ALLGEIER

Creating a Modern Data Lakehouse

with Azure Synapse Analytics

Contents

1 Workshop Preparations

The following prerequisites and setup must be done for successful completion of the course.

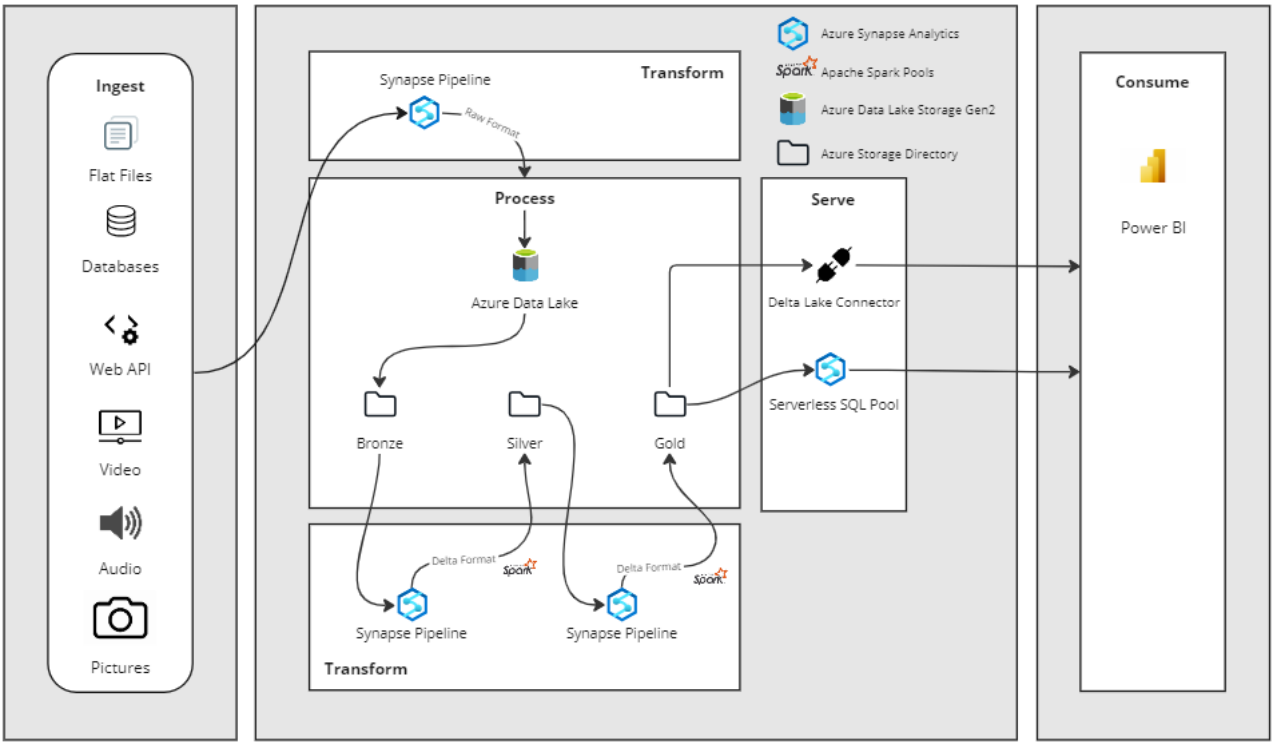
- You must be connected to the internet.
- You will need to have Microsoft Edge, or any other reliable browser installed.
- You must have started a free Azure account.
    - In this workshop we will provide you with an Azure account.
- You may have Power BI Desktop installed **(OPTIONAL)**.

2 Architecture

The Data Lakehouse is a data management architecture that combines the key features of data lakes and data warehouses to overcome their own drawbacks and offers low storage costs, ACID transactions, broad data access, data structures and governance features. These features enable business intelligence (BI) and machine learning (ML) on all data.

The architecture can be created using several cloud service providers such as: **Microsoft Azure, Databricks, Amazon Webservices, Google Cloud Platform** and **Snowflake,** to name a few.

In this workshop, we will be using **Microsoft Azure**. The architecture setup will follow the workflow of the diagram below:



2.1 Components

The architecture will be composed of the following Azure resources:

- Azure Synapse Analytics
  - Apache Spark Pool (Apache Spark Version 3.3.1, Delta Lake Version 2.2.0)
- Azure Storage Account (Azure Data Lake Gen2)

Additionally, a BI tool will be used to import and visualize the data:

- Power BI **(OPTIONAL)**

3 Provisioning Resources

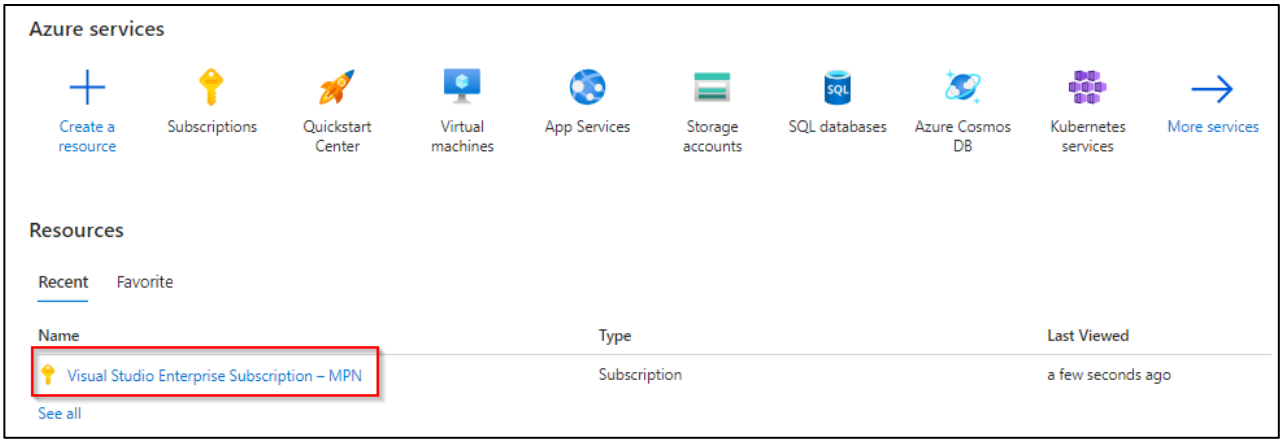
We will be creating the above Azure components using an Azure Resource Management (ARM) template. Additionally, the data being used comes from the AdventureWorks sample datasets made available by Microsoft.

All files, including the ARM template, can be found in the Git repository located here.

3.1 Azure Account & Subscription

The Azure Account you are using today has already been set up for you and the credentials can be found on the handout you have received. You can log into the Azure Portal using this account by clicking here.

The Azure Subscription you will be using today has already been provisioned for you and is named **Visual Studio Enterprise Subscription – MPN.**



After this workshop, you may create your own free Azure Account with CHF 200 worth of credits by following this link.

NOTE: You will be brought to your azure environment in chapter Create Azure Resource Group and Resources when provisioning the resources. Use the username and password supplied to you on the handout.

3.1.1 Resource Provider

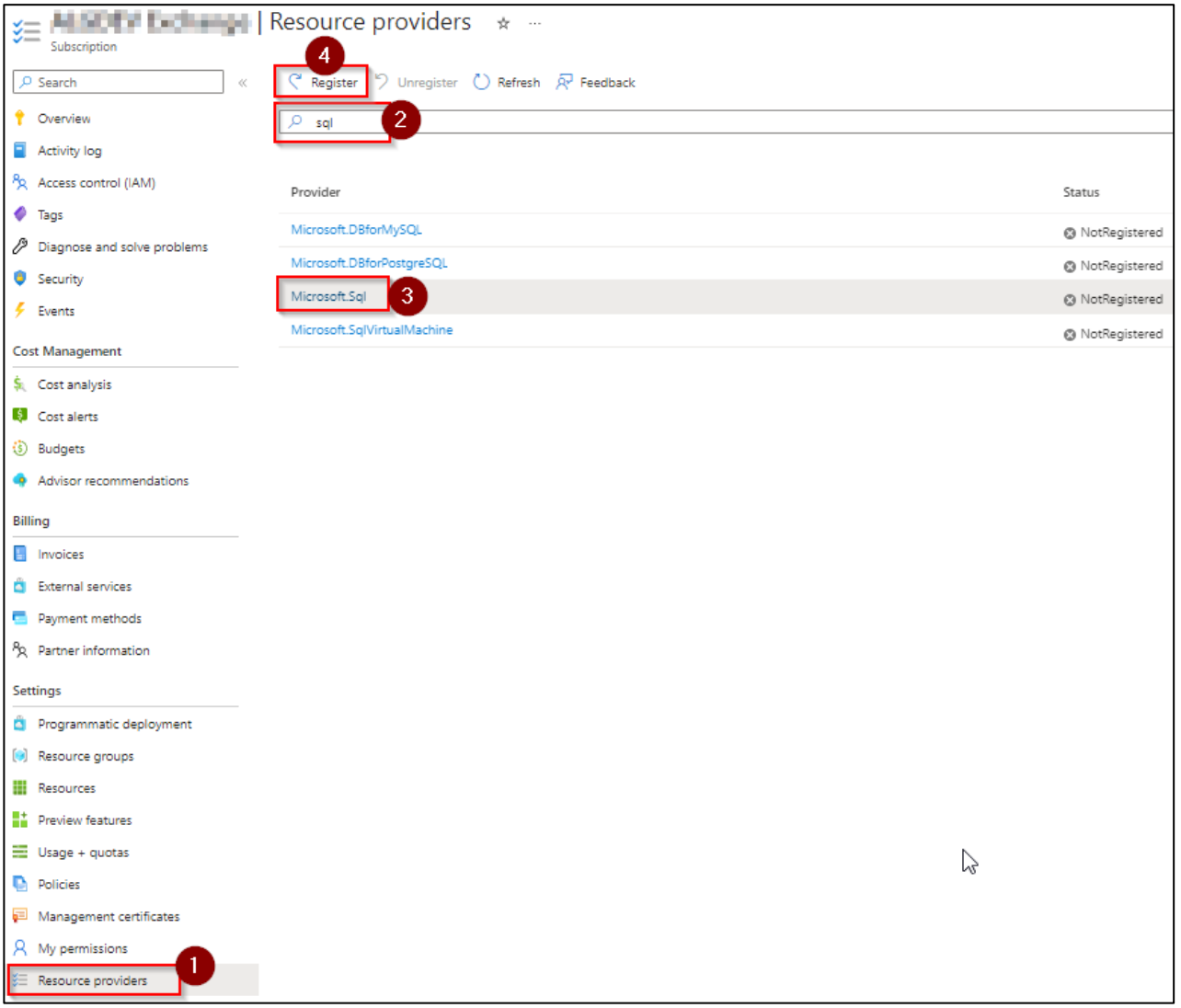
Once the subscription is created, we need to register the resource provider **Microsoft.Synapse , Microsoft.Sql** and **Microsoft.Network** to the subscription**.**

NOTE: The resource provider might already be registered, if this is the case, you may skip these steps and continue with chapter Create Azure Resource Group and Resources

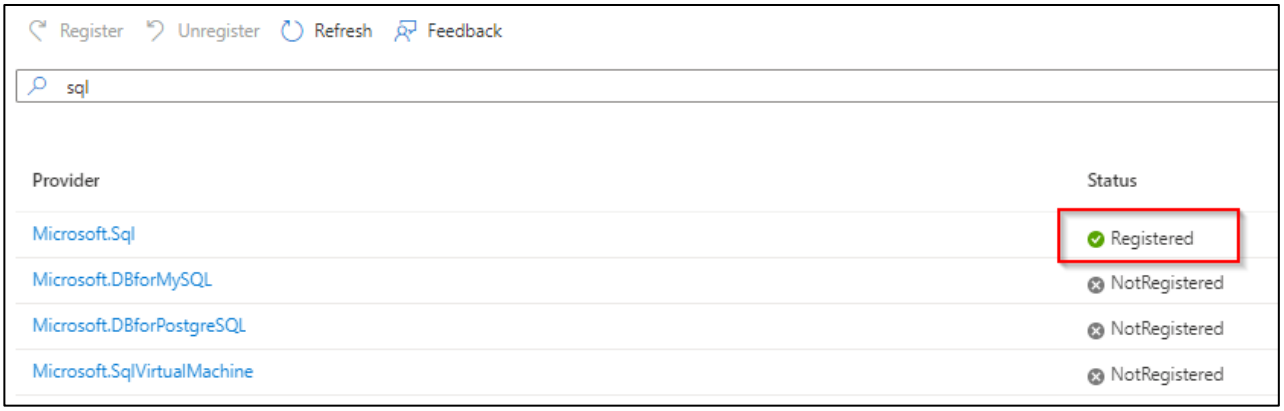
For additional information regarding what a resource provider is and the different types, see Azure resource providers and types.

The **Microsoft.Sql** resource provider is required to create an Azure Synapse Analytics workspace.

1. Go to your subscription and select **Resource providers**. Search for *Microsoft.Sql* in the search bar and highlight the row with **Microsoft.Sql** from the list below**.** Once highlighted, select **register.**



2. Once registered, the Status will change from **NotRegistered** to **Registered**.



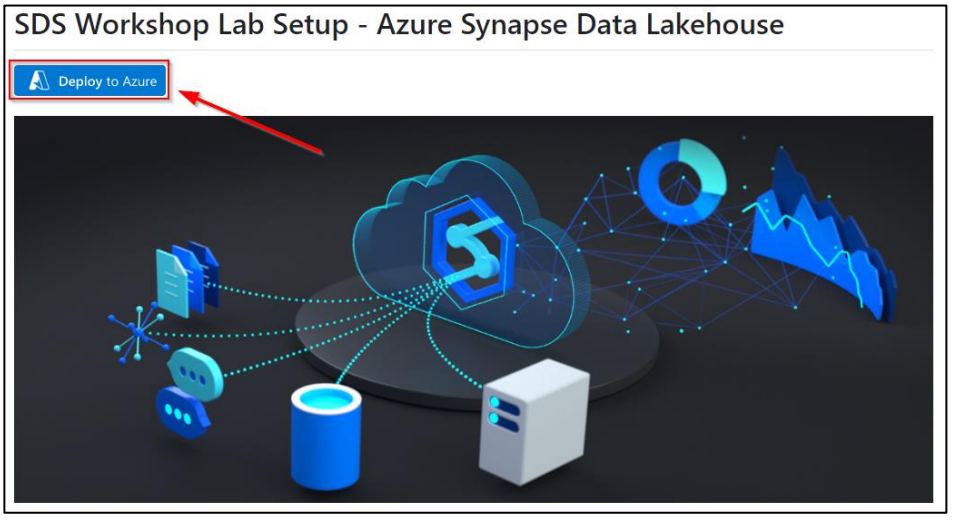
3. Repeat these same steps to register **Microsoft.Network & Microsoft.Synapse**

You are now ready to provision your Azure resources.

3.2 Create Azure Resource Group and Resources

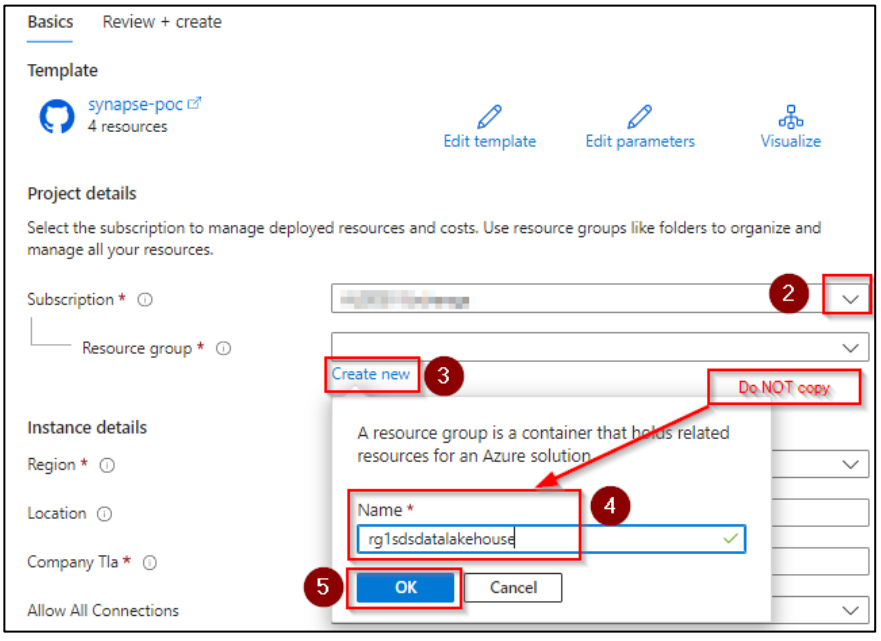
You will now create the Azure Resources required to set up our Data Lakehouse. We will use an Azure Resource Management (ARM) template to help provision the required Azure resources more efficiently.

1. Open this link and select **Deploy to Azure**. This will call the *azure.portal.com* website and start the ARM template provided by the Azure Synapse Proof-of-Concept.

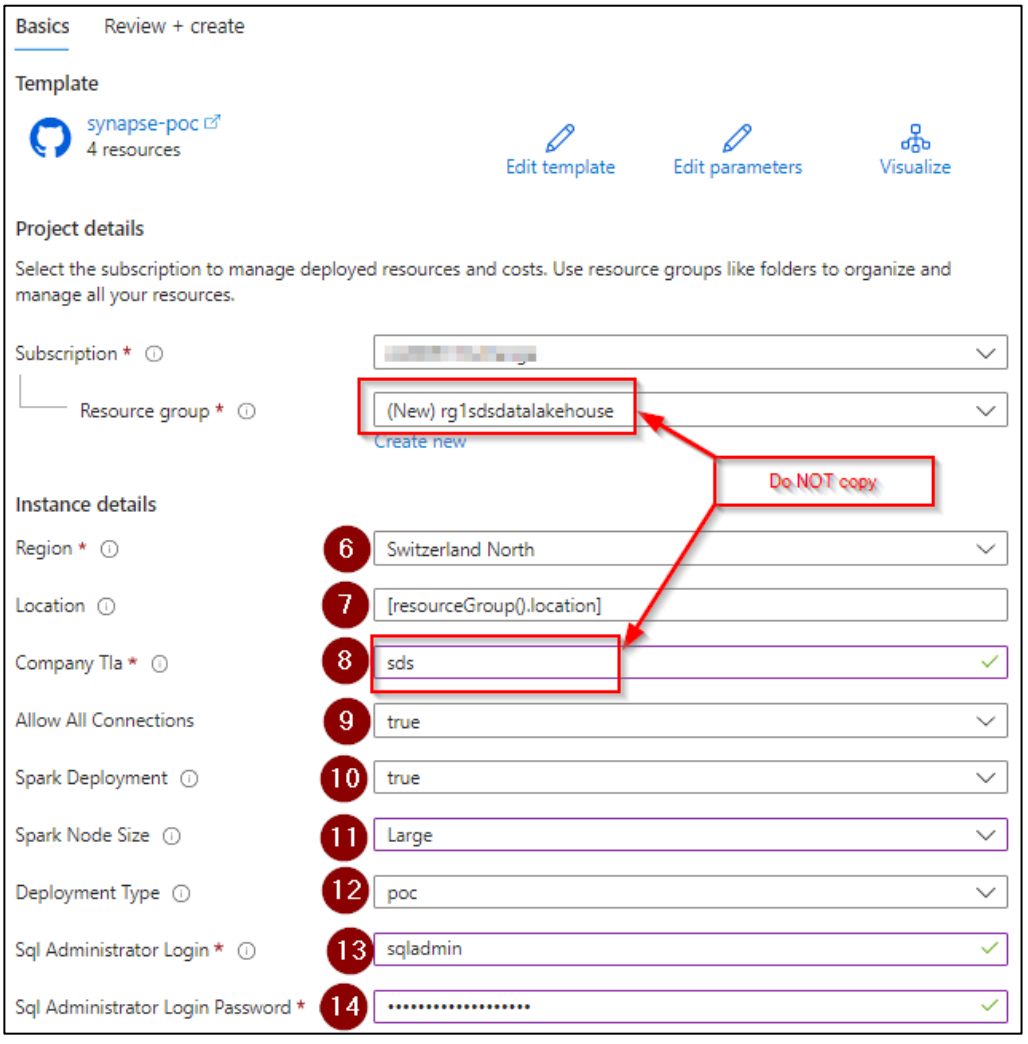


Once the template opens, there will be several entries that need to be manually input and / or changed.

2. Select the **Subscription** *Visual Studio Enterprise Subscription – MPN*.
3. Create a new **Resource Group** by selecting the **Create New** button by the Resource Group entry.
4. Add a new **Resource Group** name with the following make up:
    a. *rg1* **<sds**YOUR NUMBER ON THE HANDOUT**>** *datalakehouse* **(**If your number is 12, the full name becomes: rg1sds12datalakehouse**).**
5. Select **OK**

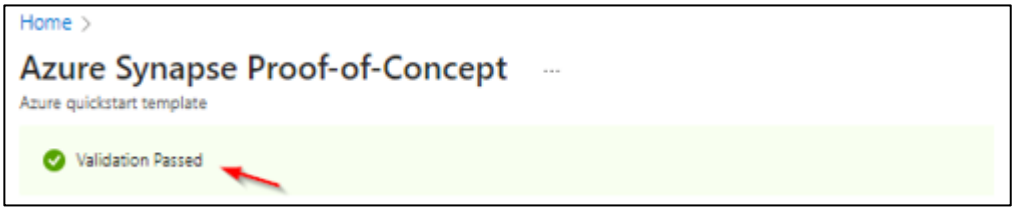


6. Set the **Region** to *the region assigned to you at the beginning of the workshop*. **(**THE REGION YOU SHOULD USE CAN BE FOUND ON THE HANDOUT GIVEN TO YOU**).**
7. Leave the **Location** as *[resourceGroup().location]*.
8. Set the **Company Tla** to **<sds**YOUR NUMBER ON THE HANDOUT**>**. **(**If your number is 12, the full name becomes: sds12**).**
9. Leave the **Allow All Connections** as *true*.
10. Leave the **Spark Deployment** as *true*.
11. Set the **Spark Node** Size as *Large*.
12. Leave / Set the **Deployment type** as *poc*.
13. Define the **Sql Administrator Login** in the template as *sqladmin*
14. Define the **Sql Administrator Login Password** in the template as *5Xb%3Ns+b{bH$7BC#$D*

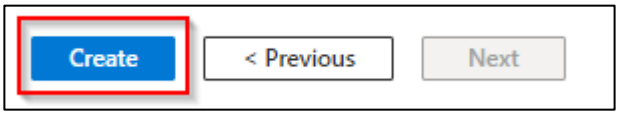


15. Select **Review + create** to start provisioning the Azure resources.

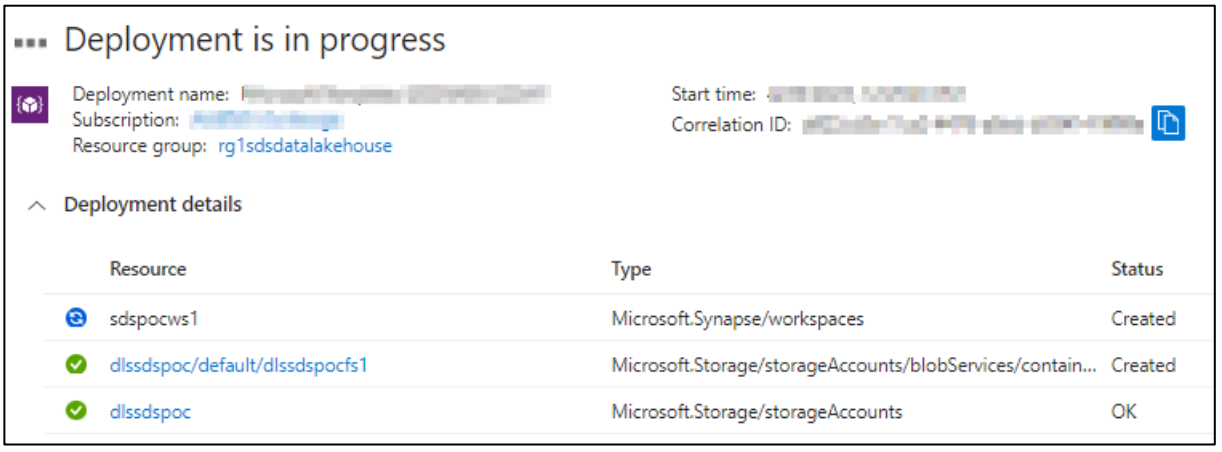
16. You will get a **Validation Passed** if all your template inputs are correct.



17. Select **Create** to start the provisioning process.

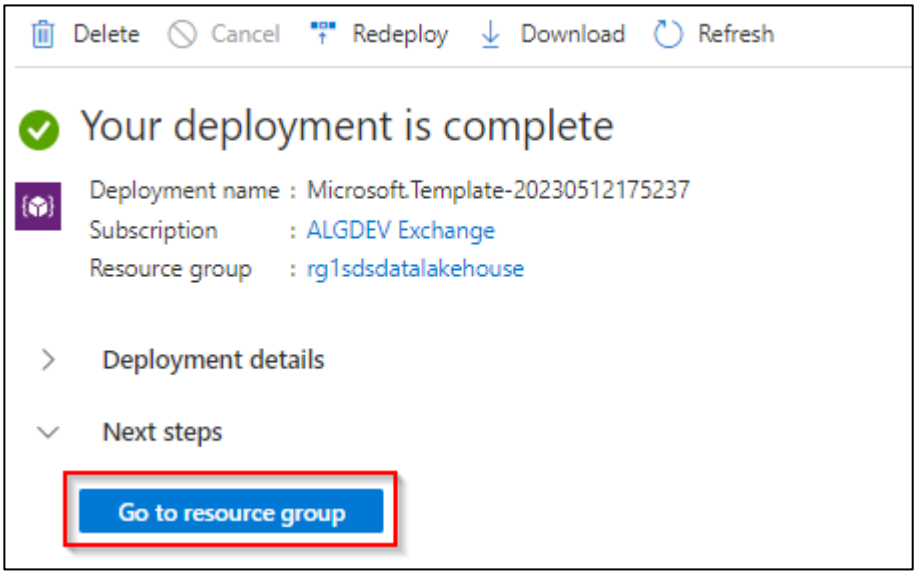


18. Once the provisioning of the resources has started, you will be brought to a page where **deployment is in progress** can be seen.

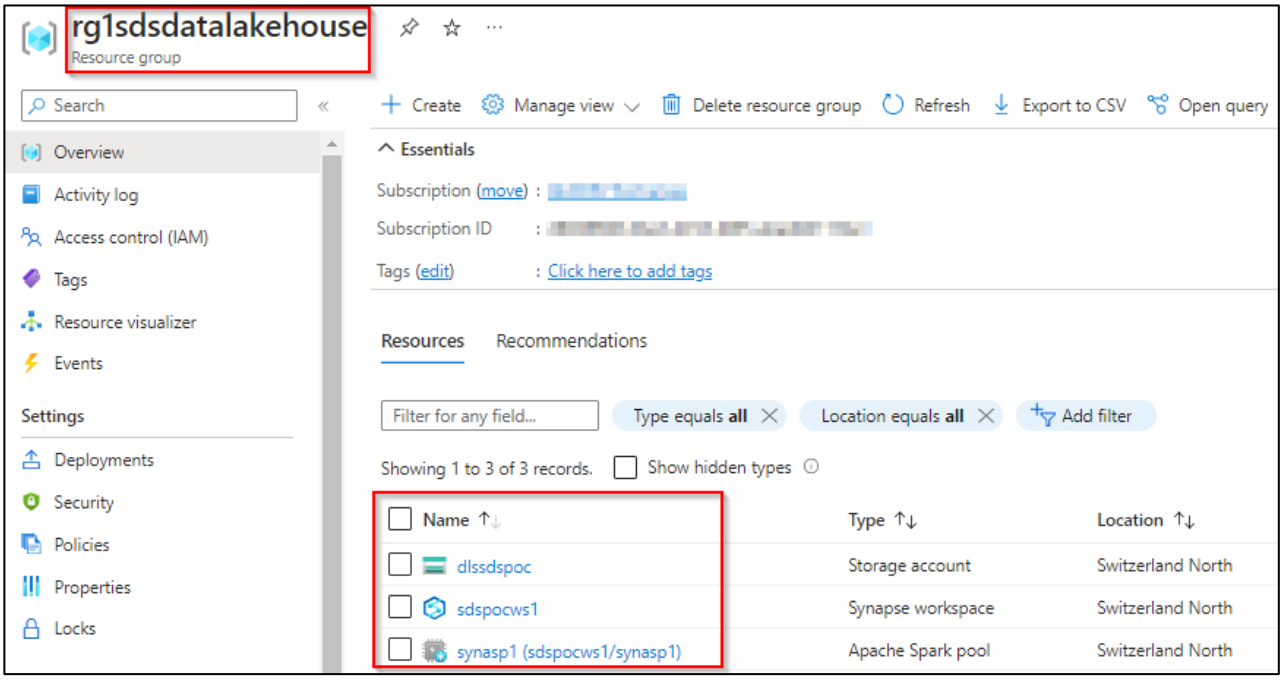


NOTE: It may take up to 10 minutes for the deployment to finalize.

19. Once the deployment is completed, select Go to **resource group** to go to the provisioned resources.



Once selected, you will be brought to the resource group where you will see the resources you created using the template.

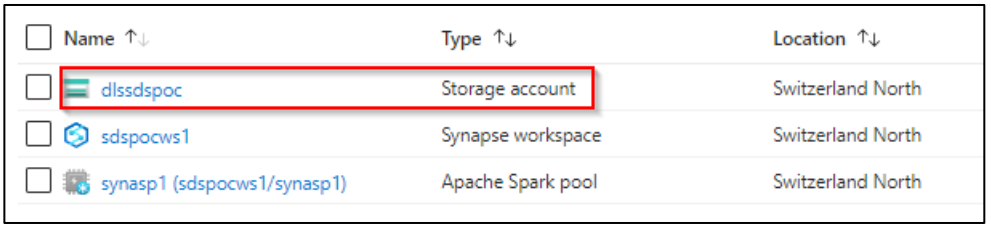


3.3 Post Deployment

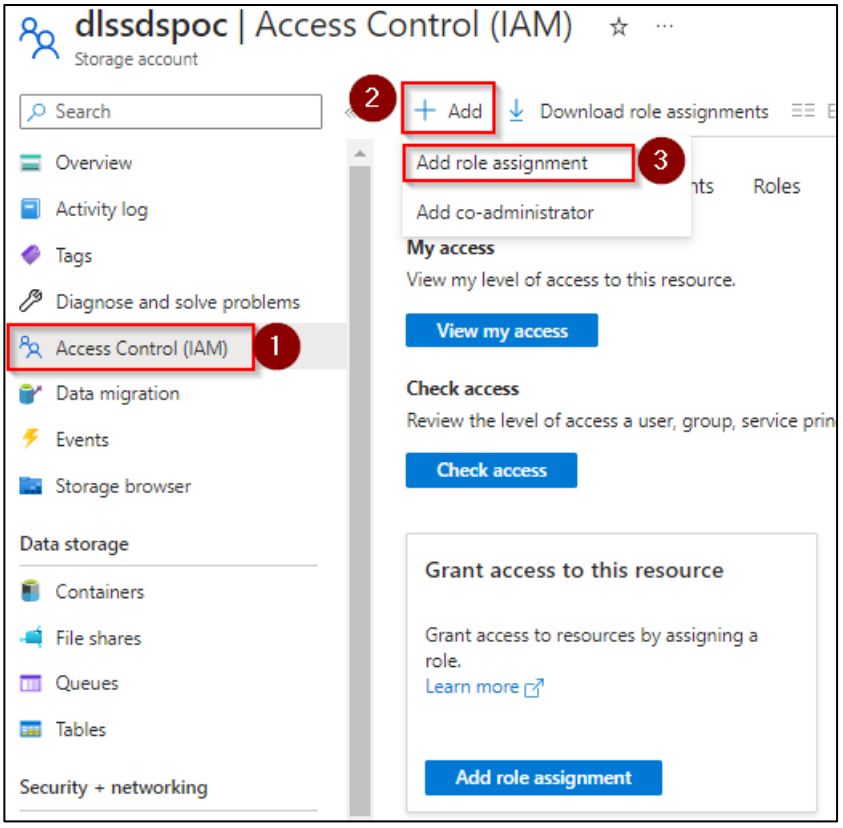
3.3.1 Role Assignment

To run queries using the Serverless SQL pool in Azure Synapse, you first need to grant yourself access to the storage account by assigning the **Storage Blob Data Contributor** role to your user (trouble shooting).

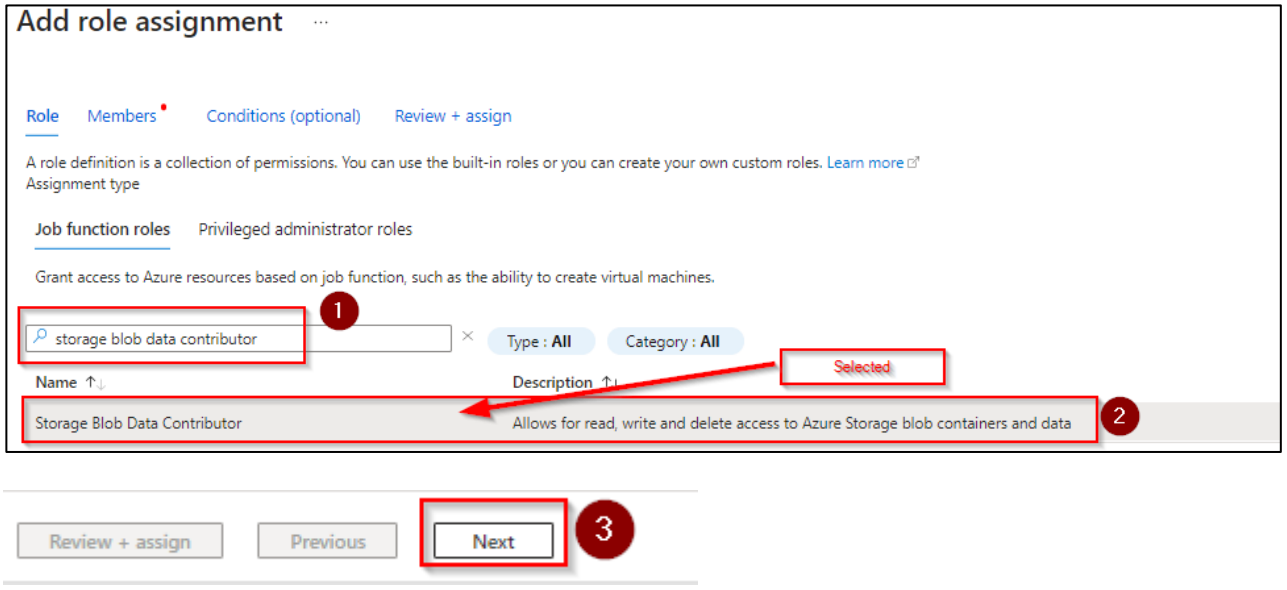
1. Within your newly created **Resource group, s**elect the resource type **Storage account**.



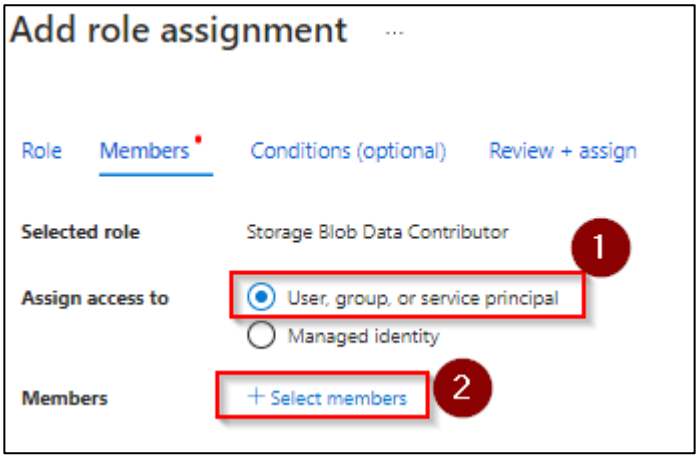
2. On the left panel select the **Access Control (IAM)** and add a new role assignment by selecting the **+ Add** button.



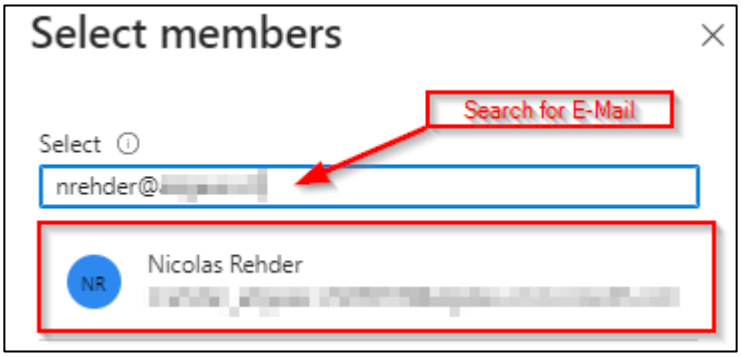
3. Search for **Storage Blob Data Contributor,** select / highlight the **Storage Blob Data Contributor** from the list and select **Next**.



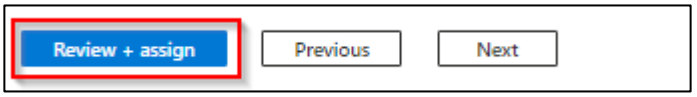
Toggle **User, group, or service principal** and then select the button **+ Select members**. A new panel will open to the far right.



4. Within this new panel, **search** for **<u>your</u>** user E-Mail address
5. Select / highlight the row with your E-mail address and click **Select**.



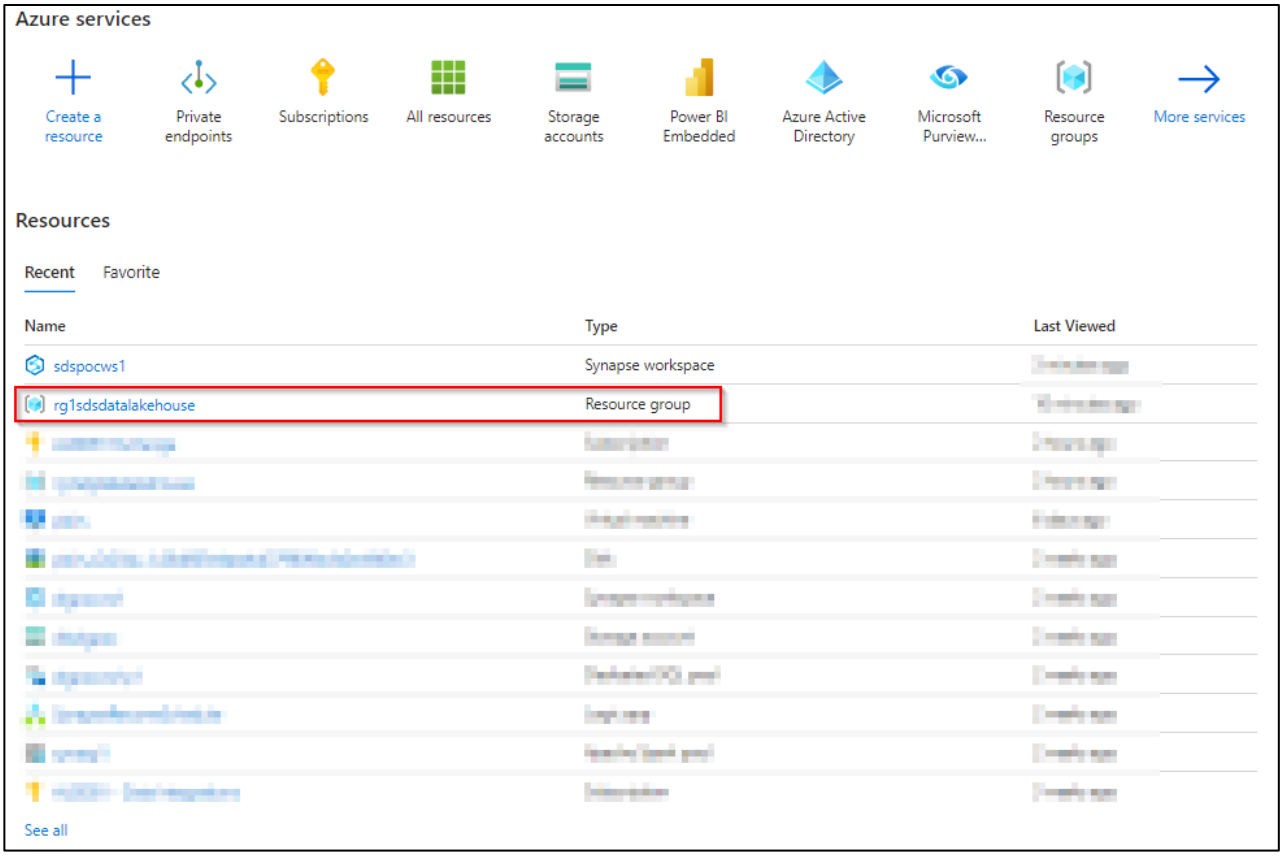
6. Once selected, you will **Review + assign** the role assignment.



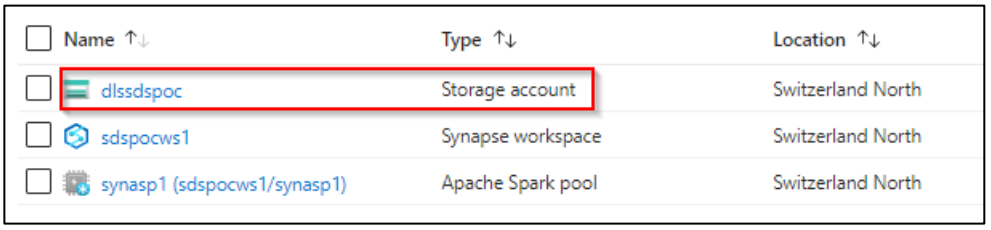
7. You will get the following banner once successful:

3.3.2 Create Directories and Load Data into the Storage Account

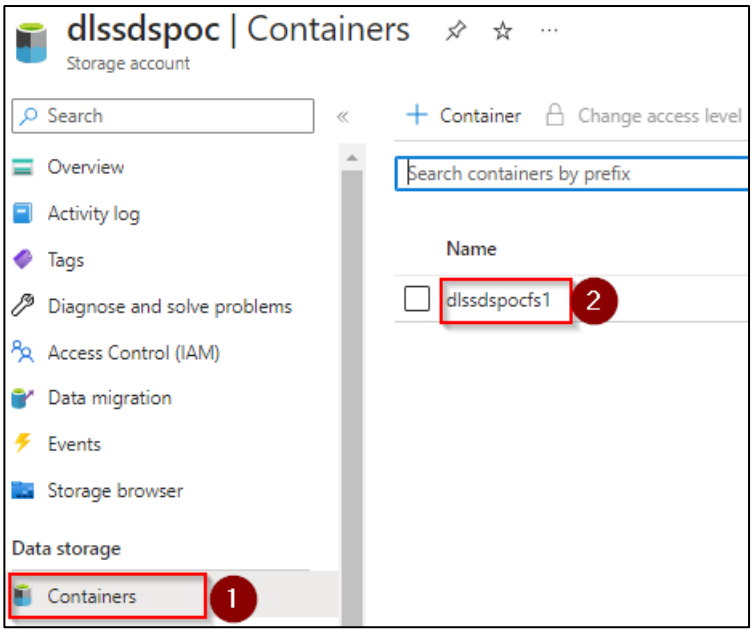
1. Go back to the Azure portal home screen and select the **Resource group** you provisioned.



