-medallian

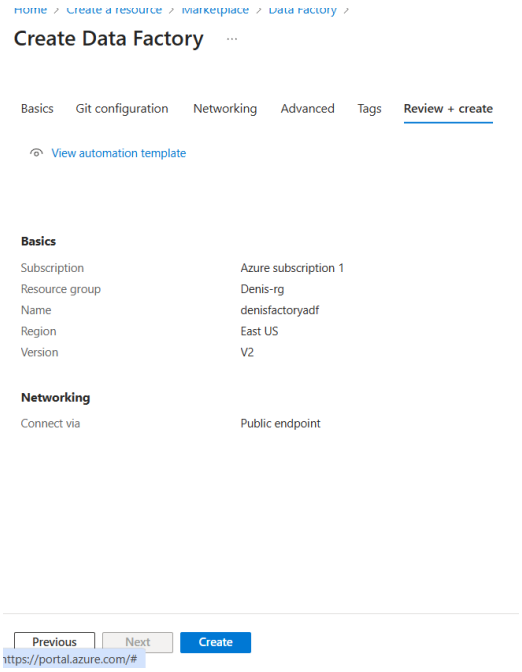


- Create the following directories:
  - `bronze`
  - `silver`
  - `gold`



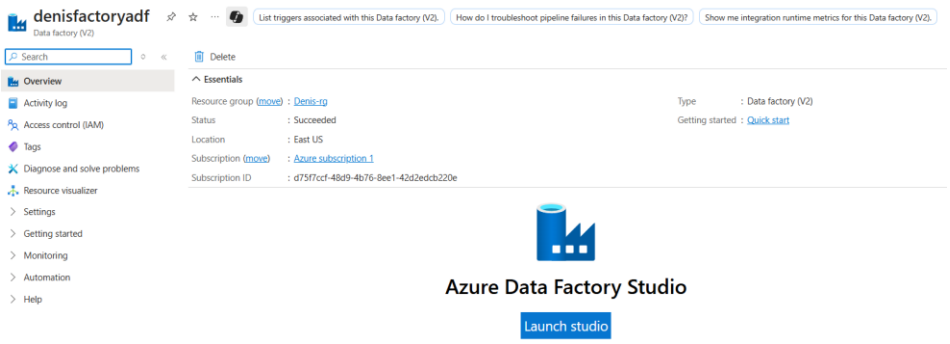
5 Azure Data Factory:-

- Azure Data Factory (ADF) is a cloud-based ETL (Extract, Transform, Load) service that helps you move and transform data between different sources and destination
  - **Data Factory Name:** denisfactoryadf
  - **Version:** V2 (latest)
  - **Region:** Choose a location close to your data and users
  - **Click "Create a resource"**
  - **Search** for **"Data Factory"** and click **Create**



Review + Create:

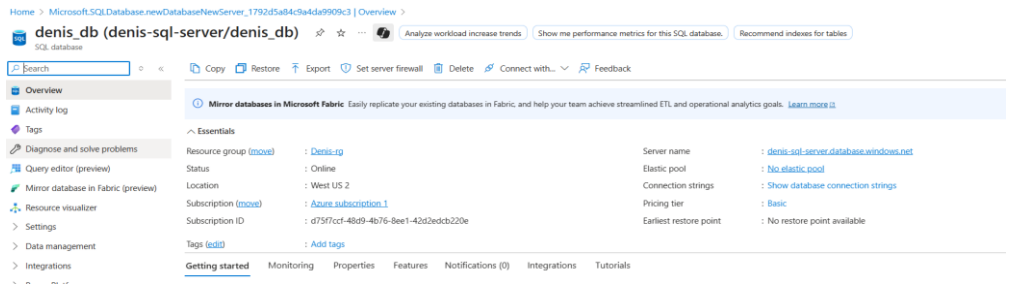
- Check all details and click **Create** to deploy the Data Factory.



6.Azure SQL Database :-

- Azure SQL Database is a fully managed cloud database service that stores structured data and supports T-SQL queries.

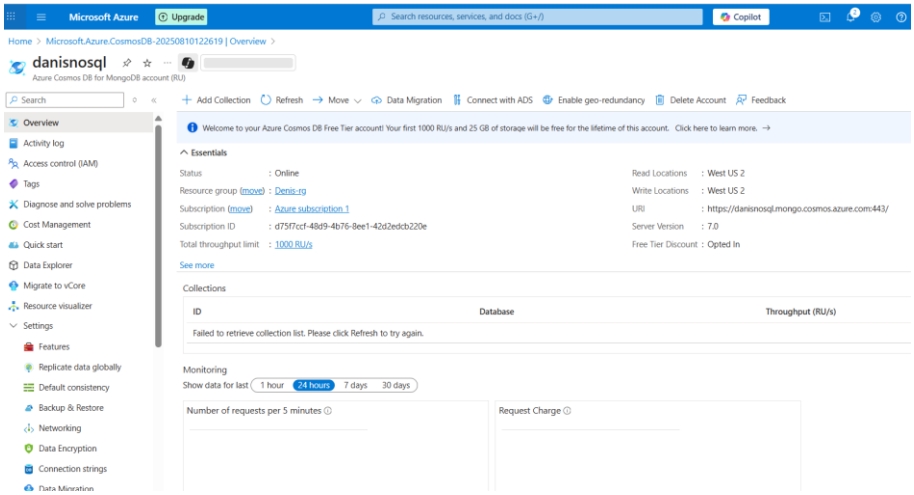
- **Database Name:** denis_db
- **Server Name:** denis-sql-server
- **Compute + Storage:** Basic/Standard/Premium (choose based on needs)
- **Click "Create a resource"**
- **Search** for **"SQL Database"** and click **Create**



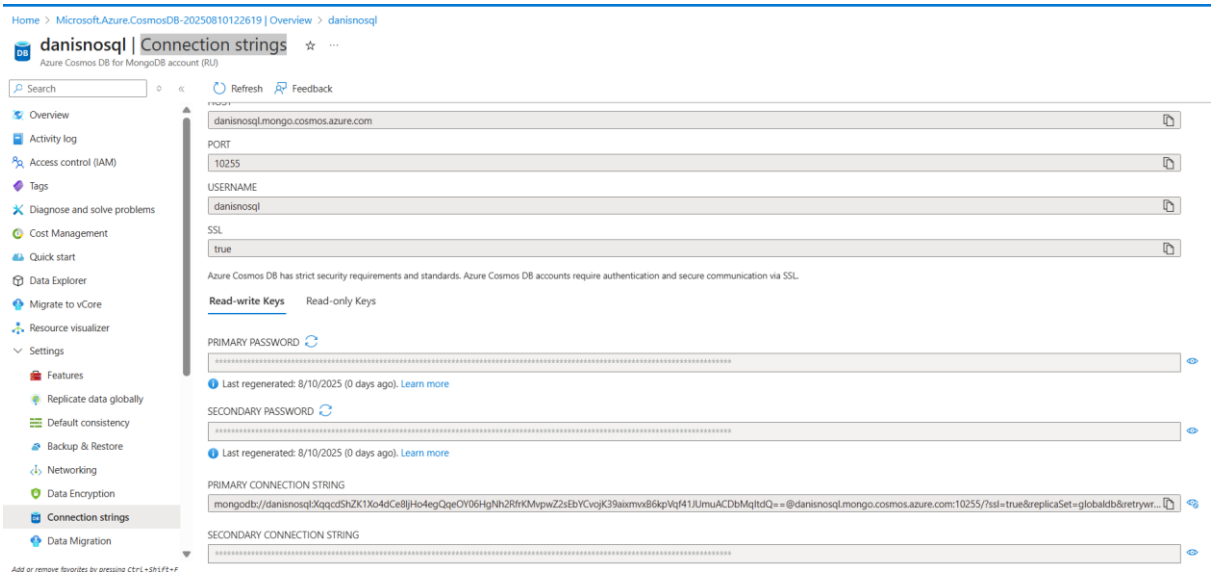
7. Azure Cosmos DB : -

• Azure Cosmos DB is a globally distributed, multi-model NoSQL database service designed for high performance and scalability.

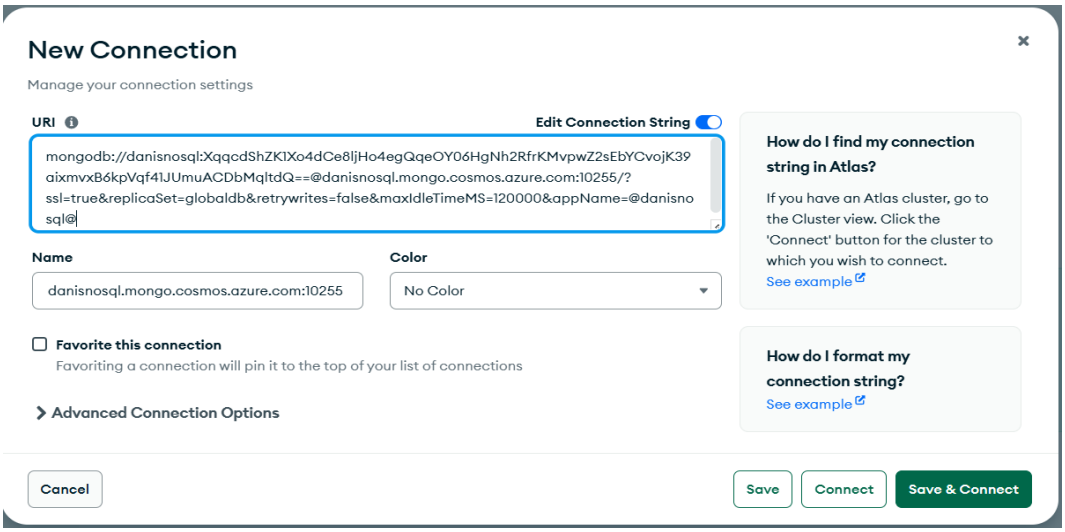
- **Account Name**: e.g., denis-cosmos
- **Workload Type:** Learning
- **API Type:** Core (SQL) – for document data in JSON format
- Capacity Mode: Provisioned throughput (or Serverless for testing)
- Click "Create a resource"
- Search for "Azure Cosmos DB" and click Create



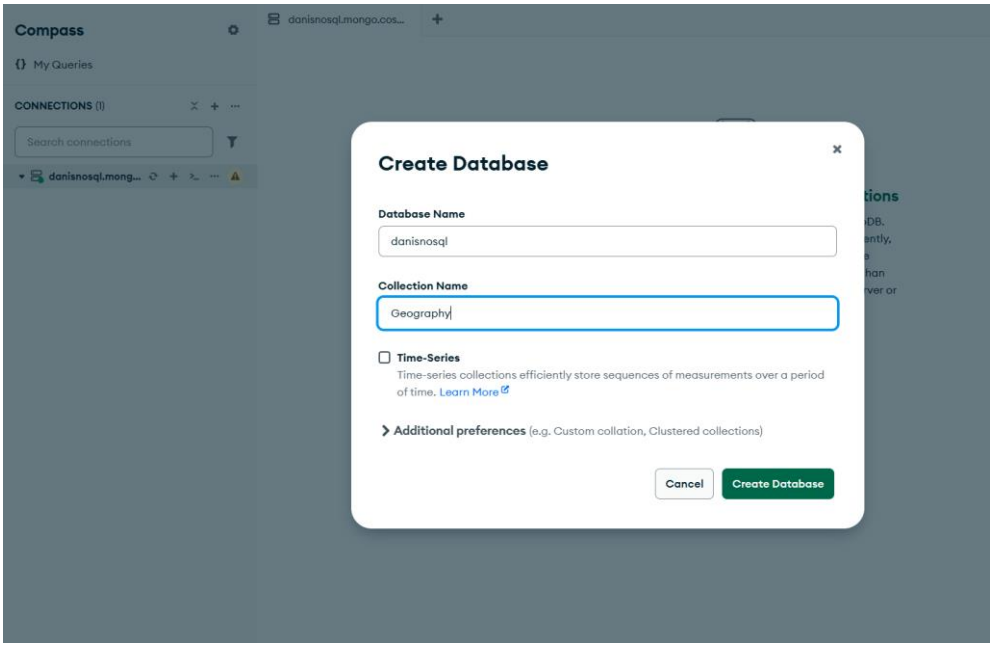
- **Connecting Cosmos DB (MongoDB API) to MongoDB Compass**



- In the Azure Portal, open your Cosmos DB account
- In the left menu, click settings in that press connecting strings.
- Under Connection String, copy the Primary Connection String (this string already contains the authentication details).
- Open MongoDB Compass.
- In the “Paste your connection string (SRV or Standard)” field, paste the copied connection string directly.