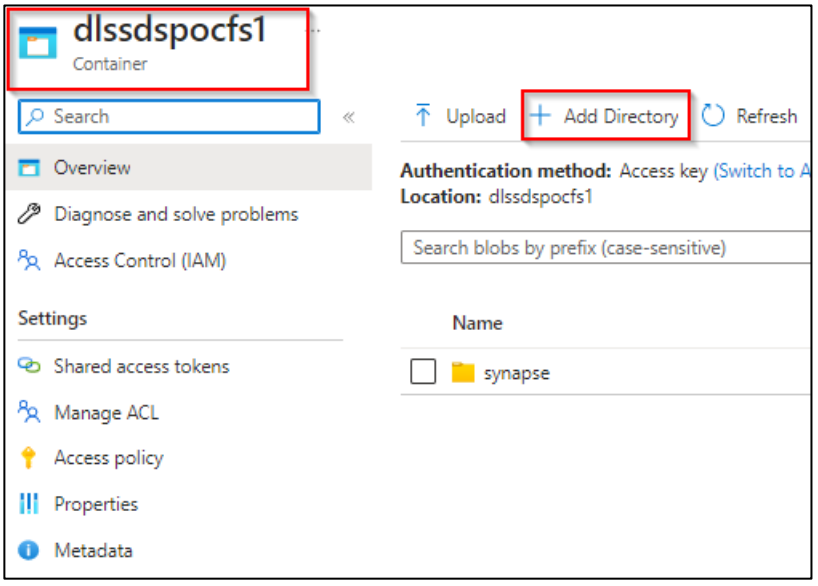
2. Once the resource group is open, select the **Storage account**.



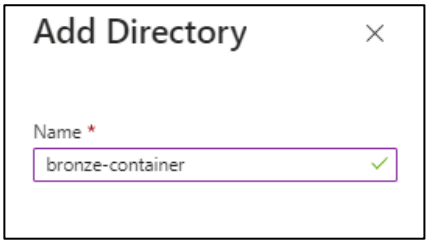
3. Select the **Container** tab on the left side and select the named container.



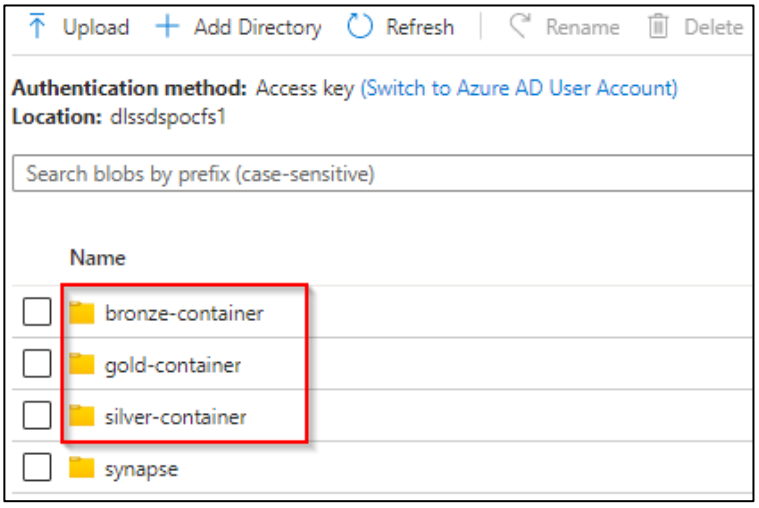
4. Within the named container, select **+ Add Directory** to create a new folder.



5. A new panel will open on the right side, where you can define the name of the new Directory. You will create 3 new directories, and label them **bronze-container, silver-container,** and **gold-container**.

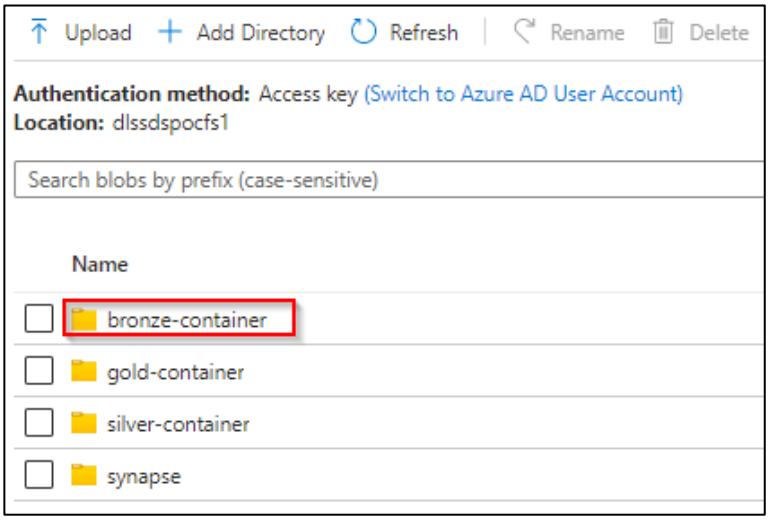


Your storage container structure should look like this:

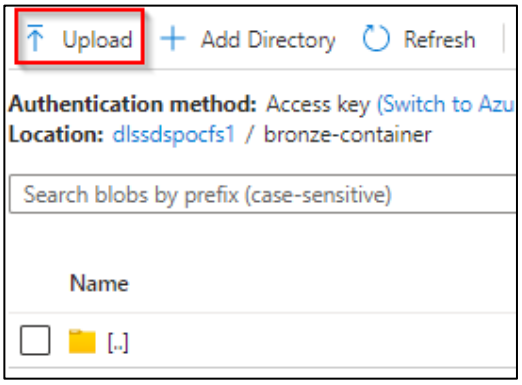


You will now add two CSV data files manually into the storage bronze-container directory.

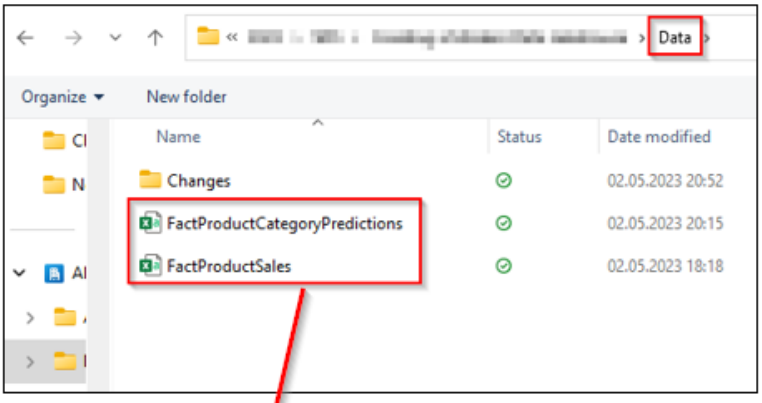
6. Within the Data Lake, first select the **bronze-container.**

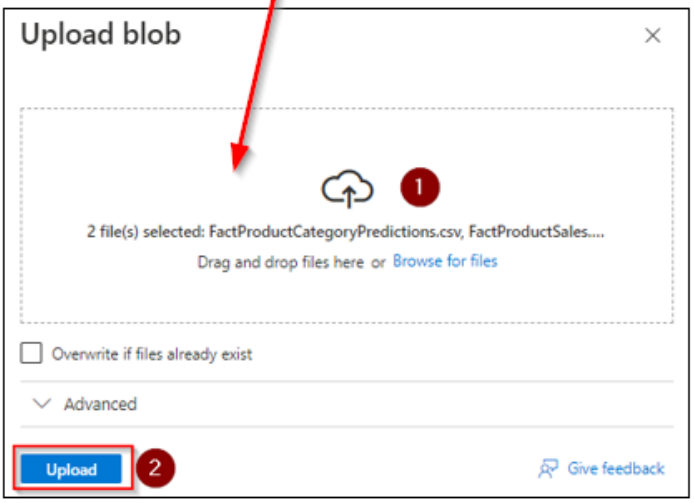


7. Within the bronze-container, select the **Upload button**.

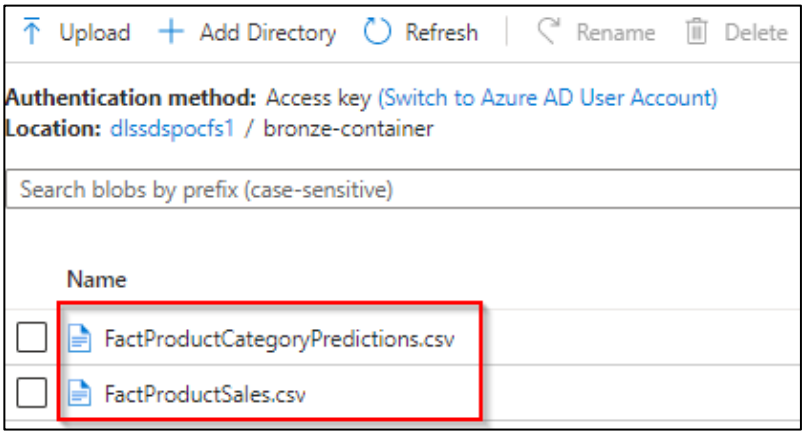


8. A new panel will open to the right. **Drag and Drop** the CSV files **FactProductCategoryPredictions.csv** and **FactProductSales.csv** and select **Upload**.



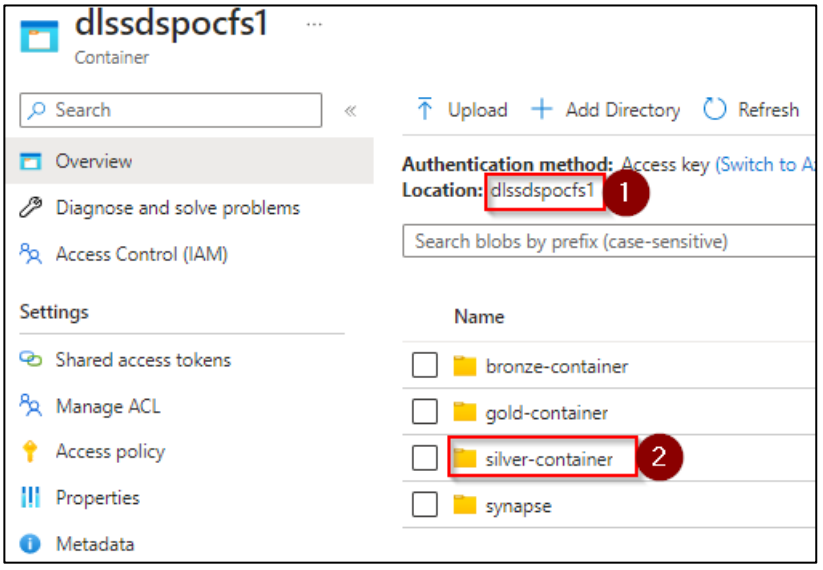


Your bonze-container directory should look like this:

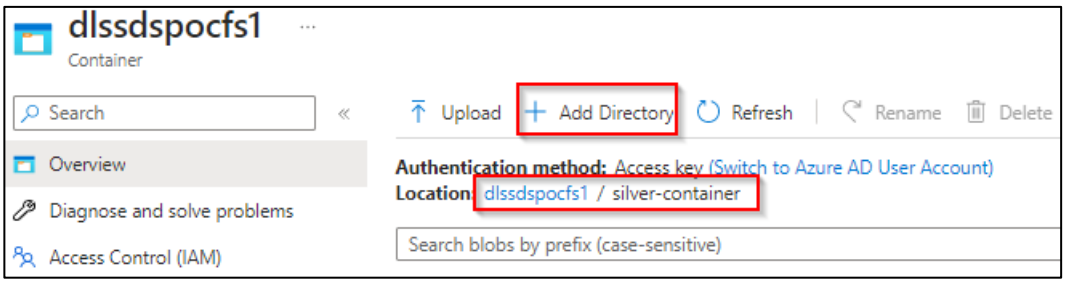


Go back to the root of the container.

9. Select the **silver-container** directory.

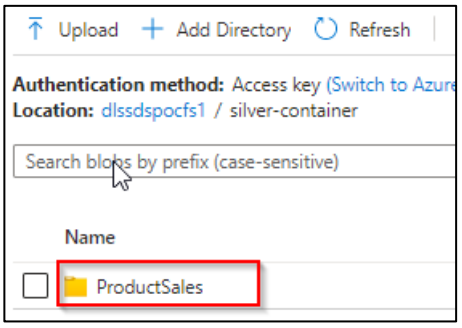


10. Within **the silver-container**, select **+ Add Directory** to create a new folder.



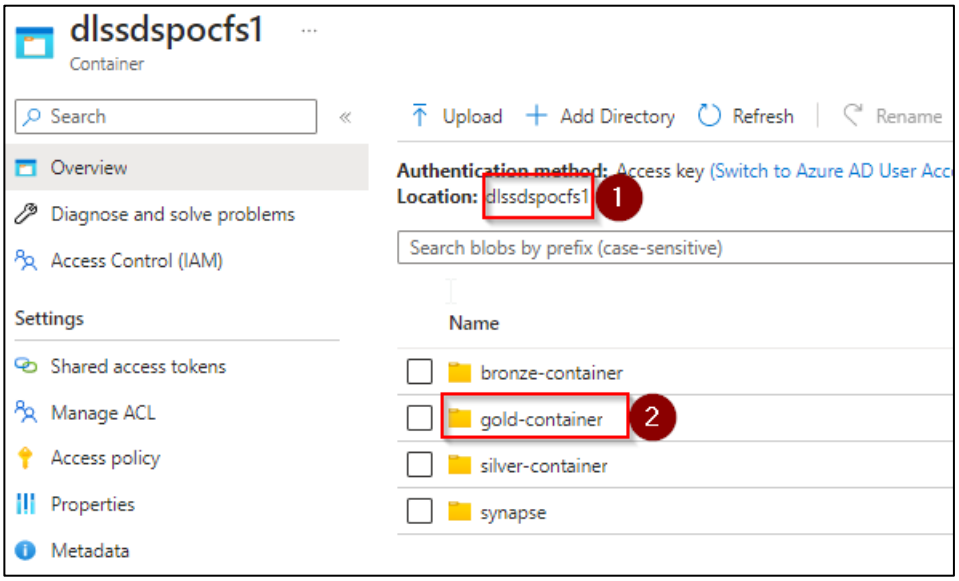
11. A new panel will open on the right side, where you can define the name of the new Directory. You will create 1 new directory called **ProductSales**.

Your storage container structure should look like this:

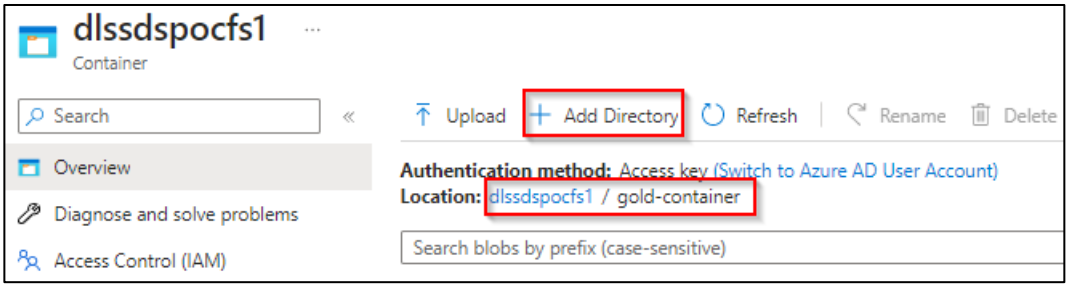


Go back to the root of the container.

12. Select the **gold-container** directory.

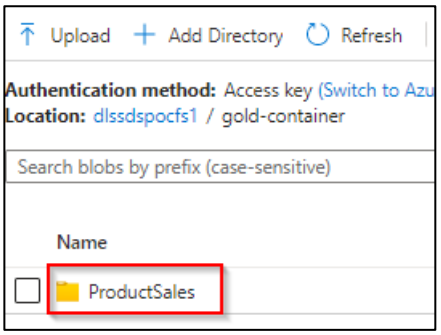


13. Within **the gold-container**, select **+ Add Directory** to create a new folder.



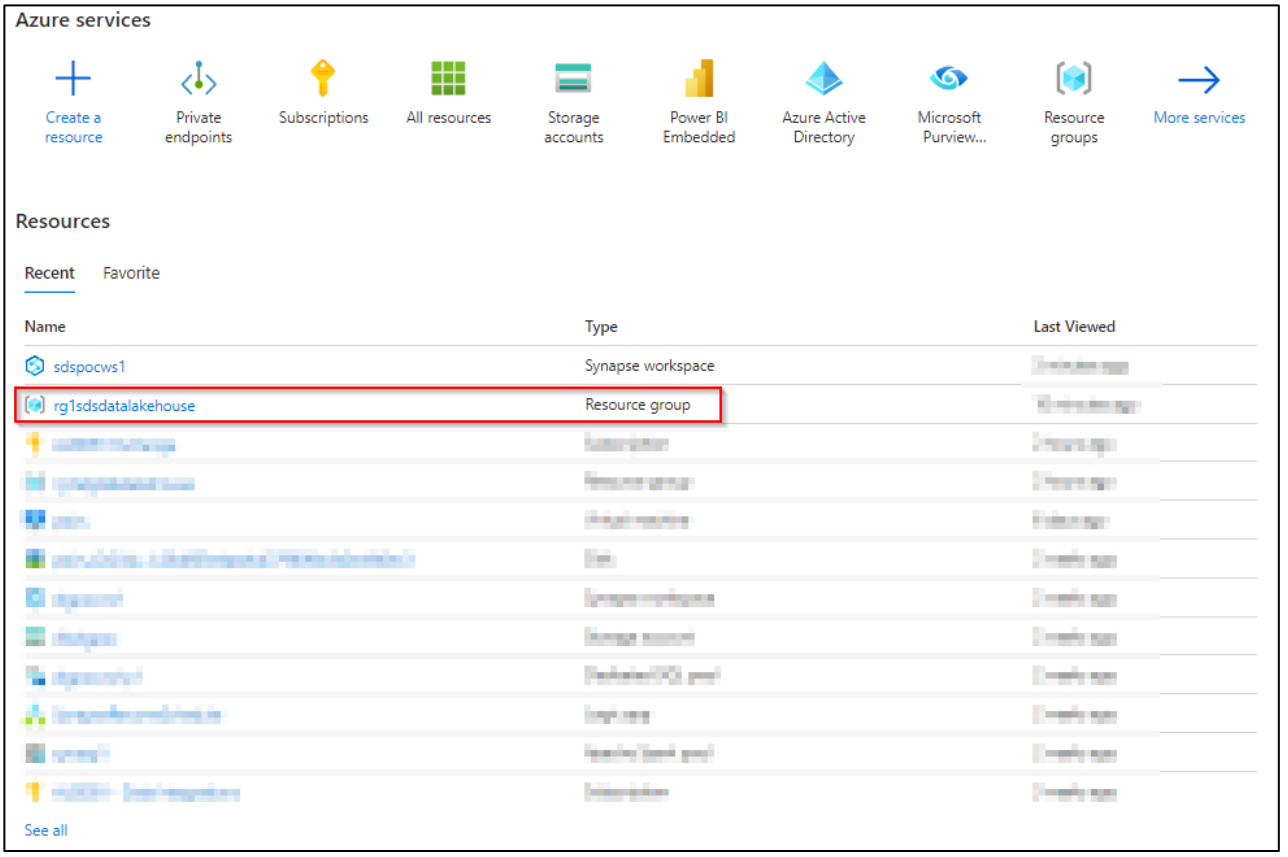
14. A new panel will open on the right side, where you can define the name of the new Directory. You will create 1 new directory called **ProductSales**.

Your storage container structure should look like this:

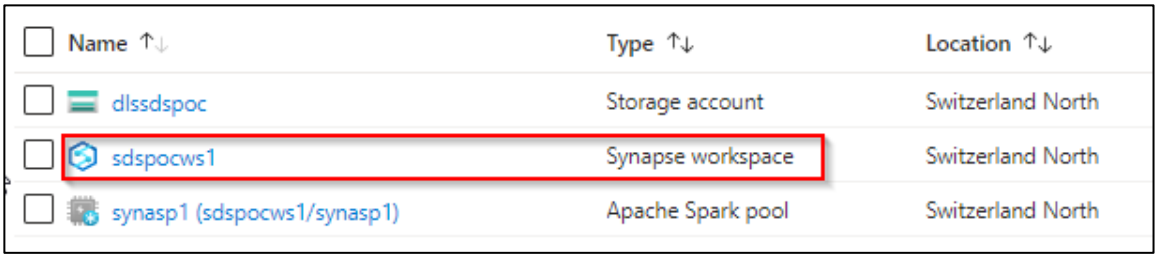


4 Starting Azure Synapse Analytics

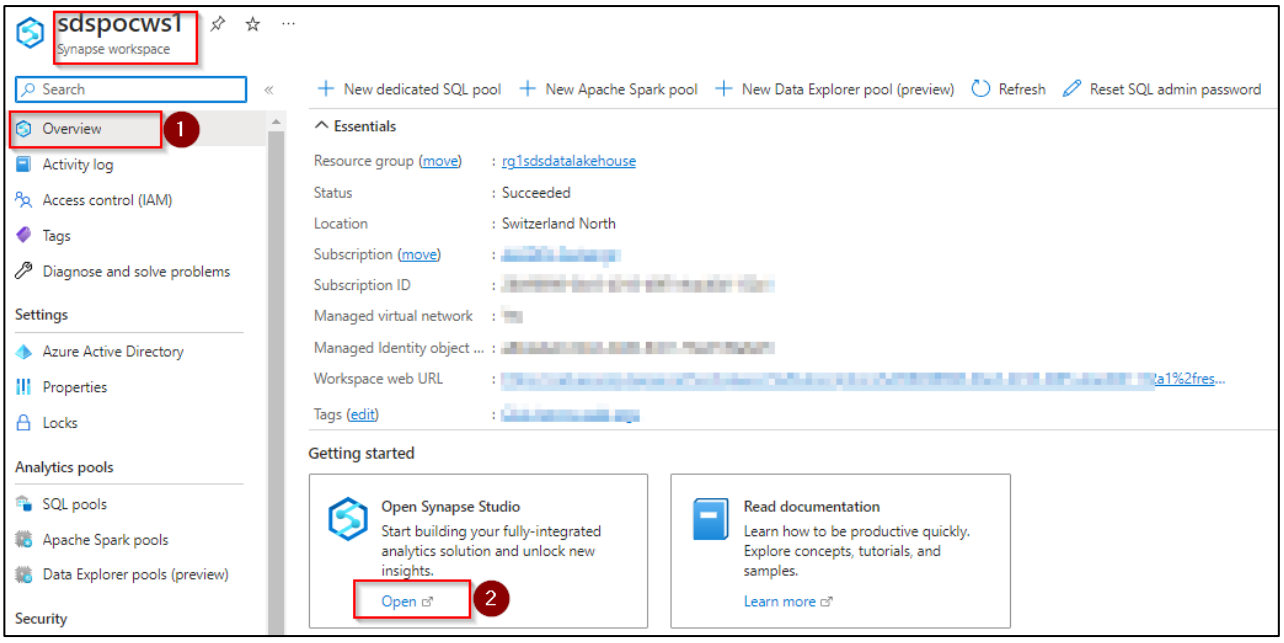
1. Go back to the Azure portal home screen and select the **Resource group** you provisioned.



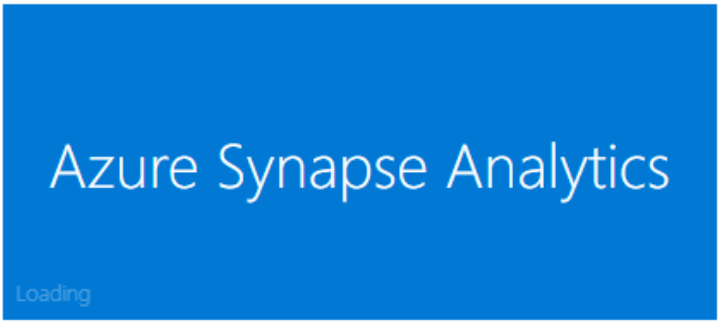
2. Once the resource group is open, **s**elect the resource type **Synapse workspace**.



3. A new page will appear. Launch **Azure Synapse Studio** by selecting **Open Synapse Studio** located in the **overview** section of the Synapse workspace.



Once selected, you will open a new webpage, launching Azure Synapse Studio. You will see the following banner:



5 Create Azure Synapse Pipelines

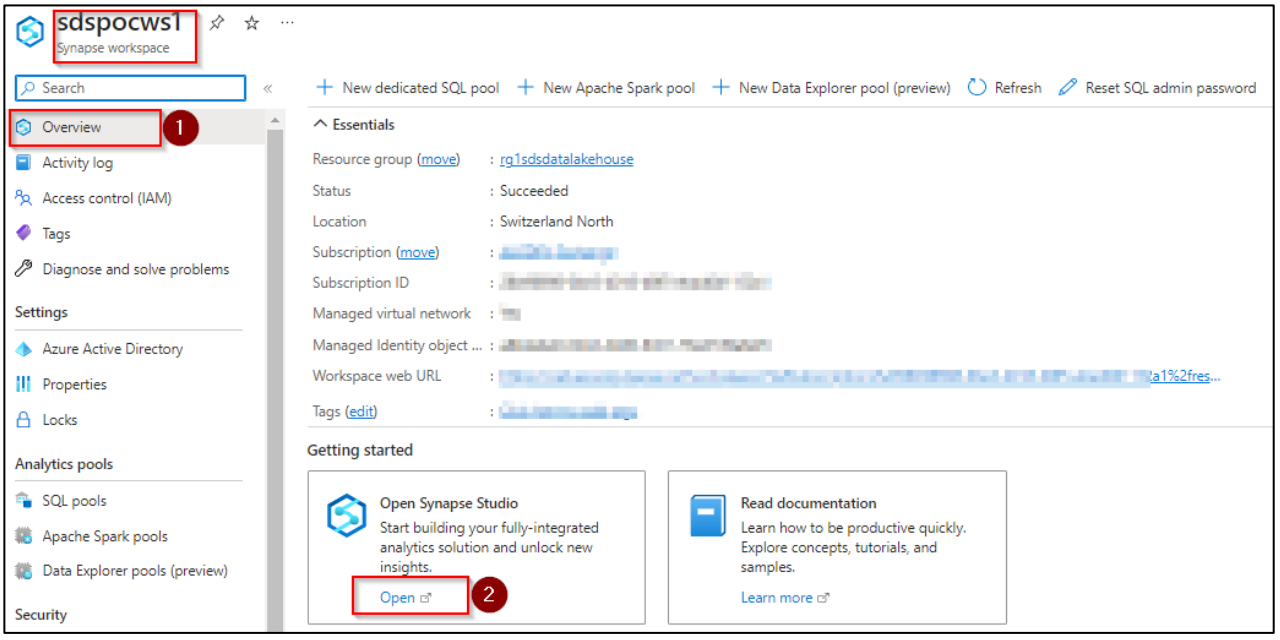
You will create two data flows using Synapse pipelines. These dataflows will move the data in the bronze container into the silver container and from the silver container into the gold containers. Each data flow will transform the data before copying it into each container in the Delta Format.

5.1 Transform Data into Delta Format

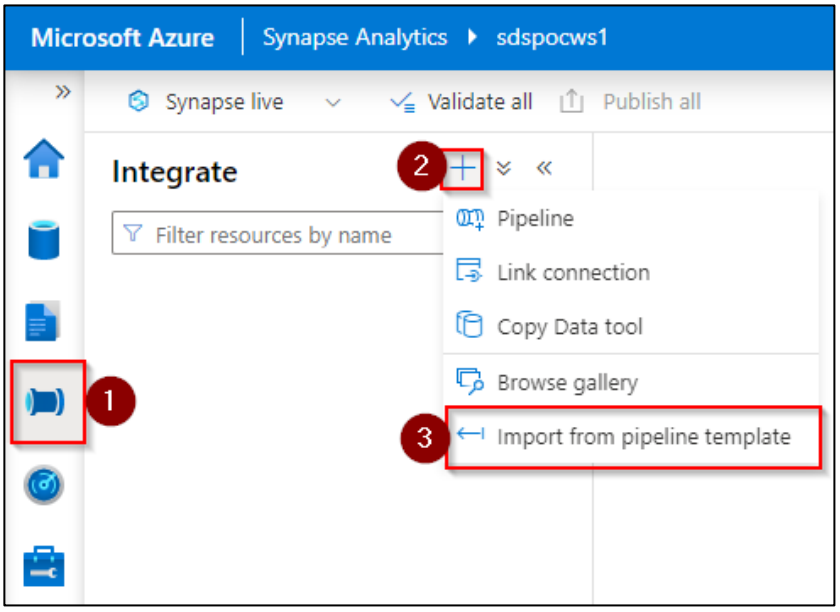
The Delta format is an open format storage layer that brings ACID (atomicity, consistency, isolation, and durability) transactions, time travelling capabilities, DML operations and many more features to Apache Spark. The Delta format stores the data in a parquet file format while also maintaining transaction logs and statistics to provide features and performance improvement over standard parquet.

5.1.1 Data Flows

1. If you haven't already, launch **Azure Synapse Studio** by selecting **Open Synapse Studio** located in the overview section of the Synapse workspace.

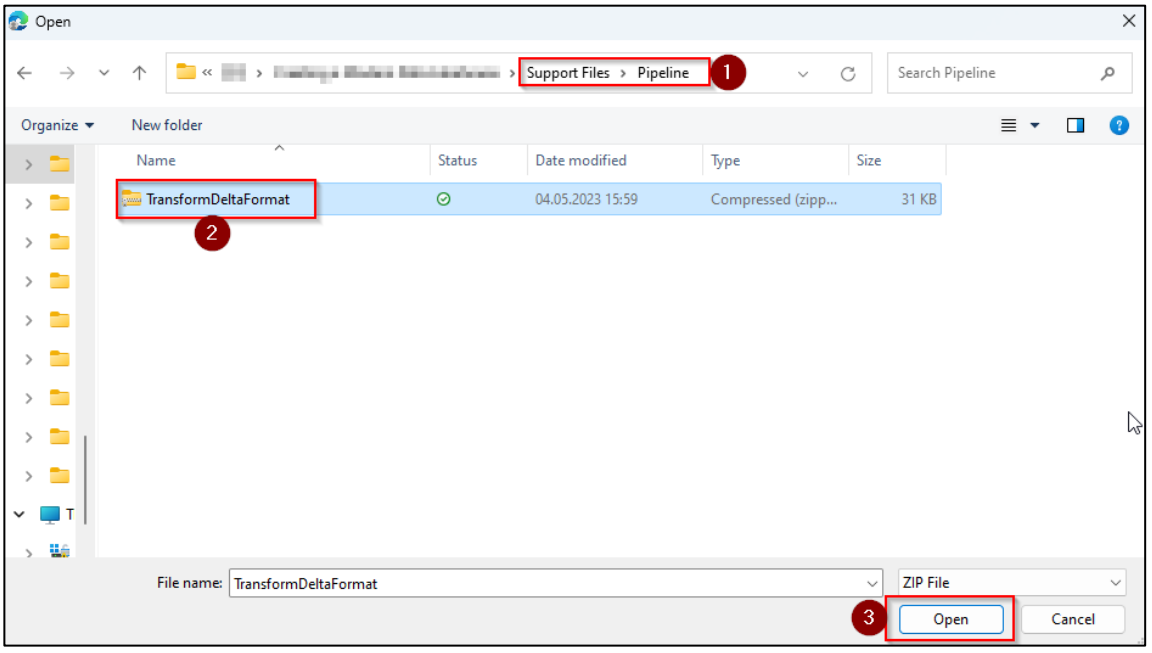


2. Once **Synapse Studio** opens, on the left side, select the **Integrate** tab, the **plus sign (+)** at the top of the new panel and **Import from Pipeline Template** in the drop down, to create a new pipeline.



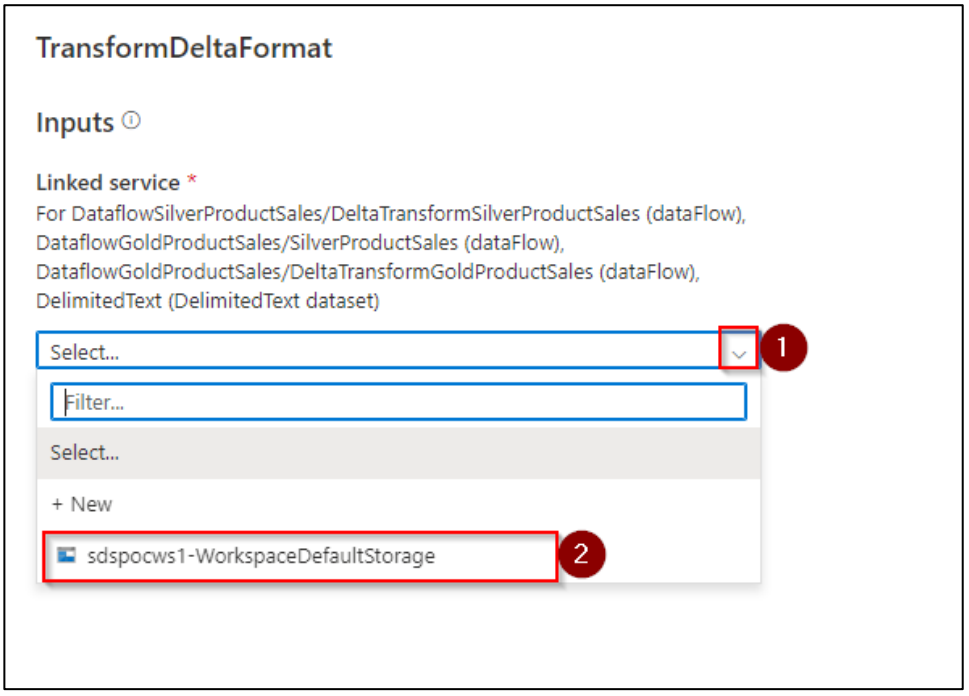
3. Once **Import from Pipeline Template** is selected, a new window to browse to a ZIP file will open.
4. Browse to the Folder **Support Files** and then **Pipeline**.
5. Select the ZIP file **TransformDeltaFormat** and click **Open**.

NOTE: The Support Files can be found in the Git repository in this link. Download these files onto your local computer.



A new window will open to configure the Linked Service.

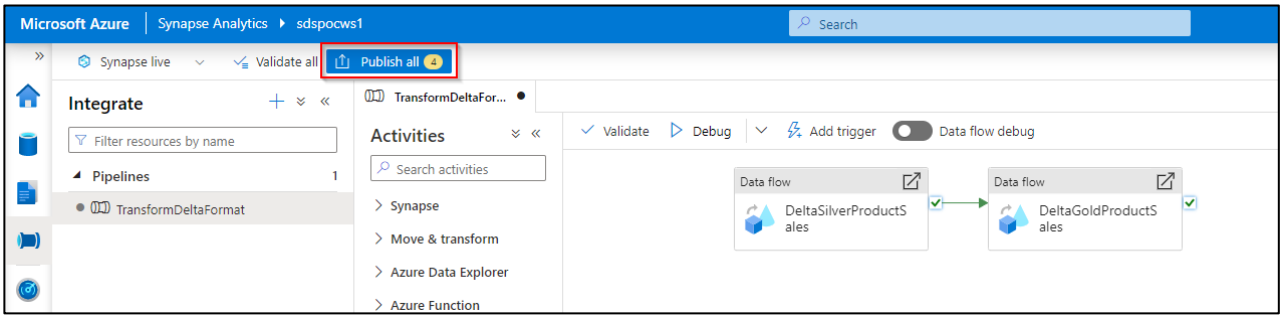
6. Within the drop box under Linked Service, select the provisioned storage account.



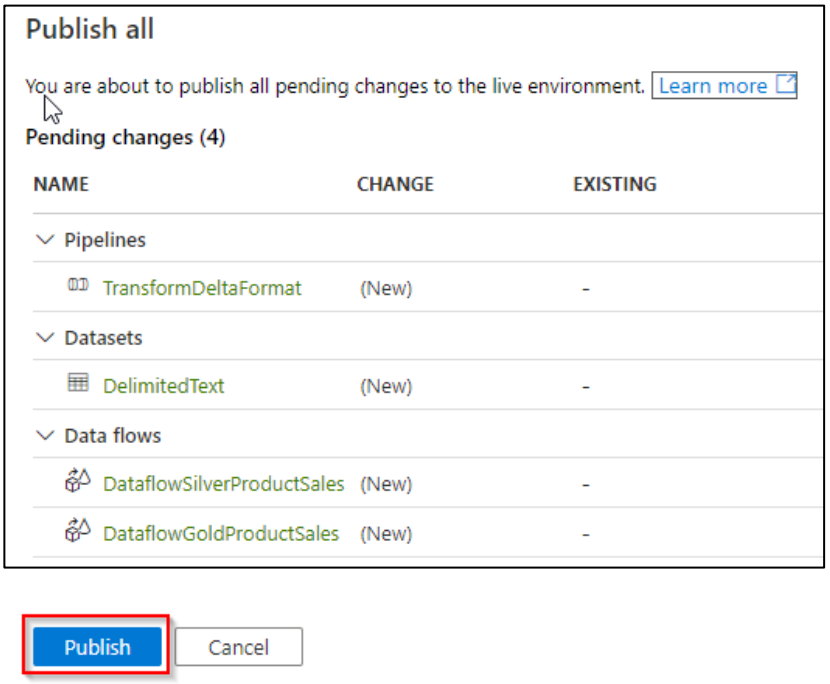
7. Once selected, click **Open pipeline** at the bottom of the page.

The pipeline will be imported, and you should see two data flows called **DeltaSilverProductSales** and **DeltaGoldProductSales**.

8. Select the **Publish all** button to save the work you have done so far.



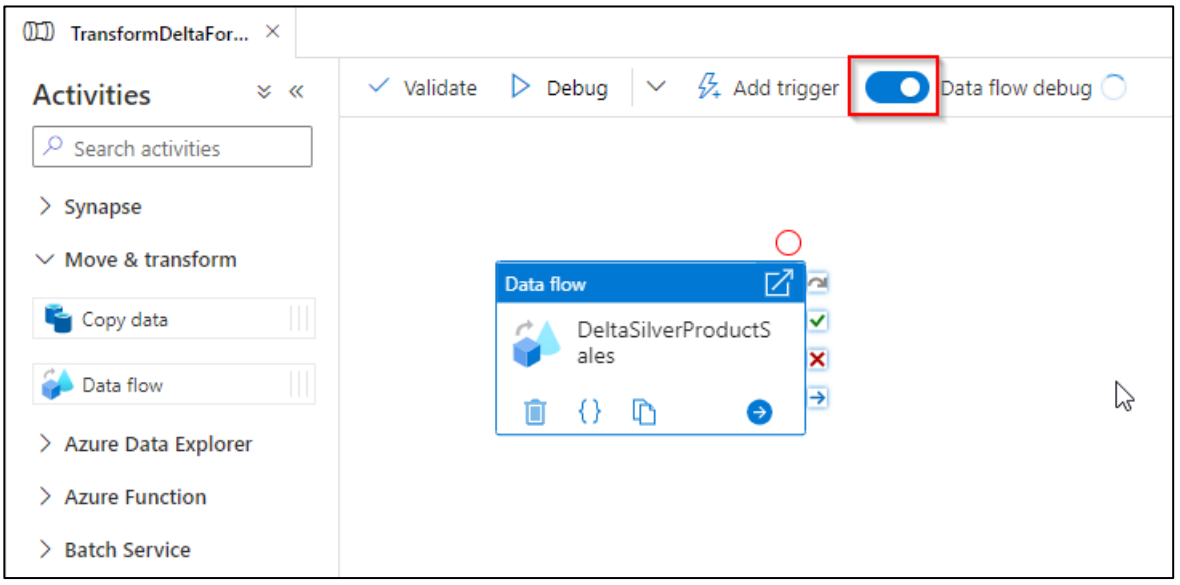
9. A new panel should open to the right. Select the **Publish** button.



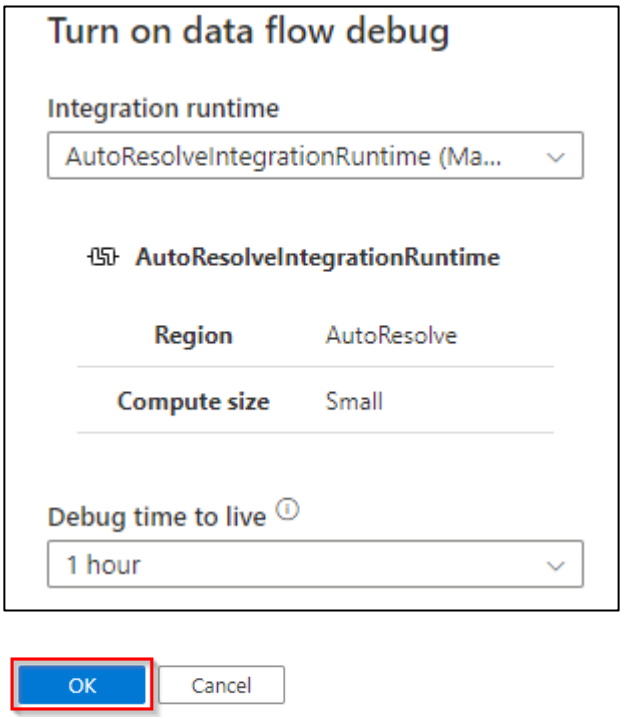
A banner should appear confirming that the content was published.

10. In the top bar of the pipeline canvas, slide the **Data flow debug** slider on.

Debug mode allows for interactive testing of transformation logic against a live Spark cluster. Data Flow clusters take 5 - 7 minutes to warm up and users are recommended to turn on debug first if they plan to do Data Flow development.

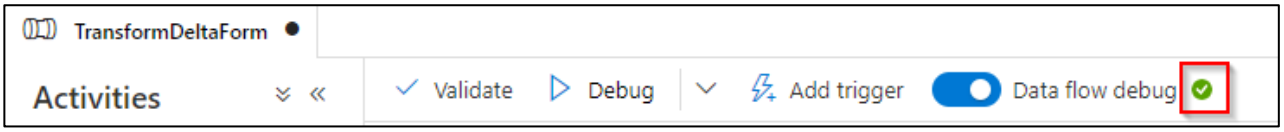


11. Once toggled, a new panel will open to the right, with the debug setting. Leave the default settings and select **OK**.



Note: The debug initialization might take 3 – 5 minutes. Please be patient!

12. You will get a green check once the debug has initialized:

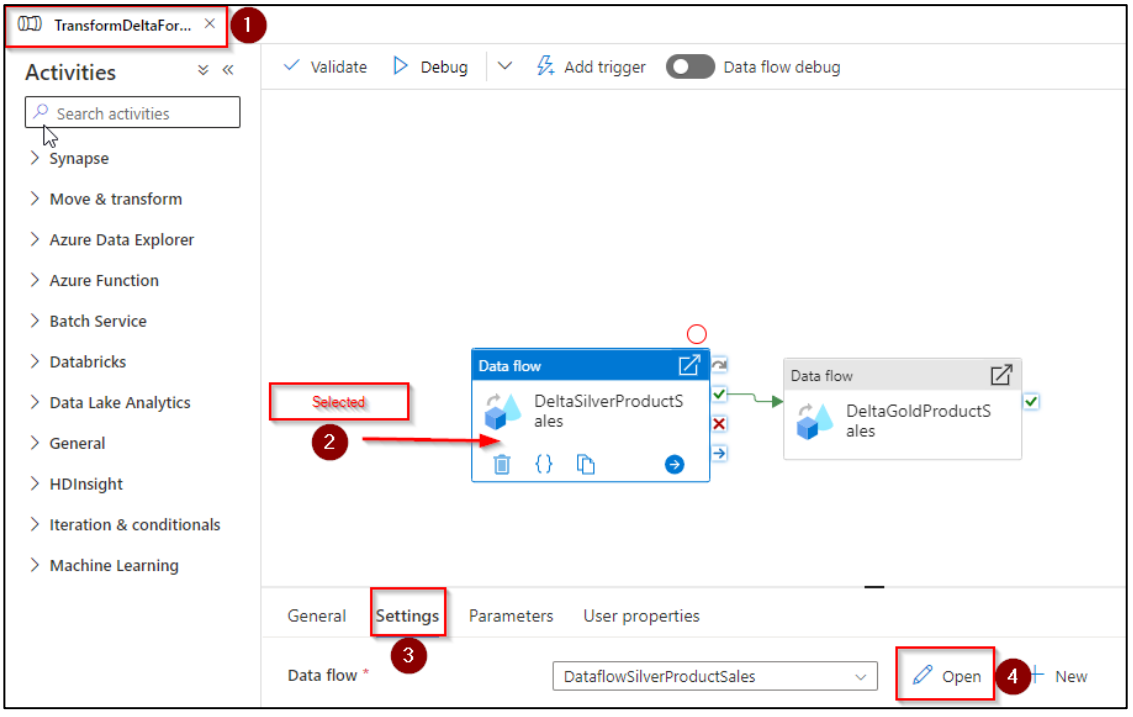


The **Debug time to live** is the amount of time that the integration runtime will wait after your last data preview before automatically shutting down your debug cluster. To avoid billing for the entire TTL, you can shut down the debug session when you are finished working.