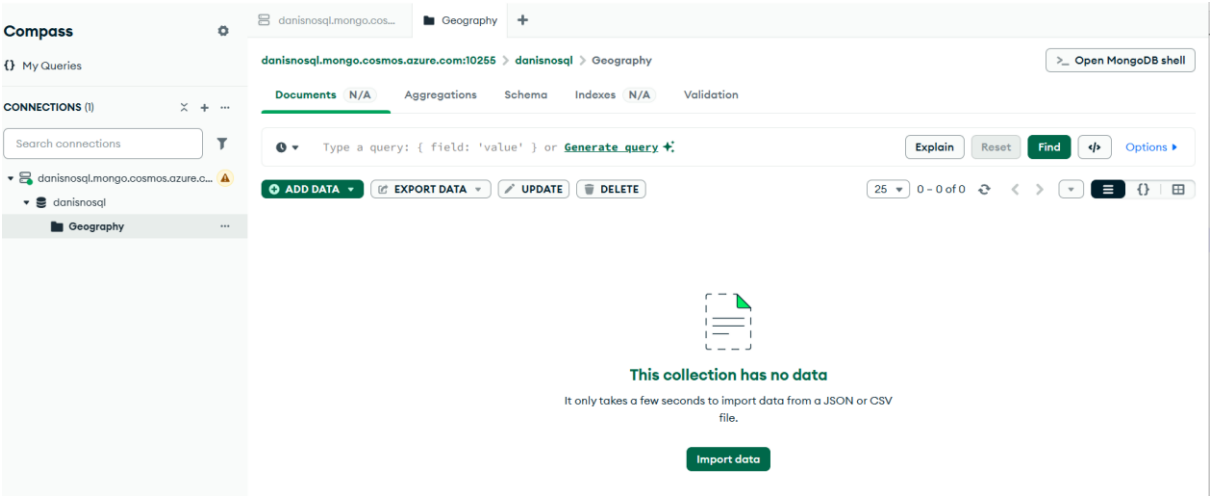
- Click Connect.
- Once connected, click **Create Database**.

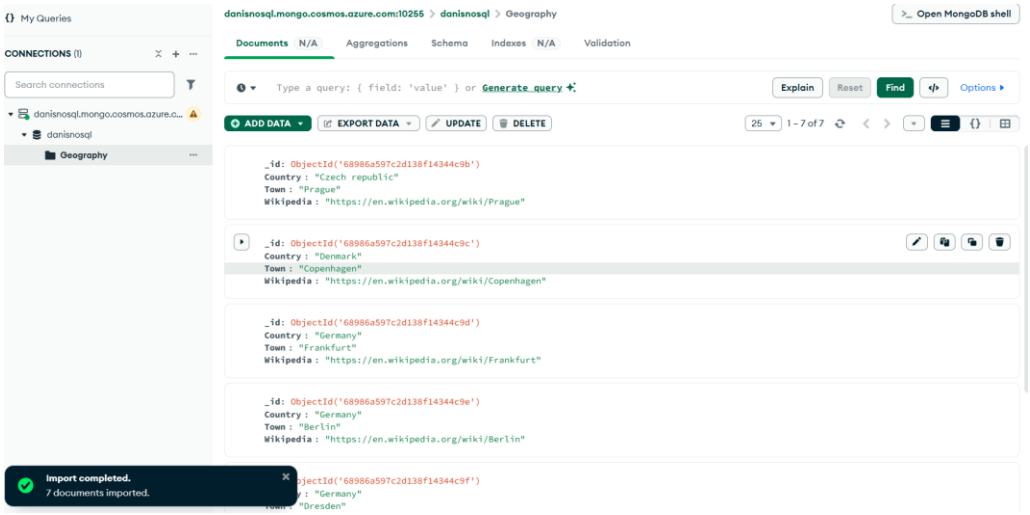


- Enter:
  - **Database Name:** danisnosql
  - **Collection Name:** Geography
- Click **Create Database**.

✅ You now have a new database called **`danisnosql`** in Cosmos DB (MongoDB API), accessible via MongoDB Compass.

- Open the geography collection.
- Click **ADD DATA → Import File**.
- Select the **Geography.json** file from your system.
- Choose **JSON** as the file type and click **Import**.





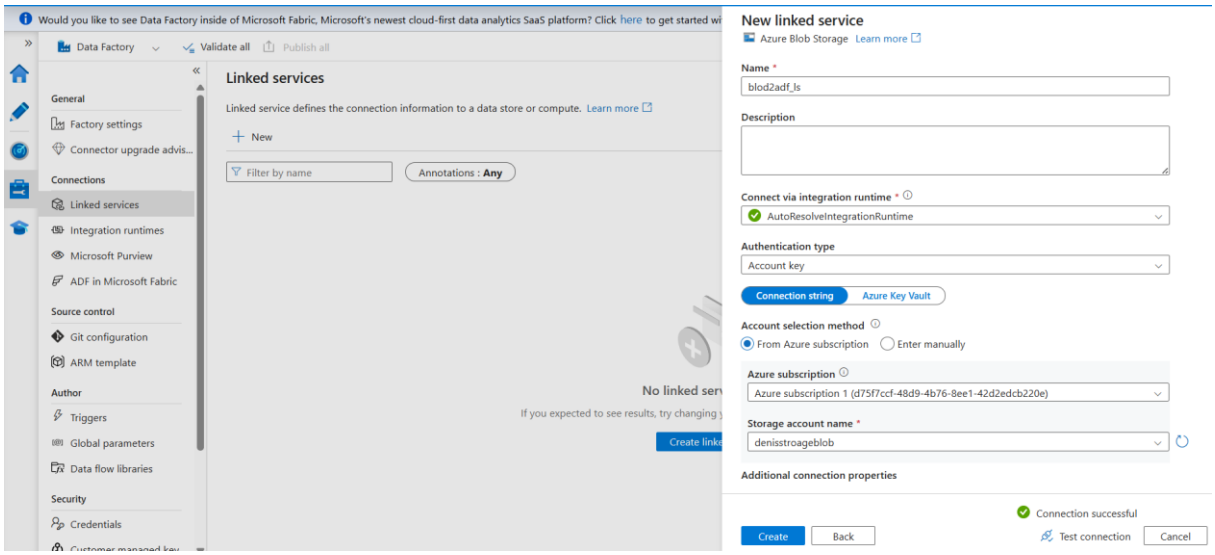
✅ The danisnosql database now contains a geography collection populated from the Geography JSON file.

Creating Linked Services in Azure Data Factory :-

1. Create Linked Service for Azure Blob Storage :-

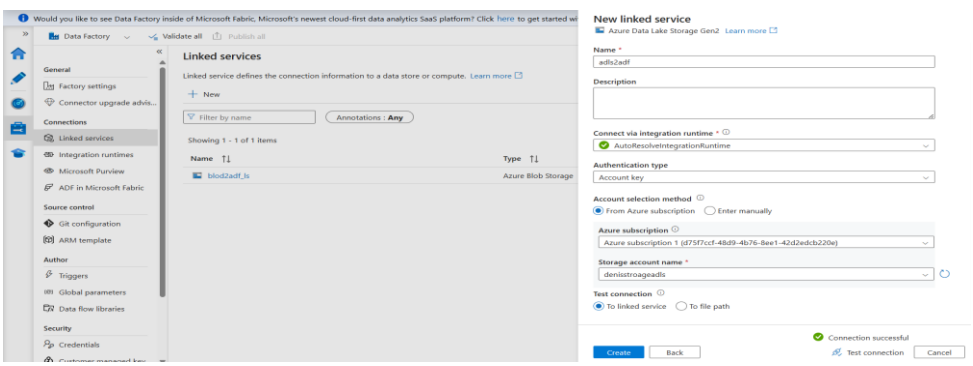
1. Open your Data Factory (denisfactoryadf) in the Azure Portal.
2. In the left menu, click Manage → Linked services.
3. Click + New.
4. Search for Azure Blob Storage
5. Select it and click Continue.
6. Fill in:
7. Name: blod2adf_ls
8. Authentication type: Account key

9. Storage account name: Select your storage account (denisstroageblob)
10. Click Test connection → Create.



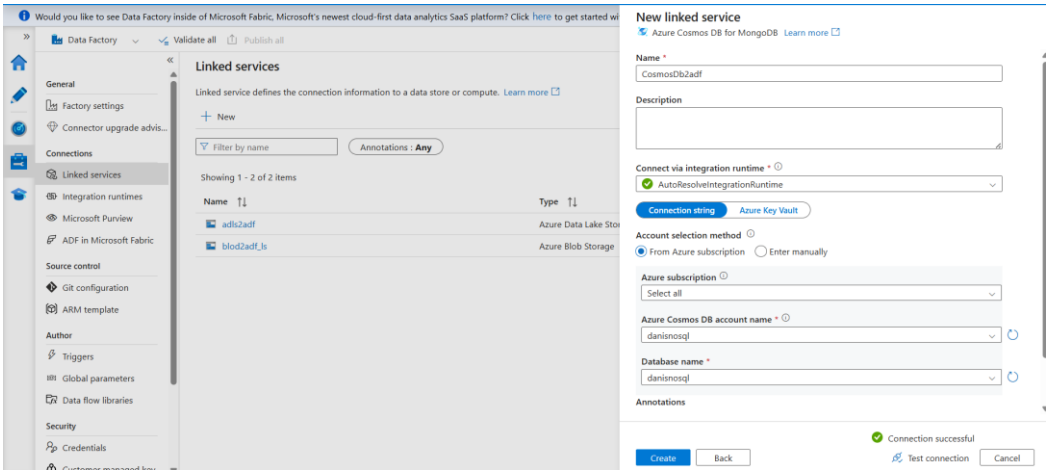
2. Create Linked Service for ADLS:-

1. Open your Data Factory (denisfactoryadls) in the Azure Portal.
2. In the left menu, click Manage → Linked services.
3. Click + New.
4. Search for Azure Data Lake Storage Gen2 (hierarchical)
5. Select it and click Continue.
6. Fill in:
7. Name: adls2adf
8. Authentication type: Account key
9. Storage account name: Select your storage account (denisstroageadls).
10. Click Test connection → Create.



3. Create Linked Service for Azure Cosmos DB (MongoDB API):-

1. Click + New.
2. In the search bar, type Cosmos.
3. Select Azure Cosmos DB (MongoDB API) and click Continue.
4. Fill in the details:
5. Name: CosmosDb2adf_ls
6. Database name: danisnosql
7. Click Test connection to ensure it works.
8. Click Create to save the linked service.

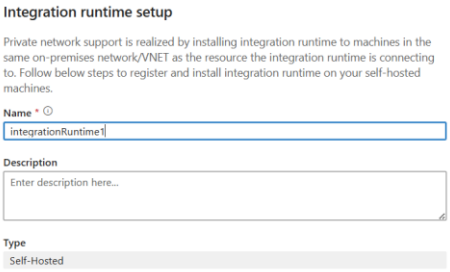


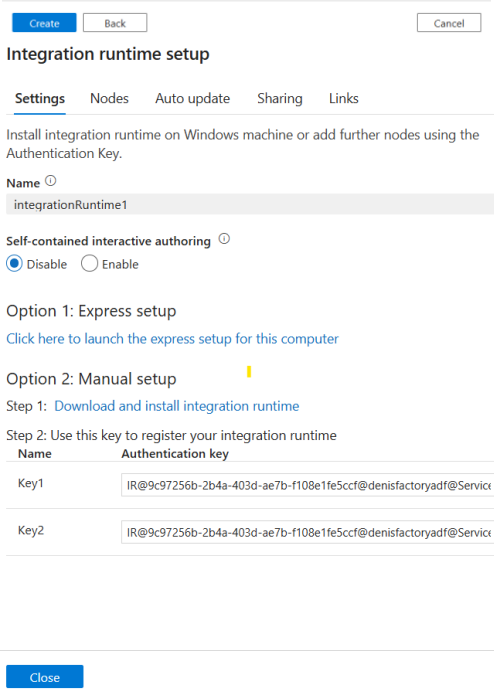
4. Create Linked Service for On-Prem MySQL:-

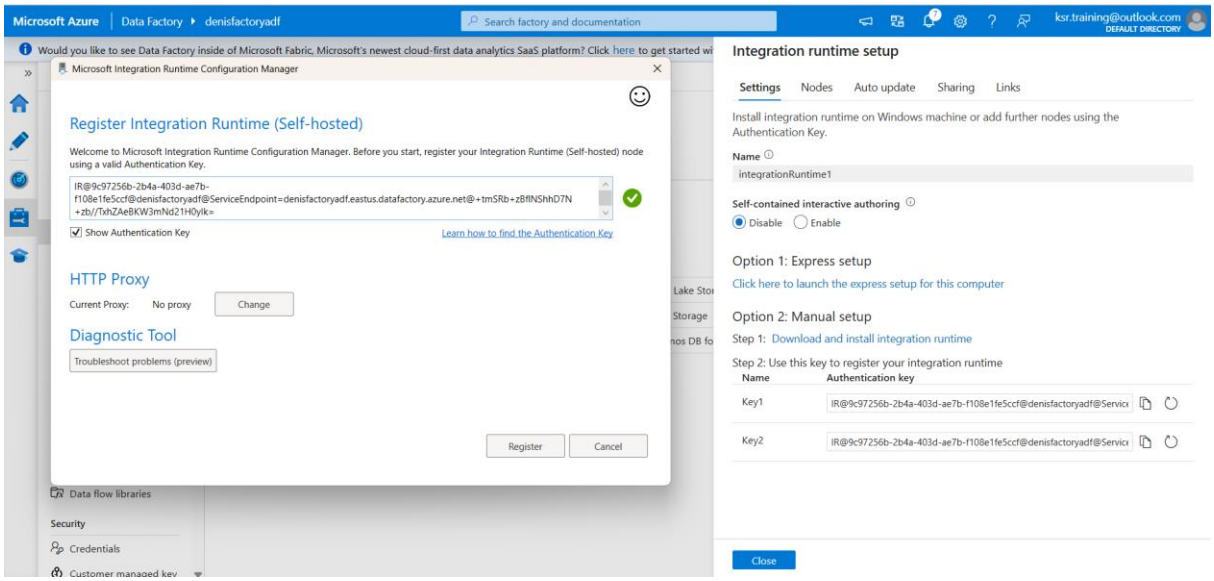
1. + New in Linked Services.
2. Search for MySQL → Continue.
3. Fill in:
4. Name: OnPrem2adf
5. **Connect via integration runtime:-As Self-Hosted:-**

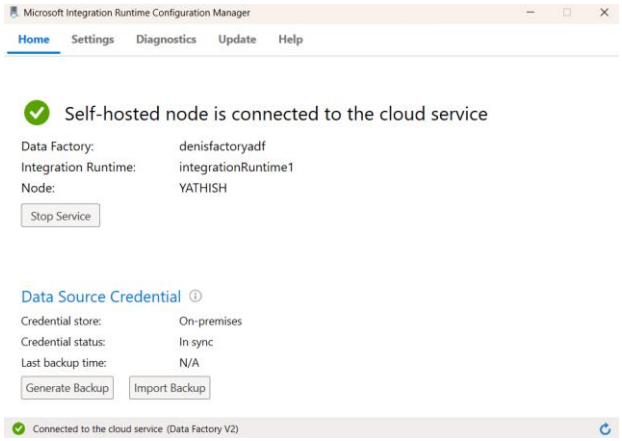
Step – Install Self-Hosted Integration Runtime:-

1. Name it: SelfHosted and click Create.
2. Once created, click Download to get the Self-Hosted Integration Runtime installer.
3. Install it on a machine that can access your On-Prem MySQL server.
4. During installation, it will ask for a key — copy the Authentication Key from ADF and paste it into the installer.
5. Finish installation and ensure the IR status in ADF shows Running.

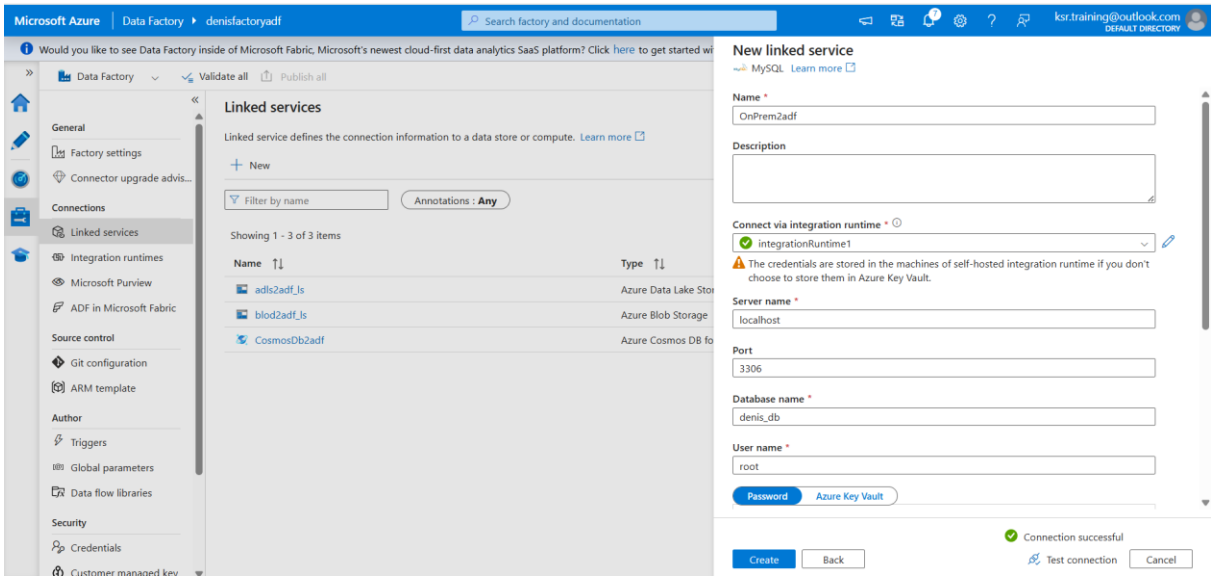

Integration runtime setup
Private network support is realized by installing integration runtime to machines in the same on-premises network/VNET as the resource the integration runtime is connecting to. Follow below steps to register and install integration runtime on your self-hosted machines.
Name *
integrationRuntime1
Description
Enter description here...
Type
Self-Hosted


Create
Back
Cancel
Integration runtime setup
Settings
Nodes
Auto update
Sharing
Links
Install integration runtime on Windows machine or add further nodes using the Authentication Key.
Name
integrationRuntime1
Self-contained interactive authoring
Disable
Enable
Option 1: Express setup
Click here to launch the express setup for this computer
Option 2: Manual setup
Step 1: Download and install integration runtime
Step 2: Use this key to register your integration runtime
Name
Authentication key
Key1
IR@9c97256b-2b4a-403d-ae7b-f108e1fe5ccf@denisfactoryadf@Servic
Key2
IR@9c97256b-2b4a-403d-ae7b-f108e1fe5ccf@denisfactoryadf@Servic
Close

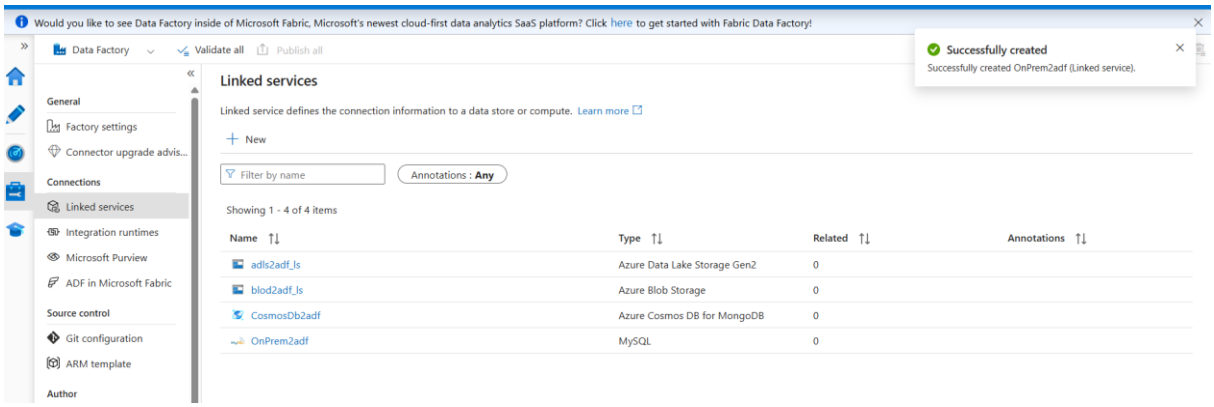

Microsoft Azure
Data Factory
denisfactoryadf
Search factory and documentation
ksr.training@outlook.com
DEFAULT DIRECTORY
Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started wi
Microsoft Integration Runtime Configuration Manager
Register Integration Runtime (Self-hosted)
Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.
IR@9c97256b-2b4a-403d-ae7b-f108e1fe5ccf@denisfactoryadf@ServiceEndpoint=denisfactoryadf.eastus.datafactory.azure.net@+tmSRb+zBflNShhD7N+zb//TxhZAeBKW3mNd21H0yIk=
Show Authentication Key
Learn how to find the Authentication Key
HTTP Proxy
Current Proxy:
No proxy
Change
Diagnostic Tool
Troubleshoot problems (preview)
Register
Cancel
Data flow libraries
Security
Credentials
Customer managed key
Integration runtime setup
Settings
Nodes
Auto update
Sharing
Links
Install integration runtime on Windows machine or add further nodes using the Authentication Key.
Name
integrationRuntime1
Self-contained interactive authoring
Disable
Enable
Option 1: Express setup
Click here to launch the express setup for this computer
Option 2: Manual setup
Step 1: Download and install integration runtime
Step 2: Use this key to register your integration runtime
Name
Authentication key
Key1
Key2
Close



After completed the setup now fill the details

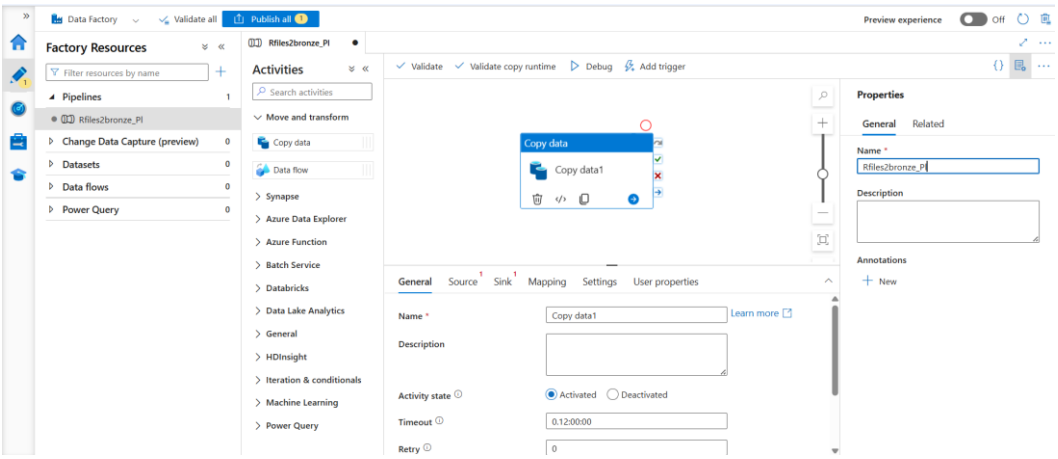


6. **Server name :-**localhost
7. **Port:-**3306
8. **Database name :-** denise_db
9. **Username :-** root
10. **Password :-** **********
11. Test connection → Create.



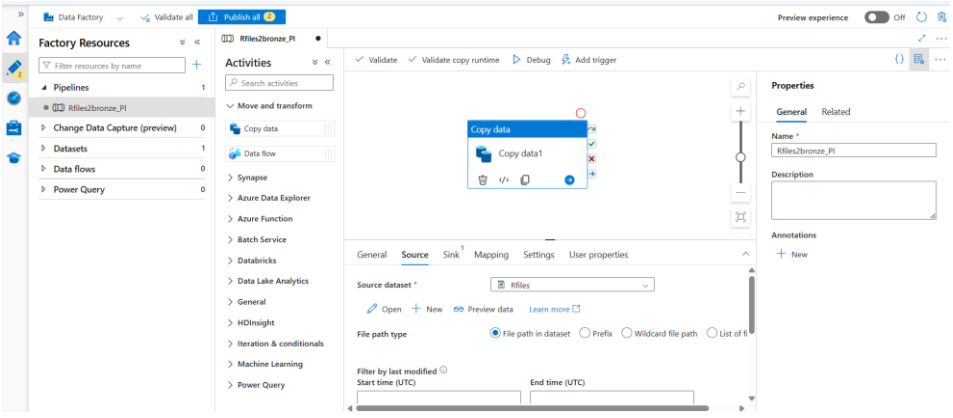
1.Creating the Rfiles2bronze_Pl Pipeline:-

1. Go to Azure Data Factory → Author tab.
2. Create new pipeline → Name it: **Rfiles2bronze_Pl**.
3. Drag a Copy Data activity into the canvas.