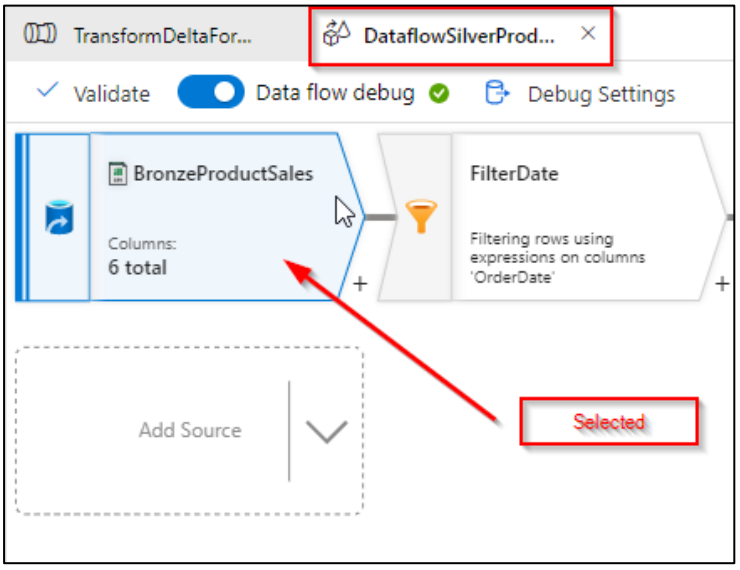
5.1.1.1 DeltaSilverProductSales

Before the data flows can be triggered, we need to make sure that the data sources i.e., the CSV files being imported into the dataflows are being found and that the sink sources are correctly mapped. This requires us to adjust parameters and change file paths.

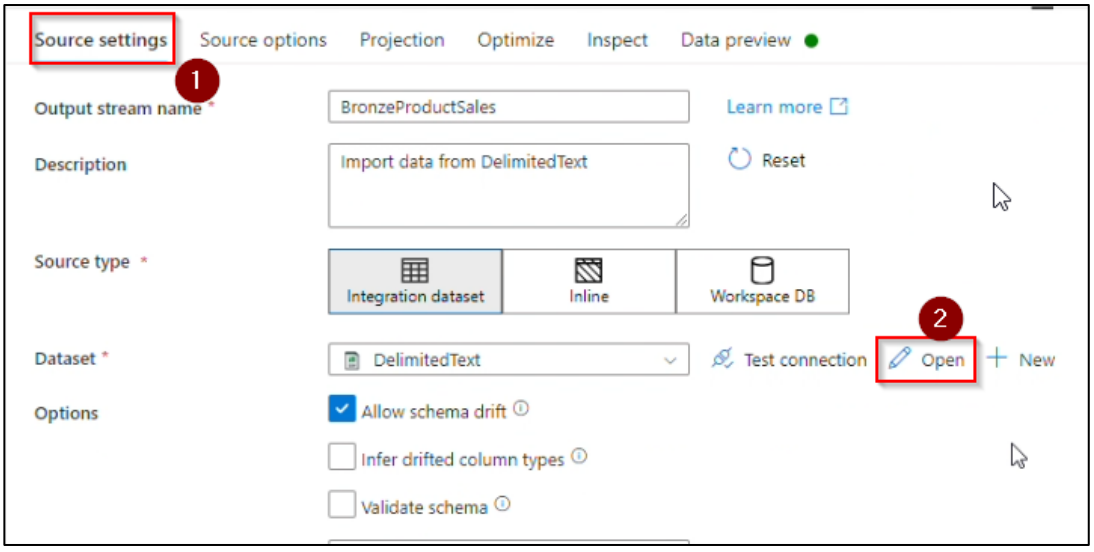
1. Within the imported **TransformDeltaFormat** pipeline, select the Data flow **DeltaSilverProductSales**.
2. In the **Settings** tab select the **Open** button next to the Data flow **DeltaSilverProductSales**.



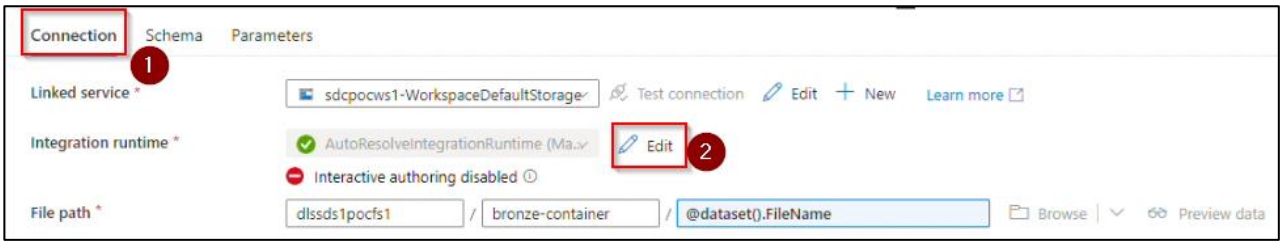
3. Within the opened data flow, select the **BronzeProductSales** tile.



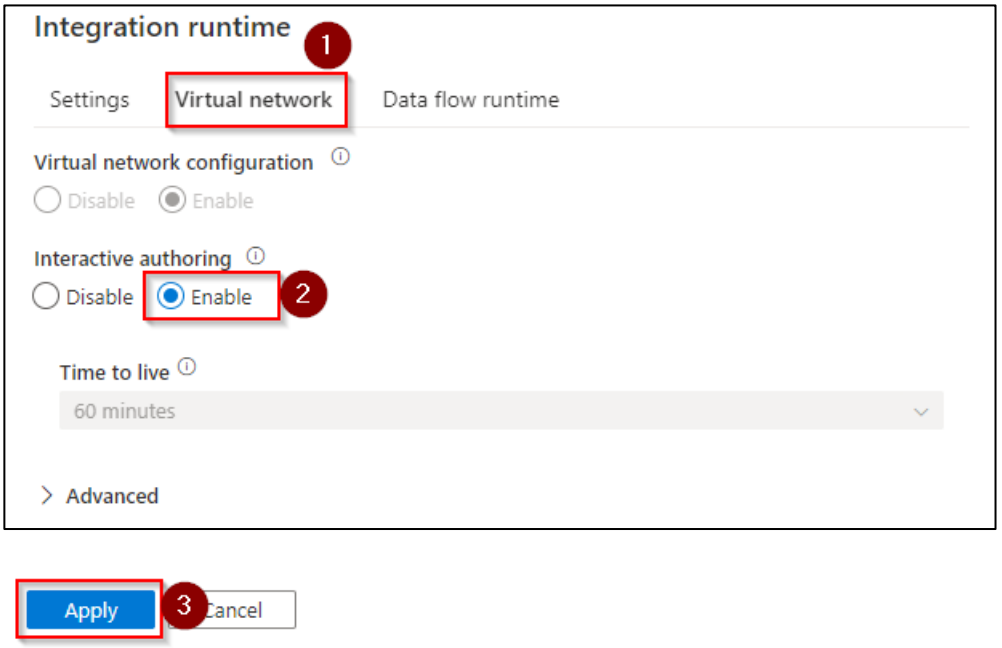
4. Under the **Source settings** tab, select the **Open** button next to **Dataset** (DelimitedText)



5. Once selected a new tab called **DelimitedText** will open.
6. Under the **Connection** tab, select the **Edit** button next to **Integration runtime**.



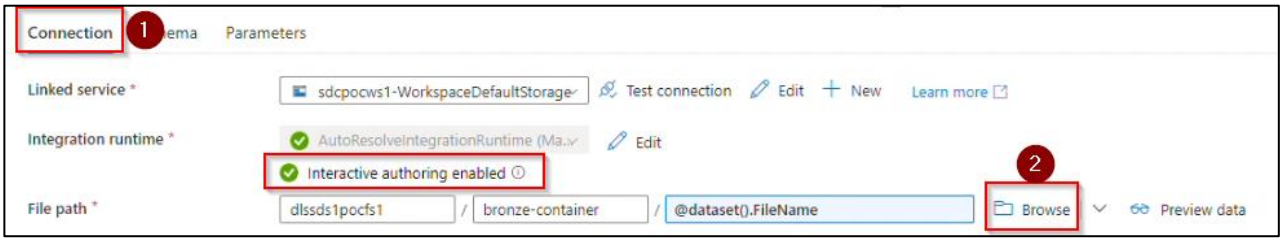
7. A new panel called **Integration runtime** will open. Go to the **Virtual Network** tab and select **Enable** under Interactive authoring. Select **Apply**.



NOTE: It may take 1 – 3 minutes for interactive authoring to be enabled. You will see the banner below:

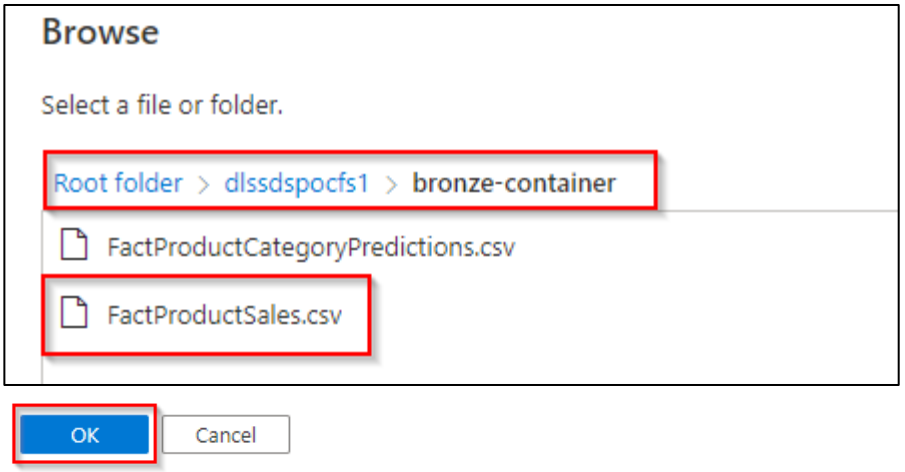
Once enabled, you will be able to Browse the directory folder.

13. Under the **Connection** tab, select the **Browse** button next to **File path**.

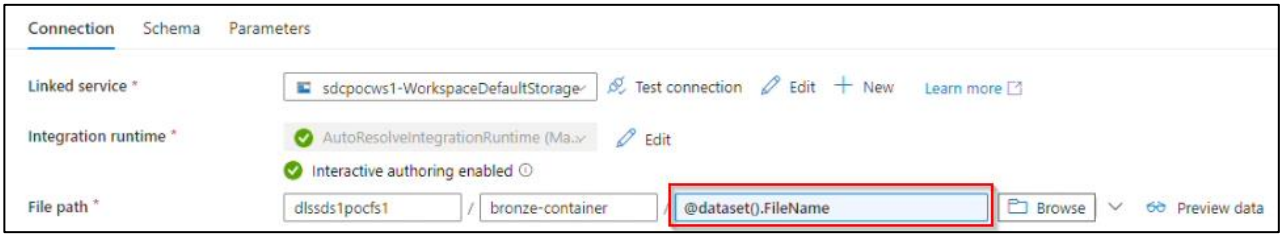


14. Browse to the **bronze-container** directory and select the **FactProductSales.csv** file.
15. Once the file path has been set, select **OK**.

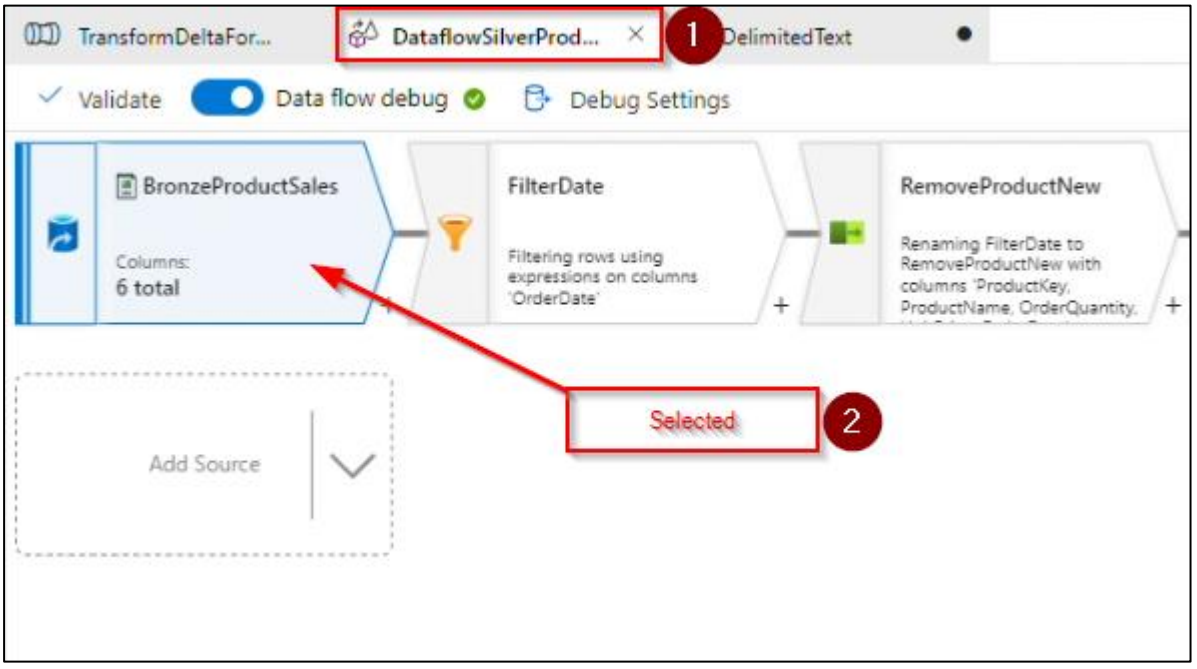
NOTE: The path should be similar to the one depicted below. Keep in mind that your storage account (Ex.: dlssdspocfs1) will have a different name.



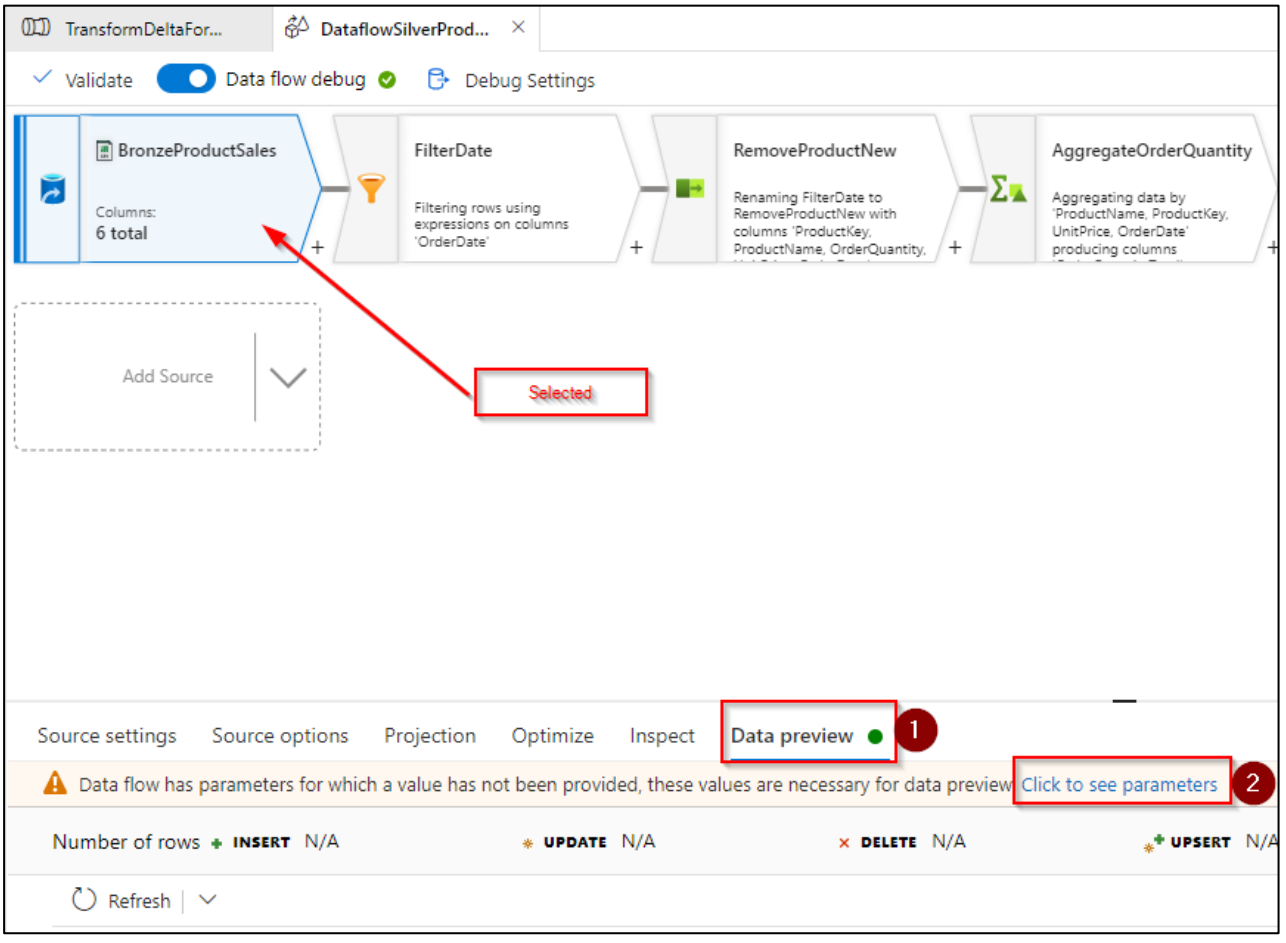
NOTE: Even though you selected the FactProductSales.csv file, your file path should retain the parameter @dataset().FileName. The only thing that should change is the storage name.



16. Go back to the **DataflowSilverProductSales** tab and select the **BronzeproductSales** tile.

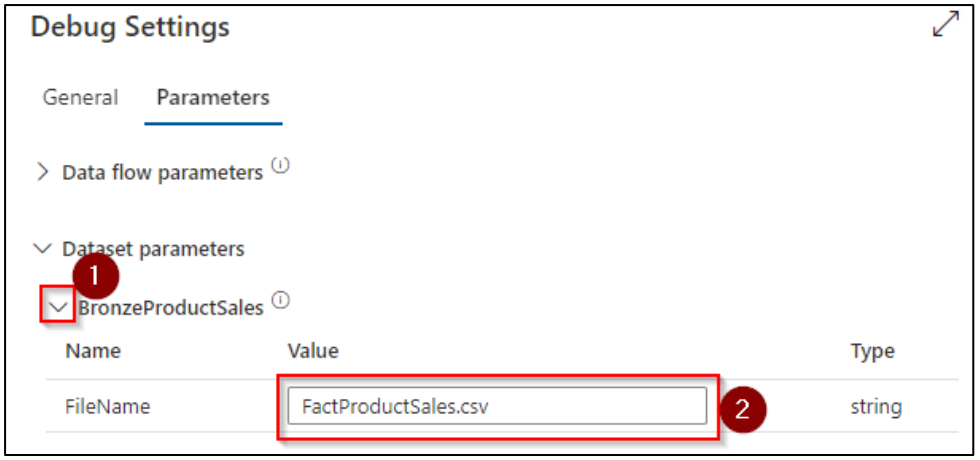


17. Select the **Data Preview** tab.
18. Once selected, there will be a yellow ribbon notifying you of parameters that require values. Select the field **Click to see parameters**.



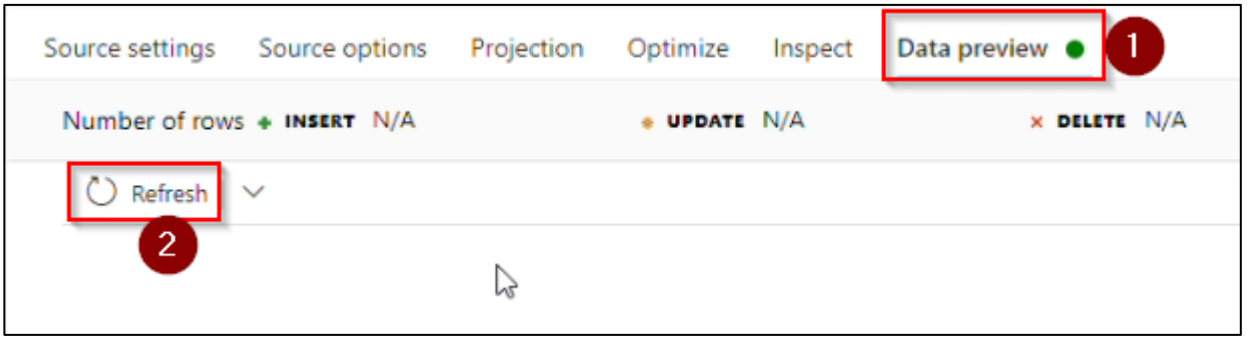
A new panel will open to the right. You will now populate these parameters.

19. Select the arrow next to **BronzeProductSales**, add the name of the Sales CSV file **FactProductSales.csv** and select **Save.**

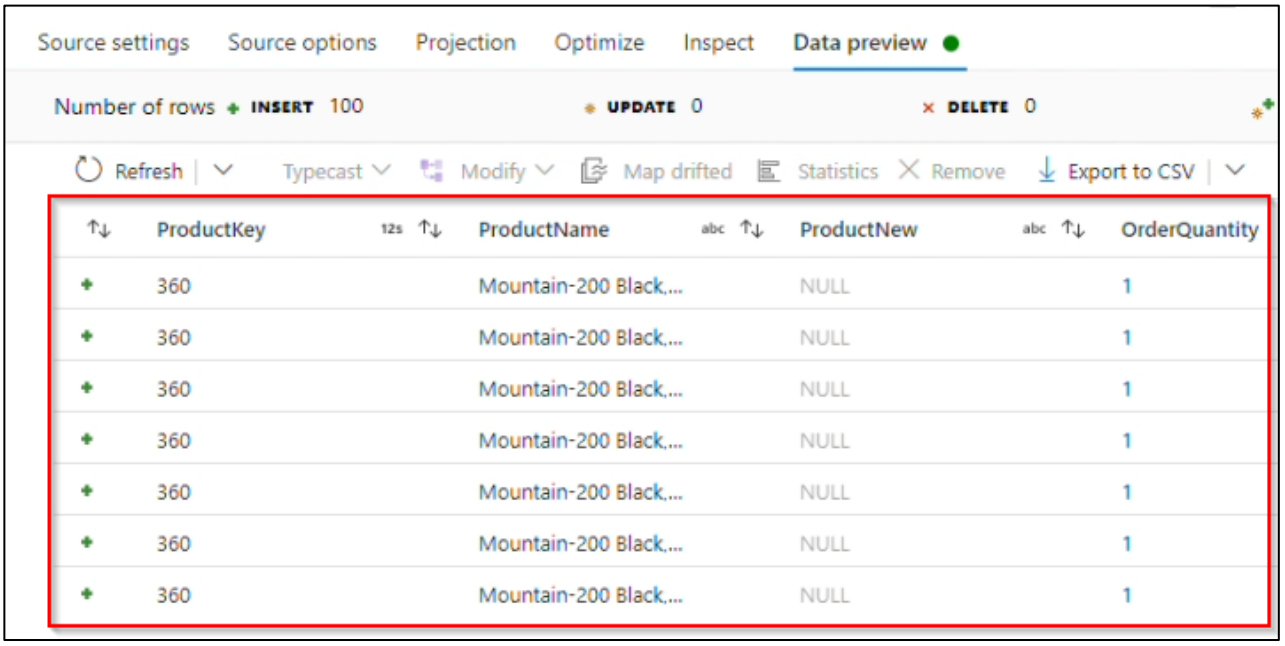


The panel will close, and you will be brought back to the Data preview tab.

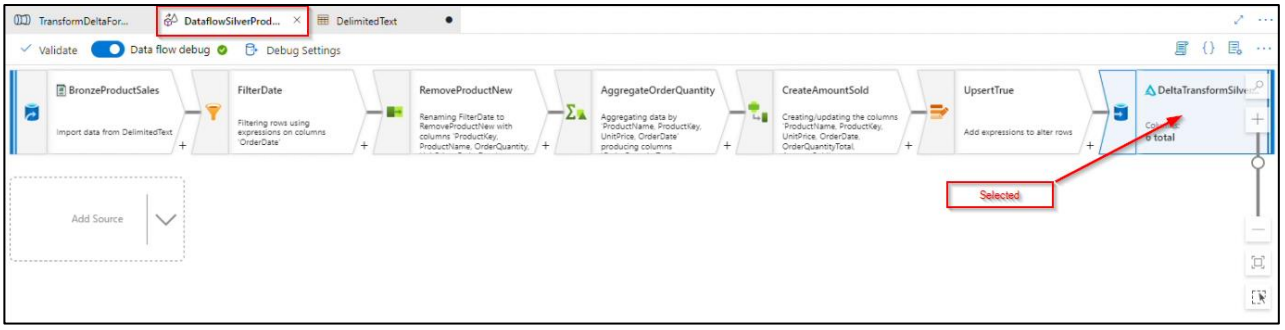
20. Under the Data preview tab, select the **Refresh** button.



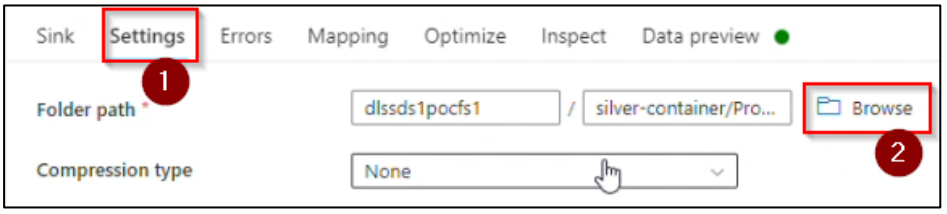
The data will be fetched, and you should see a preview of your table.



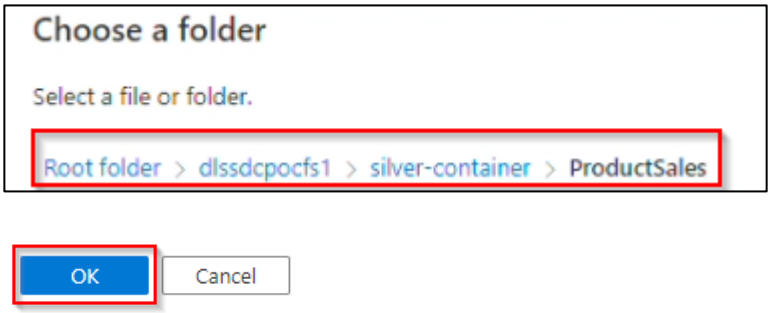
21. Within the **DataflowSilverProductSales** tab, select the **DeltaTransformSilver** tile.



22. Select the **Settings** tab and click the **Browse** button next to **Folder path**.

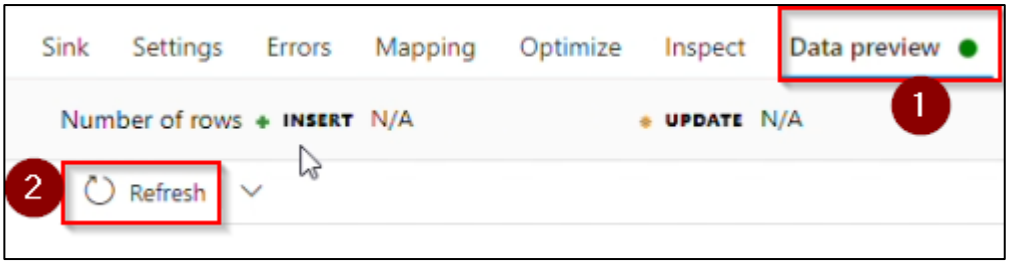


23. Browse to the **silver-container** directory and select the **ProductSales** folder.
24. Once the file path has been set, select **OK**.

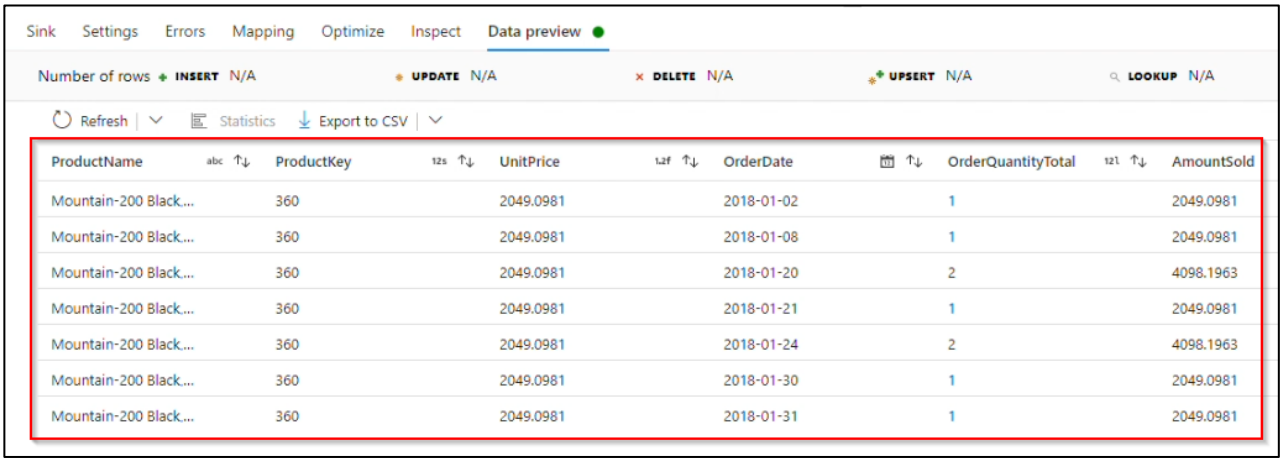


The panel will close, and you will be brought back to the Settings tab.

25. Navigate to the **Data preview** tab and select the **Refresh** button.

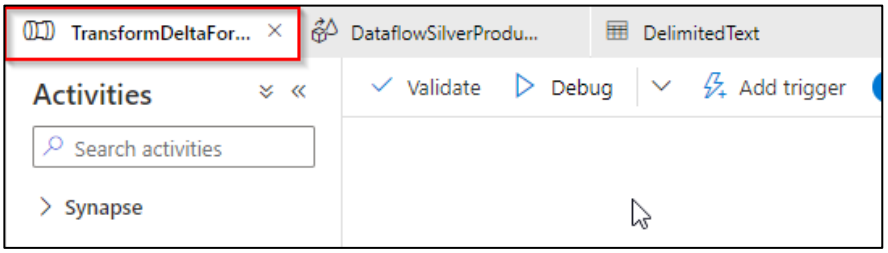


The data will be fetched, and you should see a preview of your table.

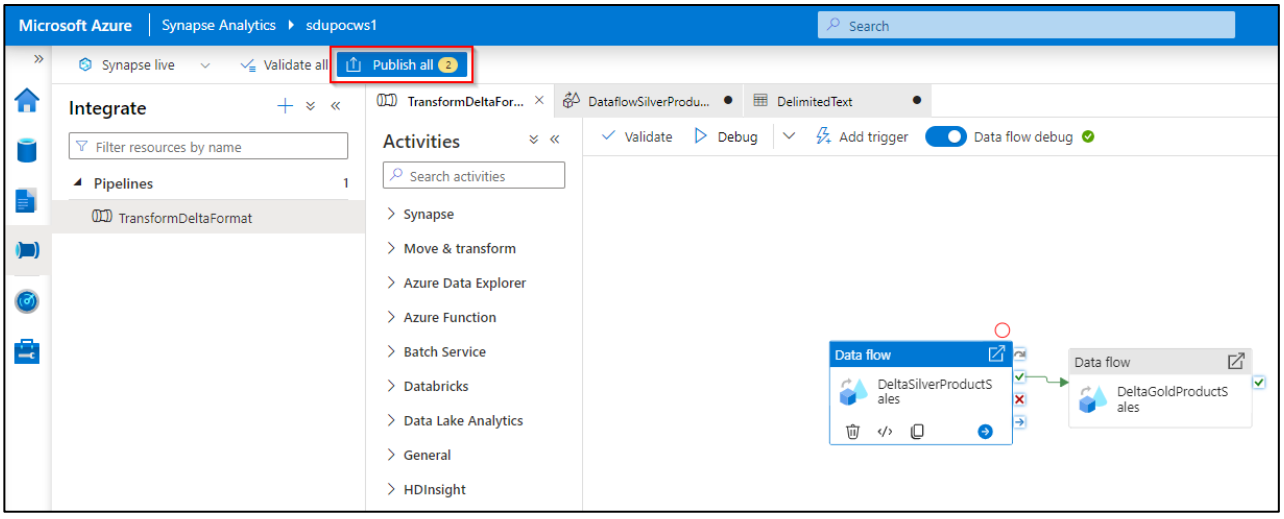


The **DeltaSilverProductSales** dataflow is ready. We now need to configure the second dataflow called **DeltaGoldProductSales** in a similar manner.

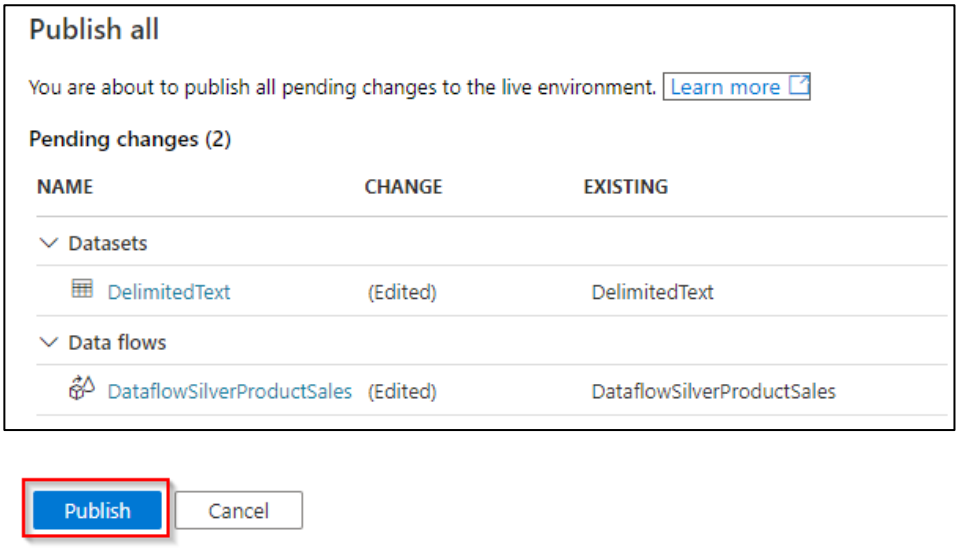
26. Go back to the **TransformDeltaFormat** tab.



27. Select the **Publish all** button to save the work you have done so far.



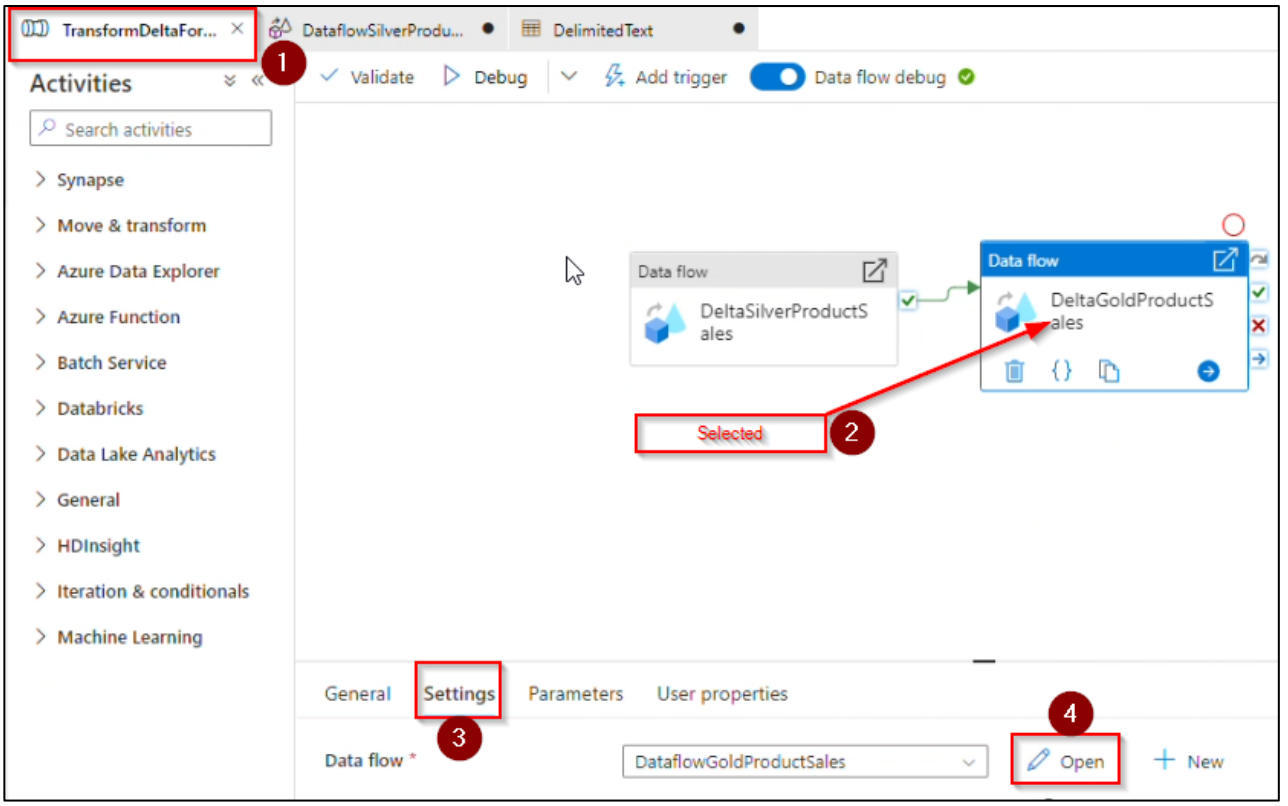
28. A new panel should open to the right. Select the **Publish** button.



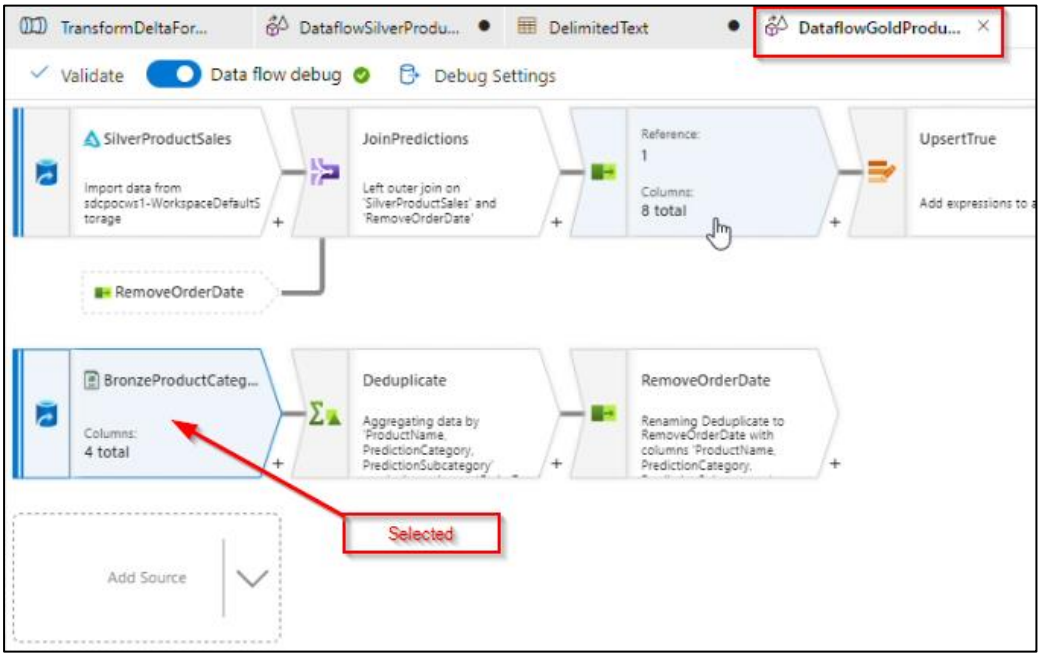
A banner should appear confirming that the content was published.

5.1.1.2 DeltaGoldProductSales

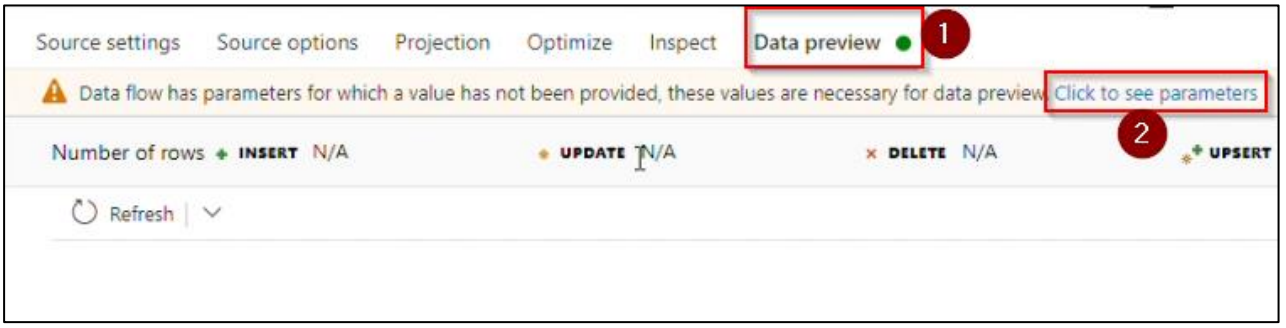
1. Go back to the **TransformDeltaFormat** tab and select the Data flow **DeltaGoldProductSales**.
2. In the **Settings** tab select the **Open** button next to the Data flow **DeltaGoldProductSales**.



3. Within the opened data flow, select the **BronzeProductCategoryPredictions** tile.

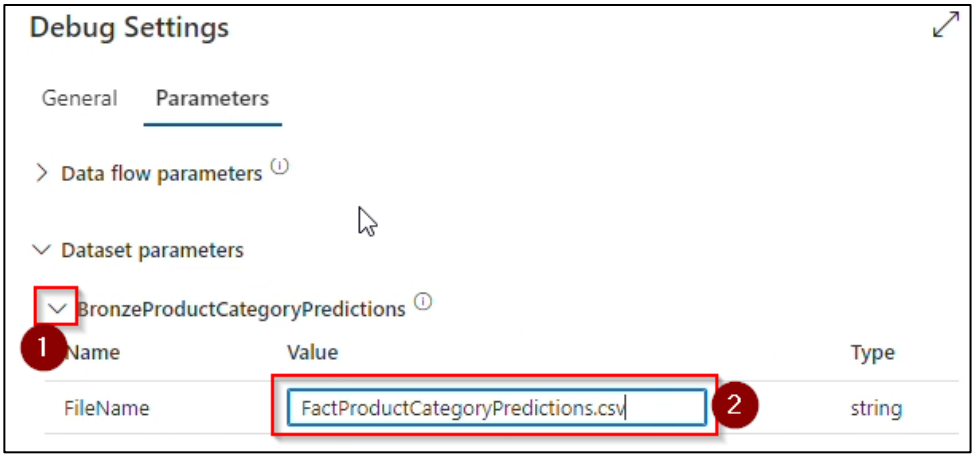


4. Select the **Data Preview** tab.
5. Once selected, there will be a yellow ribbon notifying you of parameters that require values. Select the field **Click to see parameters**.