Source Settings

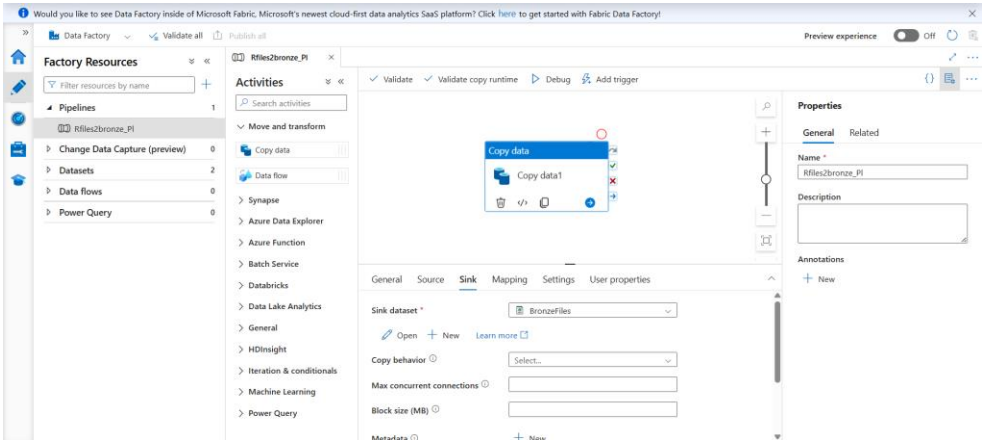
- Dataset type → Azure Blob Storage.
- Linked Service → your Blob linked service(blod2adf_ls)
- Container → Rfiles.
- File format → CSV.
- First row as header → checked.
- Wildcard path (optional) if you want to copy all CSVs in the folder: *.csv.



Sink Settings

- Dataset type → Azure Data Lake Storage Gen2.
- Linked Service → your ADLS linked service.
- Container → bronzefiles.

1. Path → folder name we created earlier for Bronze stage (e.g., denis-medallian/bronze/).
2. Copy behavior → *Preserve hierarchy* (if you want same folder structure as source)
   or *Flatten hierarchy* (if you just want all files in one place).



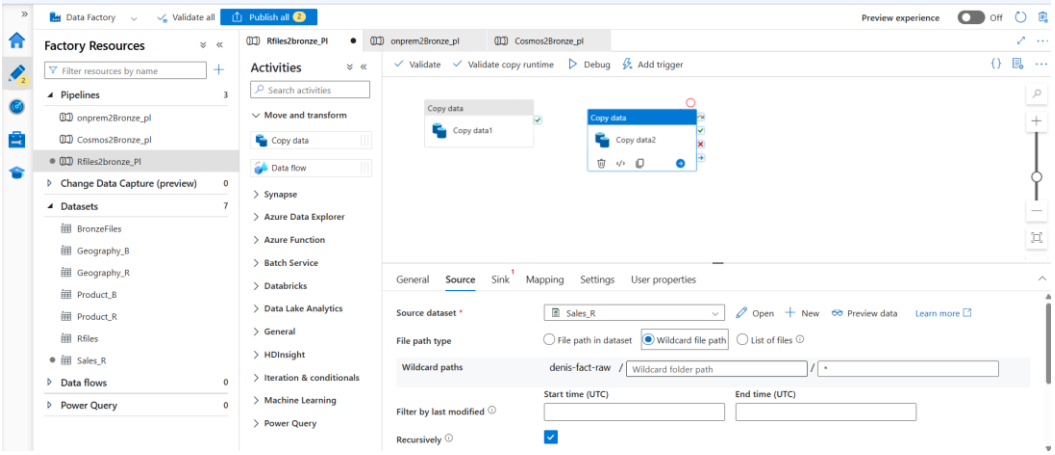
Final Steps:- Once successful → Publish All.

In same pipeline we doing another activate also

1.1ADLS2bronze_Pl

Source

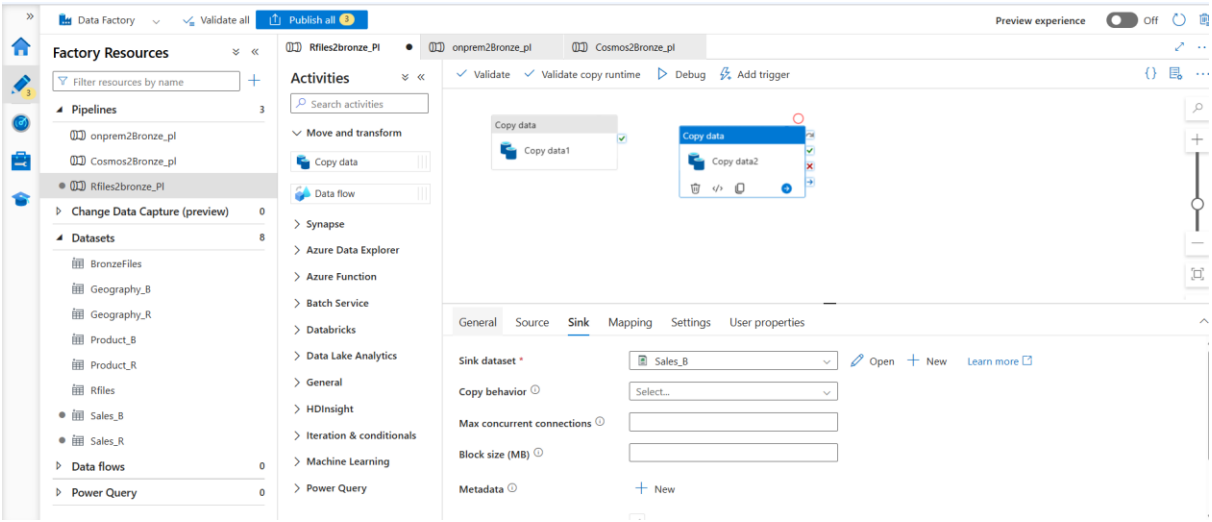
- Dataset: **Azure Data Lake Storage Gen2**.
- File format: Select CSV / JSON / Parquet depending on your file type.
- Linked Service: adls2adf_ls
- Path: *wherever your source files are*
- **If multiple files:** Enable **"Wildcard file path"** and use something like:



Sink

- Dataset: **Azure Data Lake Storage Gen2**.
- Linked Service: **Same ADLS linked service**.
- Path: / **denis-medallian**bronze/Sales/*(target folder)*.
- File format: Same as source (ADF will copy without changing type unless mapping says otherwise).

- If you want to preserve file names → enable **"Preserve hierarchy"** option in copy activity.

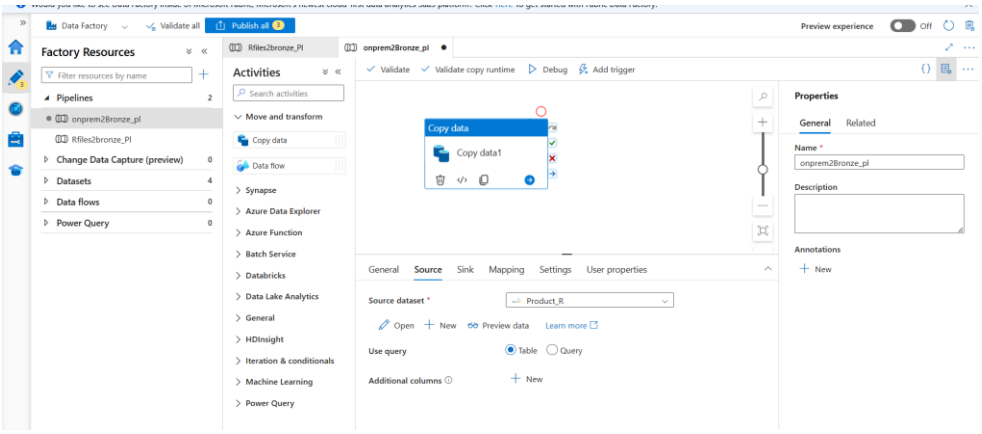


2.Pipeline: MySQL2bronze_Pl :-

- Go to Azure Data Factory → Author → New Pipeline. Name → MySQL2bronze_Pl.
- Add a Copy Data activity

Source Settings :-

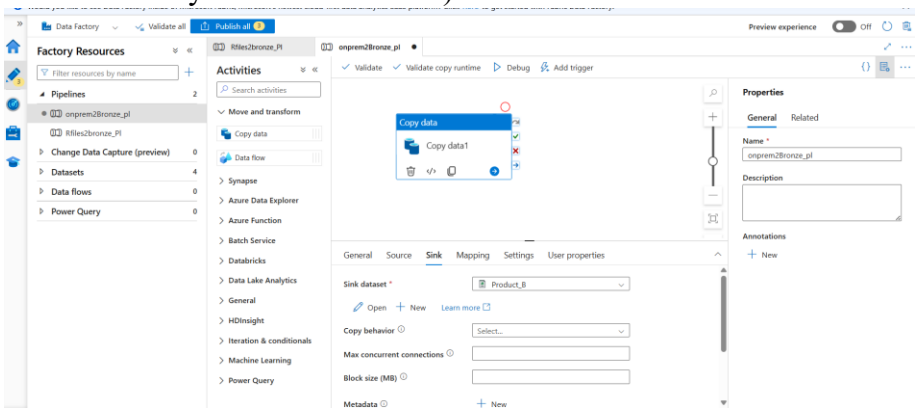
- Dataset type → MySQL(local machine).
- Linked Service → OnPrem2adf (We created)
- Table → Select the table you want to copy from MySQL**.**



Sink Settings

- Dataset type → **Azure Data Lake Storage Gen2**.
- File format → Parquet or CSV (Parquet is recommended for later transformations).
- Linked Service →adls2bronze
- Container → bronzefiles.

- Path → e.g., denis-medallian/bronze/ **Product.csv** (so you don't overwrite your blob CSV data).



Extra Note for On-Prem:- Since this is **on-prem MySQL**, you'll need:

- **Self-hosted Integration Runtime (SHIR)** installed and running.
- SHIR linked to your ()linked service.

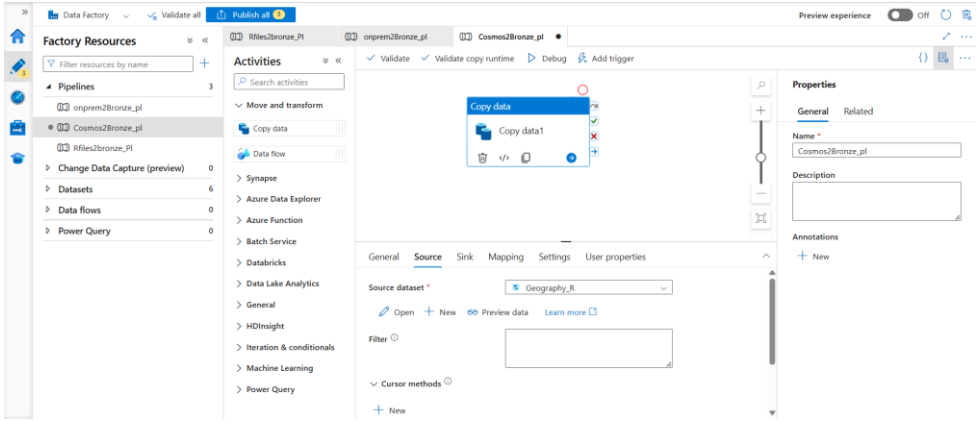
Publish All.

3.Pipeline: Cosmos2bronze_Pl

1. **Go to** Azure Data Factory → **Author** → **New Pipeline**.
   Name → Cosmos2bronze_Pl.
2. **Add** a **Copy Data** activity.

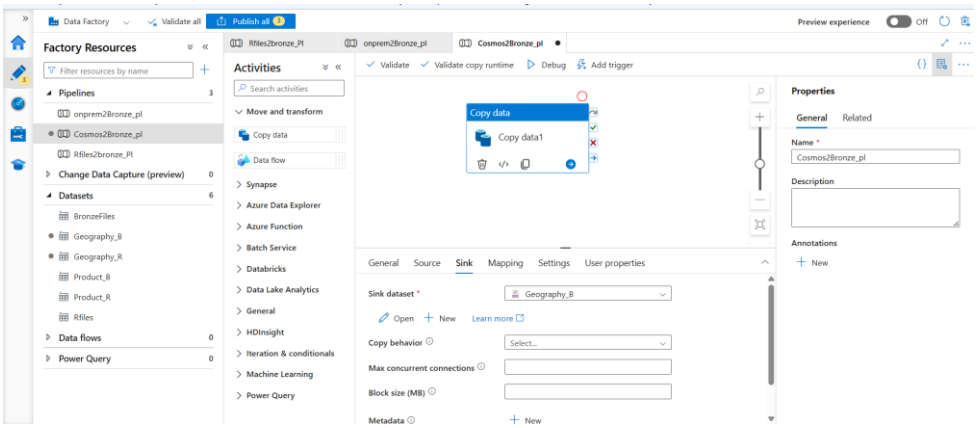
Source (Cosmos DB)

- Dataset → **Azure Cosmos DB**.
- Linked Service → CosmosDb2adf
- Choose your Database and Collection.(Json)



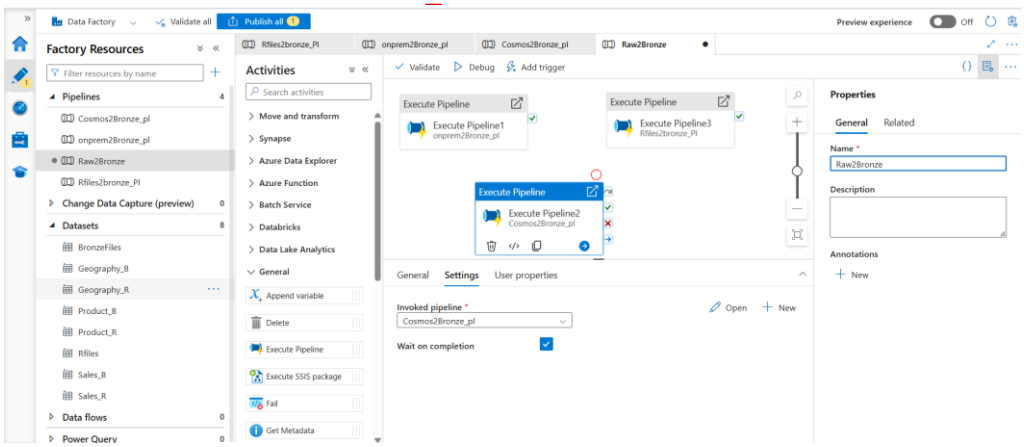
Sink (ADLS)

- Dataset → **Azure Data Lake Storage Gen2**.
- File format → JSON.
- Linked Service → adls2adf_ls.
- Path → **denis-medallian**/bronze/ **Geography.json** (to keep data separate).