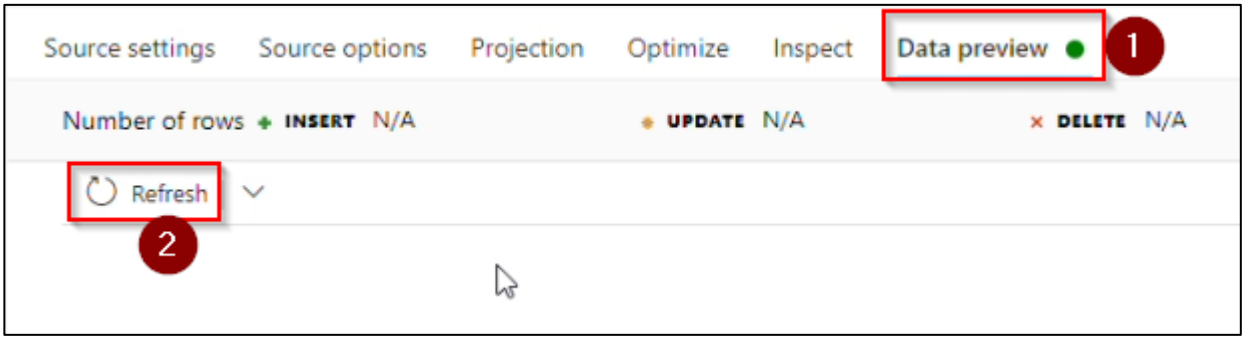
A new panel will open to the right. You will now populate these parameters.

6. Select the arrow next to **BronzeProductCategoryPredictions**, add the name of the Prediction CSV file **FactProductCategoryPredictions.csv** and select **Save.**

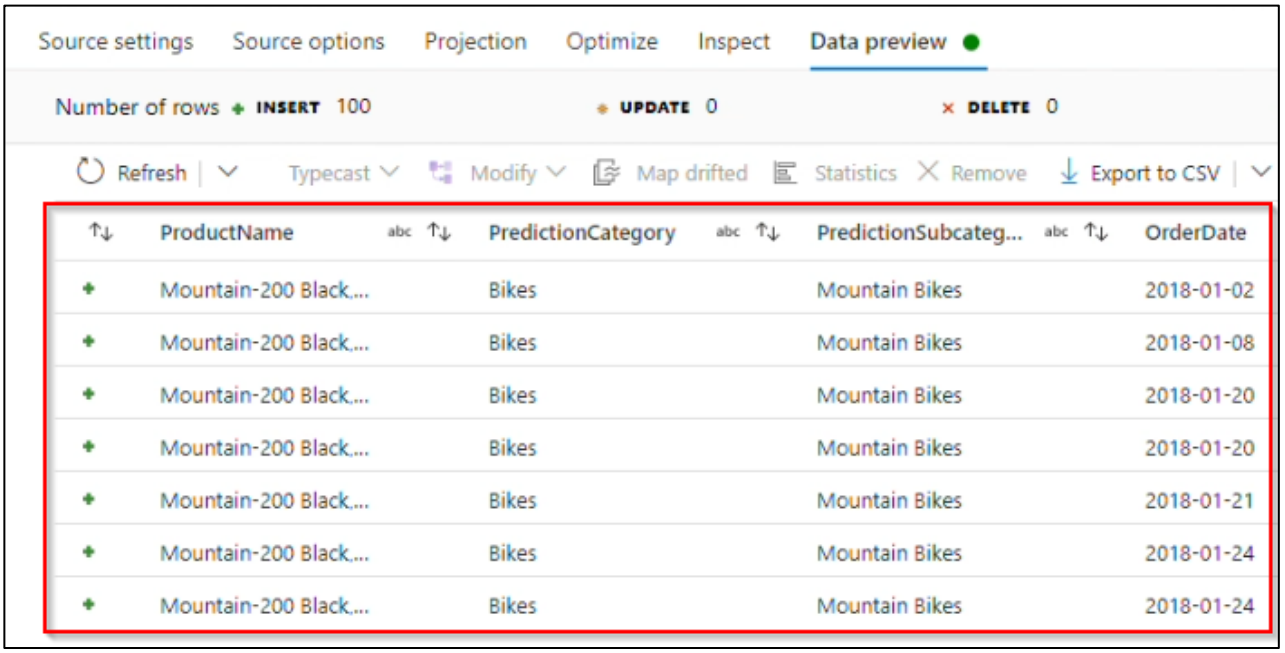


The panel will close, and you will be brought back to the Data preview tab.

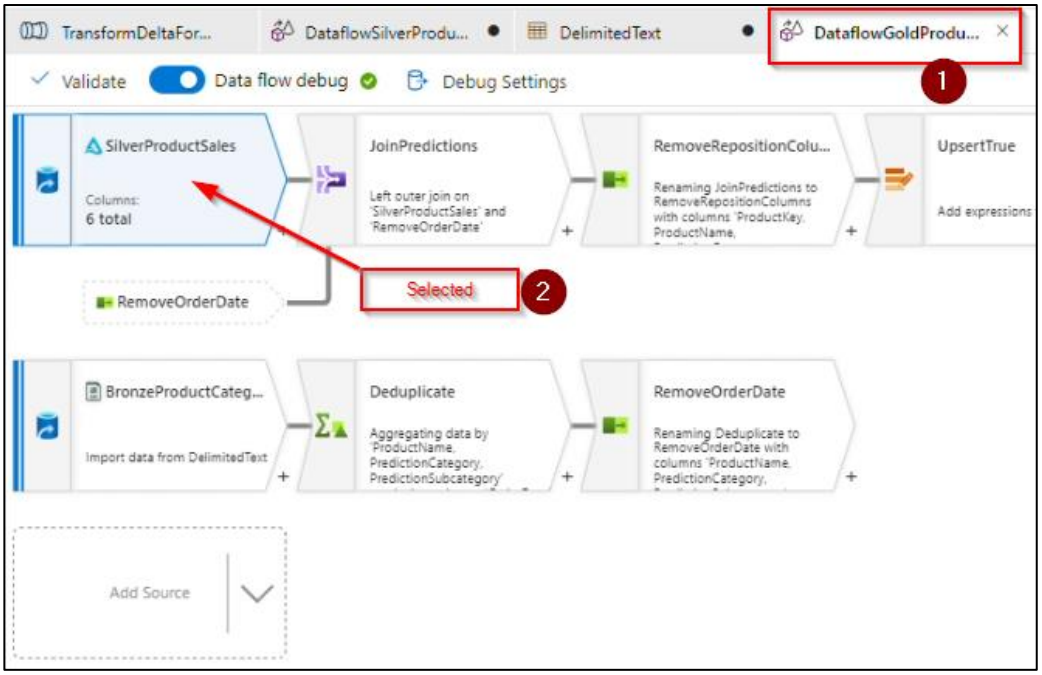
7. Under the Data preview tab, select the **Refresh** button.



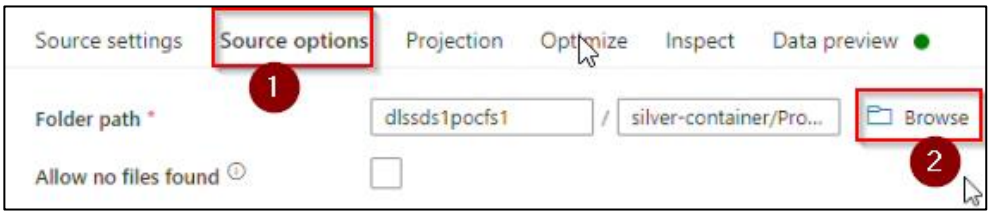
The data will be fetched, and you should see a preview of your table.



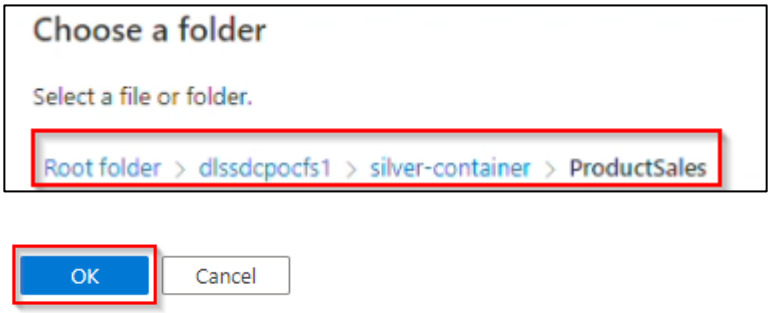
8. Within the **DeltaGoldProductSales** tab, select the **SilverProductSales** tile.



9. Select the **Source options** tab and click the **Browse** button next to **Folder path**.

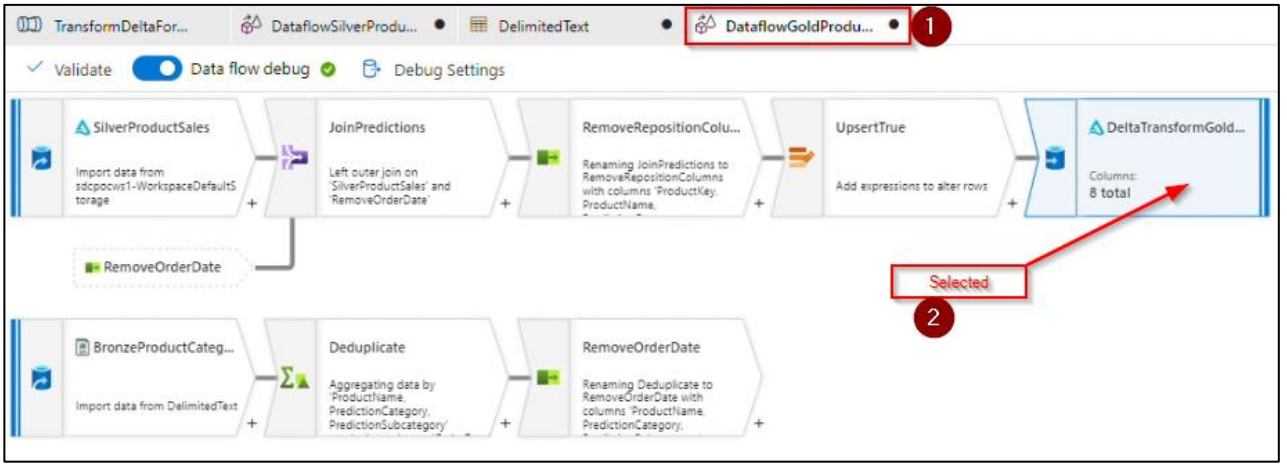


10. Browse to the **silver-container** directory and select the **ProductSales** folder.
11. Once the file path has been set, select **OK**.

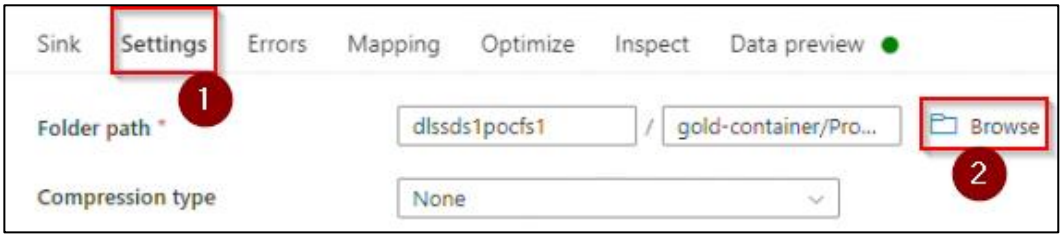


The panel will close, and you will be brought back to the Settings tab.

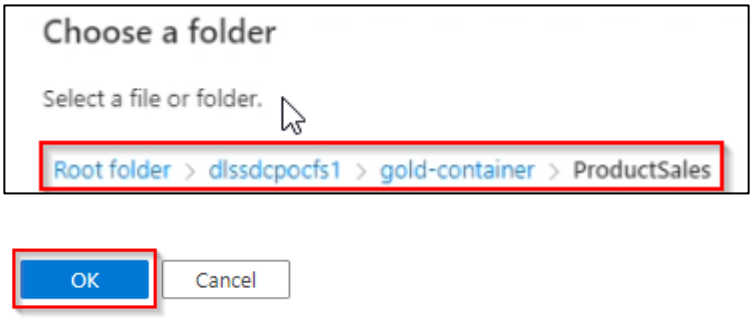
12. Within the **DeltaGoldProductSales** tab, select the **DeltaTransformGoldProductSales** tile.



13. Select the **Settings** tab and click the **Browse** button next to **Folder path**.



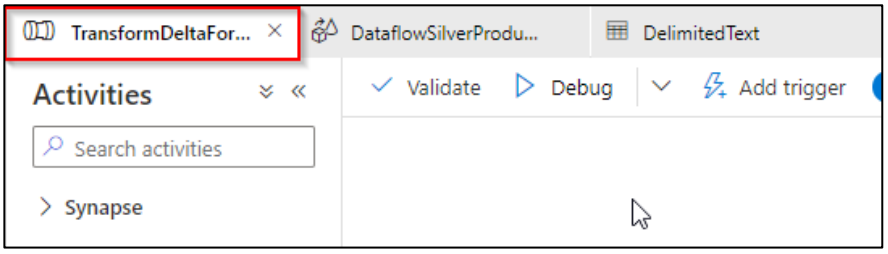
14. Browse to the **gold-container** directory and select the **ProductSales** folder.
15. Once the file path has been set, select **OK**.



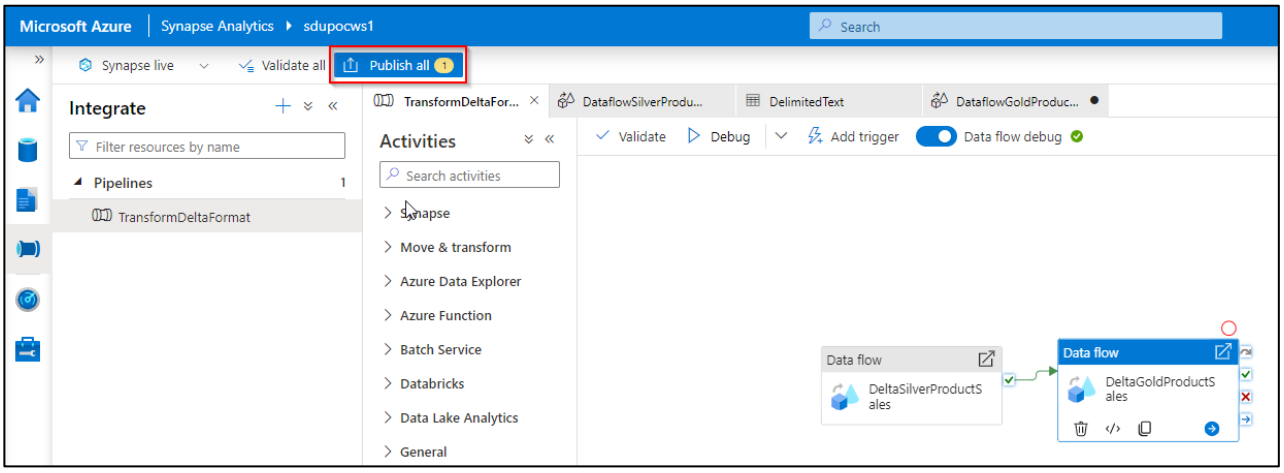
The panel will close, and you will be brought back to the Settings tab.

The **DeltaGoldProductSales** dataflow is now configured and the **TransformDeltaFormat** pipeline is ready to be triggered.

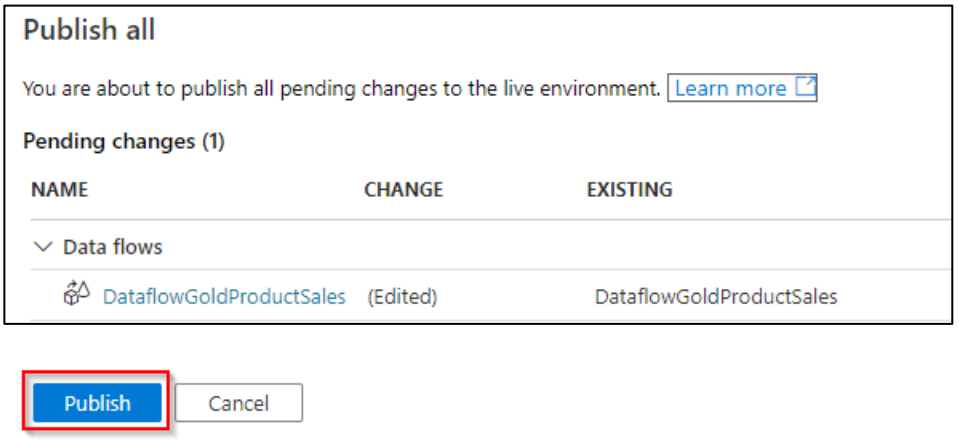
16. Go back to the **TransformDeltaFormat** tab.



17. Select the **Publish all** button to save the work you have done so far.



18. A new panel should open to the right. Select the **Publish** button.



A banner should appear confirming that the content was published.

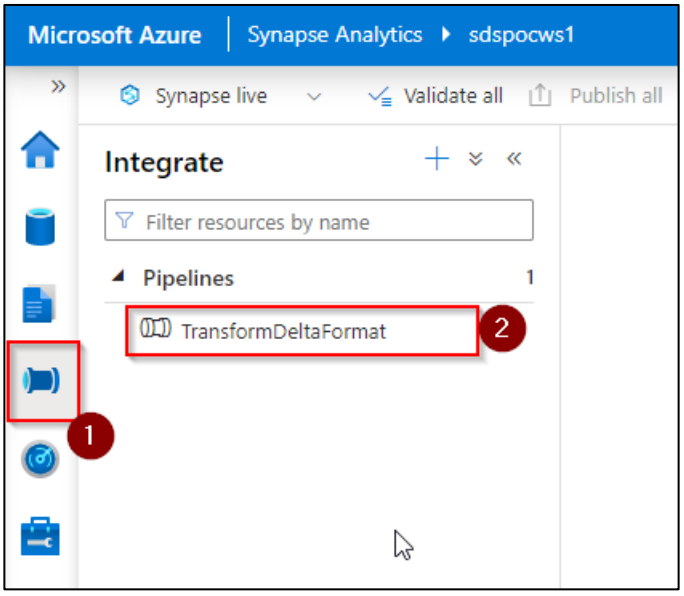
We will now manually trigger the dataflows we configured.

5.2 Manually Trigger Pipelines (Before Data Changes)

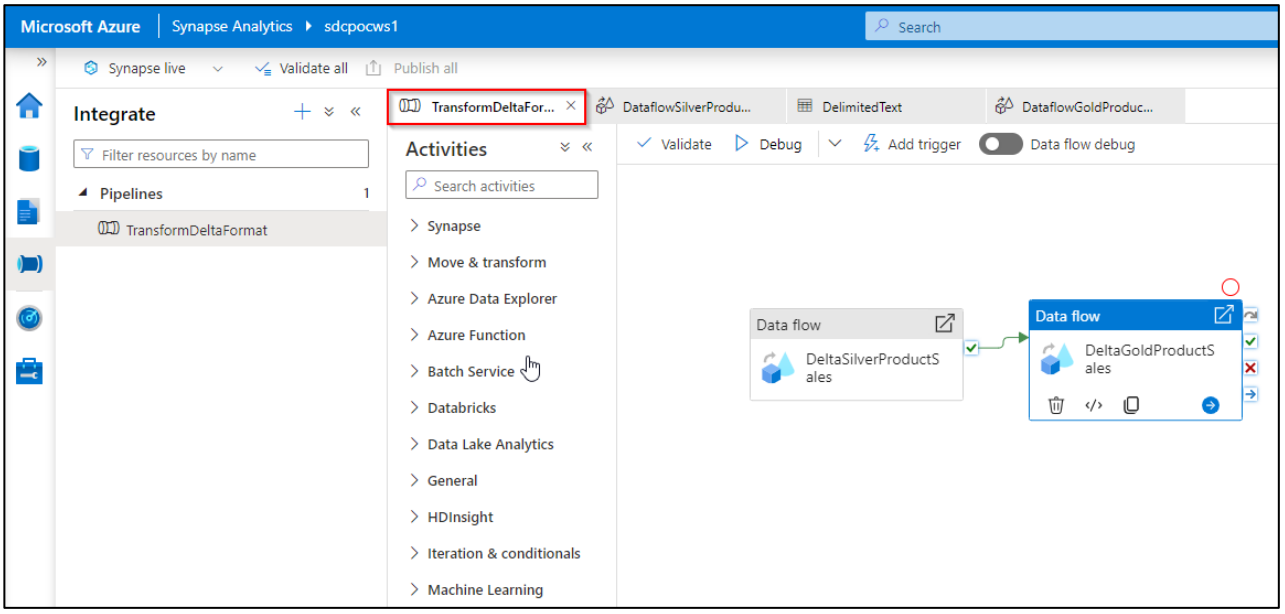
Let's trigger our synapse pipelines to extract, transform and load the raw csv data in our bronze-container as delta formatted data into our silver and gold-containers.

Azure Synapse Pipelines allow for event-based triggers, that start the pipeline when new data is added to the Azure storage. But to keep this example simple, we will be manually triggering the pipelines we have created.

1. Within the Azure Synapse Workspace, select the **Integrate** tab and select the pipeline **TransformDeltaFormat**.

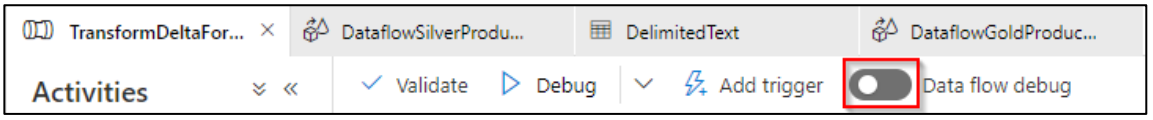


2. Once the pipeline is selected, a new tab with the dataflow activities we configured earlier will appear.

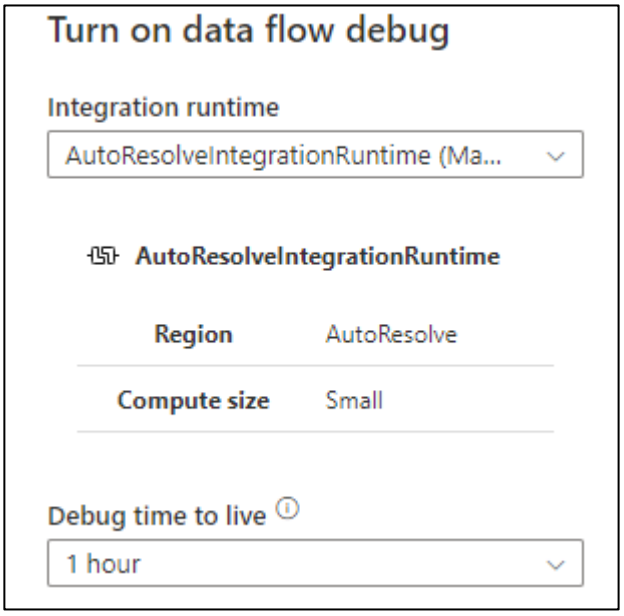


3. Select the **Data flow debug** toggle button, to initiate the integration runtime.

NOTE: If the debug is still on [green], you may continue directly on step 5

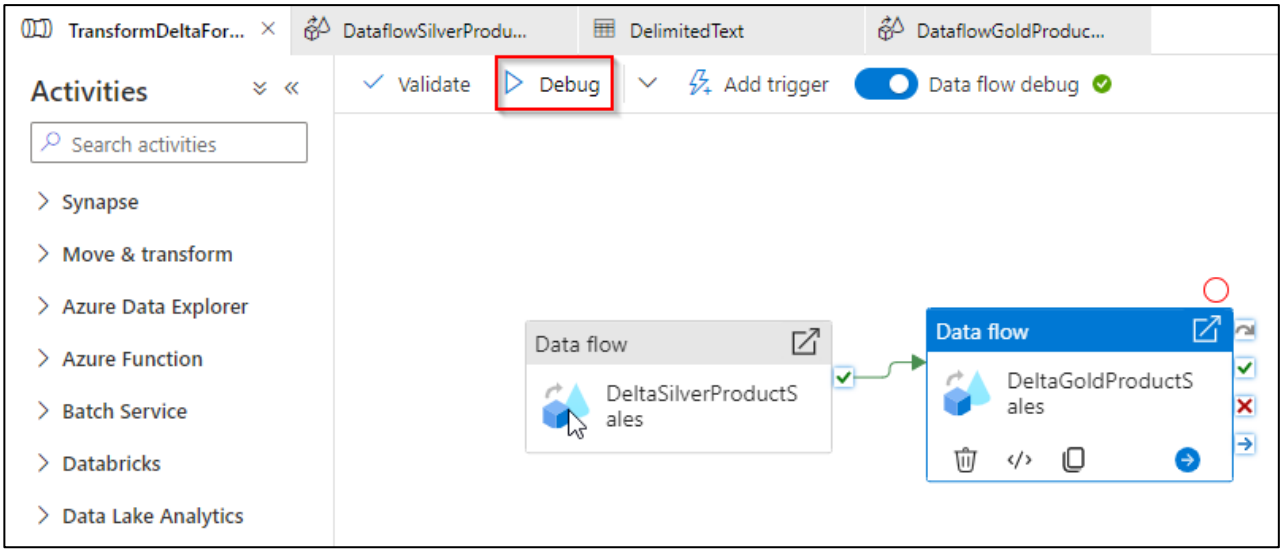


4. Once toggled, a new panel will open to the right, with the debug setting. Leave the default settings and select **OK**.

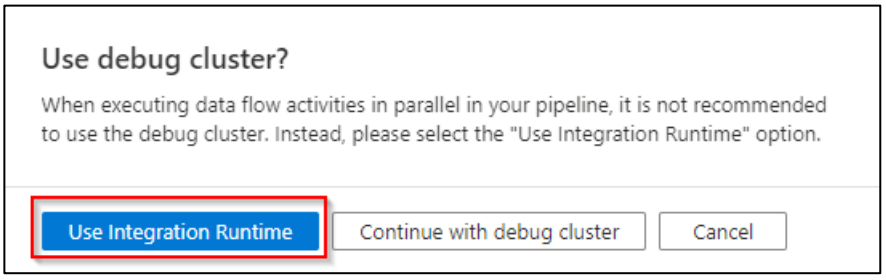


NOTE: The debug initialization might take 3 – 5 minutes. Please be patient!

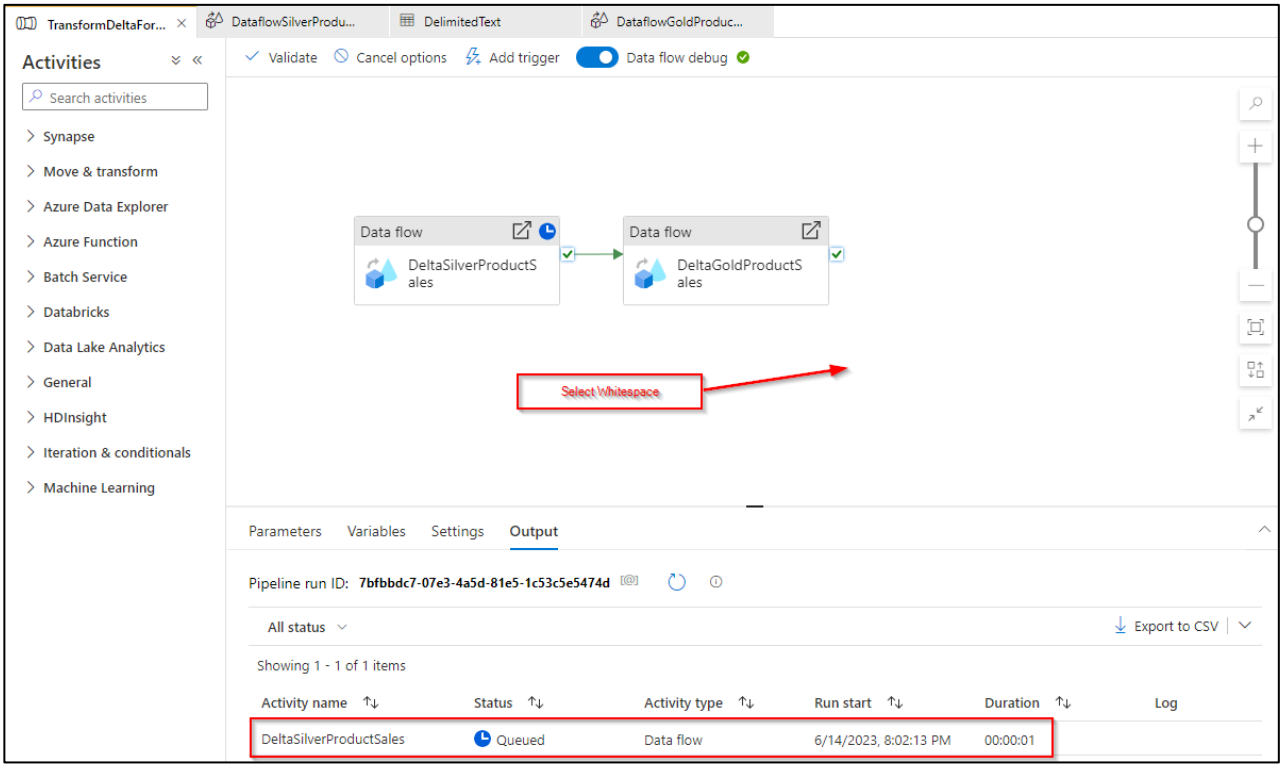
5. Select the **Debug** button to run the **TransformDeltaFormat** pipeline we configured and verify that the data flow is working.



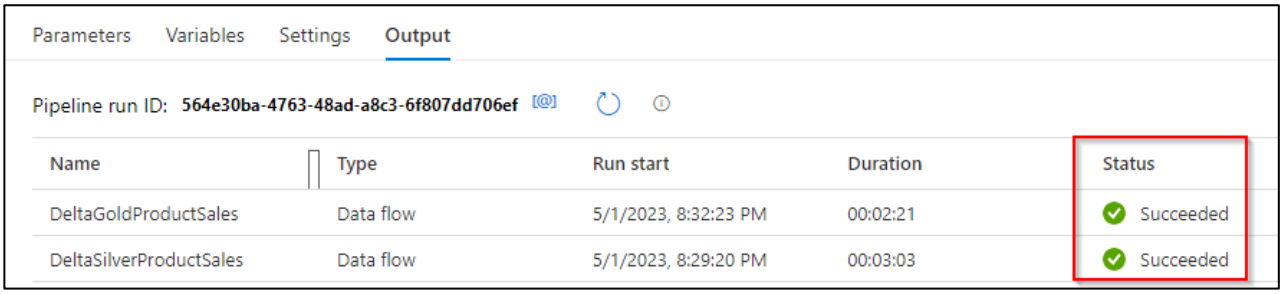
6. If prompted by the message **Use debug cluster?** Select **Use Integration Runtime.**



7. Once the Debug is selected, click on the whitespace of the **TransformDeltaFormat** canvas. You should now see the pipelines run.



8. The status of both data flows should change to Succeeded after approximately **3 - 5 minutes**.



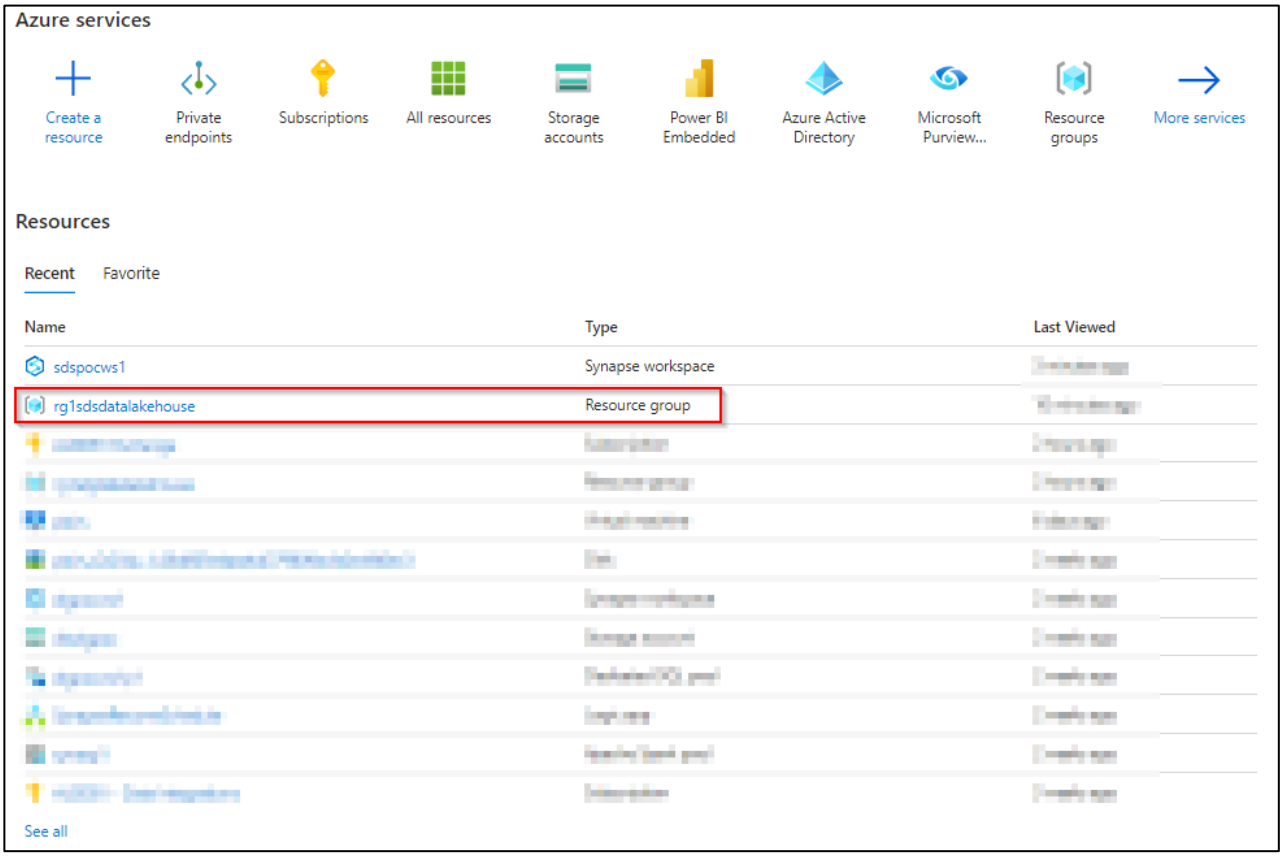
5.3 Data Changes

In real business scenarios, the data you upload will go through changes on a periodic or even aperiodic basis. To reflect those changes, we will make manual changes to our **productsales** data.

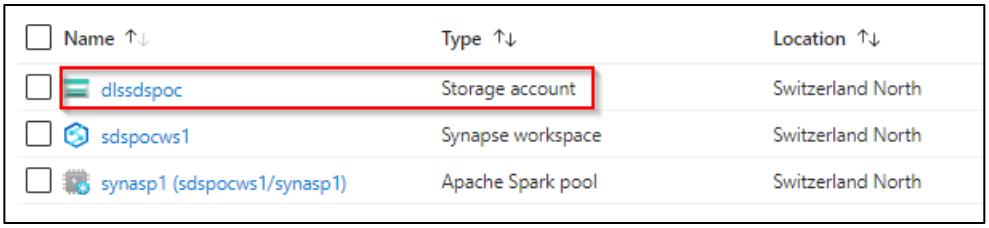
The delta format keeps track of these changes and allows us later to view these changes through audit logs and to time travel i.e., view different version of that same data.

For the sake of simplicity, we will create this change manually by overwriting the **FactProductSales.csv** file we uploaded in Create Directories and Load Data into the Storage Account with a new version containing additional rows.

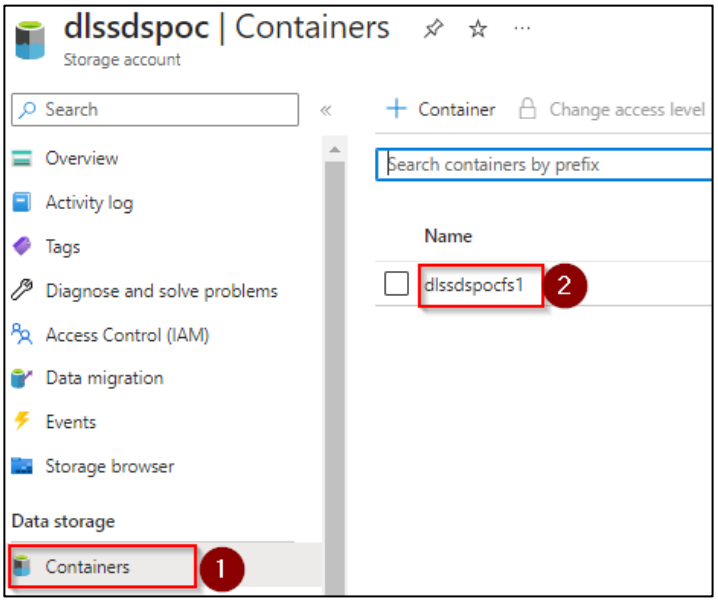
1. Go back to the Azure portal home screen and select the **Resource group** you provisioned.



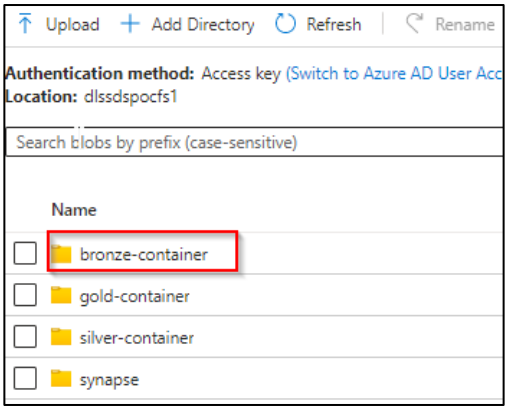
2. Once the resource group is open, select the **Storage account**.



3. Select the **Container** tab on the left side and select the named container.

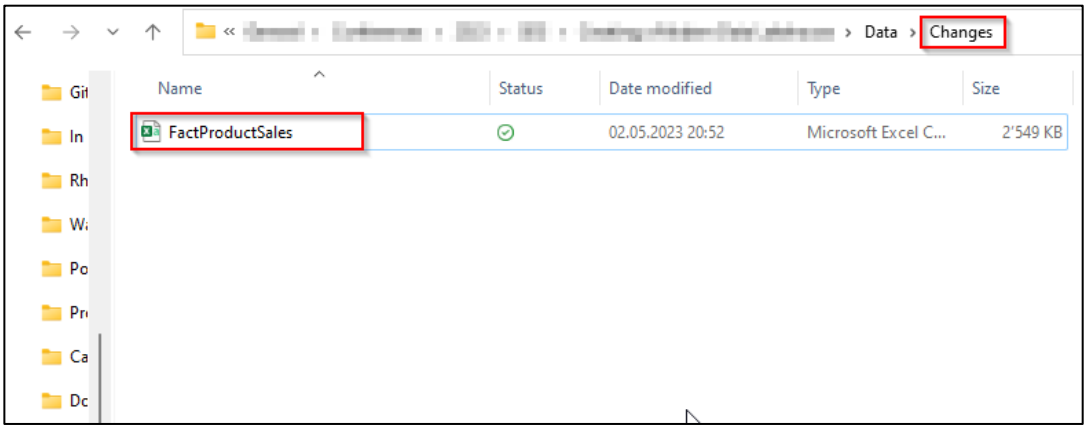


4. Within the named container, select the **bronze-container**.



5. Upload the **FactProductSales.csv** file which can be found in the **Changes** subfolder within the **Data** Folder.

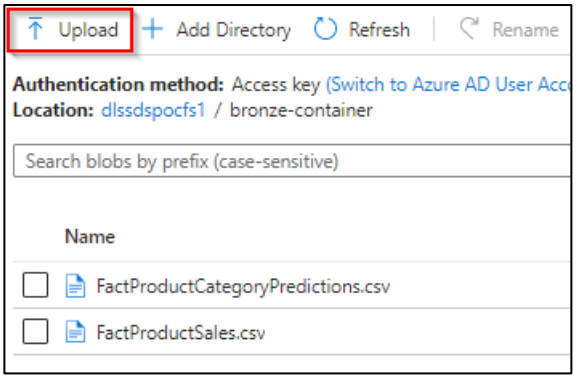
NOTE: The Support Files can be found in the Git repository in this link. Download these files onto your local computer.



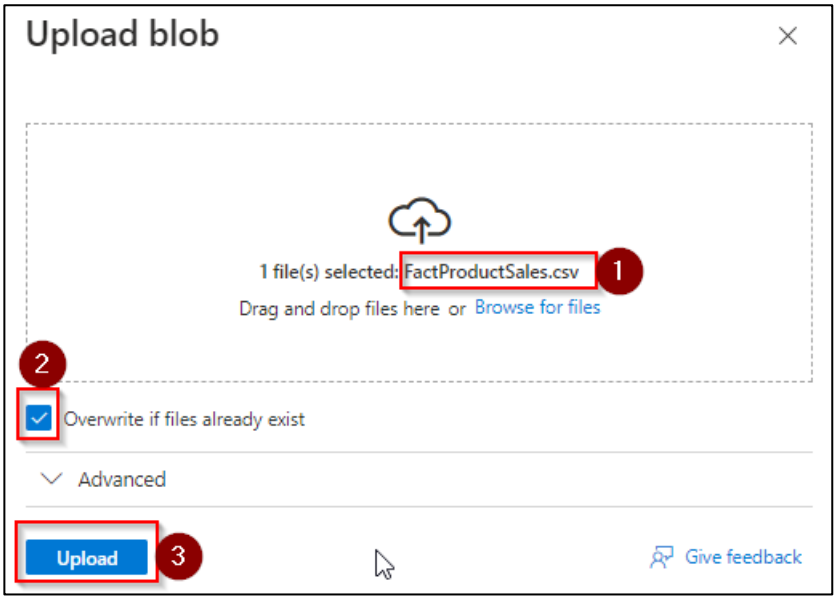
The changed **FactProductSales.csv** file contains additional rows.

	A	B	C	D	E	F
1	ProductKey	ProductName	ProductNew	OrderQuantity	UnitPrice	OrderDate
57574	486	All-Purpose Bike Stand		1	159	02.05.2023
57575	486	All-Purpose Bike Stand		1	159	02.05.2023
57576	537	HL Mountain Tire		1	35	02.05.2023
57577	225	AWC Logo Cap		1	8.99	03.05.2023
57578	485	Fender Set - Mountain		1	21.98	03.05.2023
57579	485	Fender Set - Mountain		1	21.98	04.05.2023
57580	360	Mountain-200 Black, 42		1	2049.0982	04.05.2023
57581	360	Mountain-200 Black, 42		1	2049.0982	04.05.2023

6. Within the bronze-container, select the **Upload** button.



7. A new panel will open to the right. **Drag and Drop** the **FactProductSales.csv** file from the **Changes** folder, toggle the **Overwrite if files already exist** and select **Upload**.

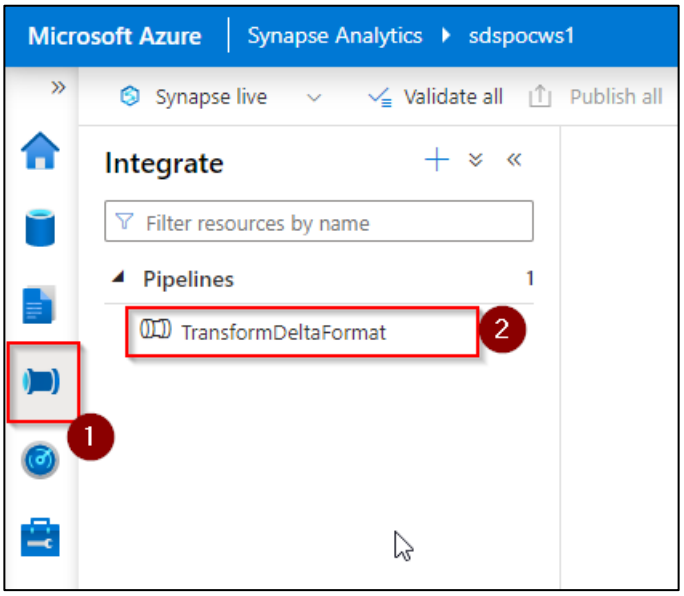


You should get a successfully uploaded banner:

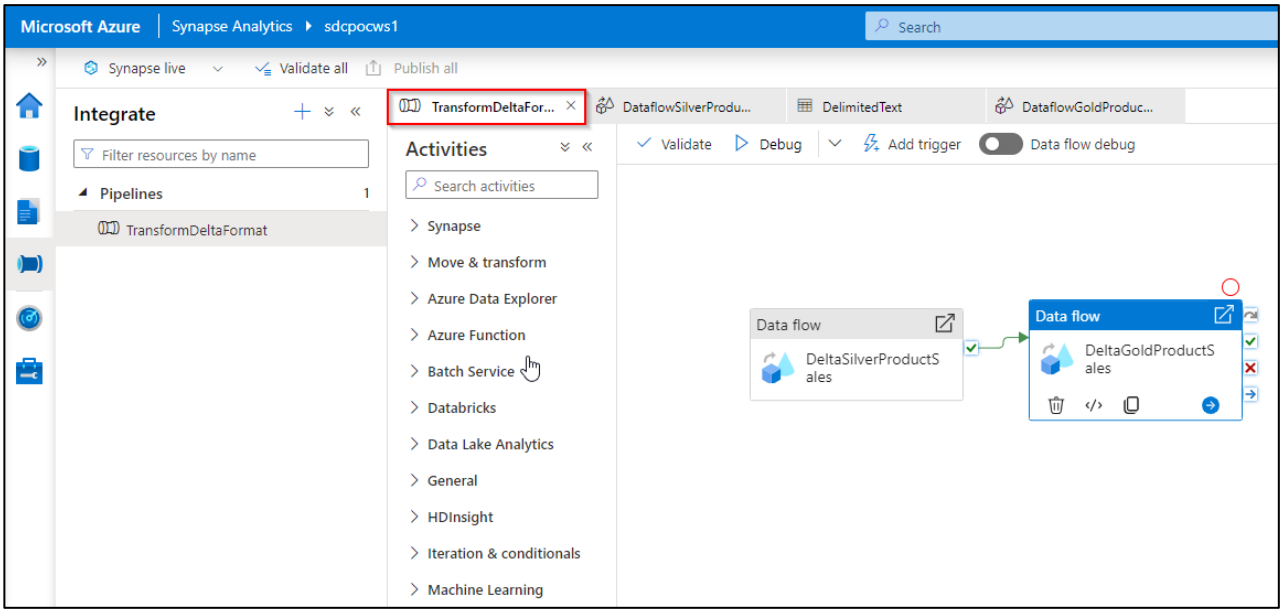
5.4 Manually Trigger Pipelines (After Data Changes)

We uploaded our changed data into the storage in chapter Data Changes. Let's trigger our synapse pipelines to extract, transform and load these changes into our silver and gold-containers.

1. Go back to your Azure Synapse Workspace and select the **Integrate** tab and select the pipeline **TransformDeltaFormat**.

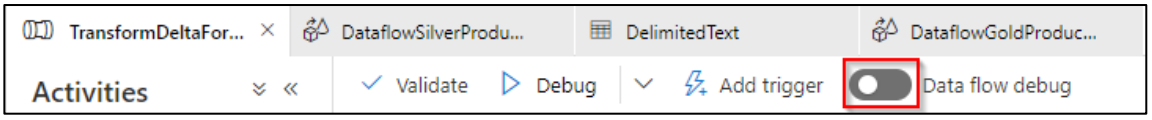


2. Once the pipeline is selected, a new tab with the dataflow activities we created earlier will appear.

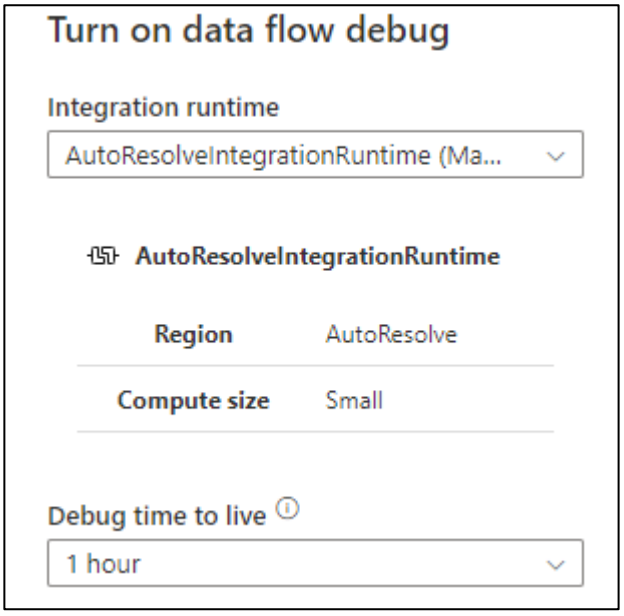


3. Select the **Data flow debug** toggle button, to initiate the integration runtime.

NOTE: If the debug is still on [green], you may continue directly on step 5.

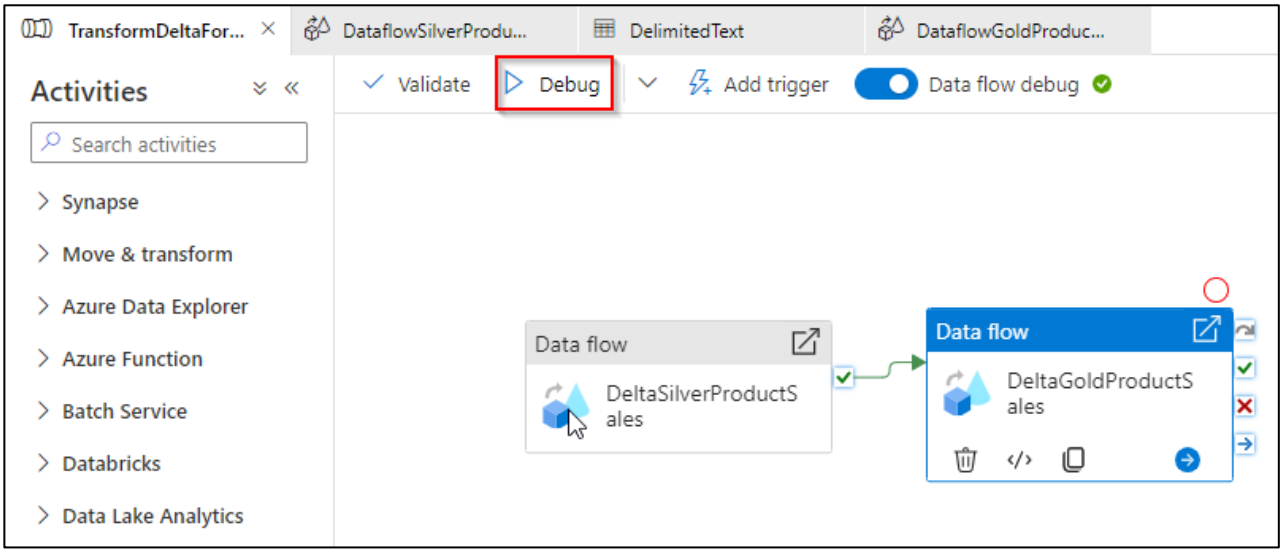


4. Once toggled, a new panel will open to the right, with the debug setting. Leave the default settings and select **OK**.

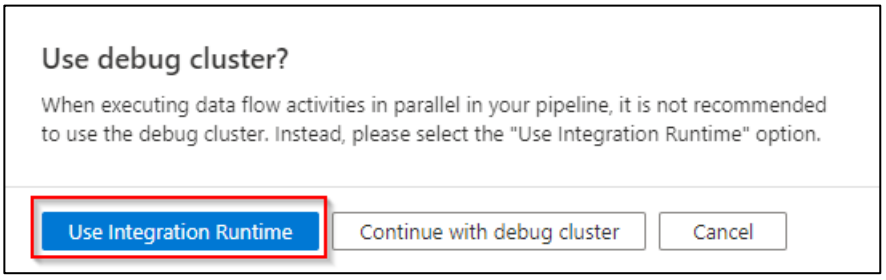


NOTE: The debug initialization might take 3 – 5 minutes. Please be patient!

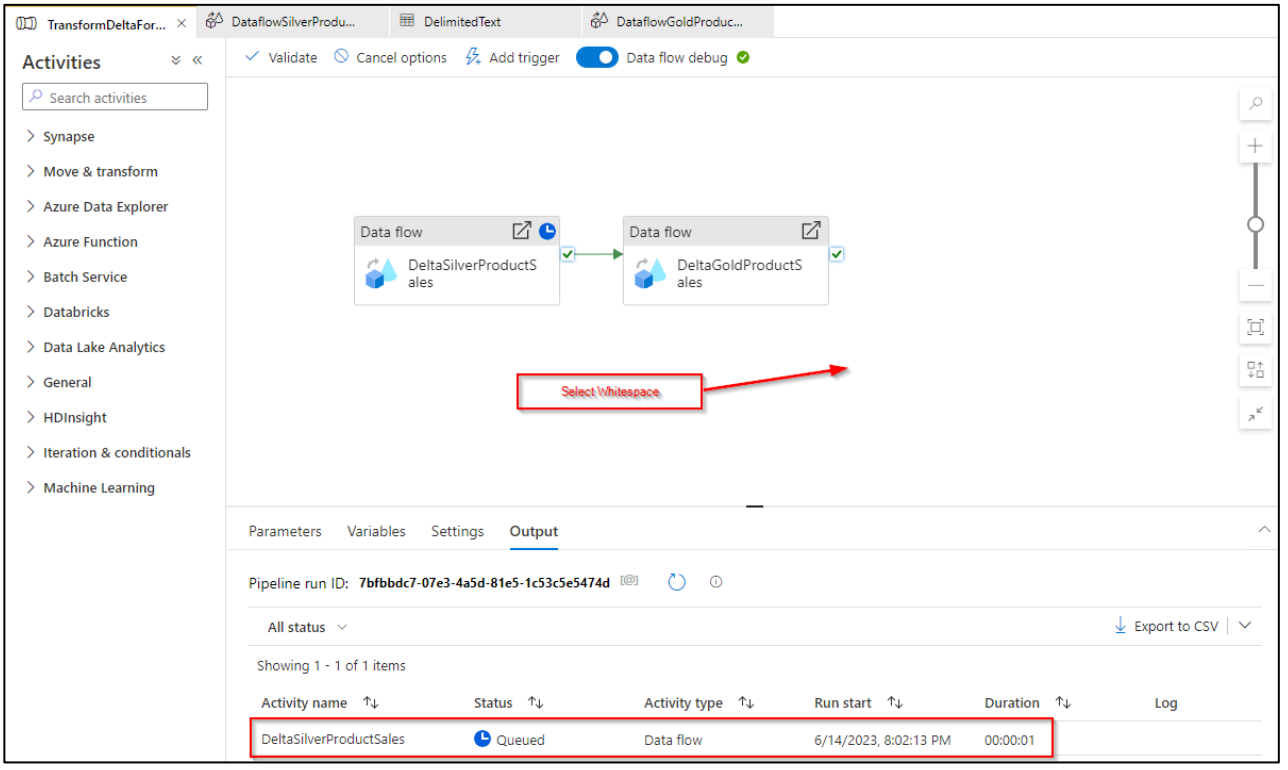
5. Select the **Debug** button to run the pipeline we created and verify that the data flow is working.



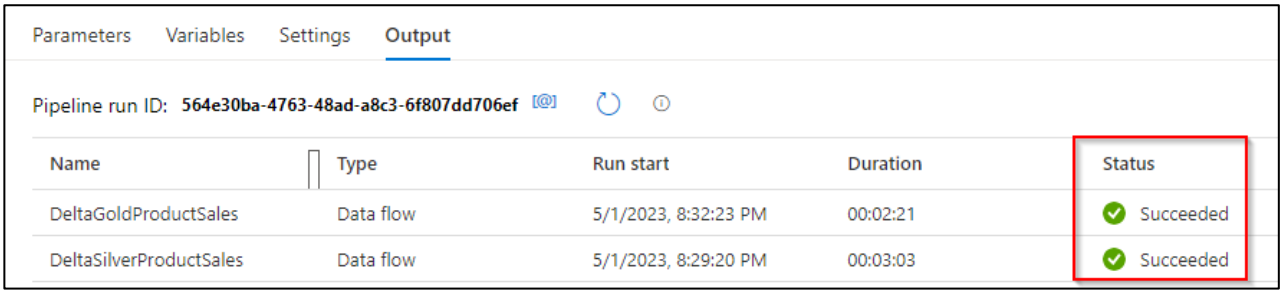
6. If prompted by the message **Use debug cluster?** Select **Use Integration Runtime.**



7. Once the Debug is selected, click on the whitespace of the **TransformDeltaFormat** canvas. You should now see the pipelines run.



8. The status of both data flows should change to Succeeded after approximately **3 - 7 minutes**.



6 Create Azure Synapse SQL Database (OPTIONAL)

This chapter is **optional**. We advise you to do this chapter after the workshop and continue the workshop on chapter Create Azure Synapse Lake Database.

The serverless SQL databases are like that of a traditional Azure SQL Database/Dedicated Pool (formerly SQL DW). The primary difference is that the serverless SQL database queries over data stored in a data lake, rather than stored on disk on the SQL Server machine. This is also the primary feature that makes the data lakehouse architecture possible.

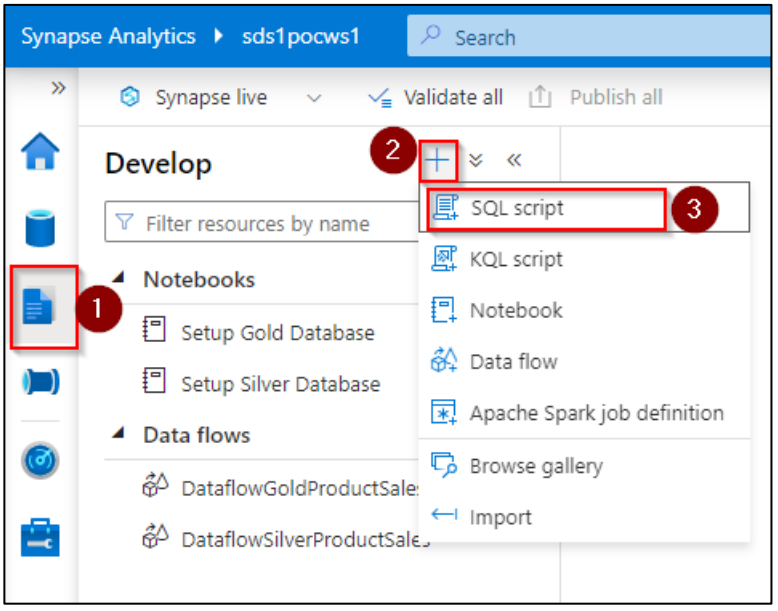
6.1 Create a SQL Script

Before we create the delta tables based off the delta formatted data in the silver-container and the gold-container, we can also create a SQL Database to query the raw CSV data in the bronze-container using a view with the serverless SQL pool.

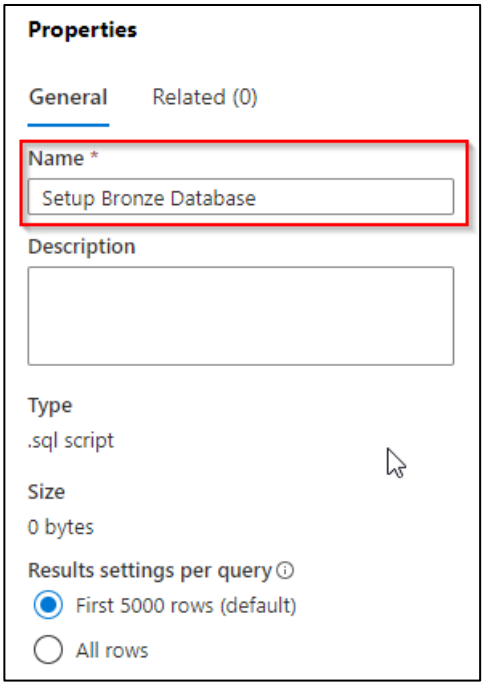
NOTE: Creating views or external tables from the raw data is usually not necessary, since they can be queried directly from the Azure storage using Serverless SQL pools as described in chapter Query Delta Formatted Data with Serverless SQL Pool (OPTIONAL).

6.1.1 Bronze Database

1. Select the **Develop** tab to the left, select the **+** button and create a new SQL script.



2. Rename the newly created SQL script to **Setup Bronze Database**.

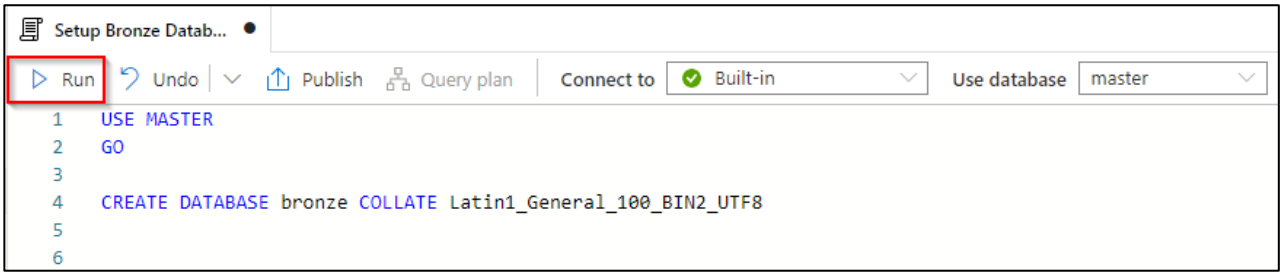


You will start by creating a bronze database using SQL.

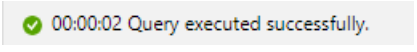
3. Within the SQL Script **Setup Bronze Database** field, add the following SQL code to create the database:

```
USE MASTER
GO

CREATE DATABASE bronze COLLATE Latin1_General_100_BIN2_UTF8
```

4. Select the Run button to run the script.



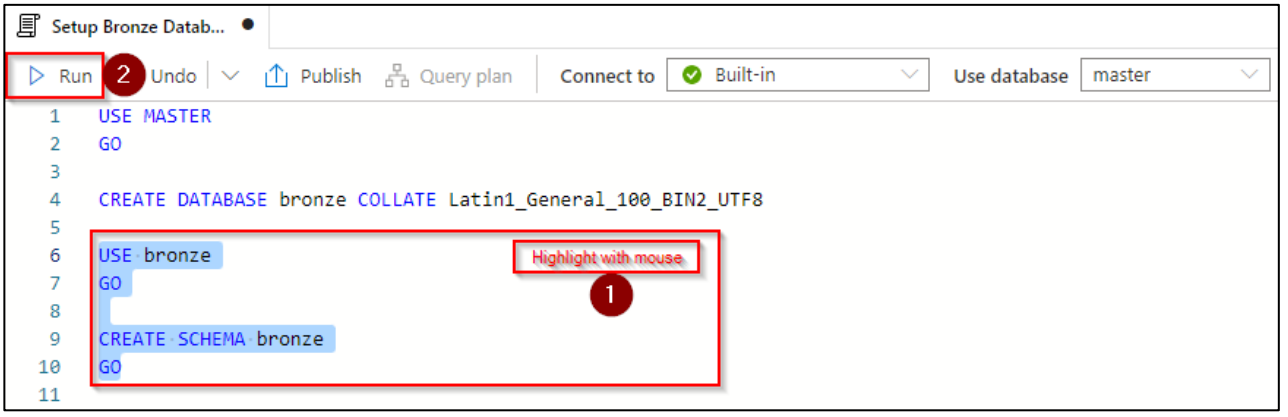
You will get a notification on the bottom that the **Query executed successfully**.



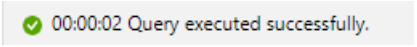
5. Below the SQL code you added in step 3, add the following SQL code to create a schema for your bronze database:

```
USE bronze
GO

CREATE SCHEMA bronze
GO
```

6. Highlight this new portion of SQL code and select the run statement.



You will get a notification on the bottom that the **Query executed successfully**.



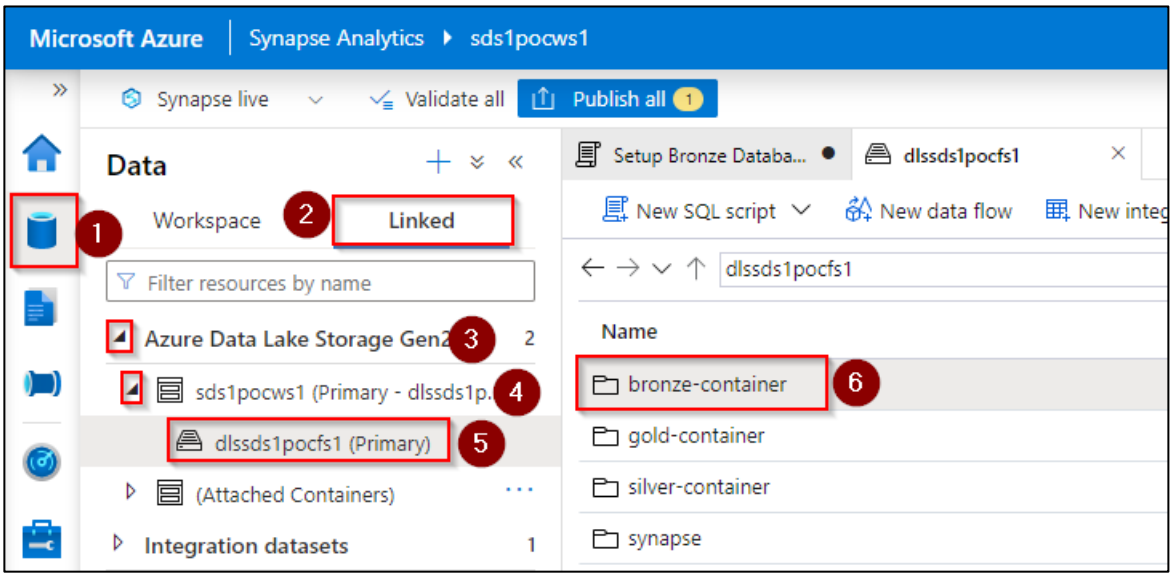
7. Below the SQL code you added in step 5 add the following SQL code to create a view of your raw data:

NOTE: The BULK command below needs to be adjusted. See Step 8 to find the correct file path.

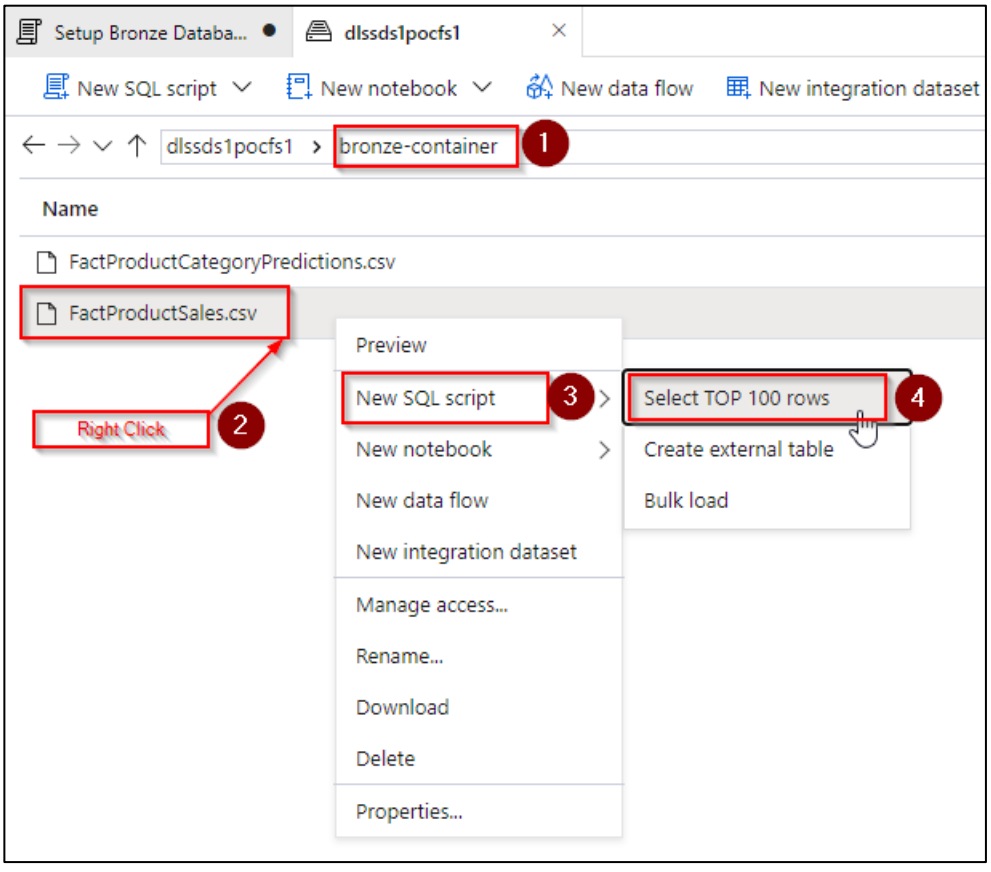
```
CREATE OR ALTER VIEW bronze.productsales AS
SELECT *
FROM
    OPENROWSET(
        BULK 'see step 8 to find correct path,
        FORMAT = 'CSV',
        PARSER_VERSION = '2.0',
        FIELDTERMINATOR =';',
        FIRSTROW = 1,
        HEADER_ROW = TRUE
    ) AS [result]
```

Before running the SQL code, we need to make sure the **BULK** command looks at the correct file path.

8. Select the **Data** tab to the left, select **Linked** and open your **primary data storage** i.e., the storage provisioned in chapter Create Resource Group and Resources.
9. Select the **named container** of this storage.
10. Select the **bronze-container.**

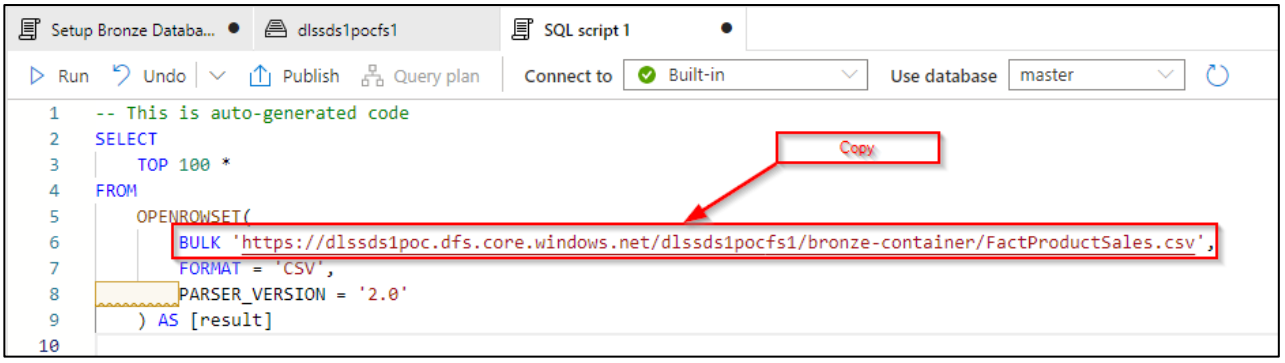


11. Within the **bronze-container**, right click the **FactproductSales.csv** file, select **New SQL Script** and select **Select TOP 100 rows**.

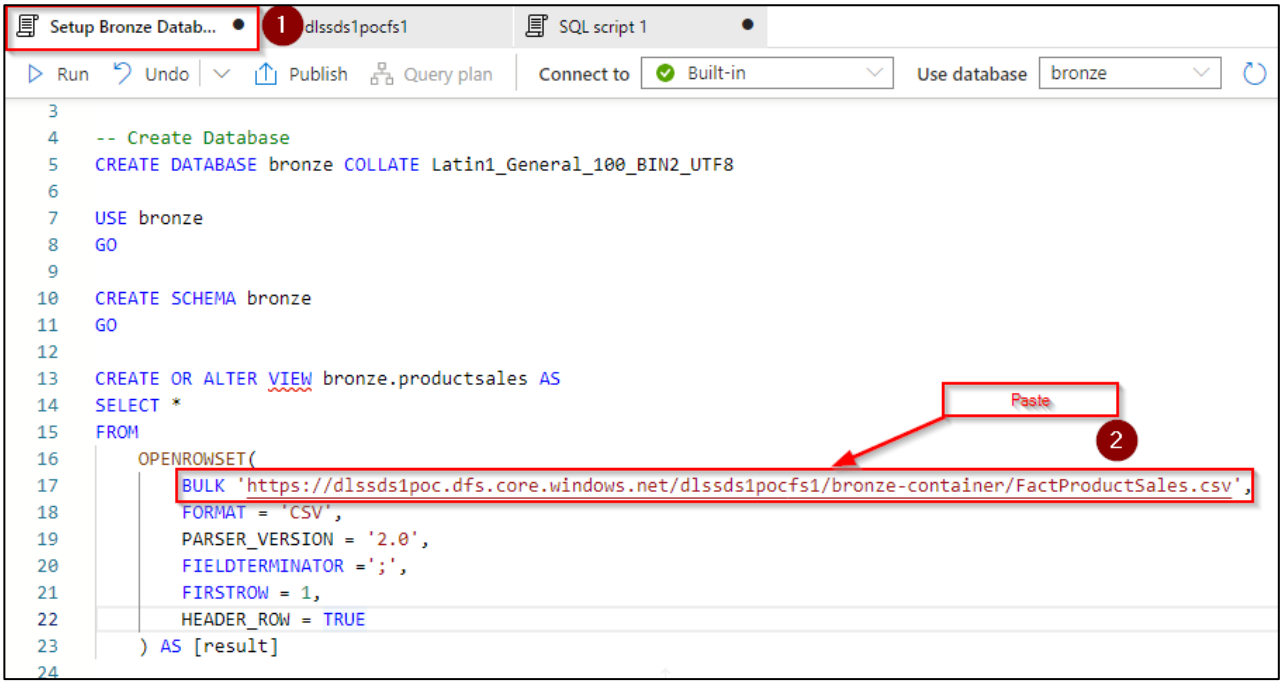


This will create a SQL statement for you, with the required OPENROWSET command to call the file.

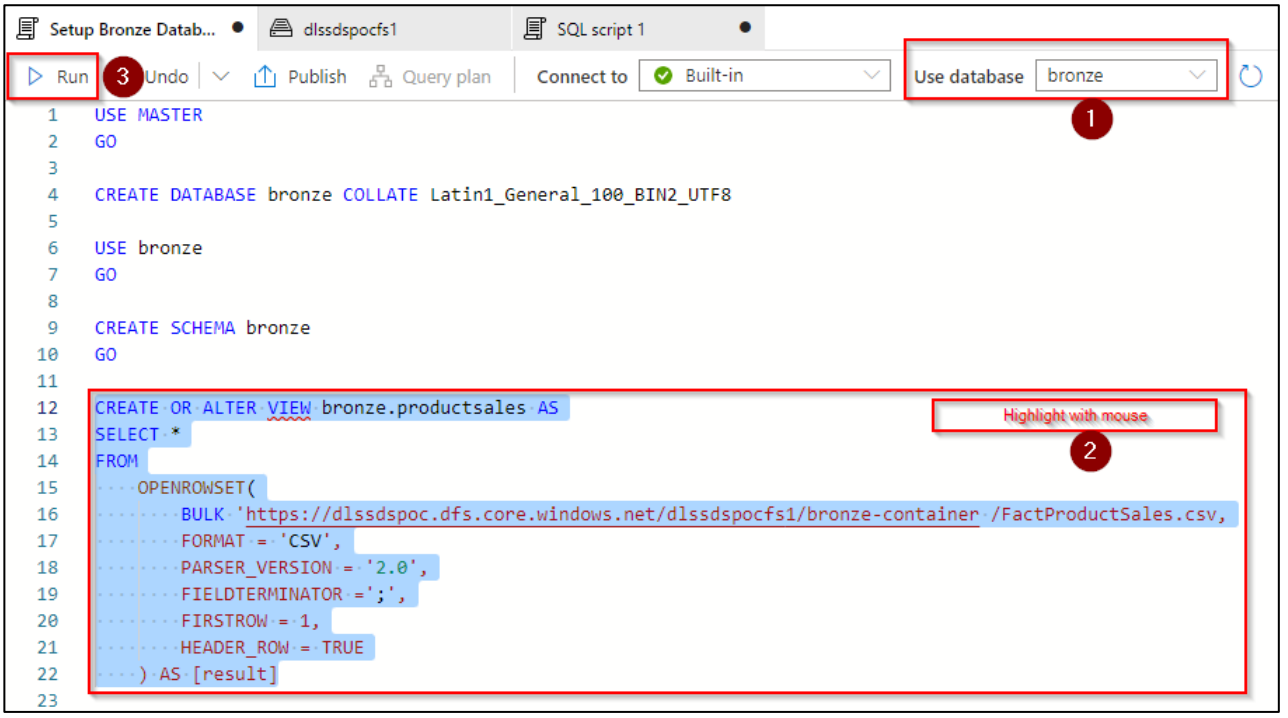
12. Copy the **BULK** statement.



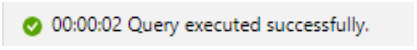
13. Go back to the **Setup Bronze Database** SQL Script by selecting the **Setup Bronze Database** tab.
14. **Paste** the copied BULK statement into the SQL code from step 7.



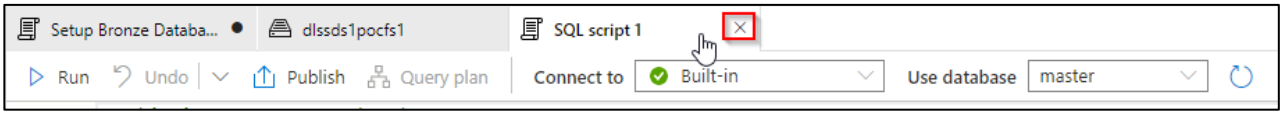
15. Make sure that the **Use database** is set to **bronze** (this should happen automatically when you run the script in step 6)
16. Highlight this new portion of SQL code you added in step 7 and select **Run**.



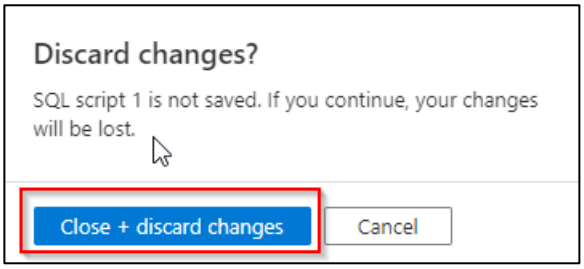
You will get a notification on the bottom that the **Query executed successfully**.



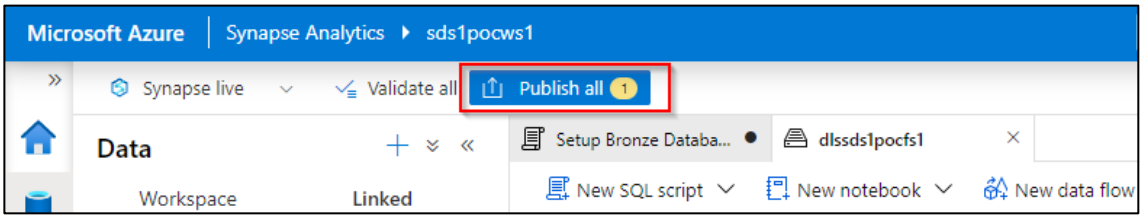
17. You may now close the tab **SQL script 1** which you created in step 11 when querying the TOP 100 rows.



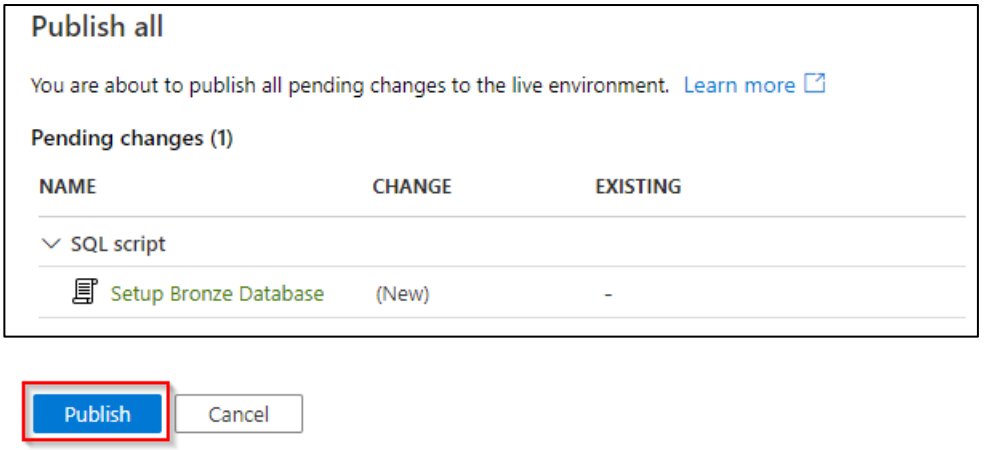
18. Select **Close + discard changes** to confirm the deletion.



19. Select the **Publish all** button to save the work you have done so far.



20. A new panel should open to the right. Select the **Publish** button.



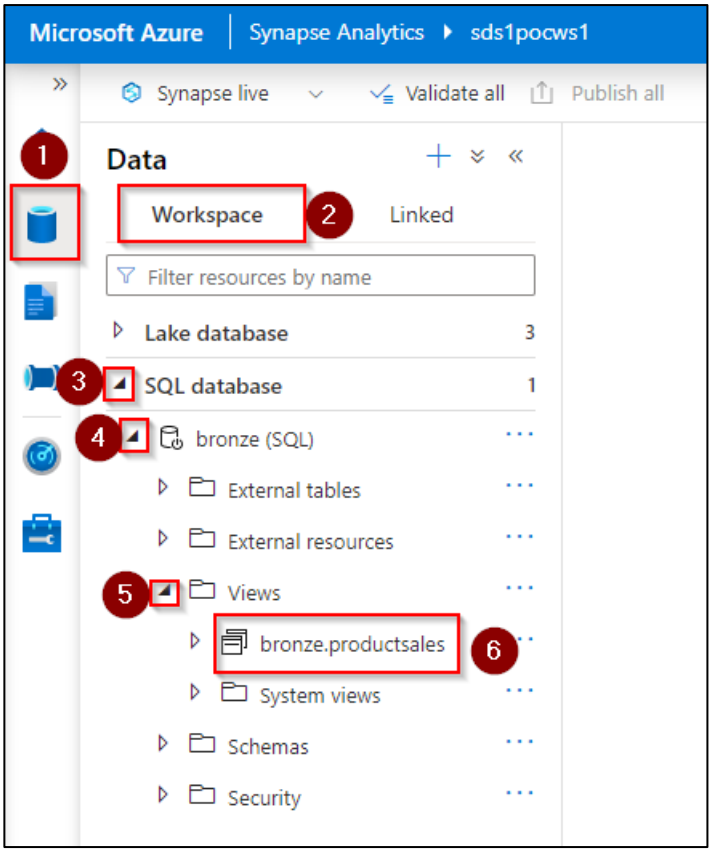
A banner should appear confirming that the content was published.

6.1.1.1 Query View with SQL Serverless Pool

We will now query the view we created from the raw file **FactproductSales.csv.**

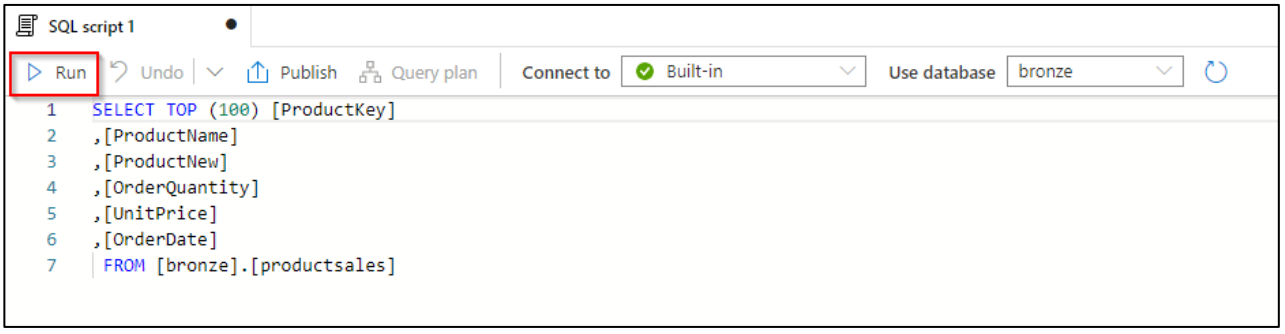
NOTE: You may need to refresh your browser to view the newly created SQL database in the Data tab. Make sure you publish everything and save your work before refreshing the browser. See Step 19.

21. Select the **Data** tab to the left, select **Workspace** and browse to the **Views** folder. Within the Views folder you will see the **bronze.productsales** view we created earlier.

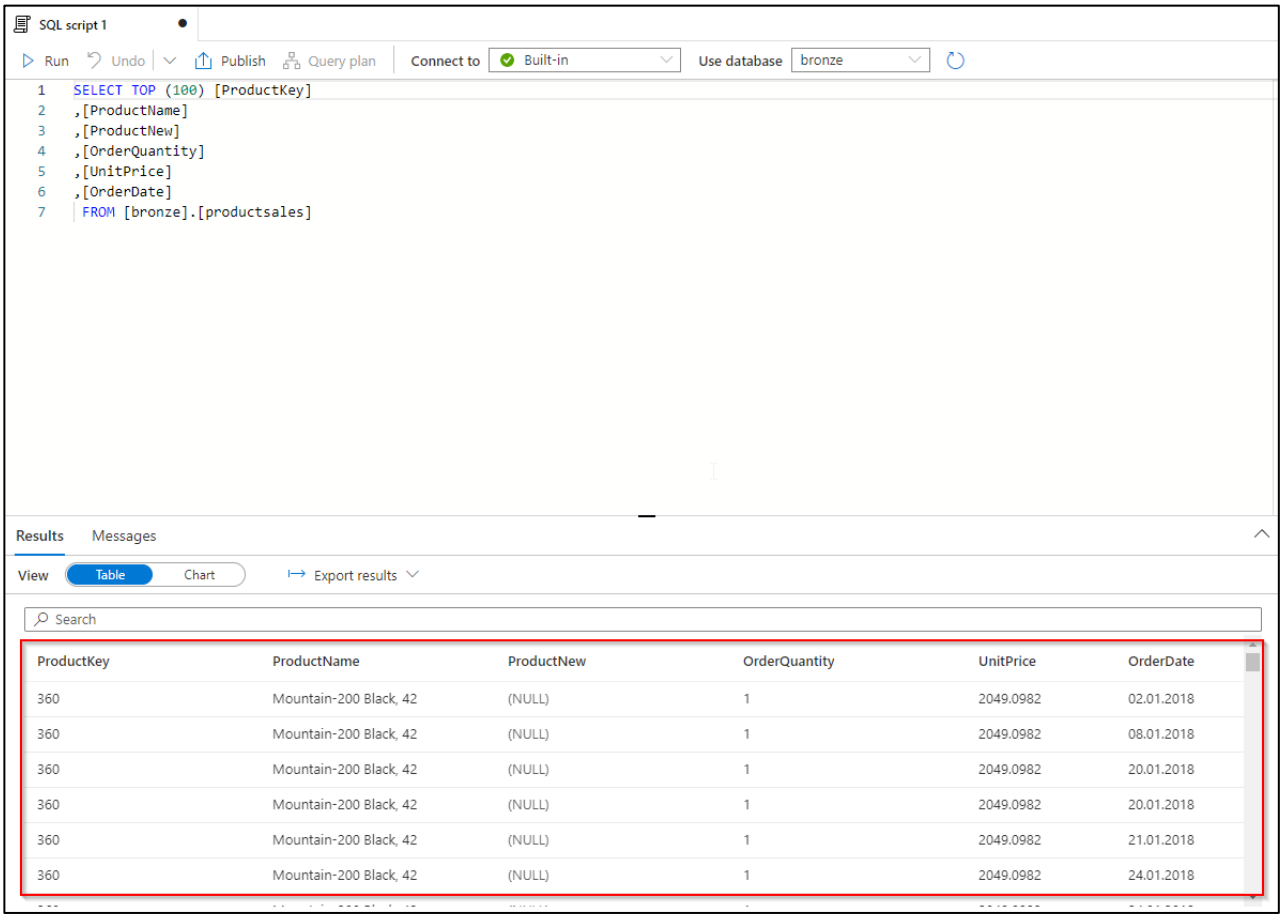


22. Right click the **bronze.productsales** and select New SQL script, followed by Select TOP 100 rows. This will create a new tab with a SQL statement to query the view.