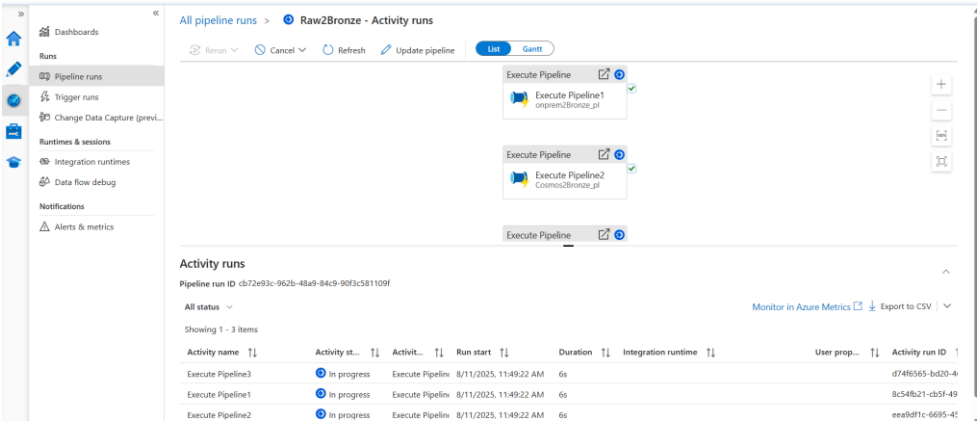


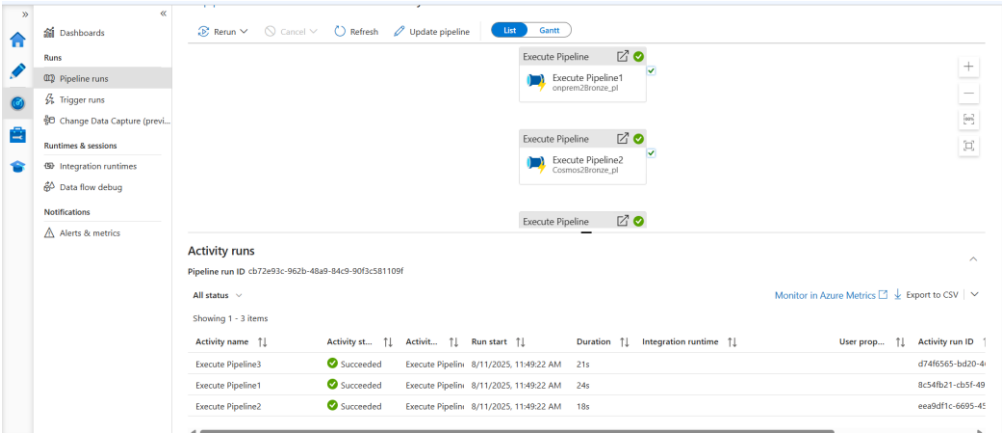
Master Pipeline Setup(Raw2Bronze)

1. **Go to** Azure Data Factory → **Author** → **New Pipeline**
   Name it → **Raw2Bronze**_PL.
2. **Add "Execute Pipeline" activities**
   - From Activities pane → Search for **Execute Pipeline**.
   - Drag one into the canvas for each pipeline you want to call.
3. **Configure Each Execute Pipeline**
   - **Settings** → Choose the pipeline you want to run.
     Example:
     - First → Rfiles2bronze_Pl same inside ADLS2bronze_Pl
     - Second → MySQL2bronze_Pl
     - Third → Cosmos2bronze_Pl

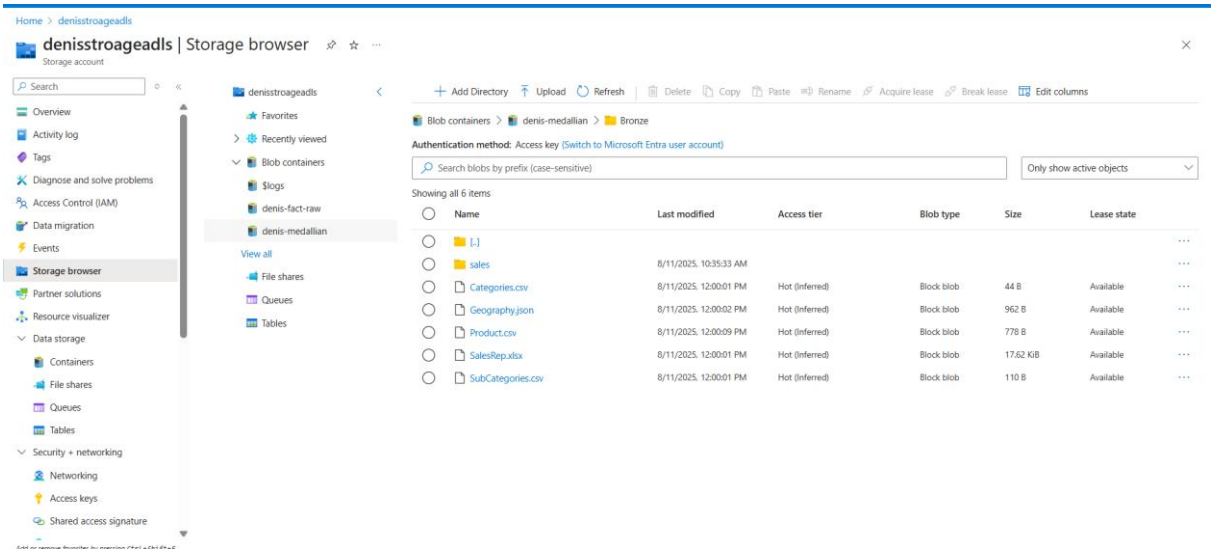


4. **Link them in sequence or parallel**
   - **If you want all at once (parallel)** → Don't connect activities with arrows, just place them side-by-side.
   - **If you want one after another (sequence)** → Connect with **success (green)** arrows.
5. **Publish & Trigger**
   - Click **Publish All**
   - Trigger the **Master_AllBronze_PL** → This will execute all child pipelines as per your arrangement.





Perfect 🚀!
So your **Raw → Bronze** data movement is fully functional now — from Blob, MySQL, and Cosmos DB all landing neatly in ADLS Bronze folders.



The logical next step in the **medallion architecture** would be:

1. **Bronze → Silver**:
   - Clean, validate, and standardize the data
   - Convert formats if needed (e.g., CSV → Parquet for analytics)

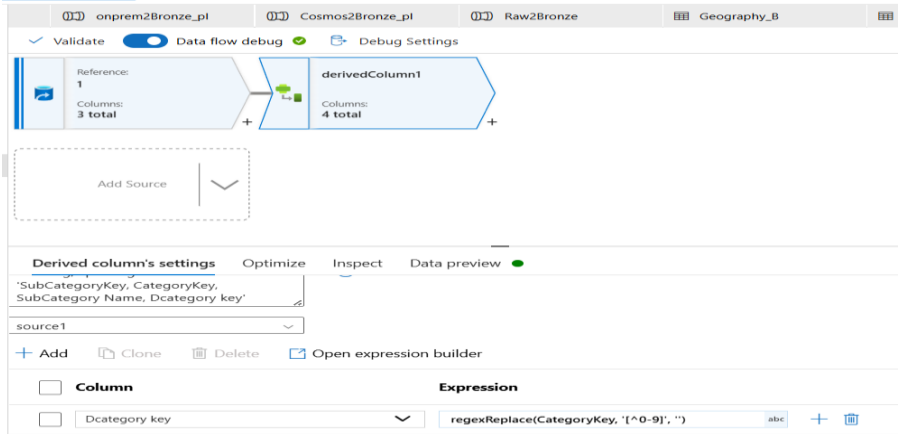
1.Data Flow for SubCategories.csv:-

1.1 Source:-

- **File:** SubCategories.csv (inside sales folder in ADLS container denis-medallian in bronze)
- Dataset: Delimited text, schema from file

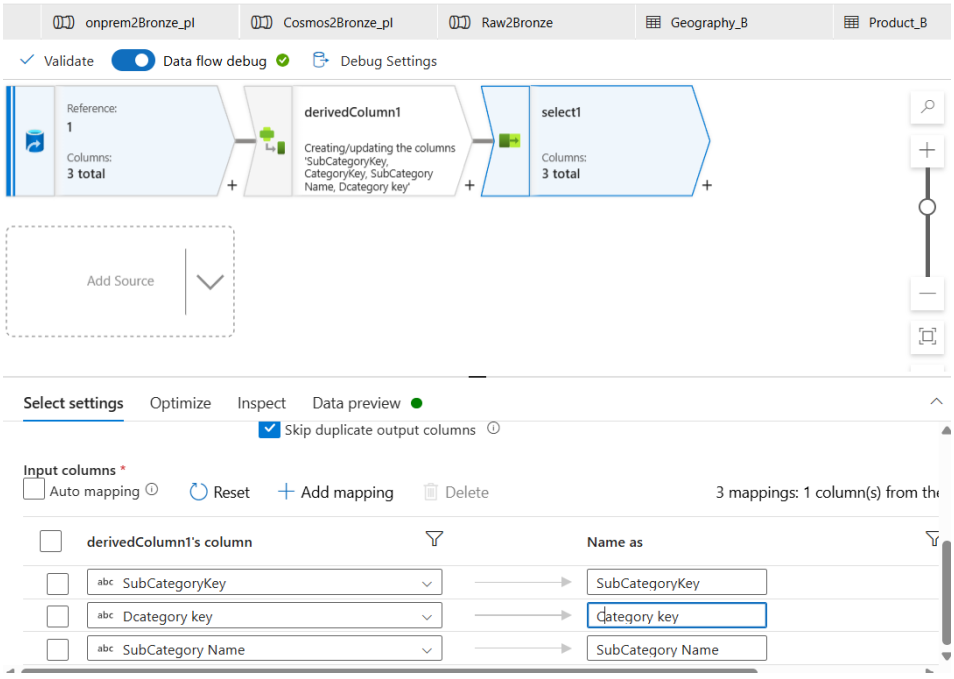
1.2 Derived Column (DerivedCategoriesKey):-

- Created a **new column**: DCategorieKey
- Expression: regexReplace(CategoryKey, '[^0-9]', '')



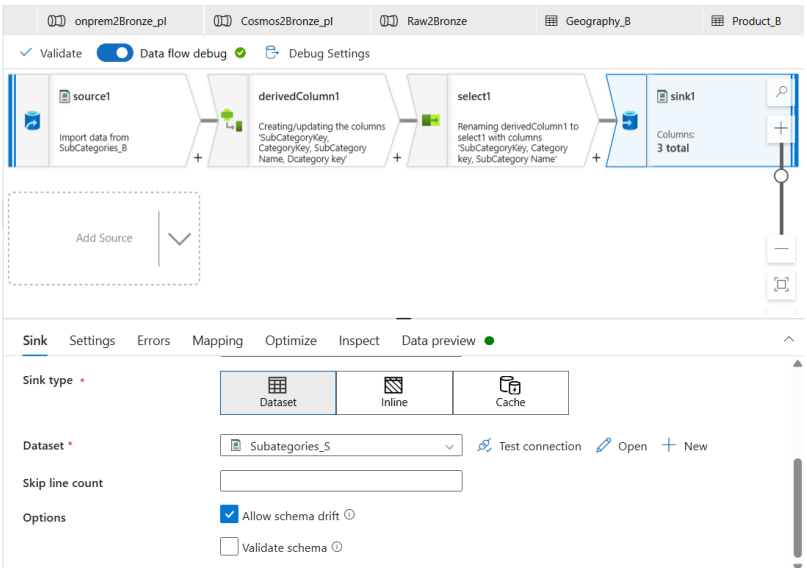
1.3 Select Transformation :-

- Removed the original CategoryKey column (since you have the derived one now)