23. Select the **Run** button to run the SQL statement.



You will now see the data related to the **FactproductSales.csv** file.



You have successfully queried a view with data directly from a data lake using a serverless SQL pool.

7 Create Azure Synapse Lake Database

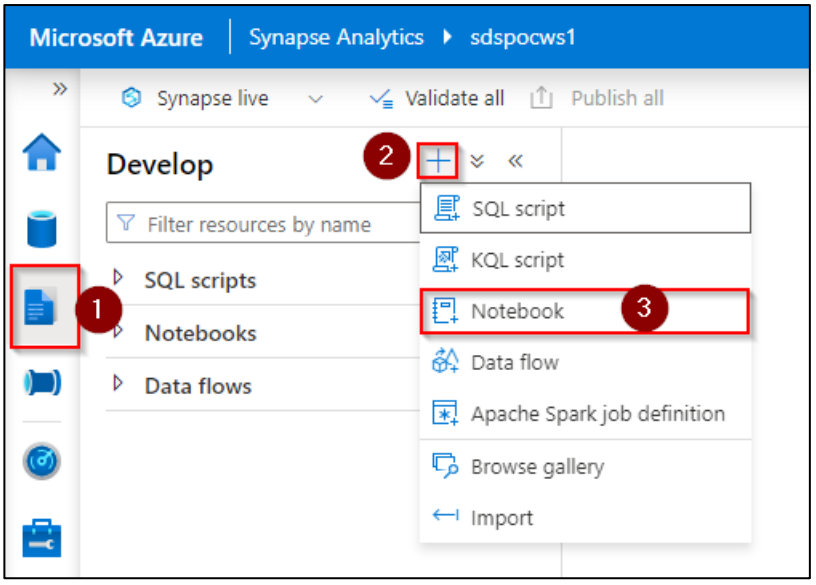
The Lake Database is a new way of defining data structures on data that is hosted on Azure Data Lake Storage and processed with Azure Synapse. This Database type is synchronized between the Spark and the SQL Serverless engines in Azure Synapse and allows interoperability between the different compute engines (Spark and SQL). It is within this database type that you can create and query delta formatted data using delta tables.

7.1 Create a Spark Notebook

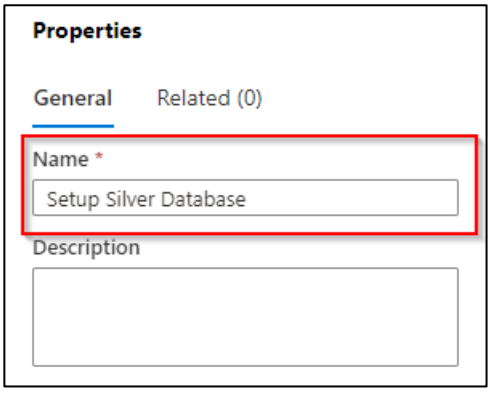
We can now create delta tables based off the delta formatted data in the silver-container and the gold-container using the Apache Spark Pool.

7.1.1 Silver Database

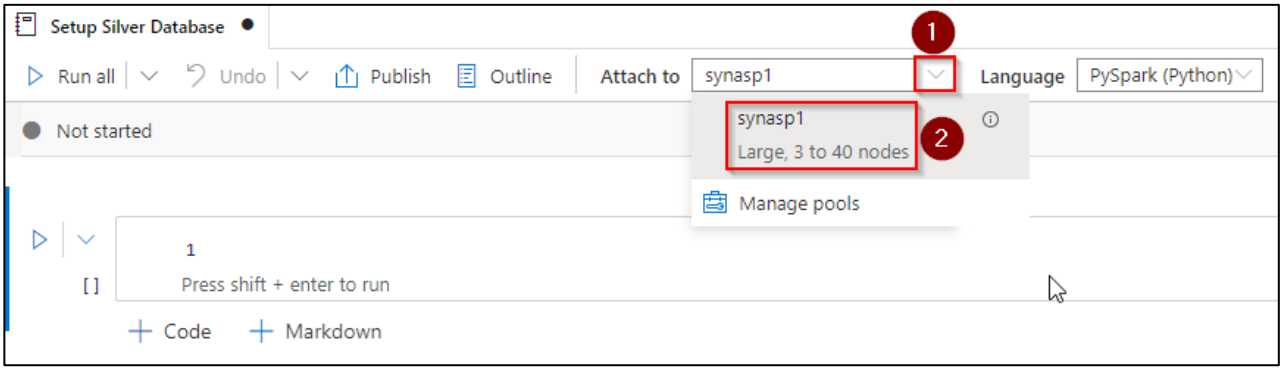
24. Select the **Develop** tab to the left, select the **+** button and create a new Notebook.



25. Rename the newly created Notebook to **Setup Silver Database**.



26. To begin using this new Notebook, attach an Apache Spark Pool by selecting the **Attach to** dropdown. Select the provisioned spark pool (Large).



The Spark Notebooks allow you to run code using Spark flavors of SQL, Python, R and Scala programming languages. In this example, we will focus on SQL and Python.

7.1.1.1 Lake Databases

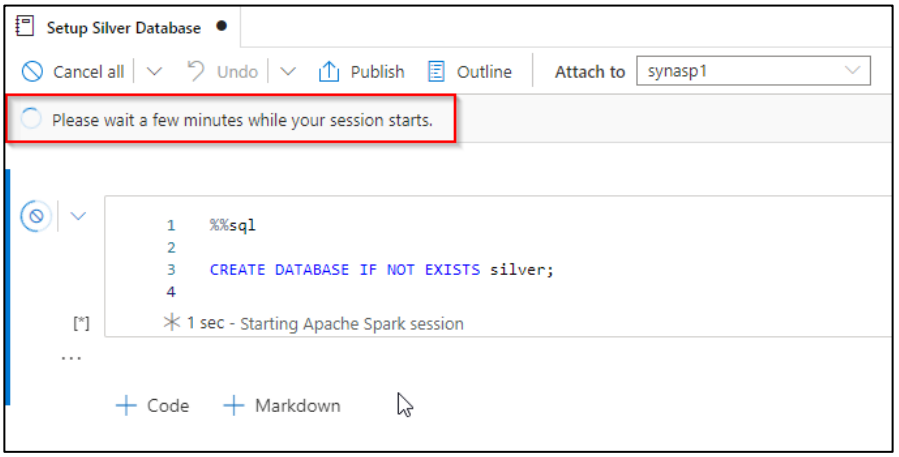
You will start by creating a silver database using Spark SQL.

27. Within the Notebook **Setup Silver Database** code field, add the following SQL code to create the databases:

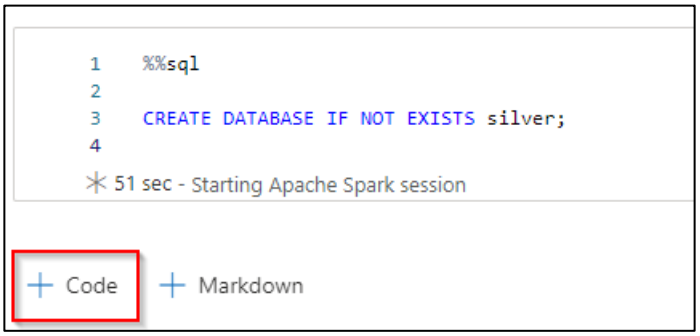
```
%%sql

CREATE DATABASE IF NOT EXISTS silver;
```

28. Select the Run button to start the Spark Pool and run the script.

NOTE: Starting the Apache Spark Pool may take 3 – 5 minutes. Please be patient!



29. Once the code has run, select the **+ Code** button, to add a new code field to the Notebook.



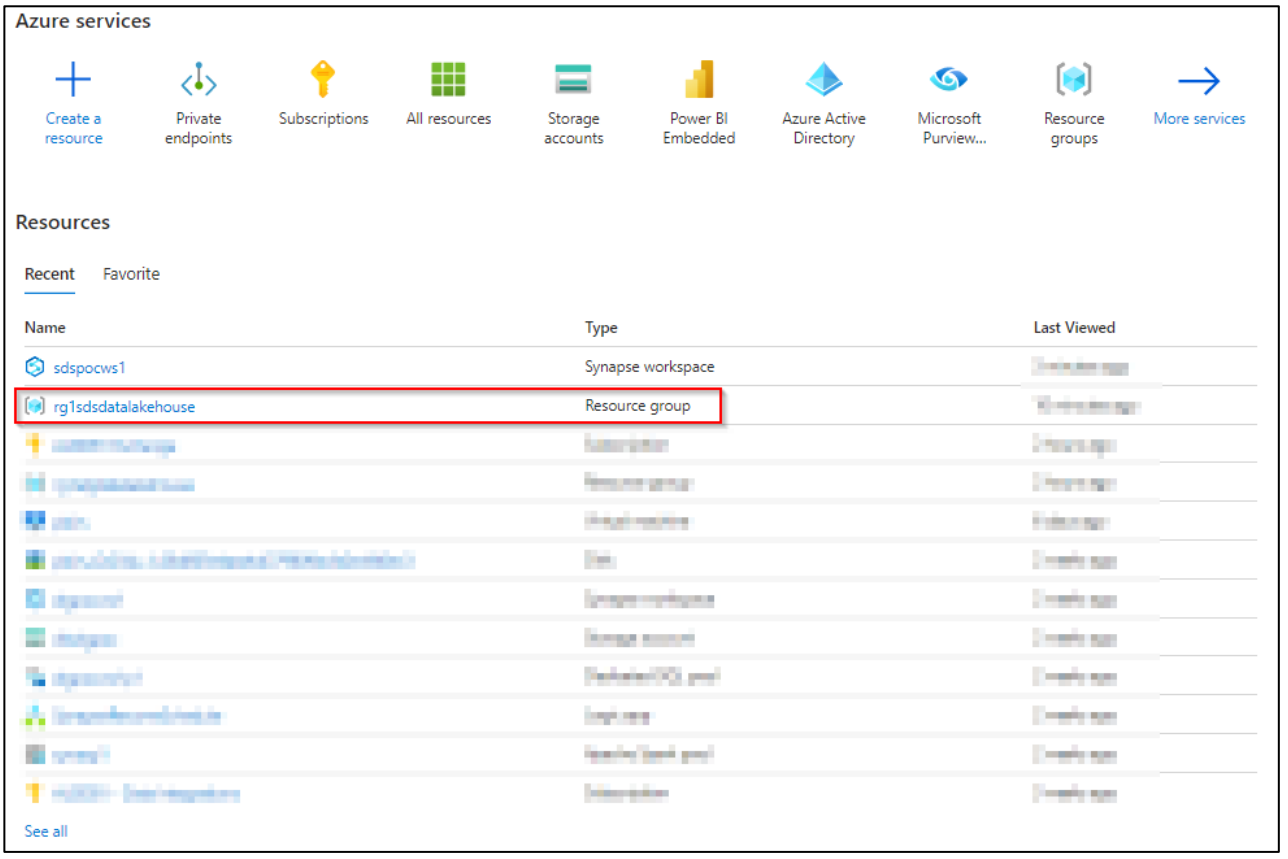
7.1.1.2 Azure Blob File System

To create a delta table, you will need the Azure Blob File System (ABFS) driver. This has the URI structure:

abfss://container_name@storage_account_name.dfs.core.windows.net/

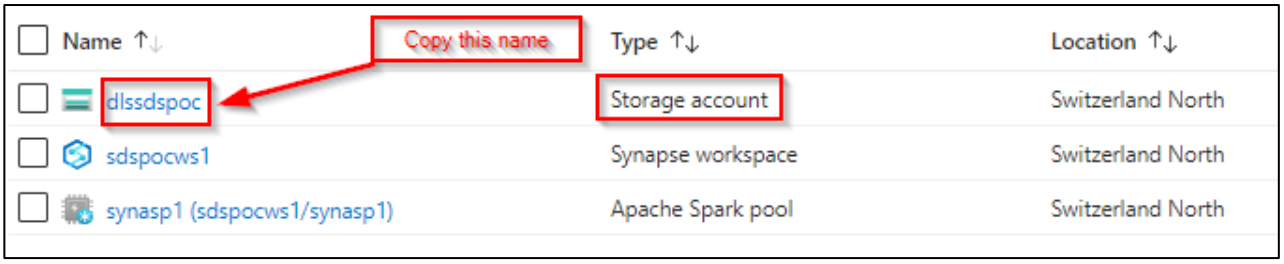
NOTE: Pay attention to the different colors in the URI above.

30. Go back to the Azure portal home screen and select the **Resource group** you provisioned.



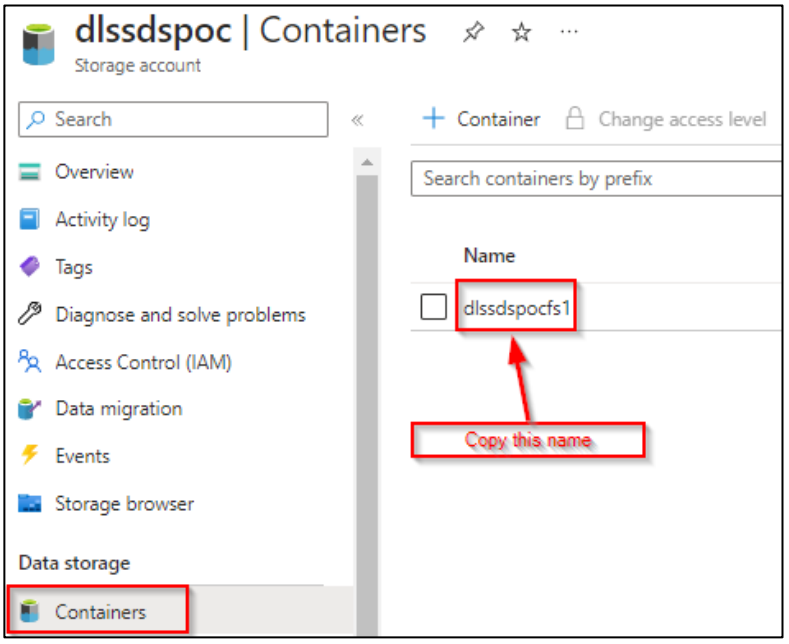
31. Once the resource group is open, find the **Storage account** and **copy** the storage name (in the example below, this would be **dlssdspoc**).

This is the **storage_account_name** from the URI.



32. Once the storage account name is copied, **select** the **Storage account** to open it**.**
33. Once the Storage account is open, select the **Container** tab and **copy** the **container name** (in the example below, this would be **dlssdspocfs1**).

This is the **container_name** from the URI.



You can now populate the URI; it should look like the one below:

- abfss://container_name@storage_account_name.dfs.core.windows.net/

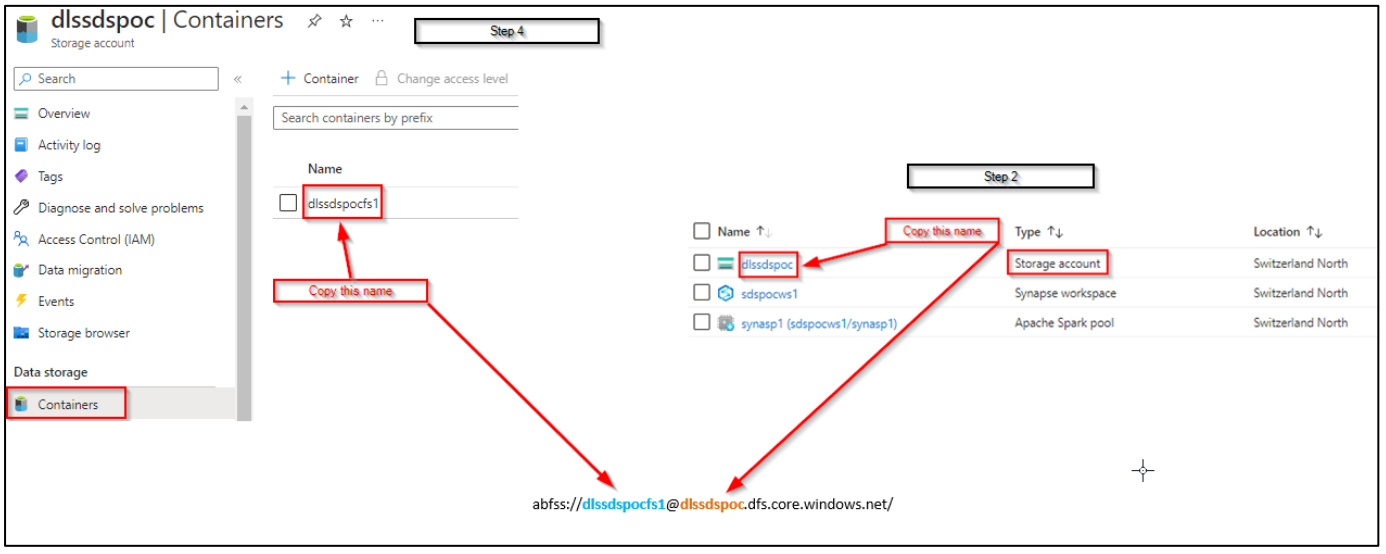
becomes in this example:

- abfss://dlssdspocfs1@dlssdspoc.dfs.core.windows.net/

This URI will be used in the **LOCATION** variable in the next code script.

NOTE: Do NOT copy the URI abfss://dlssdspocfs1@dlssdspoc.dfs.core.windows.net/ you see above. This will NOT work for you. You must input the names of the container and storage account you provisioned as described in chapter Azure Blob File System above.

Overview:



7.1.1.3 Lake Database Delta Tables

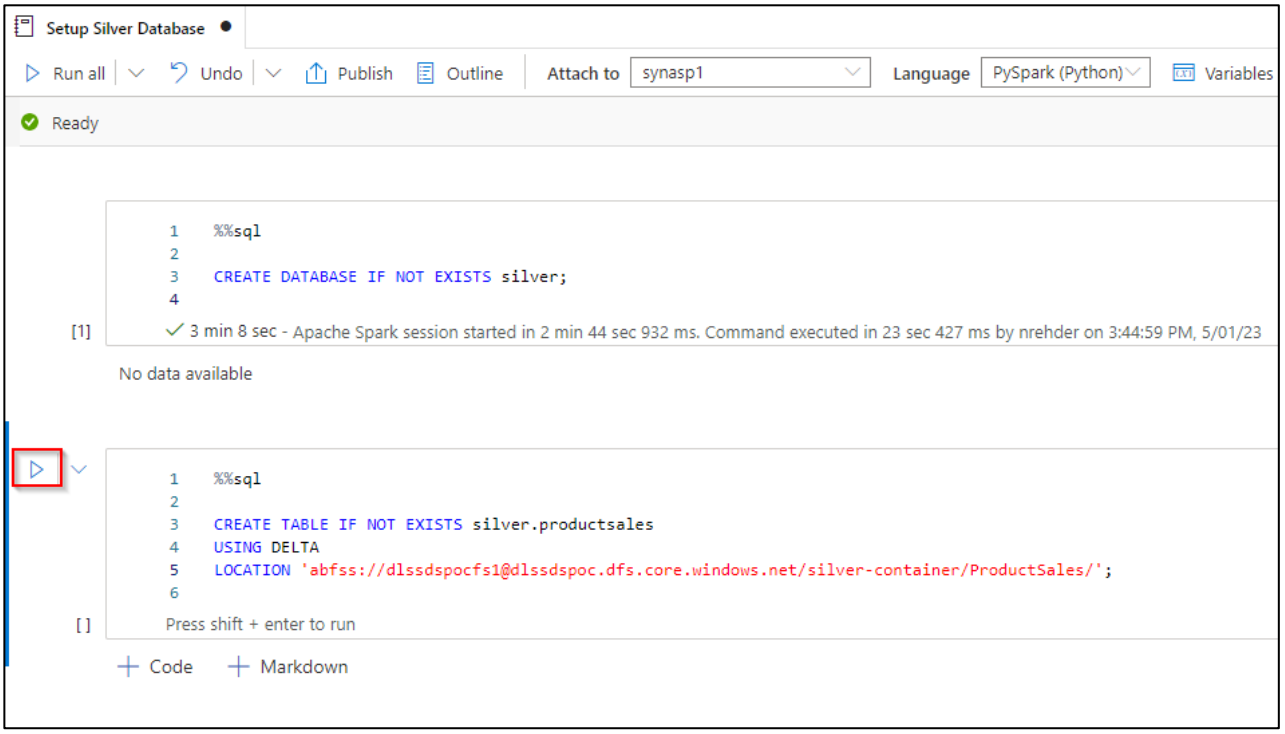
1. Go back to your Azure Synapse Workspace.
2. Within the Notebook **Setup Silver Database** code field, add the SQL code below to create the delta table.

```
%%sql

CREATE TABLE IF NOT EXISTS silver.productsales
USING DELTA
LOCATION 'abfss://dlssdspocfs1@dlssdspoc.dfs.core.windows.net/silver-
container/ProductSales/';
```

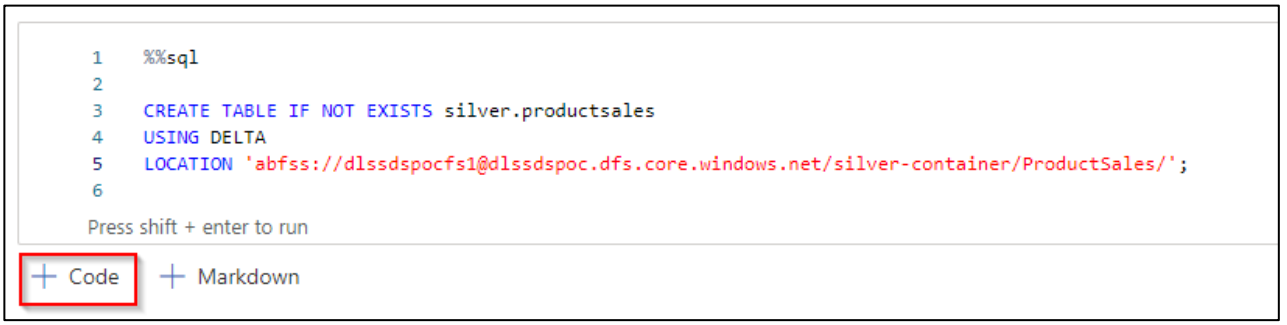
NOTE: The %%sql command explicitly tells the notebook to use the SQL language to run the code.

3. Change the **LOCATION** parameter within the SQL code to your URI as described in chapter Azure Blob File System.

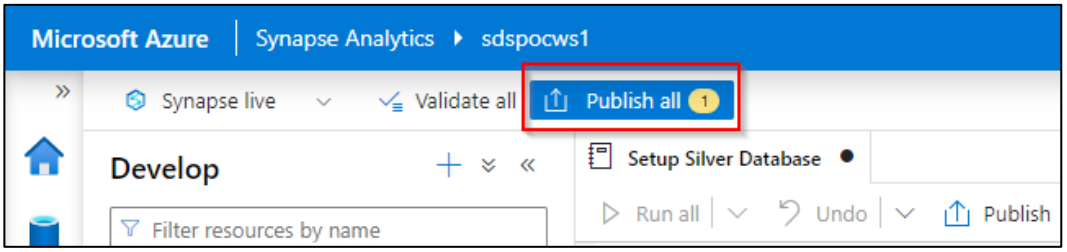
4. Select the Run button to run the script.



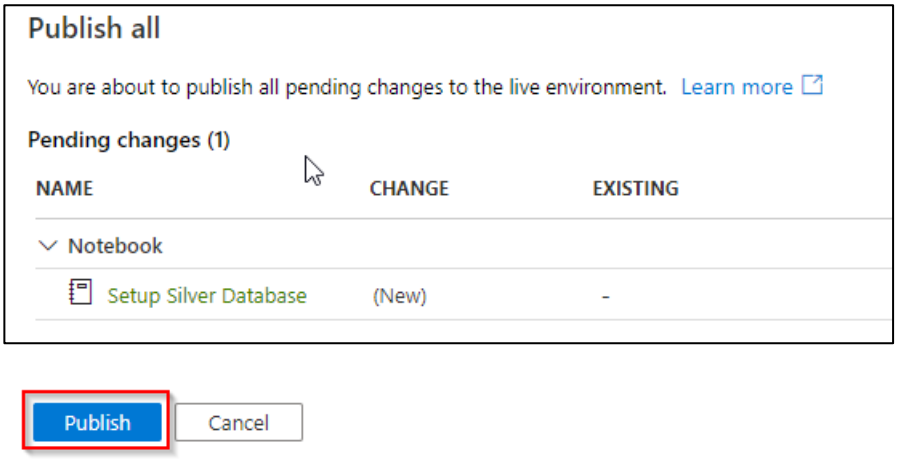
5. Once the code has run, select the **+ Code** button, to add a new code field to the Notebook.



6. Select **Publish** to save the new notebook.



7. A new panel should open to the right. Select the **Publish** button.



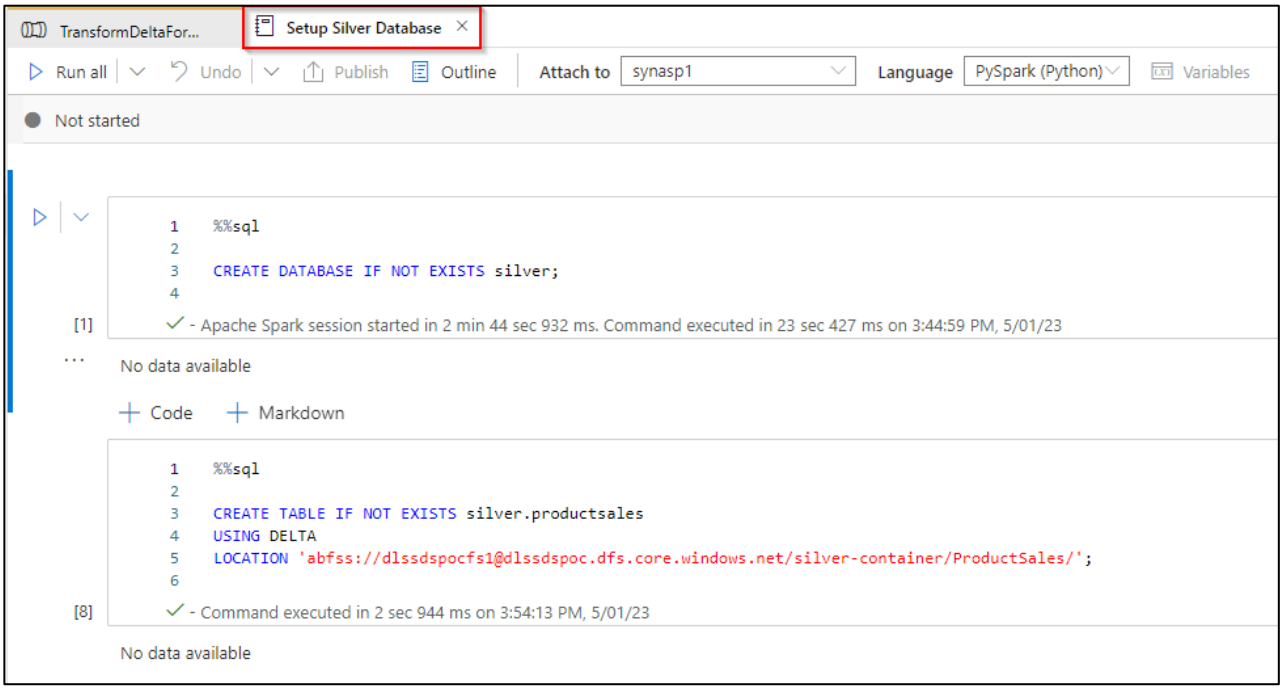
A banner will appear confirming that the content was published.

7.1.1.4 Audit Logs

Delta Lake format stores data in Parquet files and information regarding DML operations in the **_delta_log** metadata folder. With this metadata available you can view audit logs and time travel between different versions of a Delta table. Time travelling can be achieved by table version or by timestamp.

Now that the data has been changed, we want to be able to view different versions of this data i.e., time travel.

8. Go back to the **Setup Silver Database** tab. Once selected, you will see your Notebook.



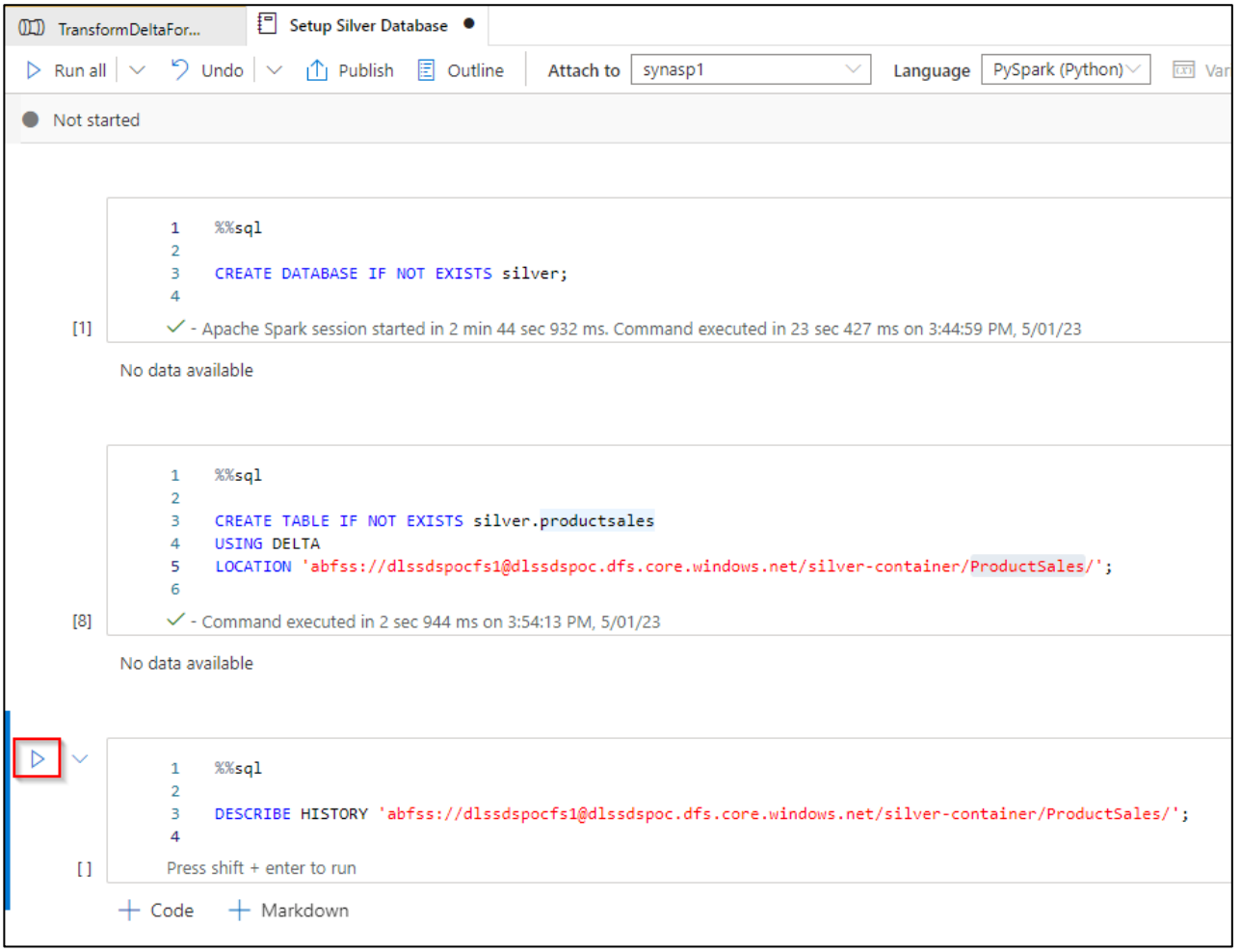
9. Within the Notebook **Setup Silver Database** code field, add the following SQL code to view the audit logs of your delta table. This gives you the change history of your data.

In the audit log you will see the operations we executed after manually changing the data in chapter Data Changes.

```
%%sql

DESCRIBE HISTORY 'abfss://dlssdspocfs1@dlssdspoc.dfs.core.windows.net/silver-container/ProductSales/';
```

10. Within the SQL code above change the **abfss URI** to your URI as described in chapter [Azure Blob File System](#).

11. Select the Run button to run the script.



You should see the Changes, specifically the operations that took place and the versions available as an output:

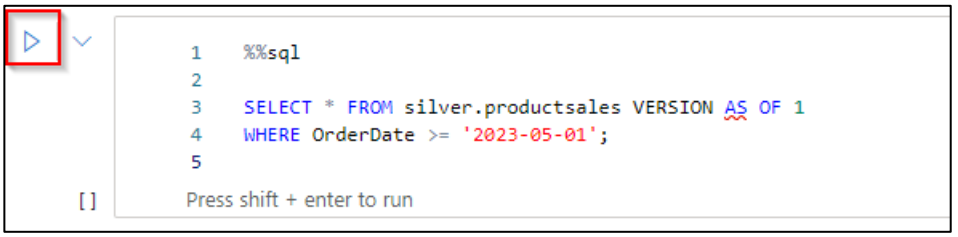
12. Once the code has run, select the **+ Code** button, to add a new code field to the Notebook.

7.1.1.5 Time Traveling

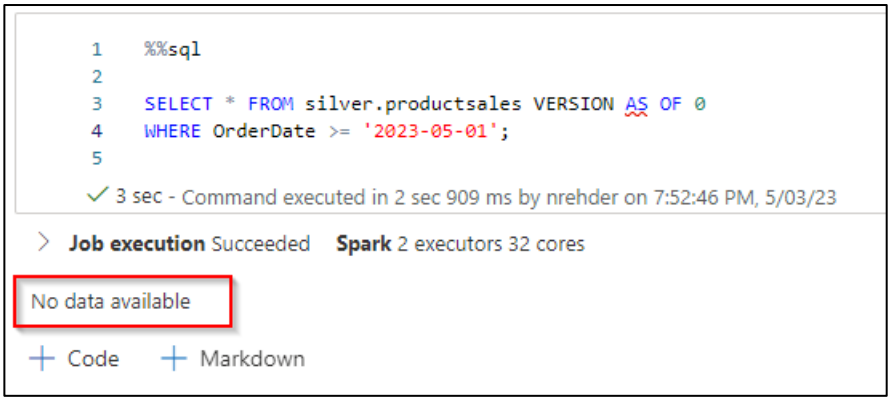
13. Add the following SQL code to view a previous version of your product sales data i.e. **version 0** or the initial state before the manual changes. For the sake of simplicity, we will filter the data.

```
%%sql

SELECT * FROM silver.productsales VERSION AS OF 0
WHERE OrderDate >= '2023-05-01';
```

14. Select the Run button to run the script.



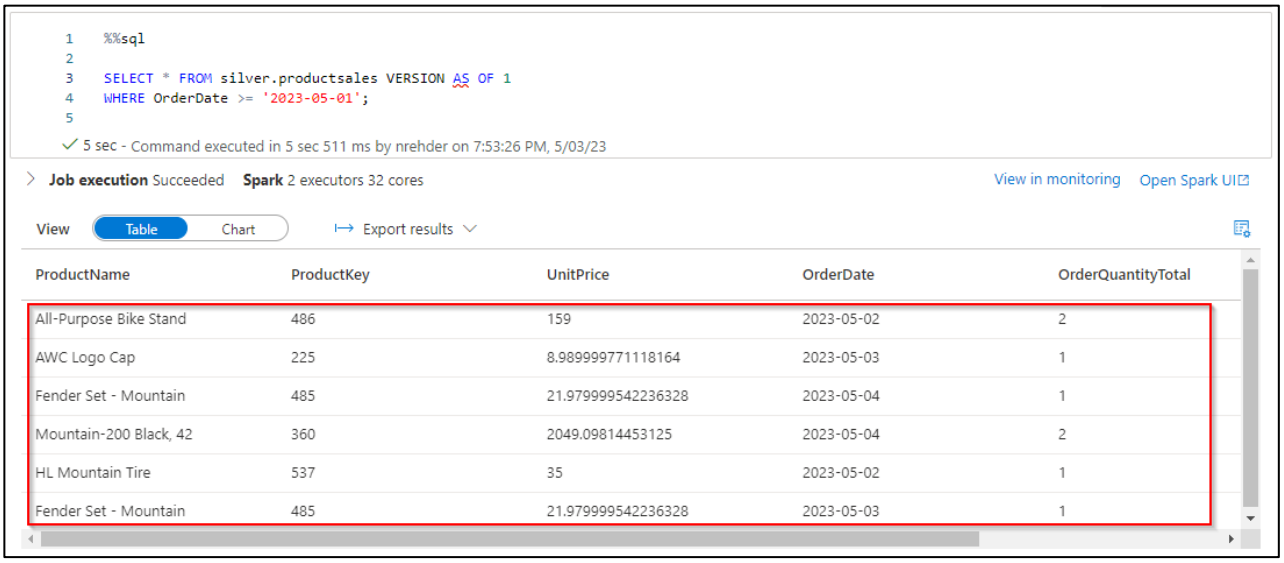
In **Version 0**, you will **not** see any values, since the previous version did not have any new OrderDate rows after May 01 2023.



15. Change the previous SQL code from **VERSION AS OF 0** to **VERSION AS OF 1** and **run** the script again.

```
%%sql

SELECT * FROM silver.productsales VERSION AS OF 1
WHERE OrderDate >= '2023-05-01';
```

You will now see the changed data from the uploaded file in chapter Data Changes together with the transformations we implemented in data flow **DeltaSilverProductSales.**



Steps 15 and 16 are **optional**. We advise you to do these steps after the workshop and continue the workshop on step 18.

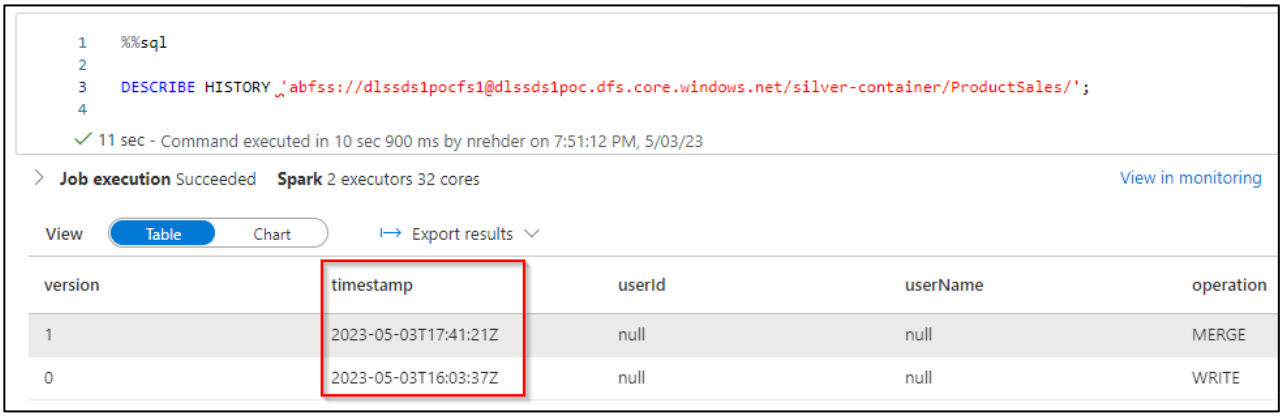
Instead of using the command **VERSION AS OF**, we can also view previous versions using a timestamp with the command **TIMESTAMP AS OF.**

16. Add the following SQL code to view the initial version of your product sales data using the **TIMESTAMP AS OF** command.

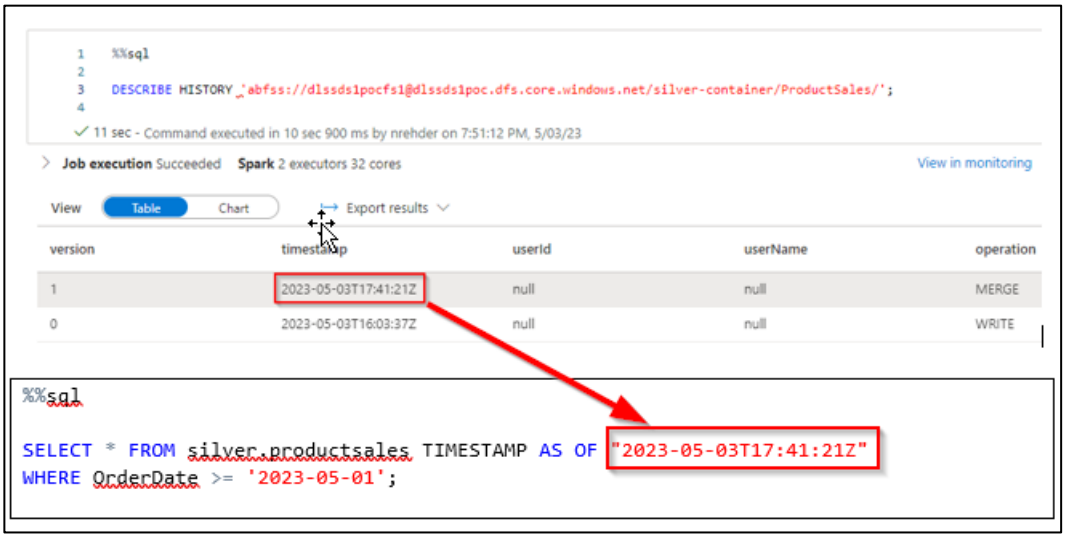
NOTE: The timestamp "2023-05-03T17:41:21Z" from the code below needs to be replaced. See the next steps.

```
%%sql

SELECT * FROM silver.productsales TIMESTAMP AS OF "2023-05-03T17:41:21Z"
WHERE OrderDate >= '2023-05-01';
```

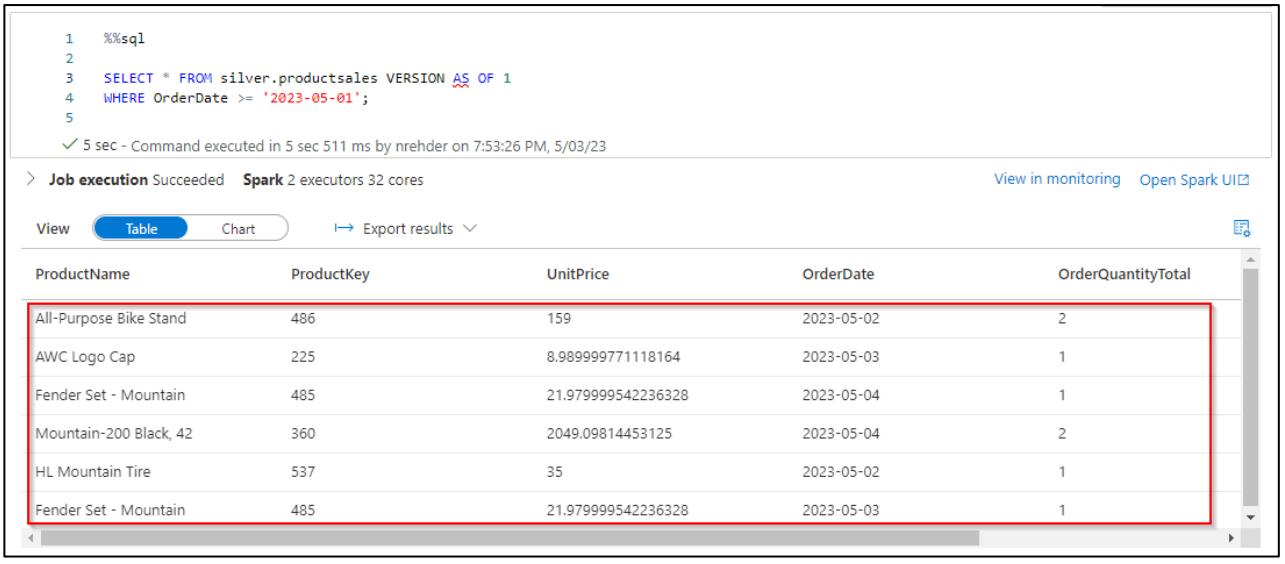
The Timestamps we can query can be found in the **DESCRIBE HISTORY** command we used previously.