1.4 Sink :-

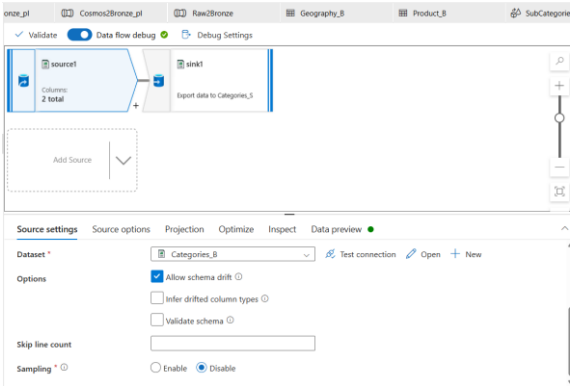
- Path: Possibly Bronze → Silver folder in ADLS
- Format: Parquet or CSV (depending on your target design)
- Note :-go to setting Select File name option in the select
- Pattern :- give name as(Subategories.csv) and select the Optimize option in that select Single partition
- Now Click **Publish All**



2.Steps for Categories_DF:-

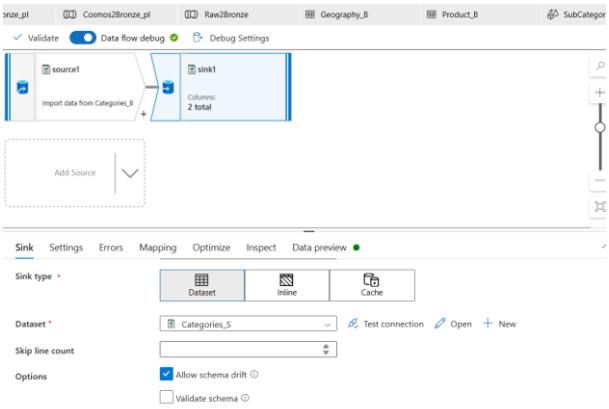
2.1Source:-

- File: Categories.csv from Bronze (in ADLS denis-medallian sales folder).
- Format: Delimited text
- First row as header ✅



2.2 Sink:-

- Silver Layer path: denis-medallian /silver/Categories



- Now Click **Publish All**

3.Steps for Geography_df :-

3.1 Source:-

- File: Geography.json from Bronze (denis-medallian/sales folder).
- Format: JSON.
- Document form: If it's a standard array of objects, set "Document per line" to false. If each object is in a separate line, set it to true.
- Projection: Enable schema drift if structure may change**.**

3.2 Select Transformation :-

- Remove these columns:
  - .wikipedia
  - _id

3.3 Surrogate Key Transformation :-

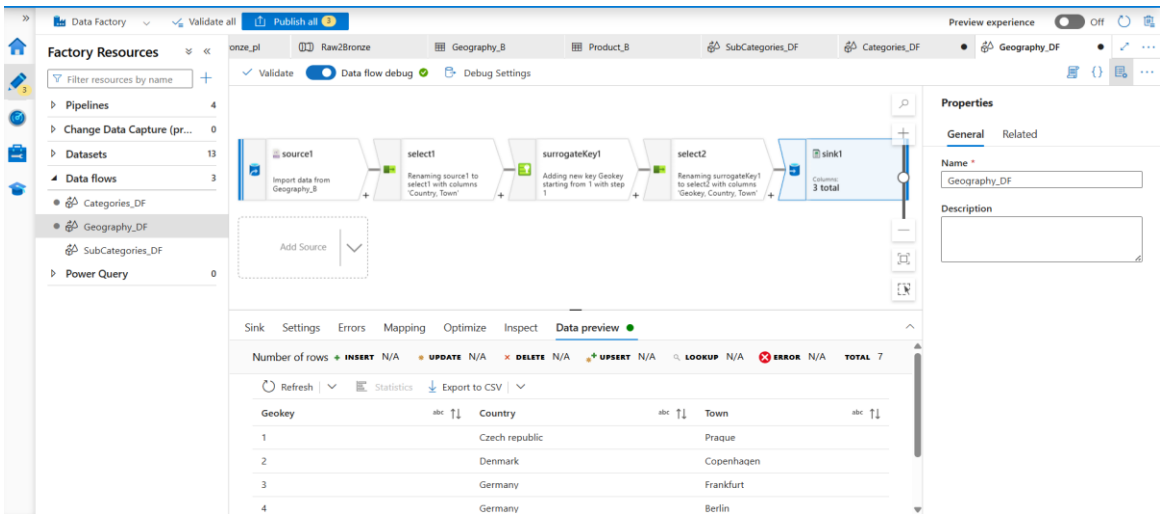
- Name: GeoKey
- This will generate unique keys for each row

3.4 Select Transformation (Reorder) :-

- Arrange the columns so GeoKey is first, followed by the rest of your business columns.

3.5 Sink:-

- Output Path: - denis-medallian/sales/silver/Geography/



4.Product_DF Pipeline Logic:-

4.1 Source:-

- File: Product.csv from Bronze folder in ADLS.
- Format: CSV.
- Enable schema drift.

4.2 Aggregate Transformation :-

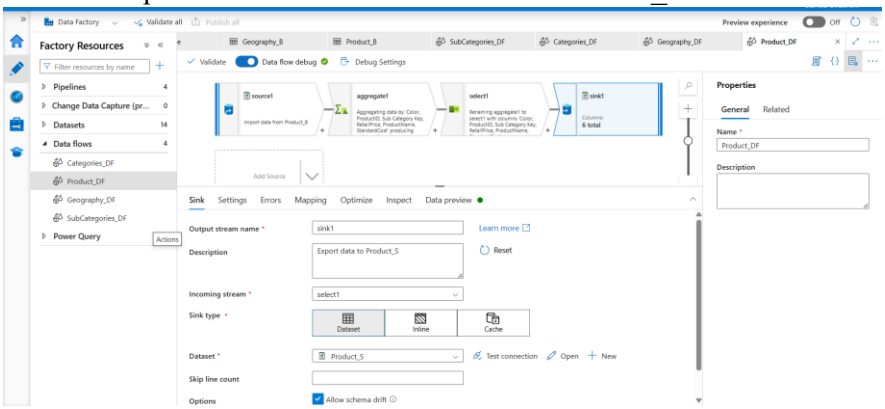
- **Group By** → all product columns (so that duplicates collapse).
- **Aggregates** →
    - Name: CountRow
    - Expression:count(ProductID)
    - This counts how many times each product appears.

4.3 Select Transformation:-

- Remove the CountRow column (since we only used it for duplicate checking).
- Keep only the grouped product columns.

4.4 Sink:-

Output Path:-denis-medallian/silver/Product_S/



5.SalesRep_DF Pipeline Logic :-

5.1Source

- File: SalesRep.xlsx from Bronze in ADLS.
- Format: Excel (point to correct sheet, enable header).
- Enable schema drift.

5.2 Derived Column Transformation:-

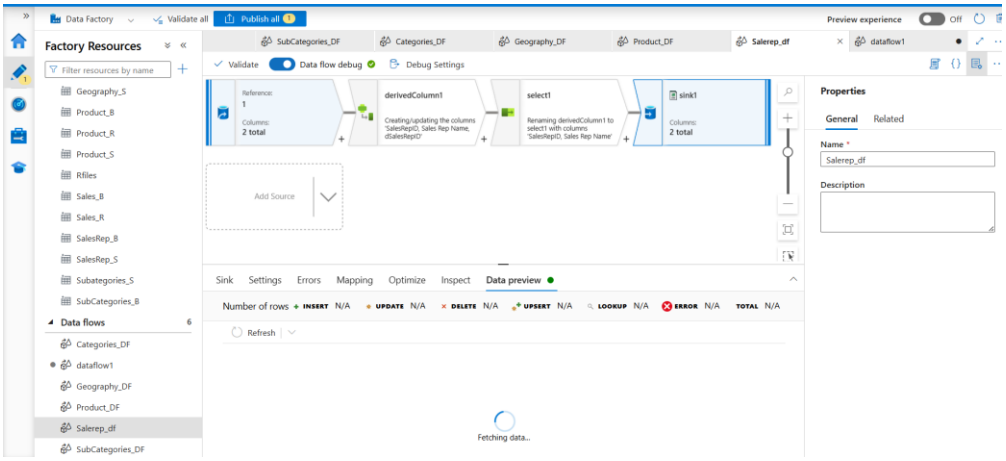
- Add column: dSalesRepID
- Expression: regexReplace(SalesRepID, '[^0-9]', '')

5.3 Select Transformation:-

- Remove original SalesRepID column.
- Rename dSalesRepID → SalesRepID

5.4 Sink:-

- Output Path: denis-medallian//silver/SalesRep_S/



6. Sales_DF Data Flow:-

6.1Source

- File: sales from Bronze in ADLS
- Format: CSV (schema drift enabled)

6.2Derived Column:-

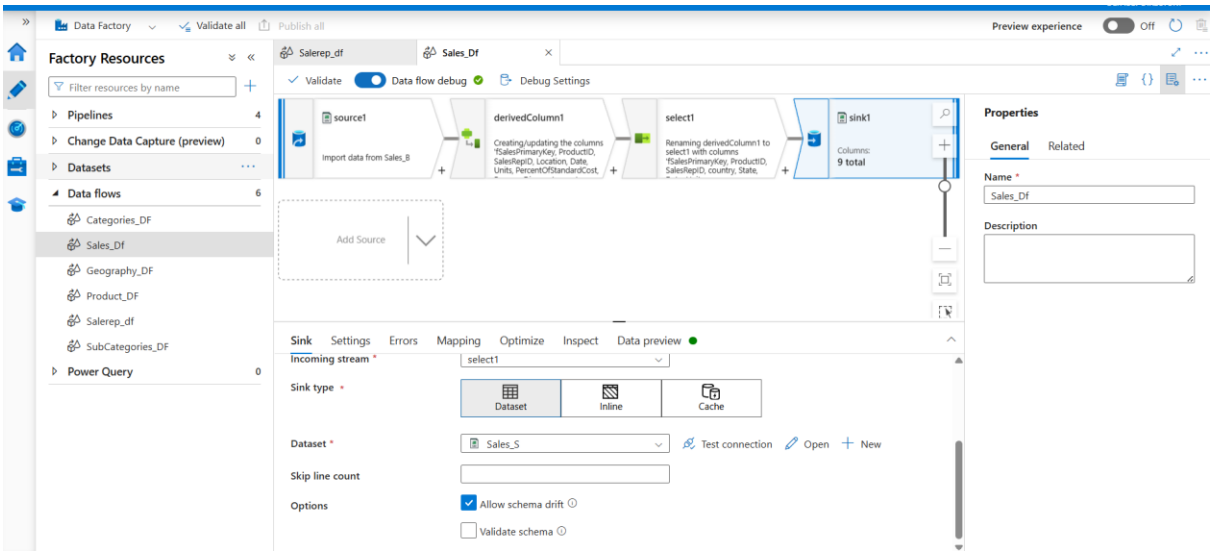
- Add column: country (1)
- Expression: split(Location, ';')[1]
- Add column: state (2)
- Expression: split(Location, ';')[2]
- Add column: Date (3)
- Expression: toDate(Date, 'dd.MM.yyyy')

6.3 Select Transformation:-

- Remove Location (original combined column)
- Keep the new country, state, and cleaned Date along with other necessary columns

6.4 Sink:-

Path:- denis-medallian/silver/Sales_S



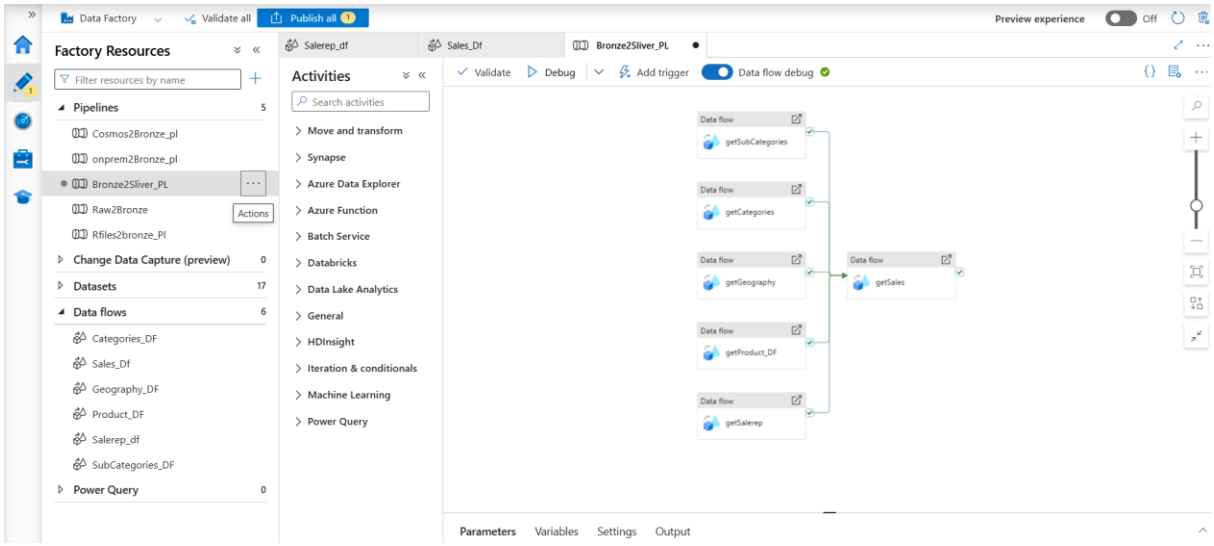
Master Pipeline Setup(Bronze2Sliver):-

- Name it something like Bronze2Sliver _PL

Add Data Flow Activities

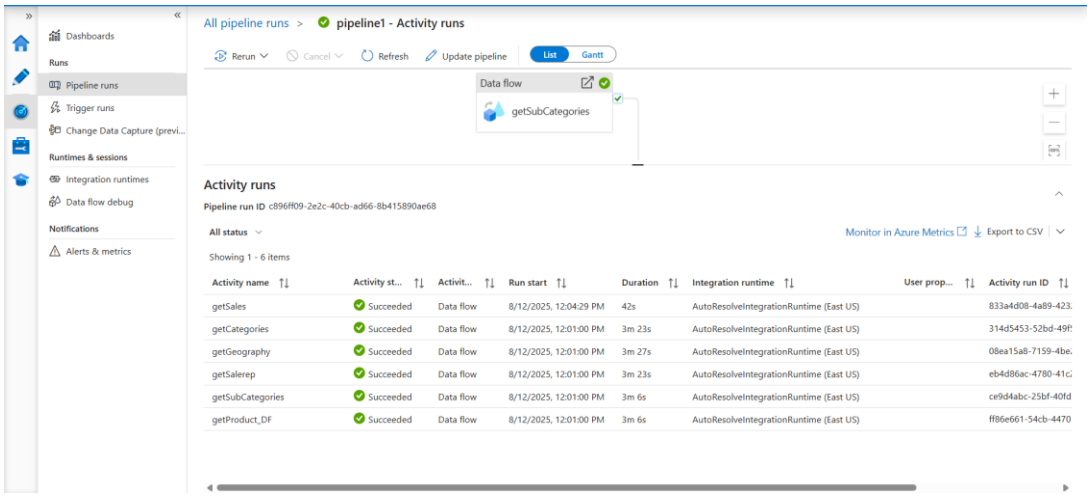
- **For each transformation you built earlier:**
  1. Drag a Data Flow activity into the pipeline canvas.
  2. Select the corresponding data flow (e.g., SubCategories_DF, Category_DF, Geography_DF, Product_DF, SalesRep_DF, Sales_DF)

Execution Order:-



Testing

- **Debug run the pipeline — check each Data Flow's output in Monitor tab.**



Nice 🎉

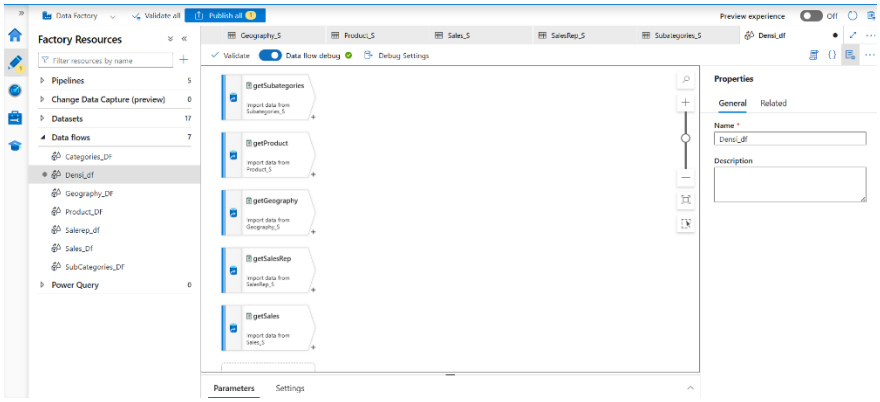
That means your bronze → silver ETL layer is done, and your silver tables are now clean, structured, and ready for joining.

Now the fun part begins — Silver → Gold:

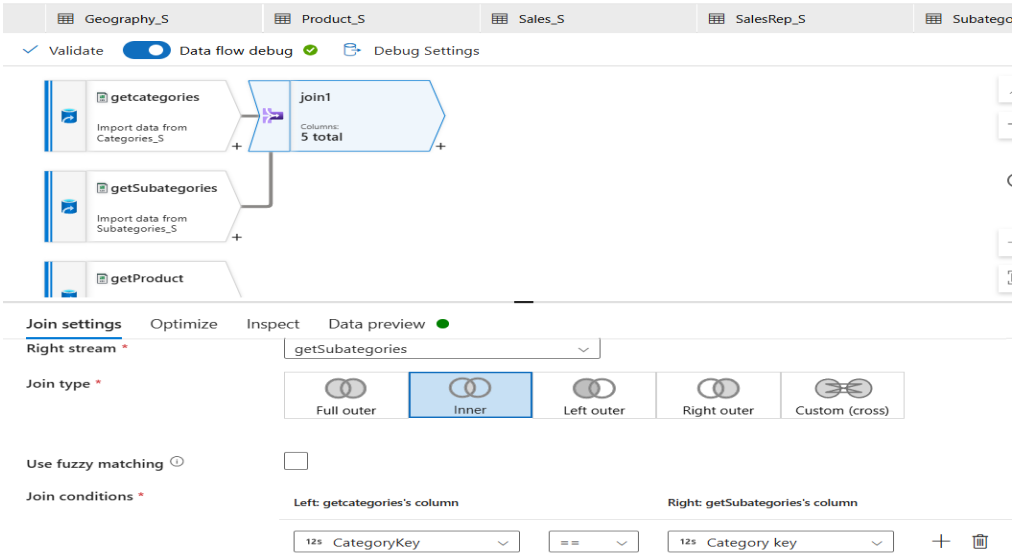
- Joining all your silver dimension tables with the Sales_DF fact staging table
- Creating the Fact_Sales table with surrogate keys and cleaned measures

Denis_DF data flow is doing this:

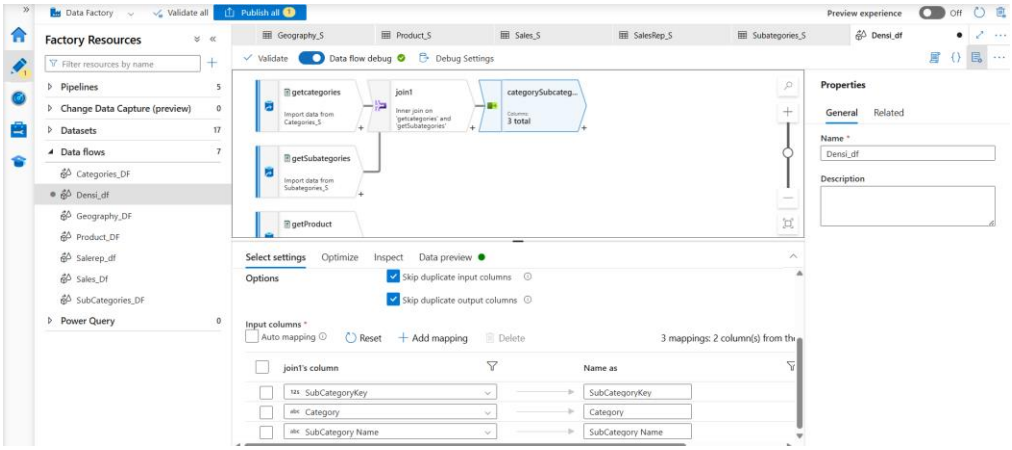
1. **Sources:**
    - Added 6 silver-layer source datasets:
        - Categories_S → getCategories_S
        - Subcategories_S → getSubcategories_S
        - Product_S → getProduct_S
        - Geography_S → getGeography_S
        - SalesRep_S → getSalesRep_S
        - Sales_S → getSales_S



2. **Join Step 1:**
    - Join: getCategories_S INNER JOIN getSubcategories_S
    - Condition: categorykey (from Categories) = categorykey (from Subcategories)



3. **Select Step 1:**
    - Removed categorykey columns from both joined streams to avoid duplication
    - Output stream name: **categorySubcategoriesData**

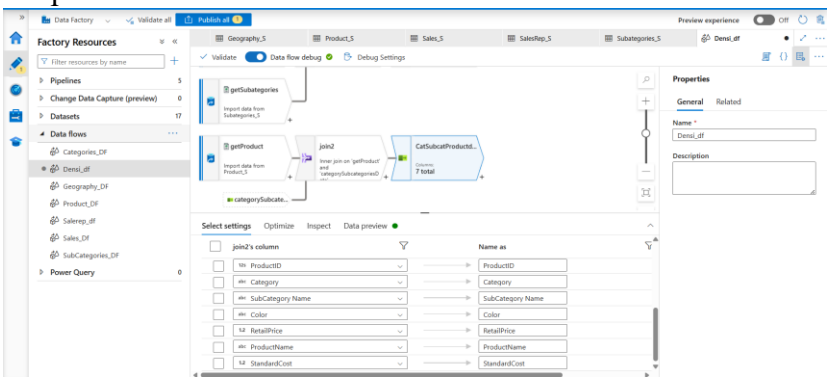


4. Join Step 2

- Join: categorySubcategoriesData INNER JOIN getProduct_S
- Condition: subcategorykey (from categorySubcategoriesData) = subcategorykey (from Product_S)

5. Select Step 2

- Removed the duplicate subcategorykey from both streams
- Output stream name: catSubcatProductData

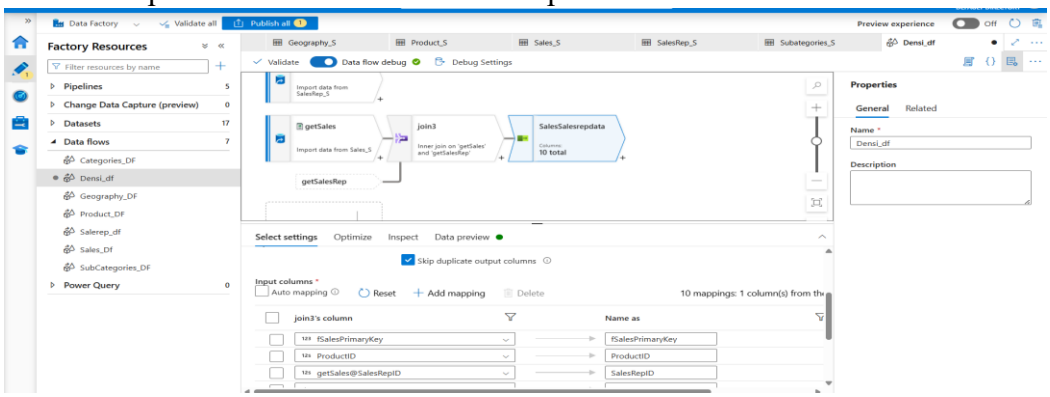


6. Join Step 3

- Join: getSales_S INNER JOIN getSalesRep_S
- Condition: SalesRepID (from Sales_S) = SalesRepID (from SalesRep_S)

7. Select Step 3

- Removed one duplicate SalesRepID column
- Output stream name: SalesSalesRepData

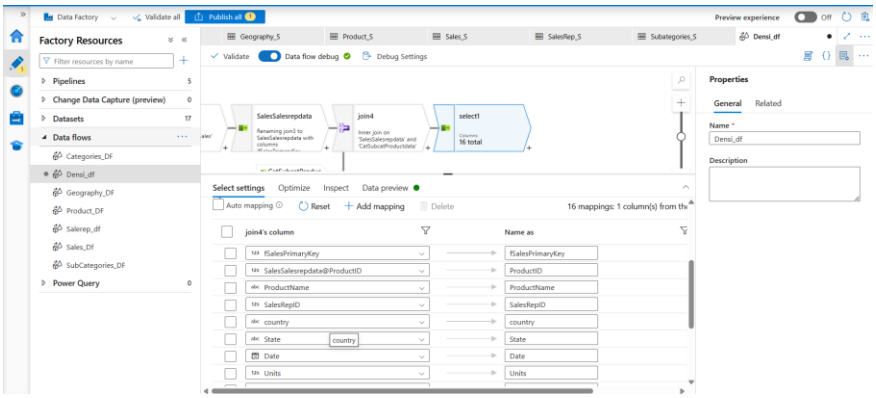


8. Join Step 4

- Join: SalesSalesRepData INNER JOIN catSubcatProductData
- Condition: productkey (from SalesSalesRepData) = productkey (from catSubcatProductData)