17. Copy a timestamp from the **DESCRIBE HISTORY** command and paste it in the SQL code from above.

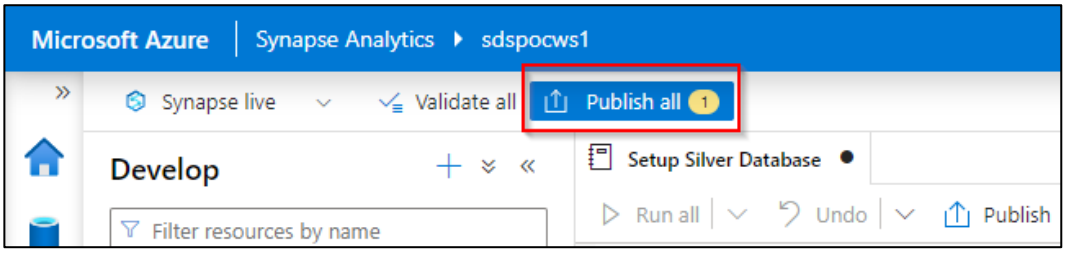


We can now see the new rows from the uploaded file in chapter [Data Changes](#) .

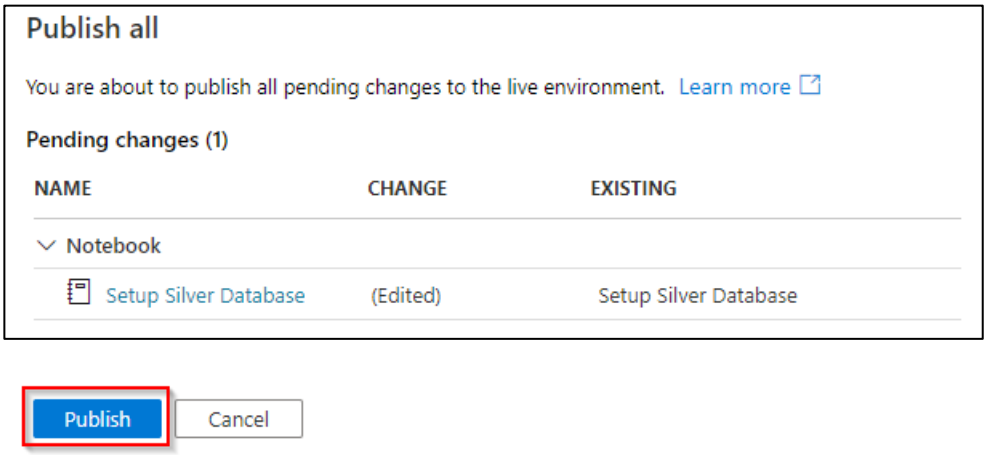


Let's save our work.

18. Select the **Publish All** button.



19. A new panel should open to the right. Select the **Publish** button.



A banner should appear confirming that the content was published.

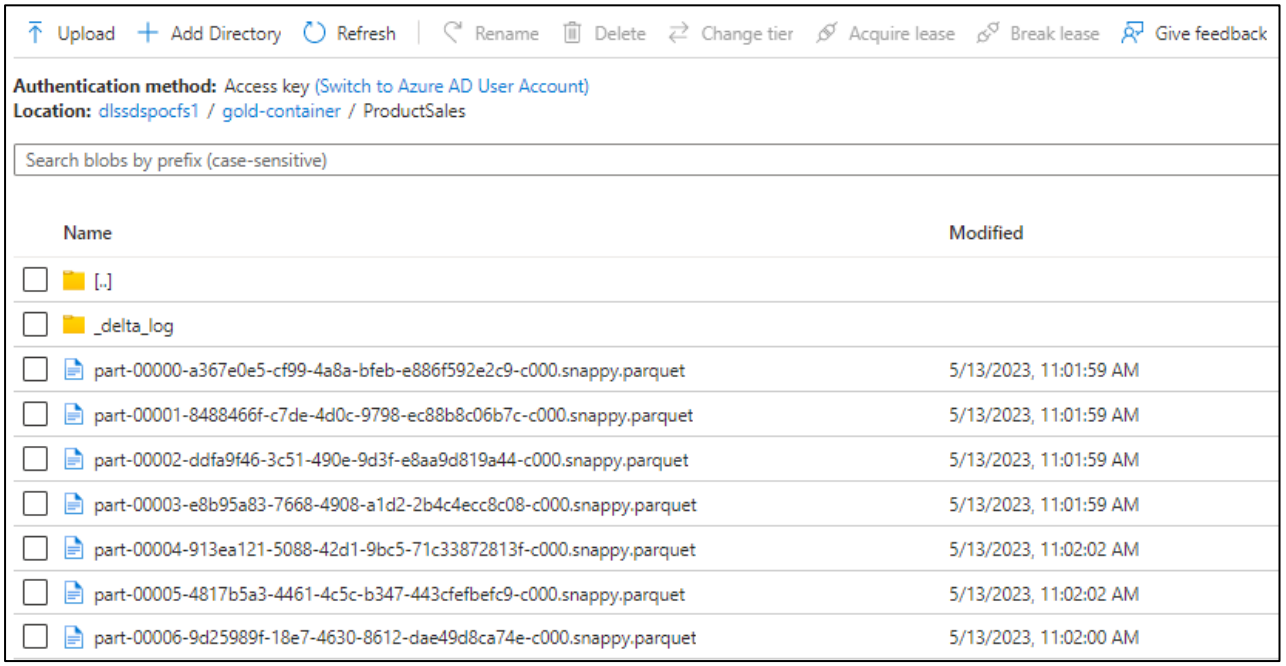
You have now learned how to view different versions of your delta formatted data being used as a delta table in your lake database using Spark SQL.

You have successfully queried a delta table using a serverless SQL pool.

7.1.1.6 Removing Historic Delta Logs (OPTIONAL)

This chapter is **optional**. We advise you to do these steps after the workshop and continue the workshop on chapter Gold Database.

Delta Lake maintains a history of all the changes by default. That means over a period, the historical data will grow. Based on your business requirements you would like to keep historical data for a past certain duration like last one month, last one year etc. to optimize your storage costs.

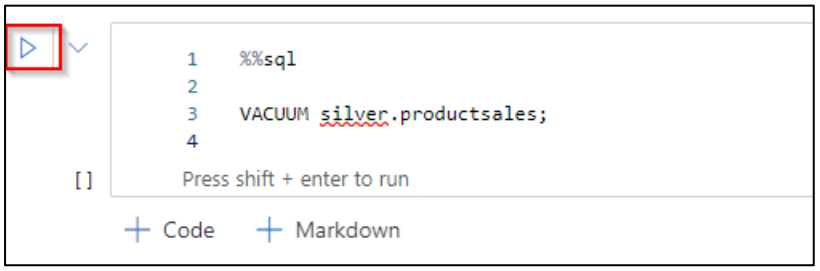


1. Within the Notebook **Setup Silver Database** code field, add the following SQL code to clear historic delta logs.

NOTE: This same command can be used later in the Setup Gold Database to clear historic delta logs.

```
%%sql

VACUUM silver.productsales;
```

2. Select the Run button to run the script.

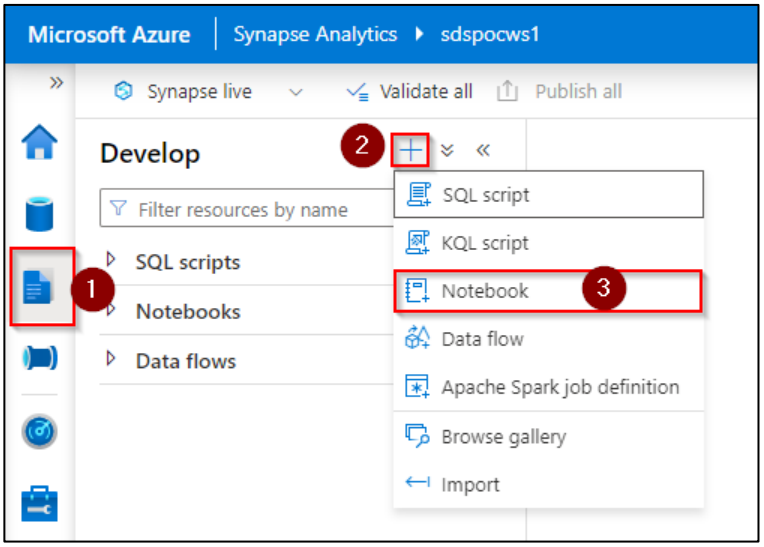


NOTE: You cannot delete historical data within the last 7 days by default and that is to maintain consistency in data.

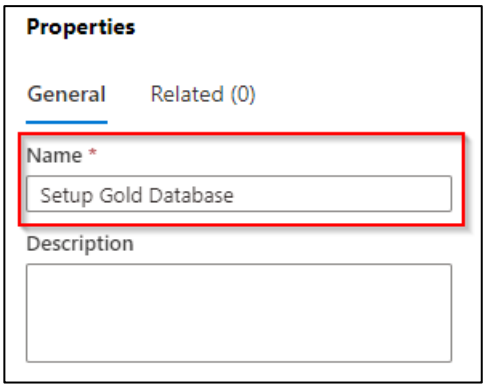
7.1.2 Gold Database

We will now create our gold database, where the data is made ready for consumption. The steps here are similar to what we configured in chapter Silver Database.

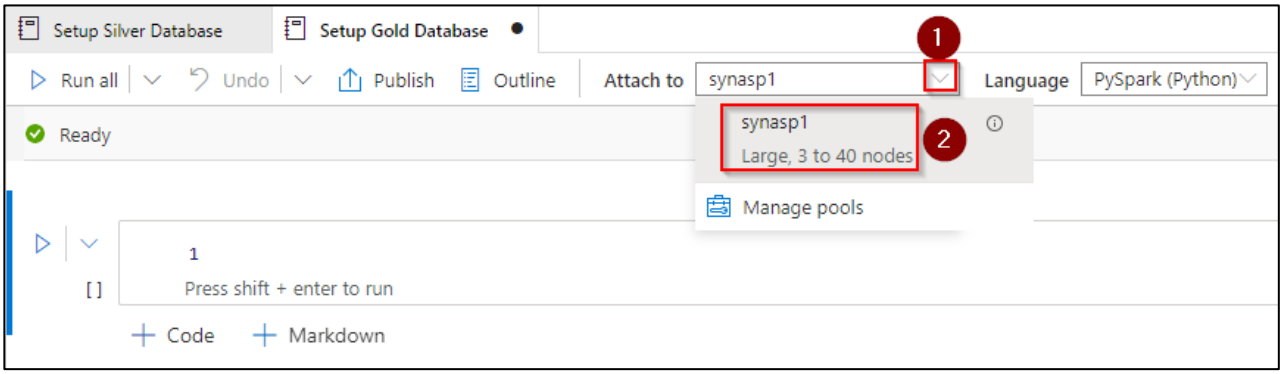
1. Select the **Develop** tab to the left, select the **+** button and create a new Notebook.



2. Rename the newly created Notebook to **Setup Gold Database**.



3. To begin using this new Notebook, attach an Apache Spark Pool by selecting the **Attach to** dropdown. Select the provisioned spark pool.



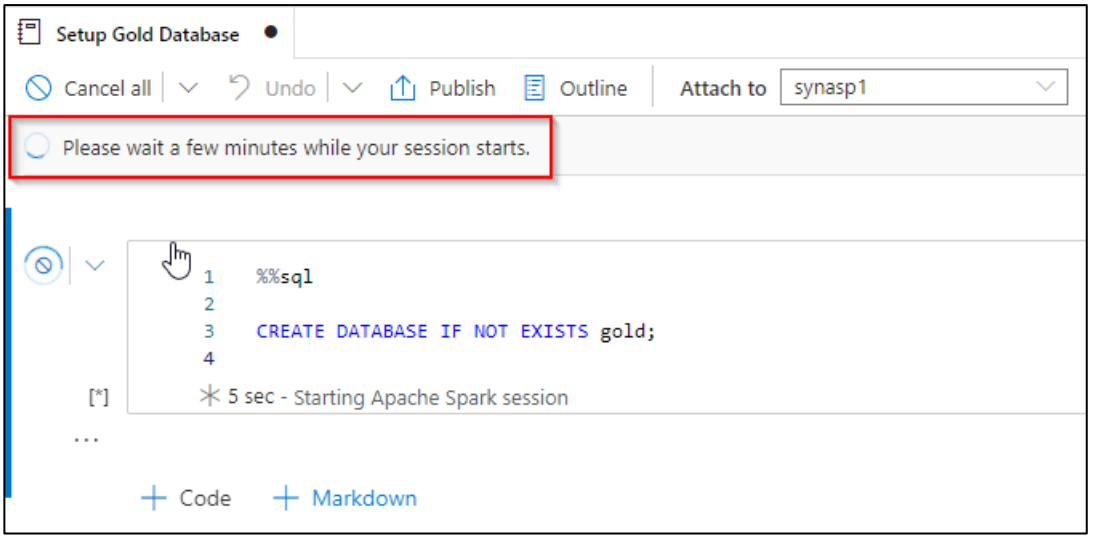
7.1.2.1 Lake Database

4. Within the Notebook **Setup Gold Database** code field, add the following SQL code to create the database:

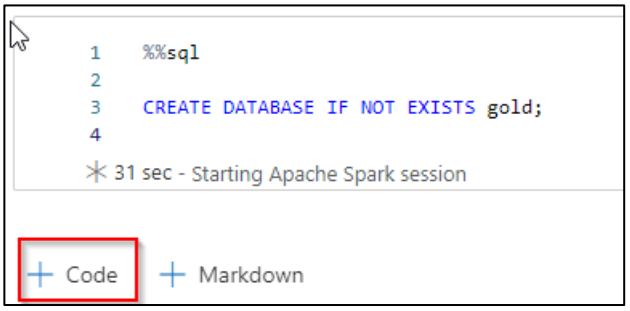
```
%%sql

CREATE DATABASE IF NOT EXISTS gold;
```

5. Select the Run button to start the Spark Pool and run the script.

Note: Starting the Spark Pool may take 3 – 5 minutes. Please be patient!



6. Once the code has run, beneath the initial code field we created in step 4. Select the **+ Code** button, to add a new code field to the Notebook.



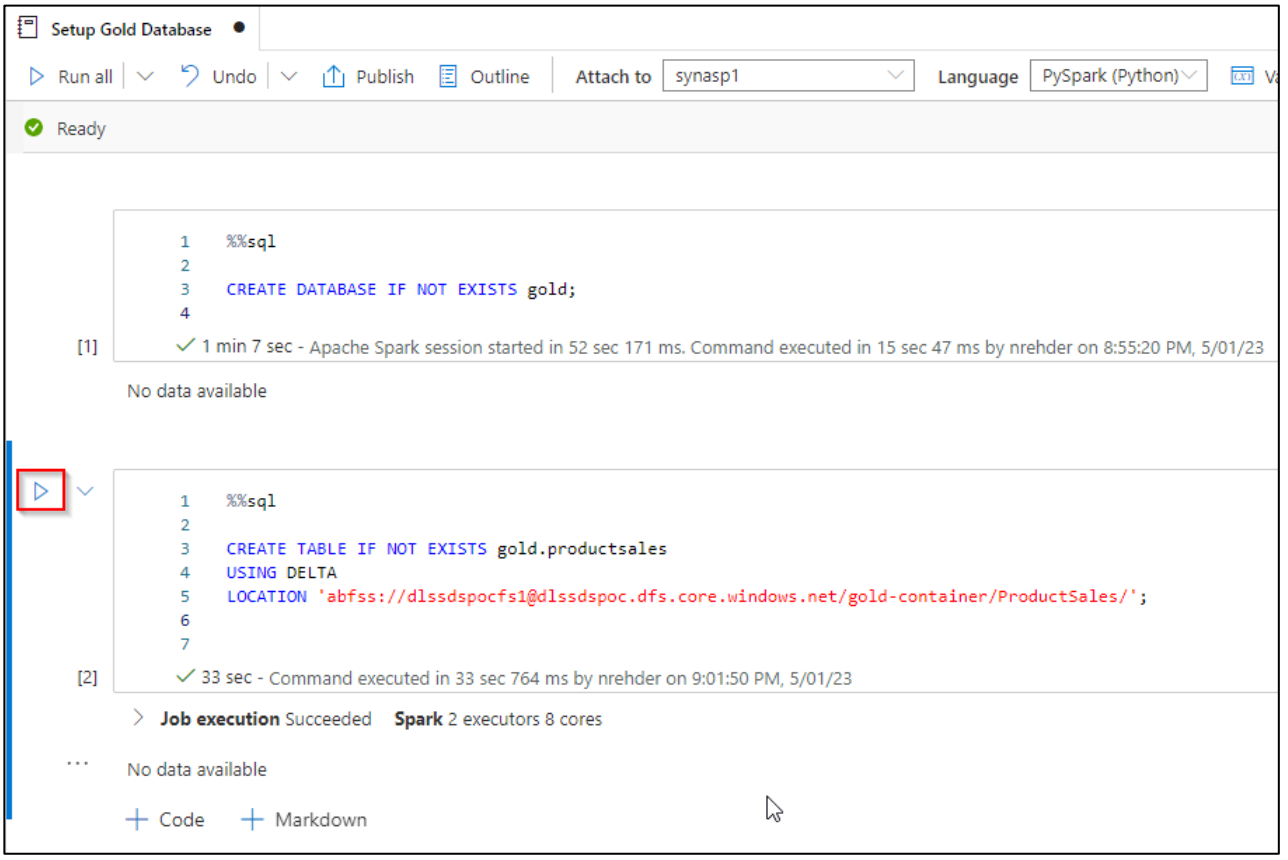
7.1.2.2 Lake Database Delta Tables

7. Within the Notebook **Setup Gold Database** code field, add the following SQL code to create the delta tables in the gold lake database.

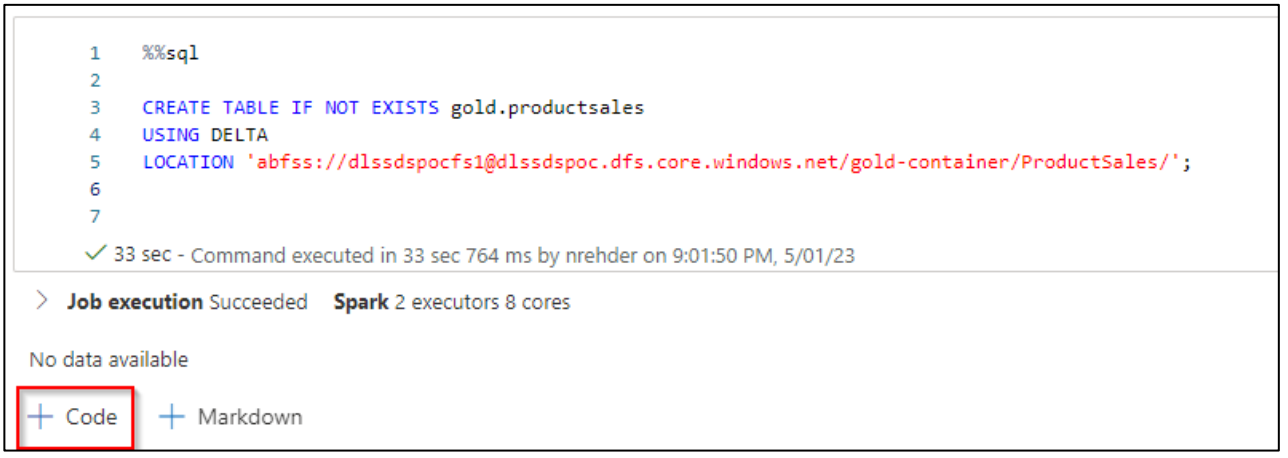
```
%%sql

CREATE TABLE IF NOT EXISTS gold.productsales
USING DELTA
LOCATION 'abfss://dlssdspocfs1@dlssdspoc.dfs.core.windows.net/gold-
container/ProductSales/';
```

8. Change the **LOCATION** parameter within the SQL code to your URI as described in chapter [Azure Blob File System](#).

9. Select the Run button to run the script.



10. Once the code has run, select the **+ Code** button, to add a new code field to the Notebook.



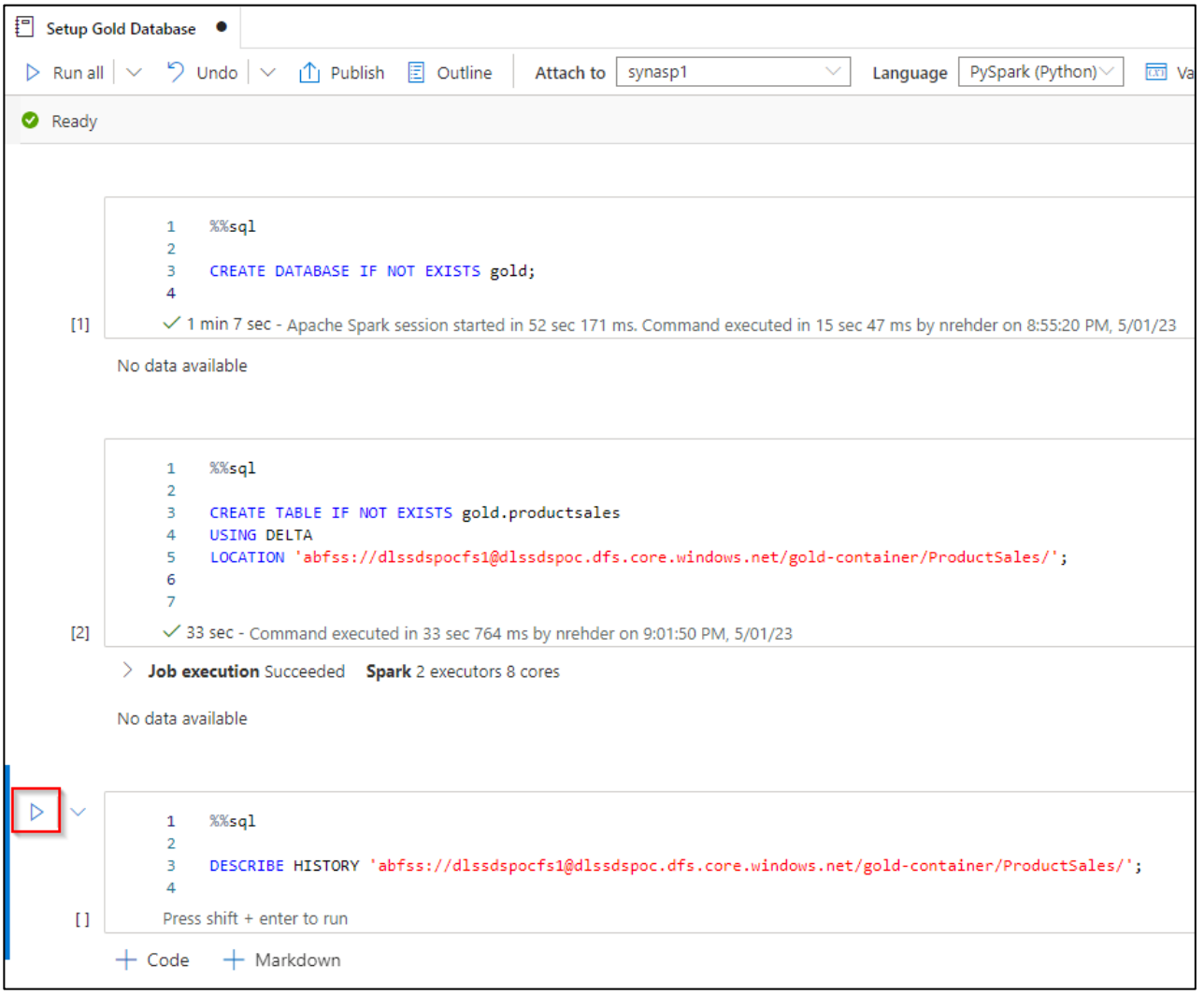
7.1.2.3 Audit Logs

11. Add the following SQL code to view the change history of your product data i.e., the data that we manually changed in chapter Data Changes.

```
%%sql

DESCRIBE HISTORY 'abfss://dlssdspocfs1@dlssdspoc.dfs.core.windows.net/gold-
container/ProductSales/';
```

12. Within the SQL code above change the **abfss URI** to your URI as described in chapter Azure Blob File System.

13. Select the Run button to run the script.



Once the code has run you will see the change operations WRITE (the initial trigger) and MERGE (the second trigger after the data changes)

14. Select the **+ Code** button, to add a new code field to the Notebook.

7.1.2.4 Optimization (Z-Ordering)

Z-ordering will allow for greater read performance by taking advantage of data skipping. One or multiple columns can be specified for a Z-order. Ideal column choices are those that are commonly used as filters when reading data.

15. Add the following SQL code to implement Z-ordering optimization using the **OrderDate** column.

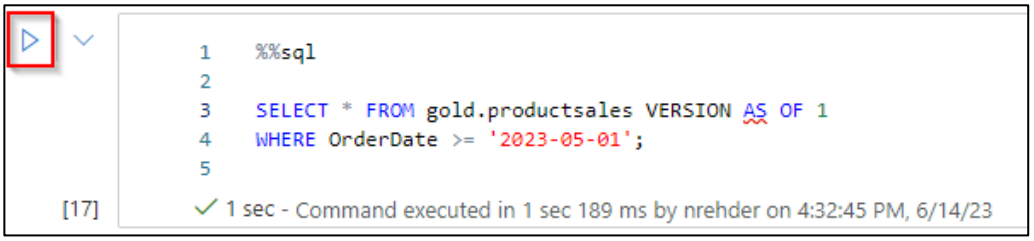
```
%%sql

OPTIMIZE gold.productsales ZORDER BY (OrderDate);
```

7.1.2.5 Time Traveling

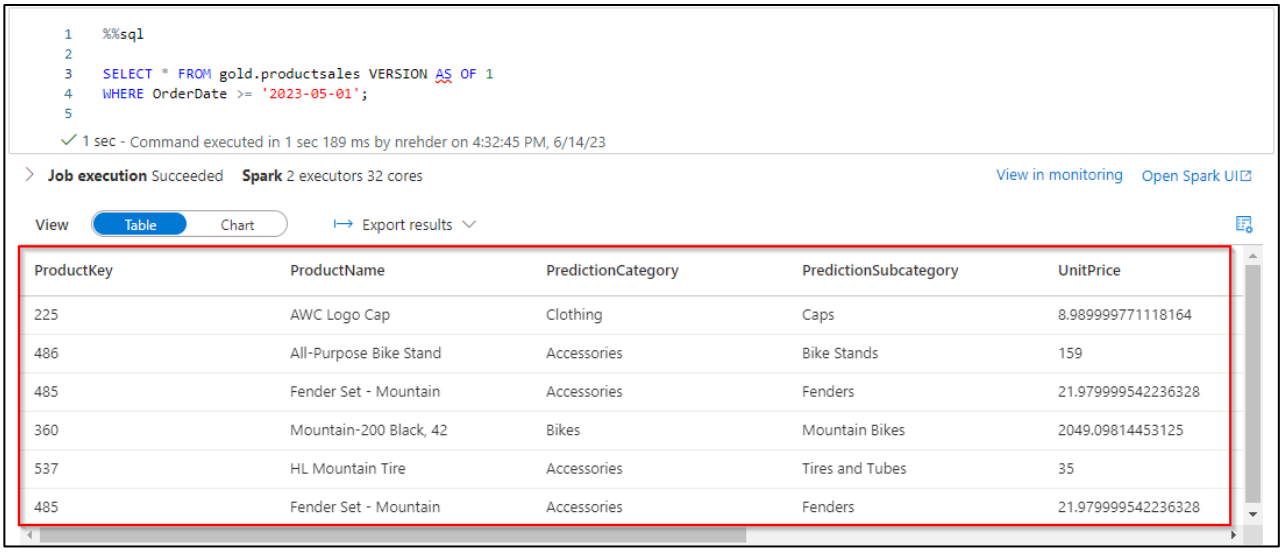
16. Add the following SQL code to view **version 1**.

```
%%sql

SELECT * FROM gold.productsales VERSION AS OF 1
WHERE OrderDate >= '2023-05-01';
```

17. Select the **Run** button to run the script.



You will now see the same changed data we queried previously in chapter Silver Database, but transformed based on the transformations activities we implemented in data flow **DeltaGoldProductSales.**



7.1.2.6 Restoring

Let's assume, we are not happy with the changes made and would like to revert back to version 0 i.e. without the data changes we implemented in chapter Data Changes.

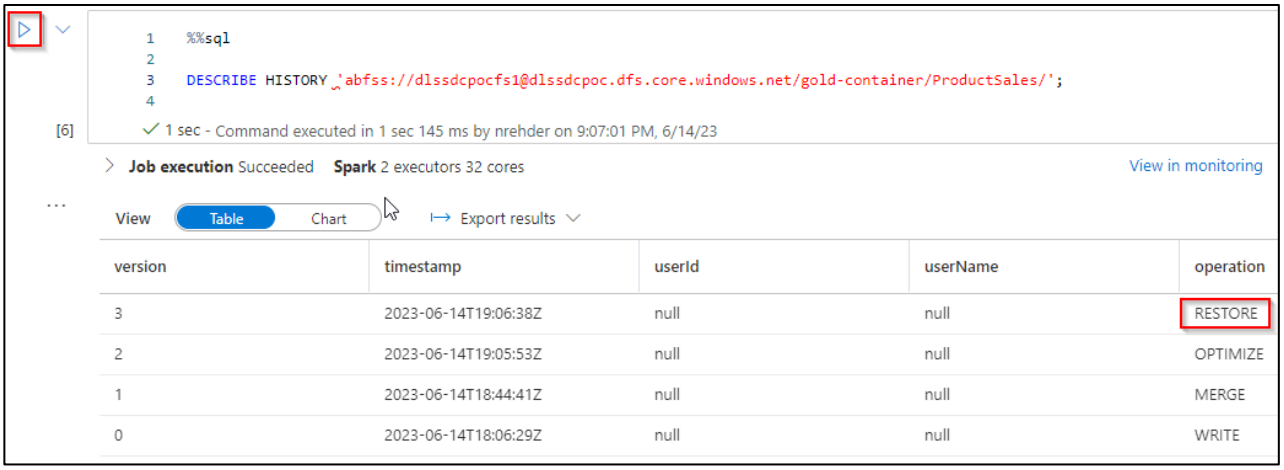
18. Select the **+ Code** button, to add a new code field to the Notebook.
19. Add the following SQL code to restore **version 0** as the primary data structure.

```
%%sql

RESTORE gold.productsales TO VERSION AS OF 0;
```

20. Select the **Run** button to run the script.

We can now view this restoration process in our audit log. Re-run the DESCRIBE HISTORY script from step 13. You should now see the RESTORE command in the logs.

NOTE: Restoring is database specific. If you restore a version within the gold database, you will also have to do so manually in the silver database.



Let's verify that we are viewing the initial data structure (version 0).

21. Select the **+ Code** button, to add a new code field to the Notebook.
22. Add the following SQL code to view the data.

```
%%sql

SELECT * FROM gold.productsales VERSION AS OF 3
WHERE OrderDate >= '2023-05-01';
```

23. Select the **Run** button to run the script.

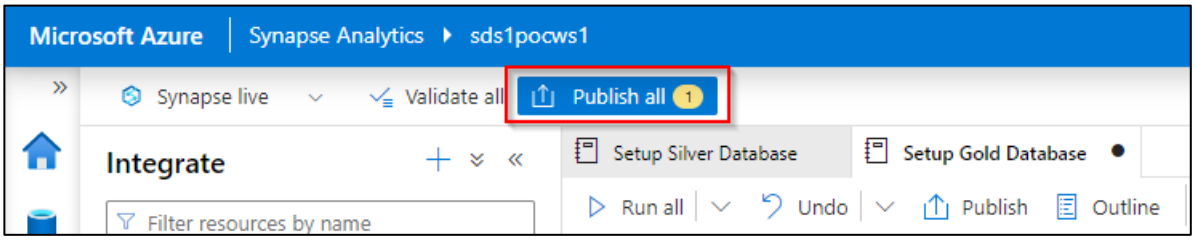
NOTE: We select VERSION AS OF 3 because version 2 is the Z-Ordering optimization we implemented in chapter Optimization (Z-Ordering)

Running this script should return no data since the initial data i.e., version 0 did not have any new OrderDate rows after May 01, 2023.

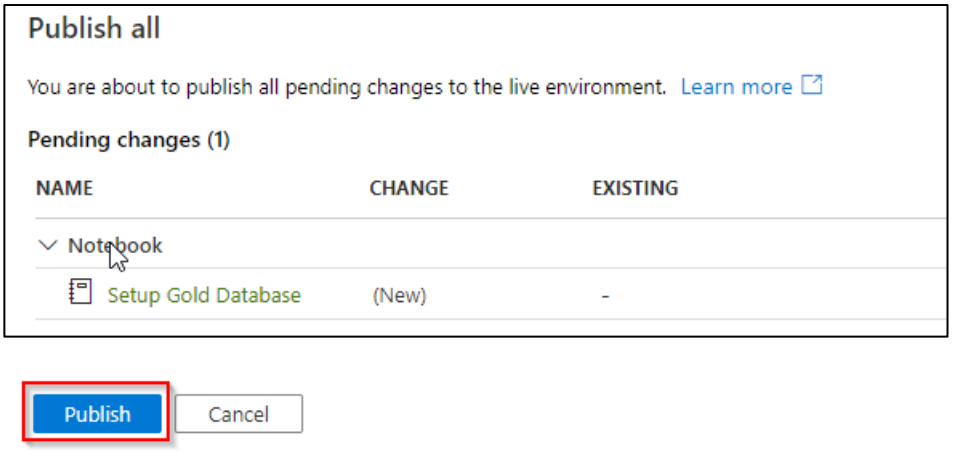
We have therefore successfully reversed our data changes.

Let's save our work.

24. Select the Publish All button.



25. A new panel should open to the right. Select the **Publish** button.



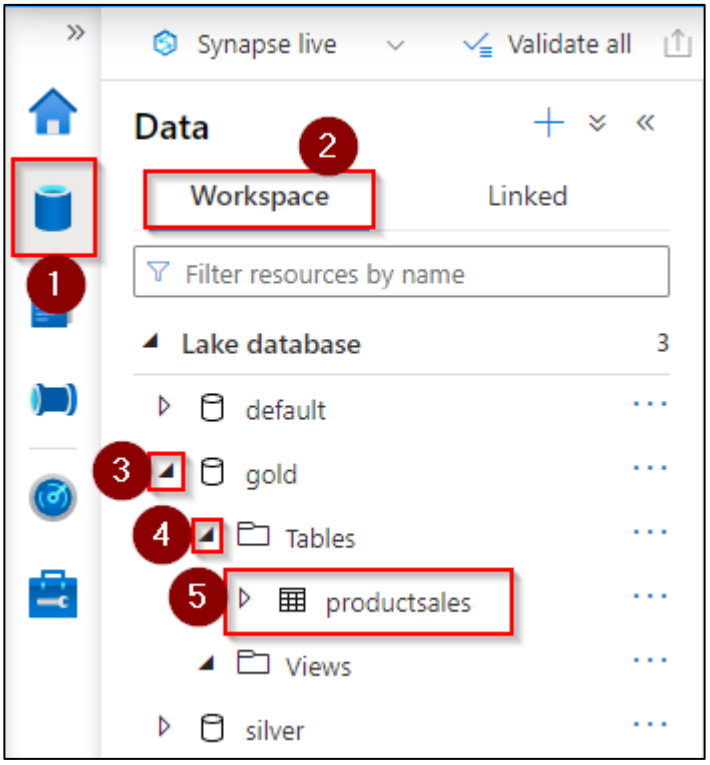
A banner should appear confirming that the content was published.

7.1.2.7 Query Delta Table with SQL Serverless Pool

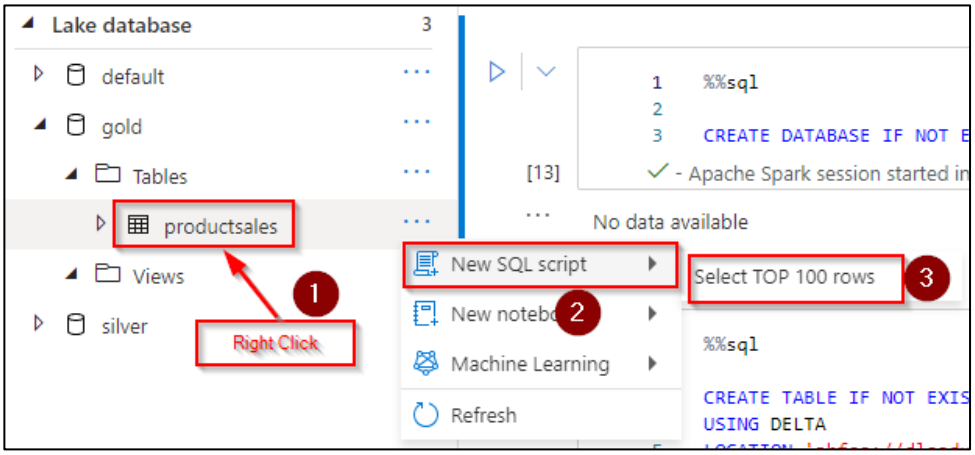
We will now query the delta table we created from the delta formatted data located in the gold-container in chapter Gold Database.

NOTE: You may need to refresh your browser to view the newly created Lake Database in the Data tab. Make sure you publish everything and save your work before refreshing the browser. See Step 24.

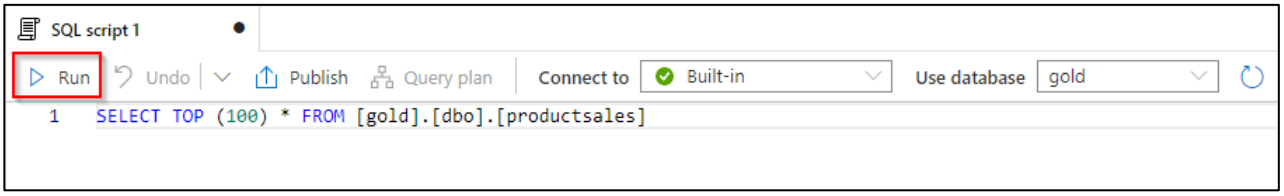
34. Select the **Data** tab to the left, select **Workspace** and browse to the **Tables** folder. Within the Tables folder you will see the **productsales** table we created earlier.



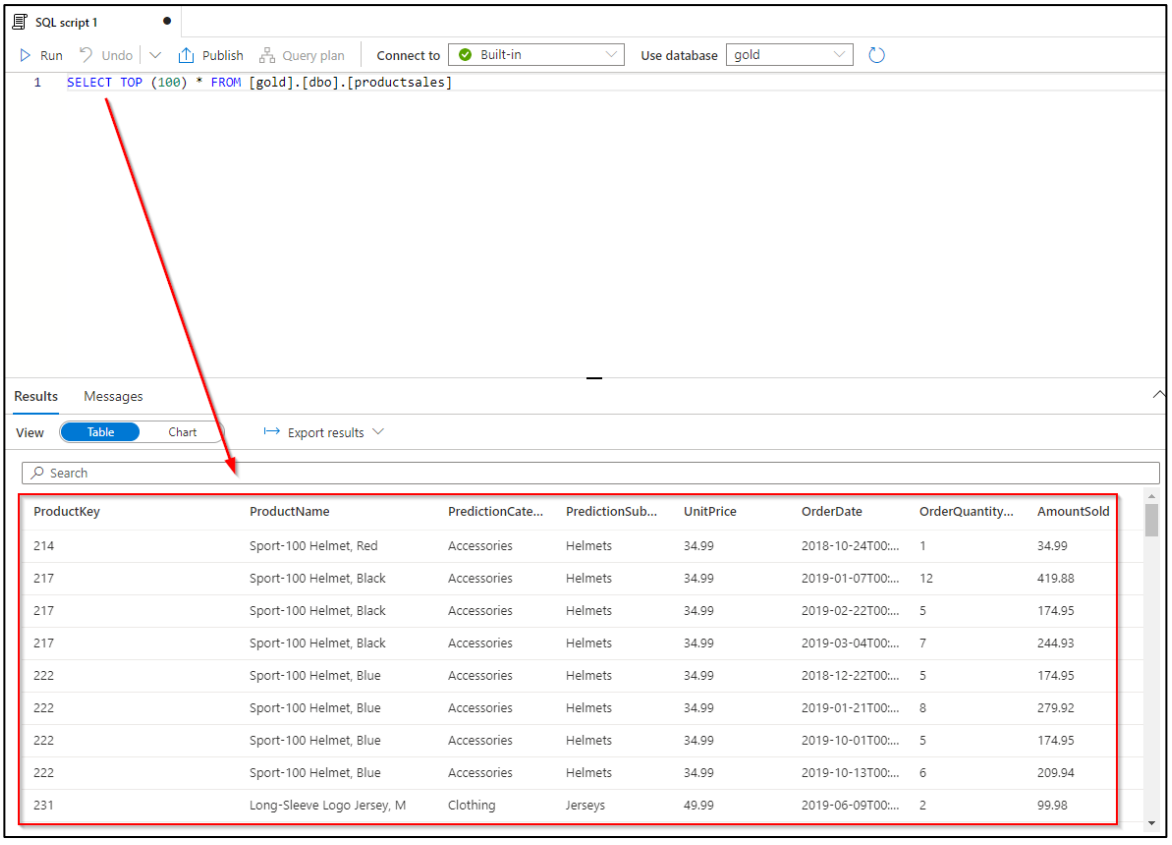
35. Right click the **productsales** table and select **New SQL script**, followed by **Select TOP 100 rows**. This will create a new tab with a SQL statement to query the table.



36. Select the **Run** button to run the SQL statement.



After query execution, you will now see the data related to the **productsales** delta table in the gold database.



8 Query Delta Formatted Data in Azure Storage with Serverless SQL Pool (OPTIONAL)

This chapter is **optional**. We advise you to do this chapter after the workshop.

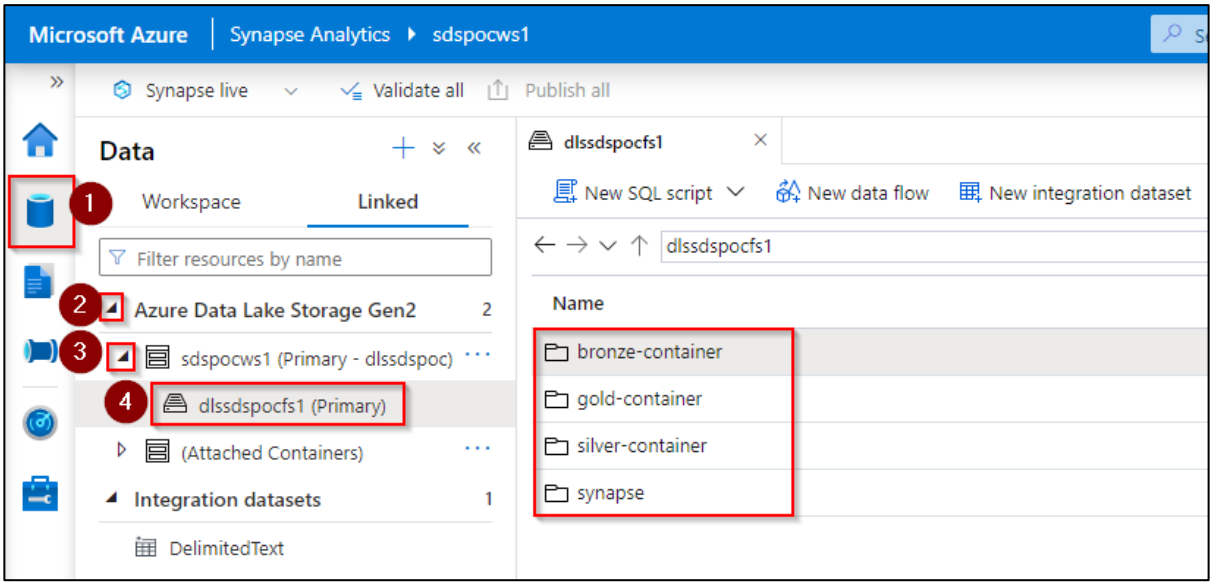
We have been querying the delta tables using the Apache Spark Pool and the Serverless SQL Pool. Additionally, we can also query the delta formatted files directly from the Azure storage using the Serverless SQL Pool. This method is usually reserved for ad-hoc queries.