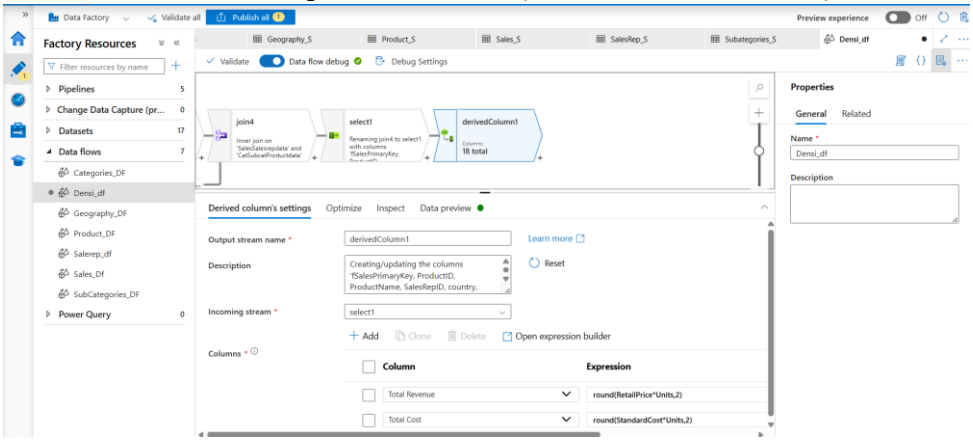
9. Select Step 4

- Removed one duplicate productkey column



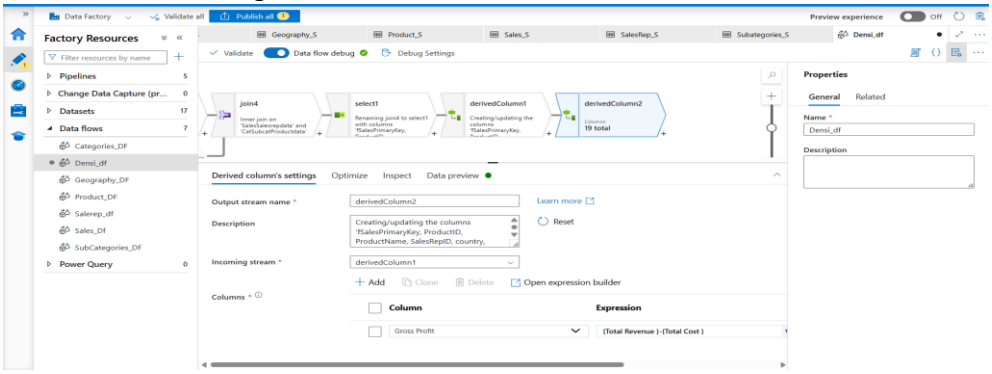
10. Derived Column Step

- New Columns:
  - TotalRevenue → Expression:Round(RetailPrice * Units,2)
  - TotalCost → Expression: Round(StandardCost * Units,2)



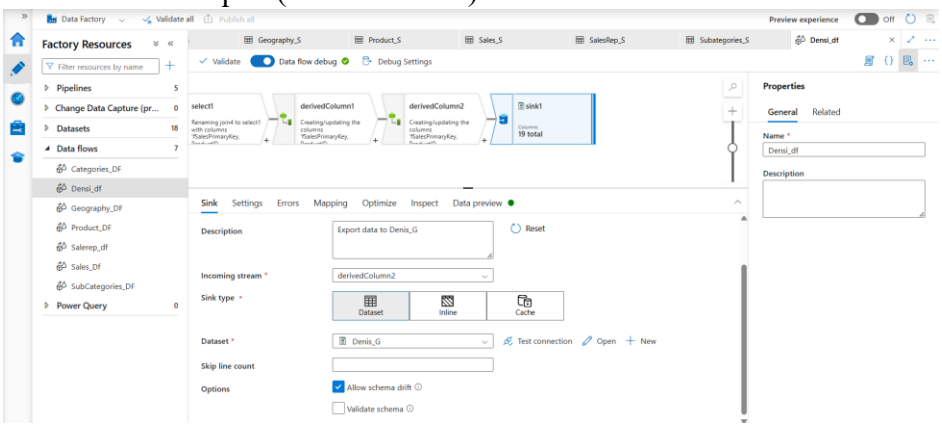
11.Derived Column (Profit Calculation)

- Input stream: from previous step (with TotalRevenue and TotalCost columns).
- New Column:
  - Profit → Expression: TotalRevenue - TotalCost



12. Sink (Gold Layer)

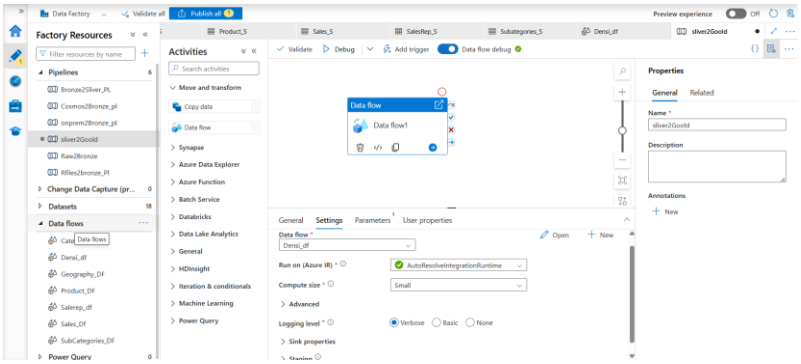
- Sink Type: ADLS Gen2
- Folder Path: denis-medallian/gold/Denis_G
- File Format: Parquet (recommended) or CSV



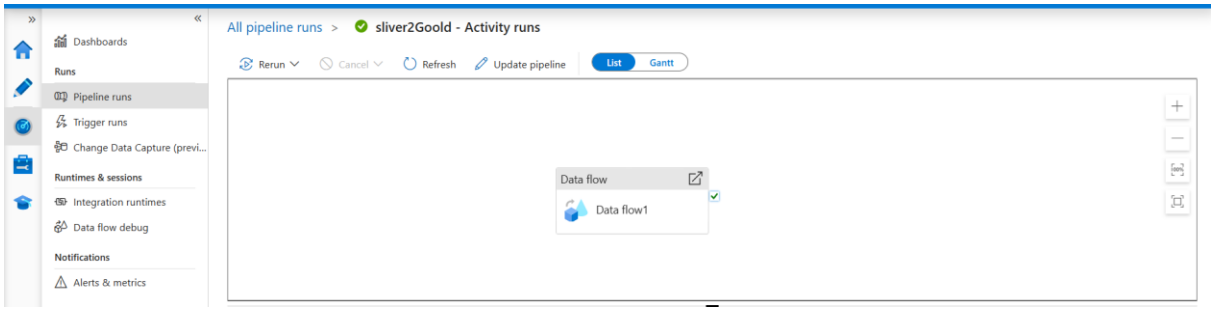
Now publish it

Silver2Gold pipeline:-

1.Add Data Flow activity → Select your Denis_DF data flow.



1. Set parameters (if you made any dynamic paths or filters in the Data Flow).
2. Publish All so it's saved in the Data Factory.

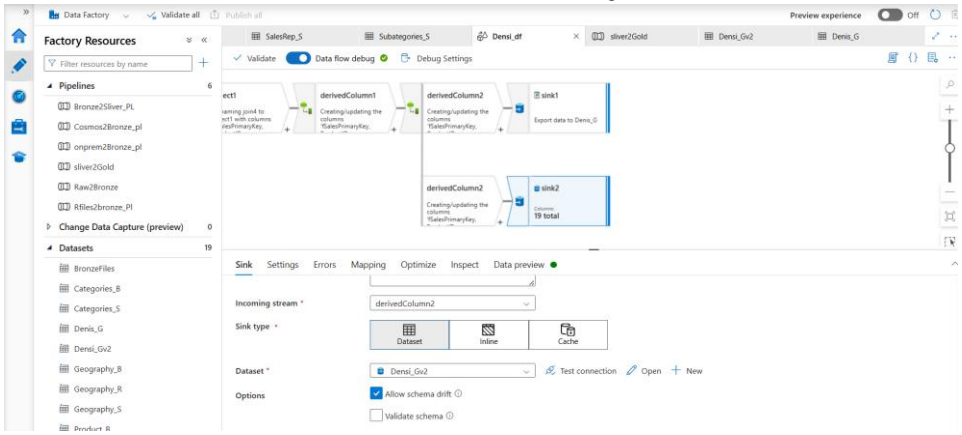


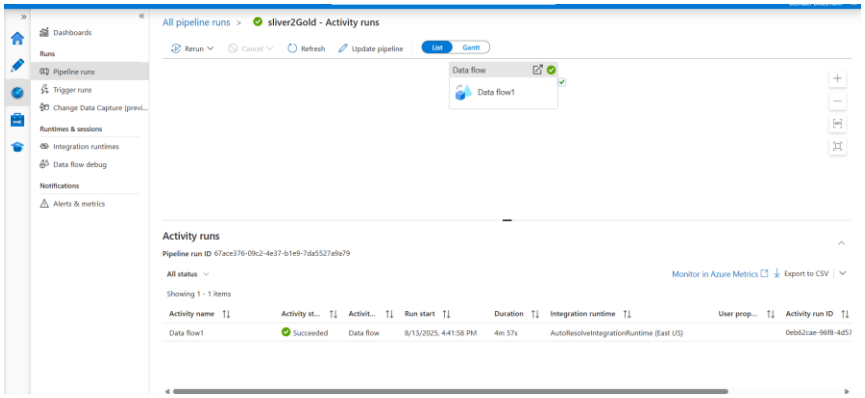
full Medallion architecture in ADF:

Raw → Bronze → Silver → Gold ✅

Add Sink in Data Flow

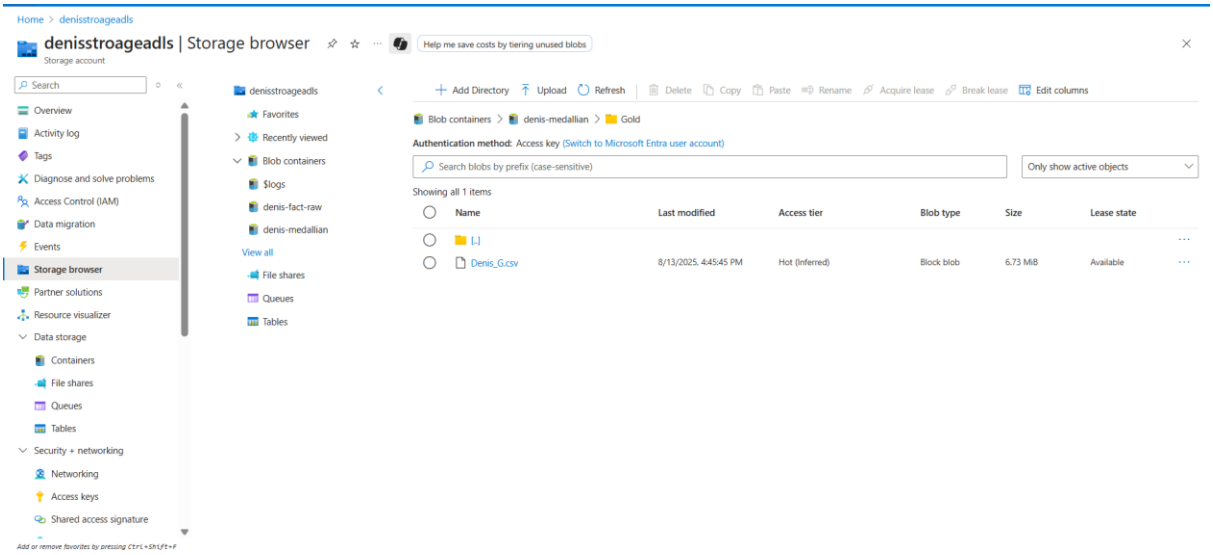
- Open your Denis_DF Data Flow
- After your last Derived Column (where you calculate profit), add a Sink transformation.
- Sink type → Azure SQL Database
- Linked service → Denis_G
- Table name → Create a new table in SQL





✅ Perfect — so after running your Silver2Gold pipeline, you've verified:

- Gold layer in ADLS → Denis-medallian/gold/Denis_G contains the final Fact_Sales file.



- Azure SQL Database → Table (e.g., Fact_Sales_Gold) contains the same transformed data.