NOTE: Querying the delta files using the Serverless SQL Pool comes with limitations. One of those is the inability to time travel. The data queried is always in the most resent state. Additional limitations can be found here.

NOTE: We are querying the gold-container in this chapter, but any container with delta formatted data could be used.

1. Select the **Data** tab to the left, select **Linked** and open your **primary data storage** i.e., the storage you provisioned in chapter Create Resource Group and Resources . Select the **named container** of this storage.

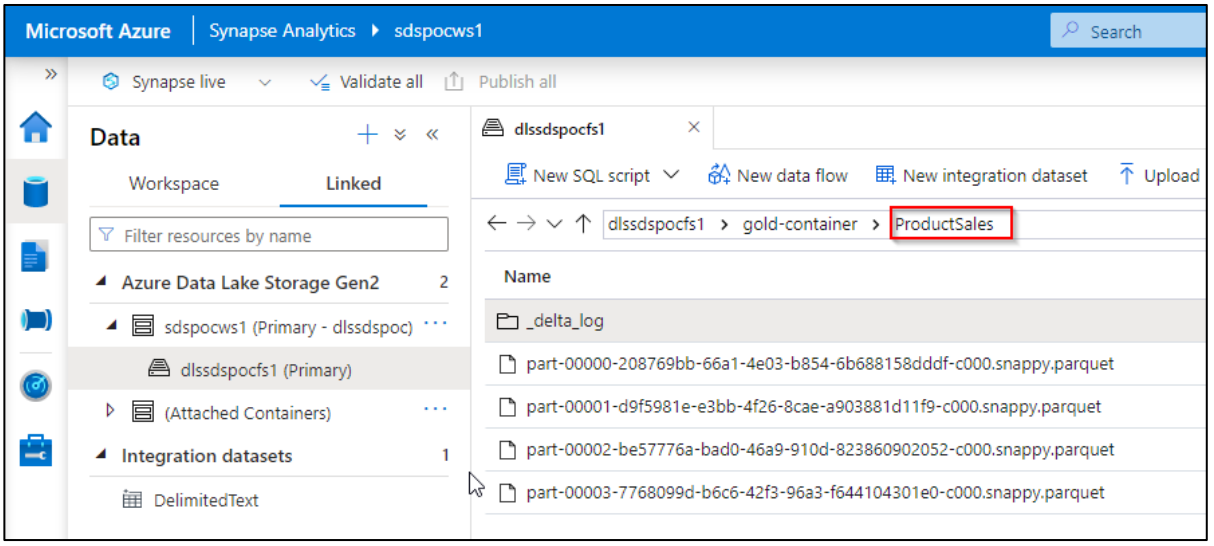
Selecting the container will open a **new tab** on the Synapse canvas. You will see the medallion structure you created earlier (*bronze-container, silver-container, and gold-container*).



The first data flow we created (**DeltaSilverProductSales**) imported data from the *bronze-container*, transformed it and saved it to the Delta format in the *silver-container*. Later we created a second data flow (**DeltaGoldProductSales**) that imported data from the *silver-container* and joined it with predicted data, which in turn was saved in the Delta format in the *gold-container*. To review these steps, see chapter Data Flows.

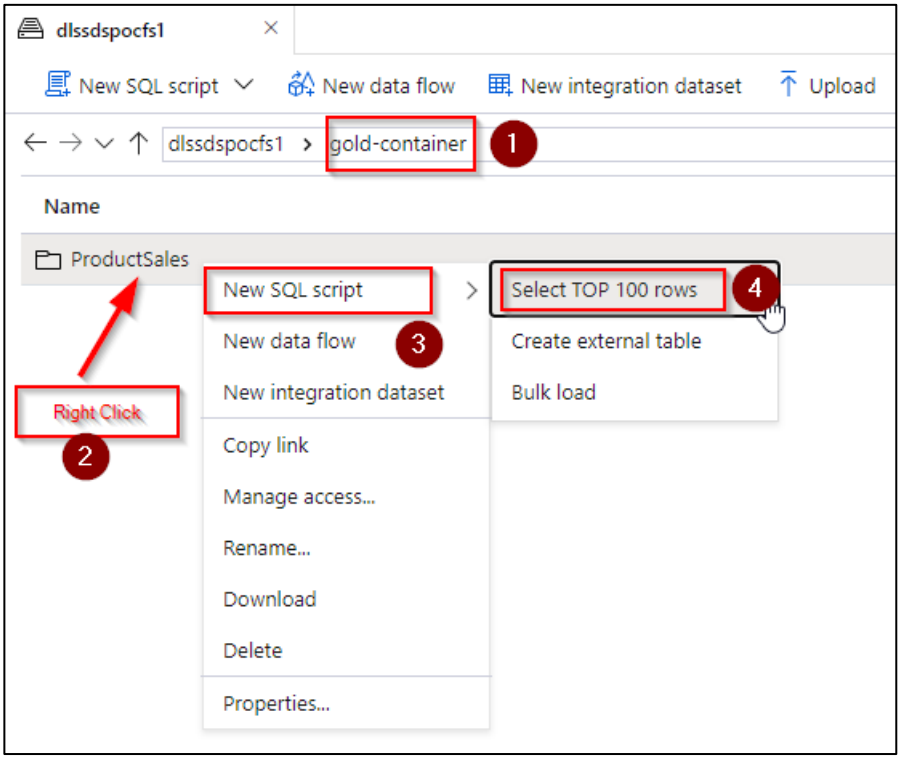
2. Within the **gold-container**, select the **ProductSales** directory.

You will see that a **delta log folder** (_delta_log) was created together with a delta formatted file.

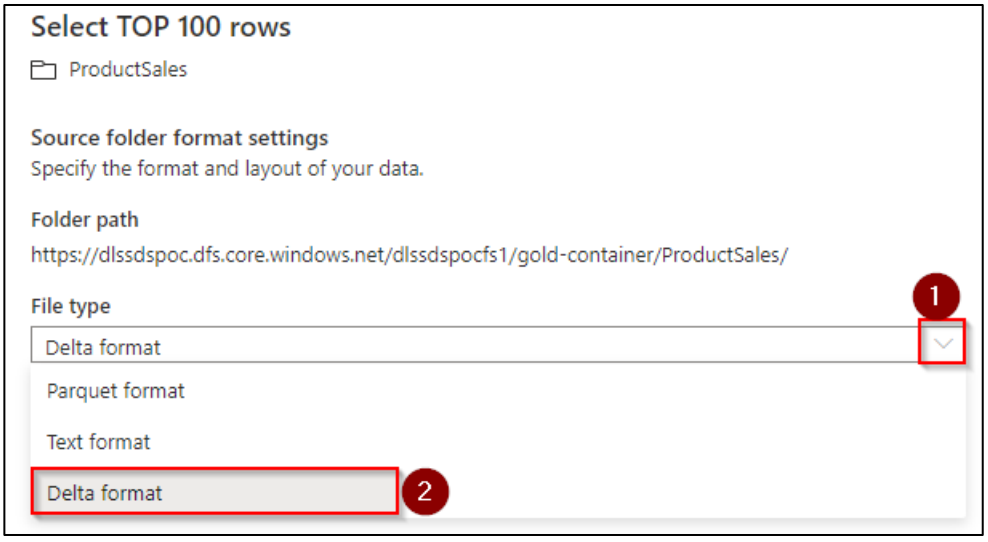


We can now query this delta formatted file using the SQL Serverless pool directory from the data lake.

3. Go back to the **gold-container**, right-click on the **ProductSales** directory, select **New SQL script** and **Select Top 100 rows.**

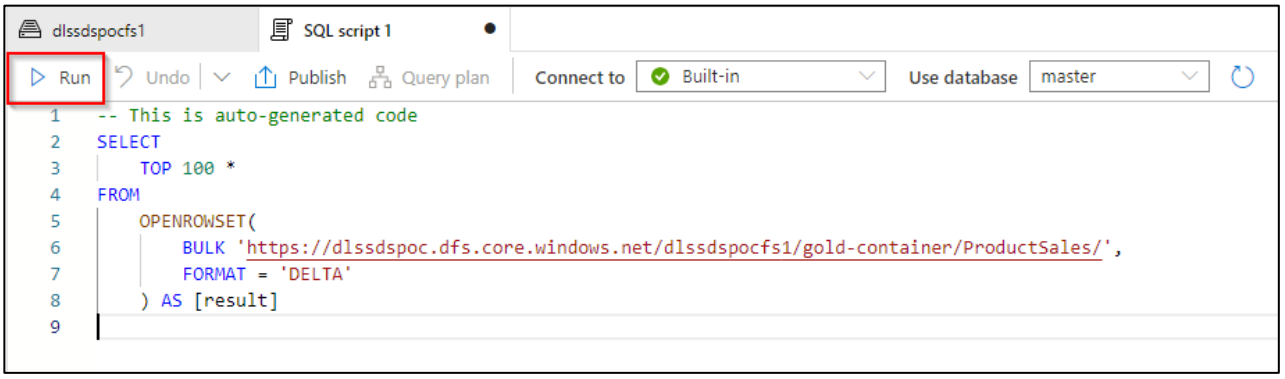


4. Once selected, a new panel will open to the right. Select the File Type **Delta format** and select **Apply**.

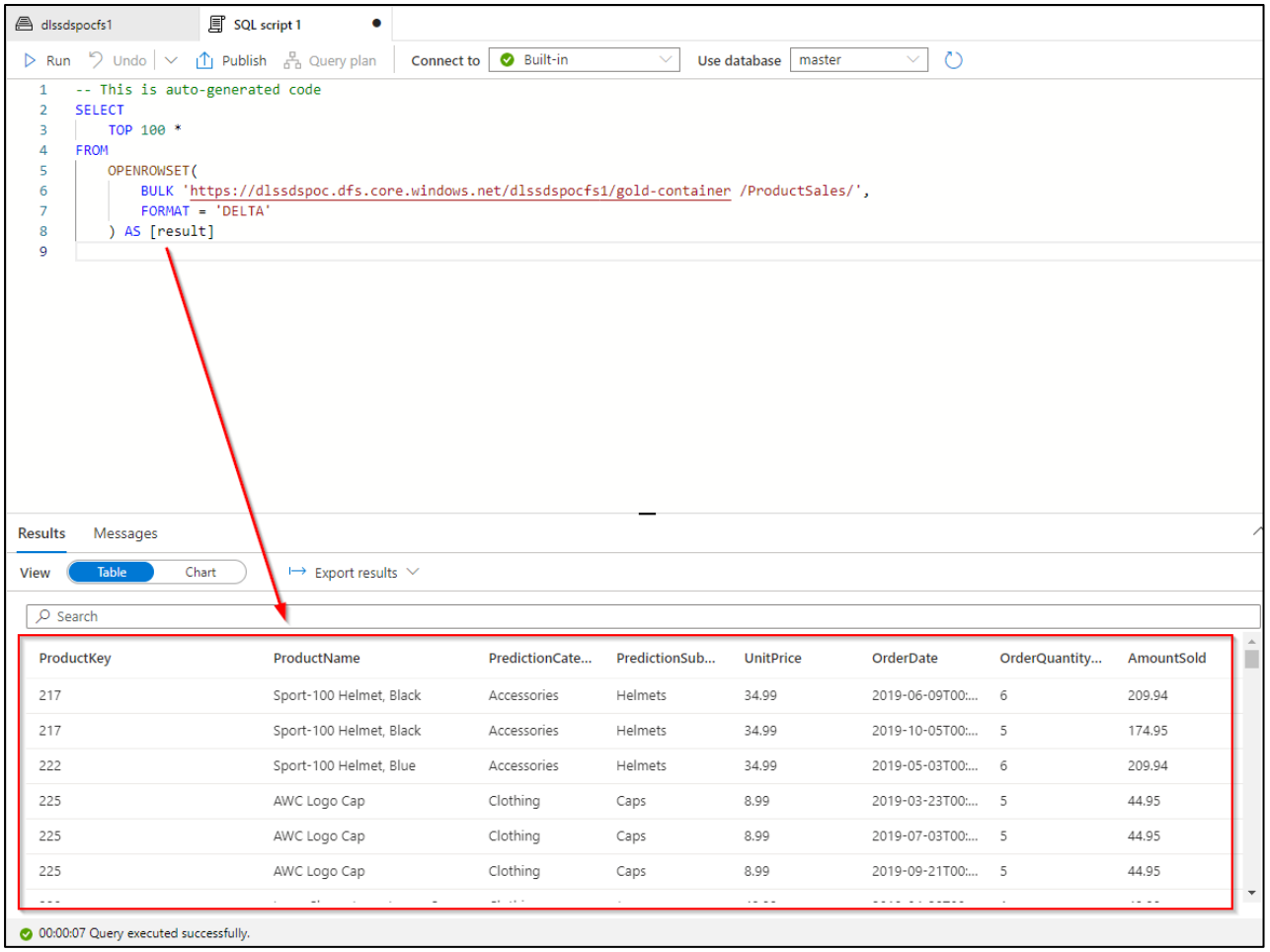


This will create a SQL statement querying the first 100 rows of ProductSales directory located in the gold-container using the delta format.

5. Run the query.



You will see a query output below:


dlssdspocfs1
SQL script 1
Run
Undo
Publish
Query plan
Connect to
Built-in
Use database
master
-- This is auto-generated code
SELECT
TOP 100 *
FROM
OPENROWSET(
BULK 'https://dlssdspoc.dfs.core.windows.net/dlssdspocfs1/gold-container /ProductSales/',
FORMAT = 'DELTA'
) AS [result]
Results
Messages
View
Table
Chart
Export results
Search
ProductKey
ProductName
PredictionCate...
PredictionSub...
UnitPrice
OrderDate
OrderQuantity...
AmountSold
217
Sport-100 Helmet, Black
Accessories
Helmets
34.99
2019-06-09T00:...
6
209.94
217
Sport-100 Helmet, Black
Accessories
Helmets
34.99
2019-10-05T00:...
5
174.95
222
Sport-100 Helmet, Blue
Accessories
Helmets
34.99
2019-05-03T00:...
6
209.94
225
AWC Logo Cap
Clothing
Caps
8.99
2019-03-23T00:...
5
44.95
225
AWC Logo Cap
Clothing
Caps
8.99
2019-07-03T00:...
5
44.95
225
AWC Logo Cap
Clothing
Caps
8.99
2019-09-21T00:...
5
44.95
00:00:07 Query executed successfully.

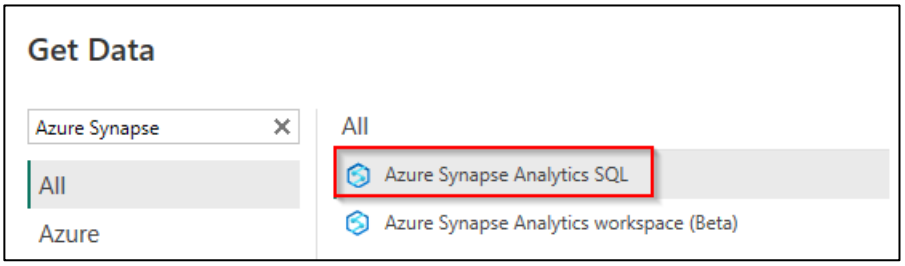
9 Connect to Power BI (OPTIONAL)

This chapter is **optional**. We advise you to do this chapter after the workshop.

We have created a data lakehouse, the next step is to import and visualize the data using a BI Tool such as Power BI.

There are currently two methods of importing delta formatted data into Power BI if you've created the data lakehouse using Azure Synapse Analytics:

1. **Azure Synapse Analytics SQL Connector**



This native connector uses the **Serverless SQL endpoint** for Azure Synapse to query the delta tables we created in chapter Create Azure Synapse Lake Database. This is a great option if the report consumers expect live data, as you can connect to the Serverless SQL endpoint using the **DirectQuery mode**.

Since we are using the Serverless SQL endpoint, we cannot time travel. The delta formatted data we query using this connector will always be the most current version.

In the future, Microsoft will most likely make a Serverless Spark endpoint for Azure Synapse available, allowing us to query the delta formatted data using spark, which enables time travel. But for now, this is not yet available.

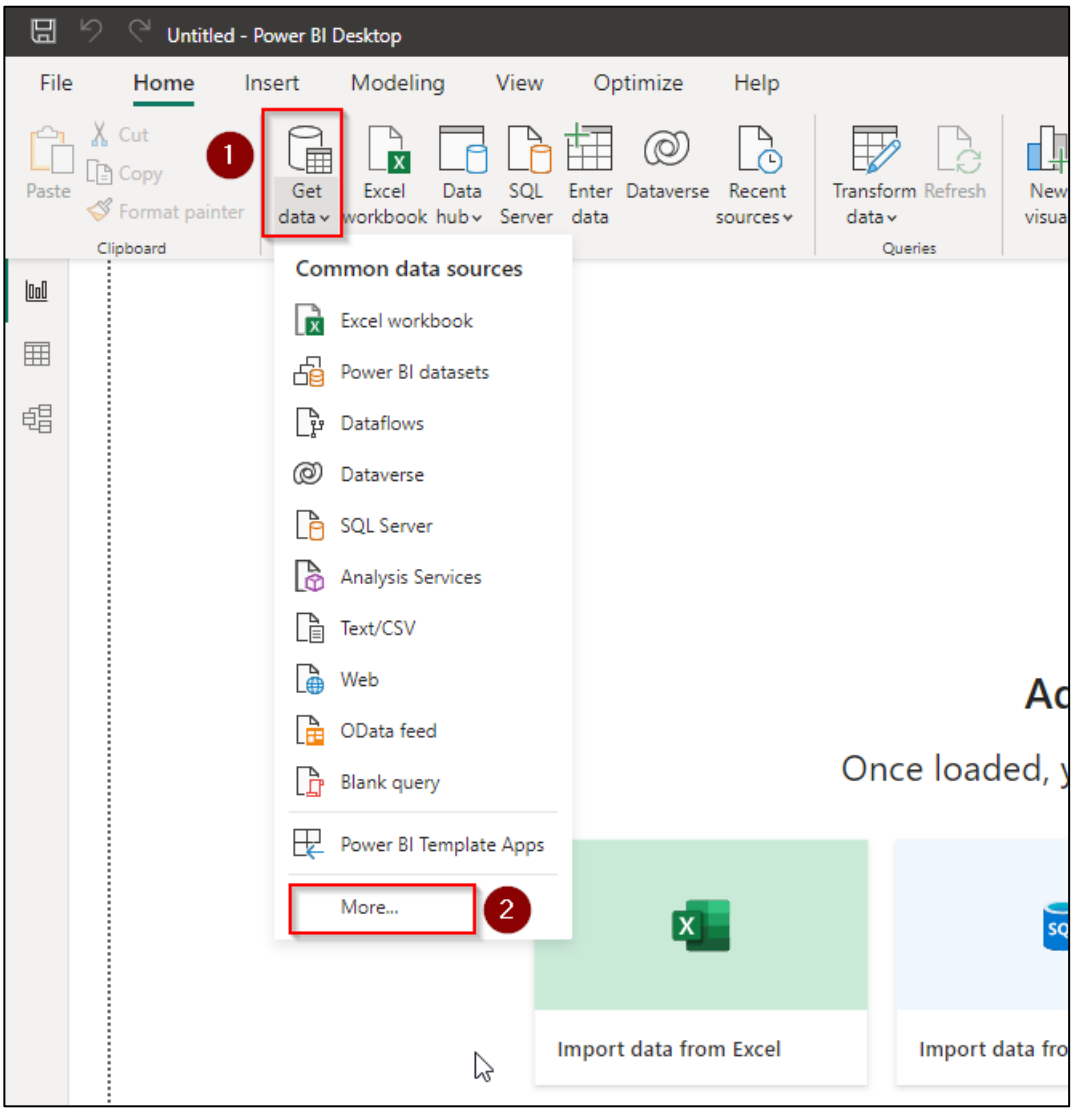
2. **Custom Delta Lake Connector**

This custom connector allows you to import your delta lake files directly from your storage account. This option allows you to define the version number directly in the Power Query Editor in Power BI. Keep in mind that you can only use the **import mode** with this option.

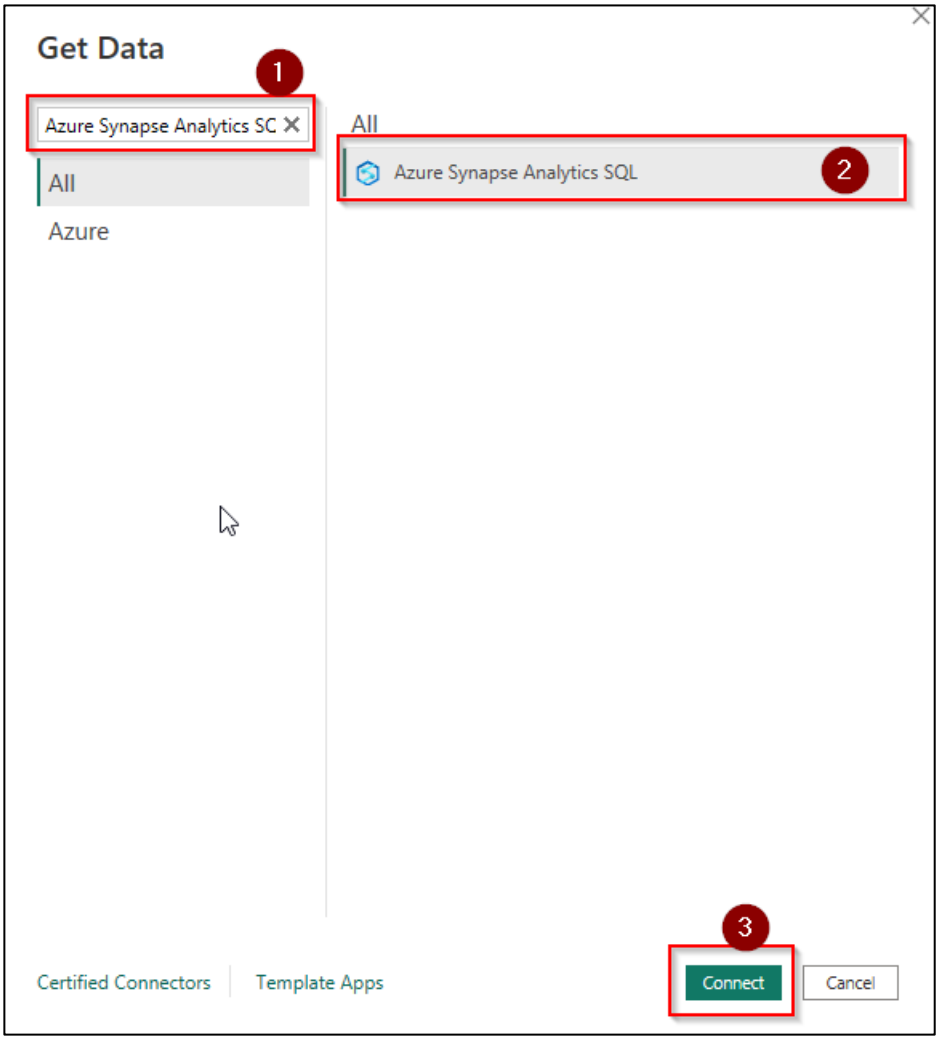
9.1.1 Azure Synapse Analytics SQL Connector

Let's connect our delta table in the gold database located in the Azure Synapse using the Power BI Azure Synapse Analytics SQL Connector.

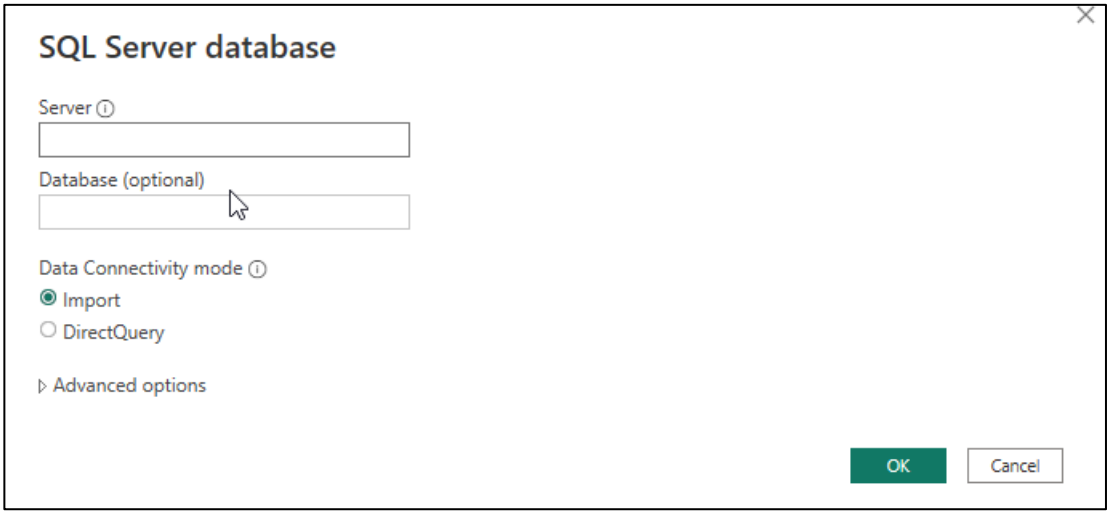
1. Open the **Power BI Desktop** application.
2. Once opened, select the **Get data** button and select **More** to view all the available connectors.



3. A new window will open, search for the **Azure Synapse Analytics SQL** and select it from the list**.**
4. Select the **Connect** button.

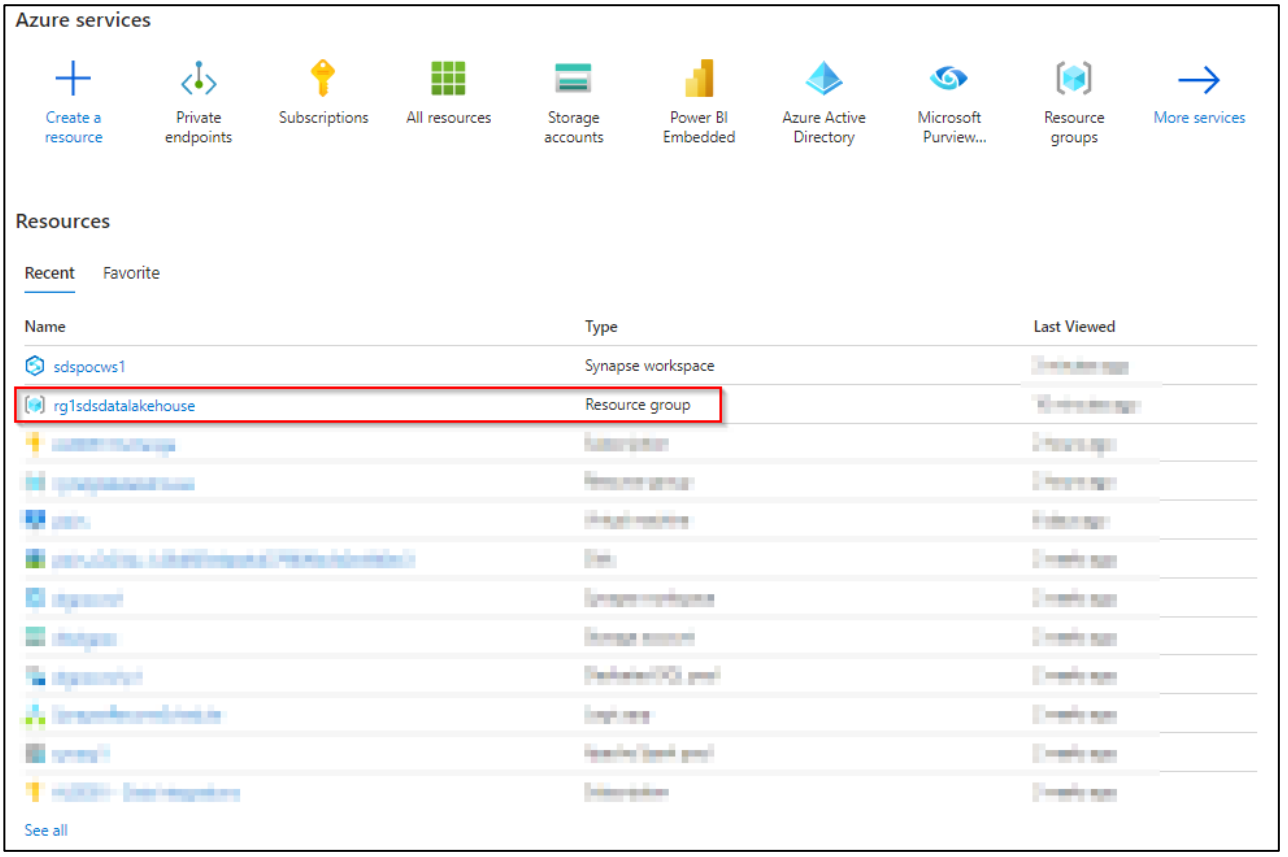


A new window will appear requesting the SQL Server database information.

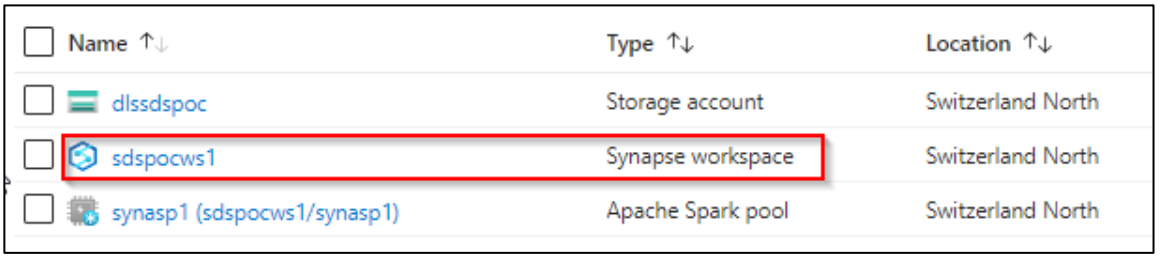


We require the SQL Serverless Pool endpoint for Azure Synapse.

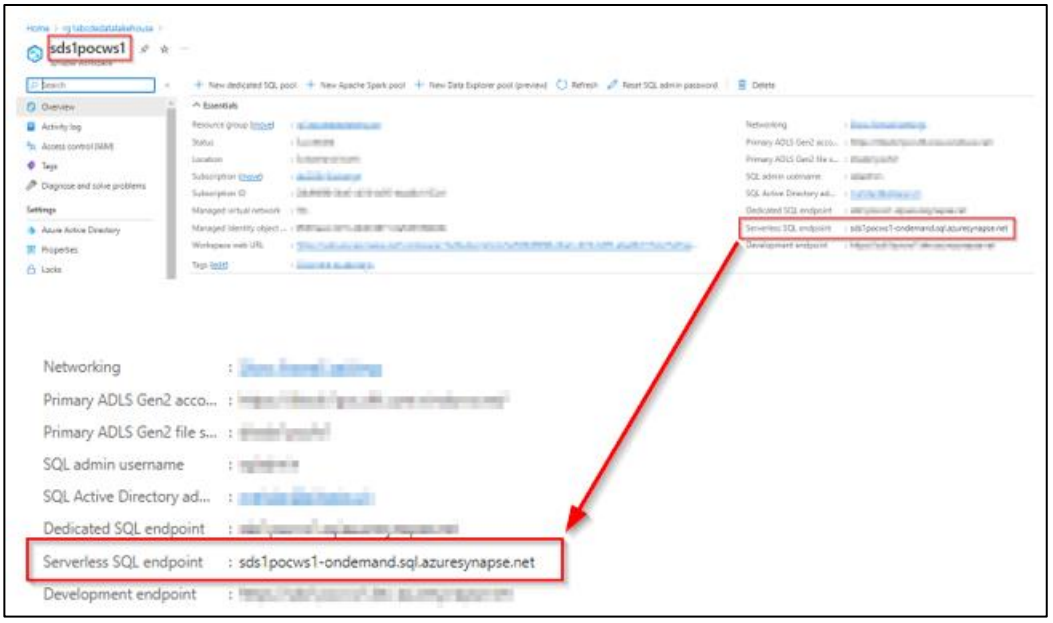
3. Go back to the Azure portal home screen and select the **Resource group** you provisioned.



4. Within the **Resource group, s**elect the resource type **Synapse workspace**.

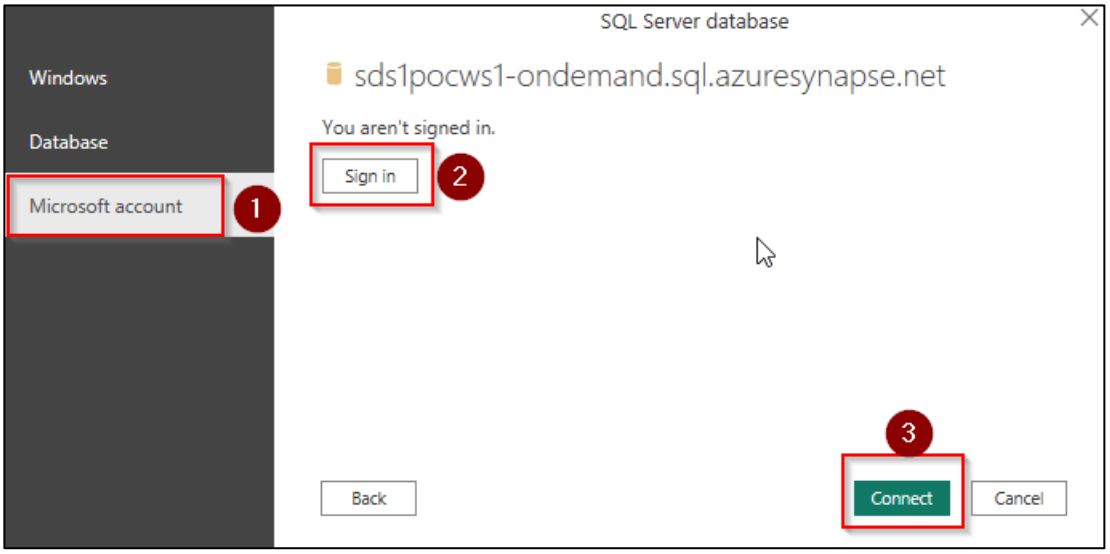


5. A new page will appear. The Serverless SQL endpoint is in the **overview** section of the Synapse workspace.

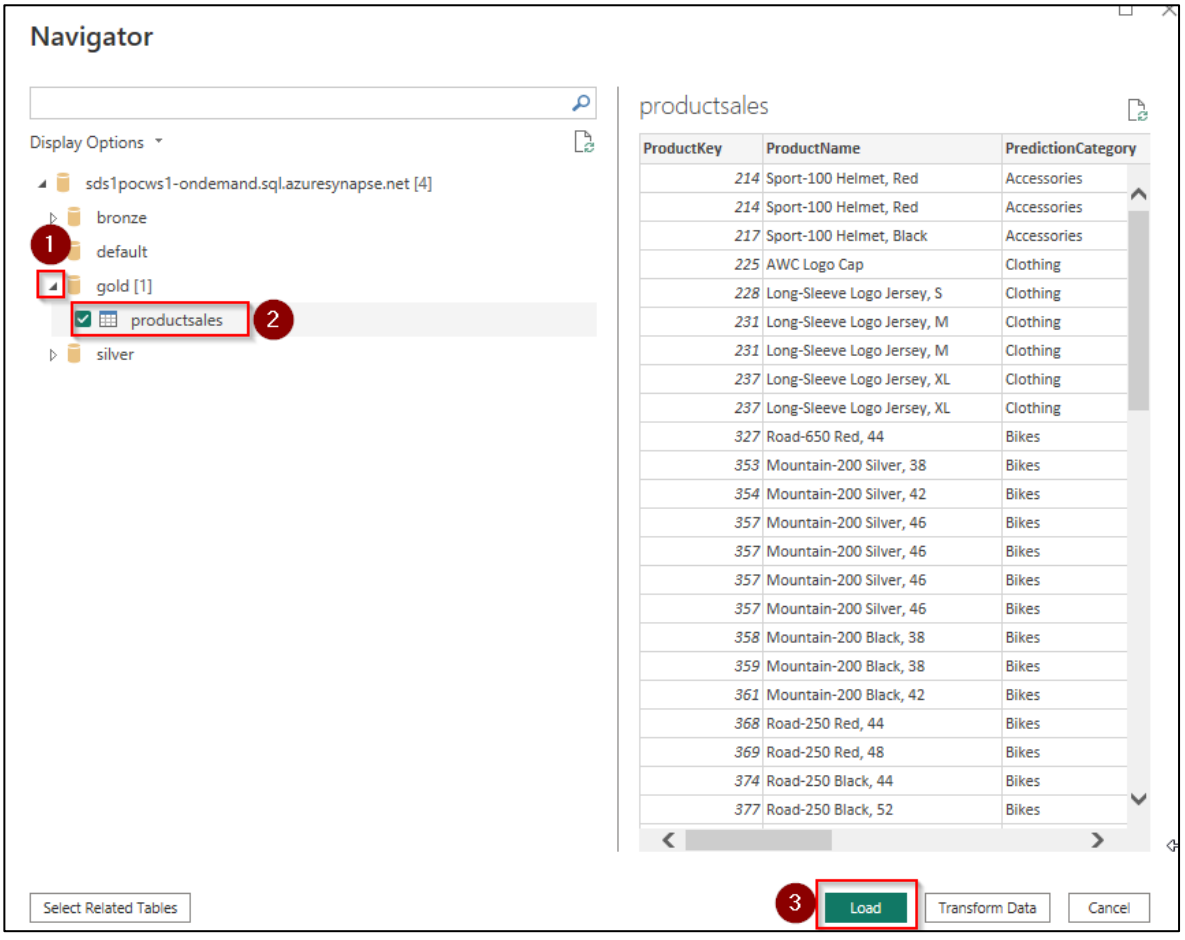


6. **Copy** the Serverless SQL endpoint.
7. Go back to your Power BI Desktop application and paste the Serverless SQL endpoint in the **Server** field and select **OK**.

8. A new window will open asking for the credentials. Select **Microsoft account**, select **Sign in** and select the **Connect** button.

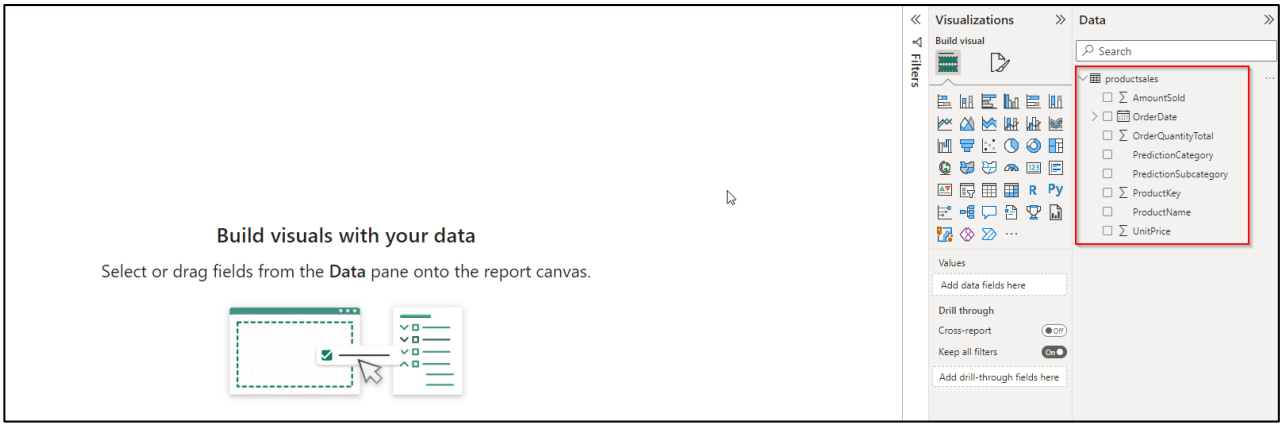


9. A new window will open showing the databases we created in chapters Create Azure Synapse SQL Database (OPTIONAL) and Create Azure Synapse Lake Database.
10. Open the **gold database** and select the **productsales** delta table.
11. Select **Load** to import the data directly into Power BI without calling the Power Query Editor.



12. A loading screen will appear.

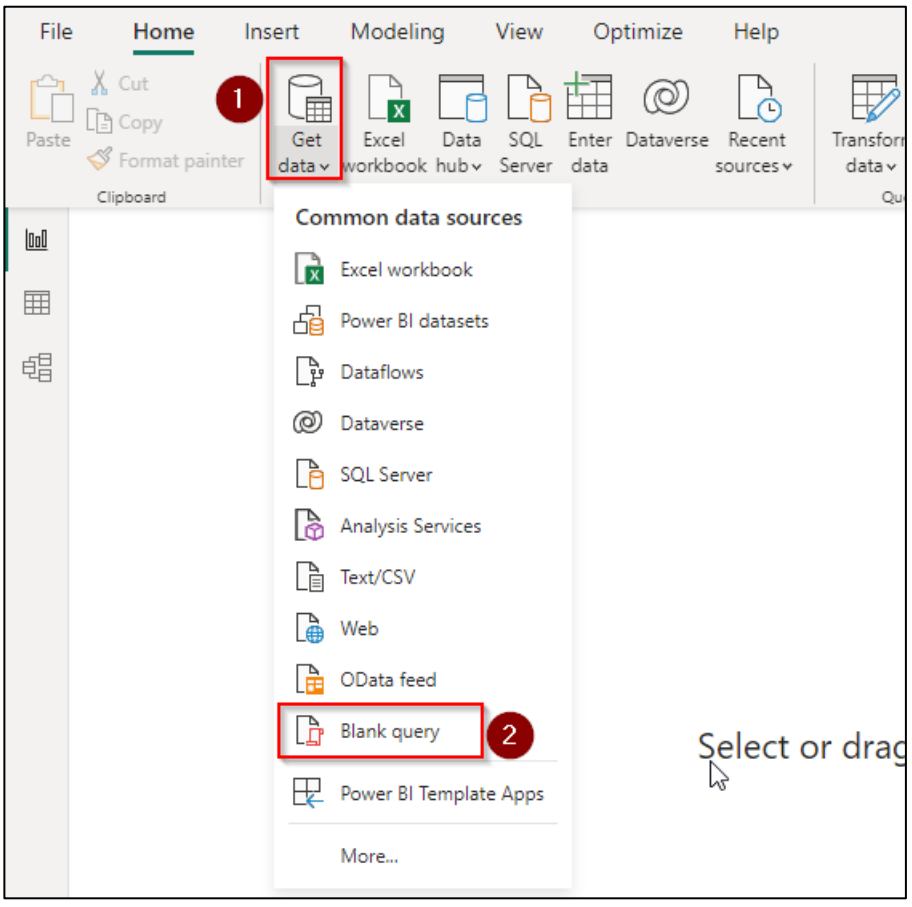
13. The data is now imported and available to be used for visualizations in Power BI.



9.1.2 Custom Delta Lake Connector

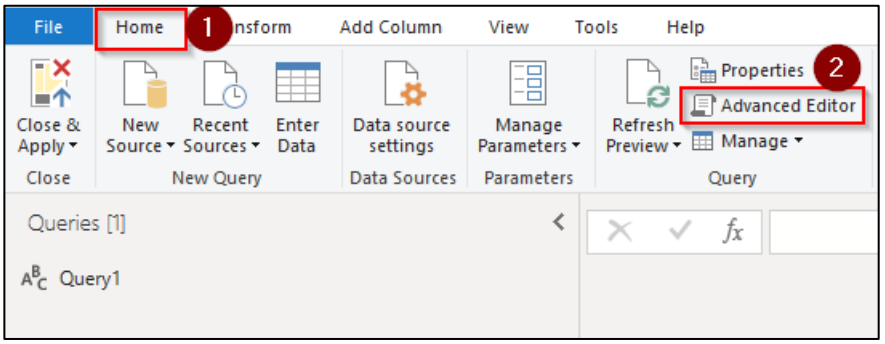
The provided connector allows you to read a Delta Lake table directly from any storage supported by Power BI such as the Azure Data Lake Gen2 we provisioned. Additional information can be found here.

1. Open the **Power BI Desktop** application.
2. Once opened, select the **Get data** button and select **Blank query**.

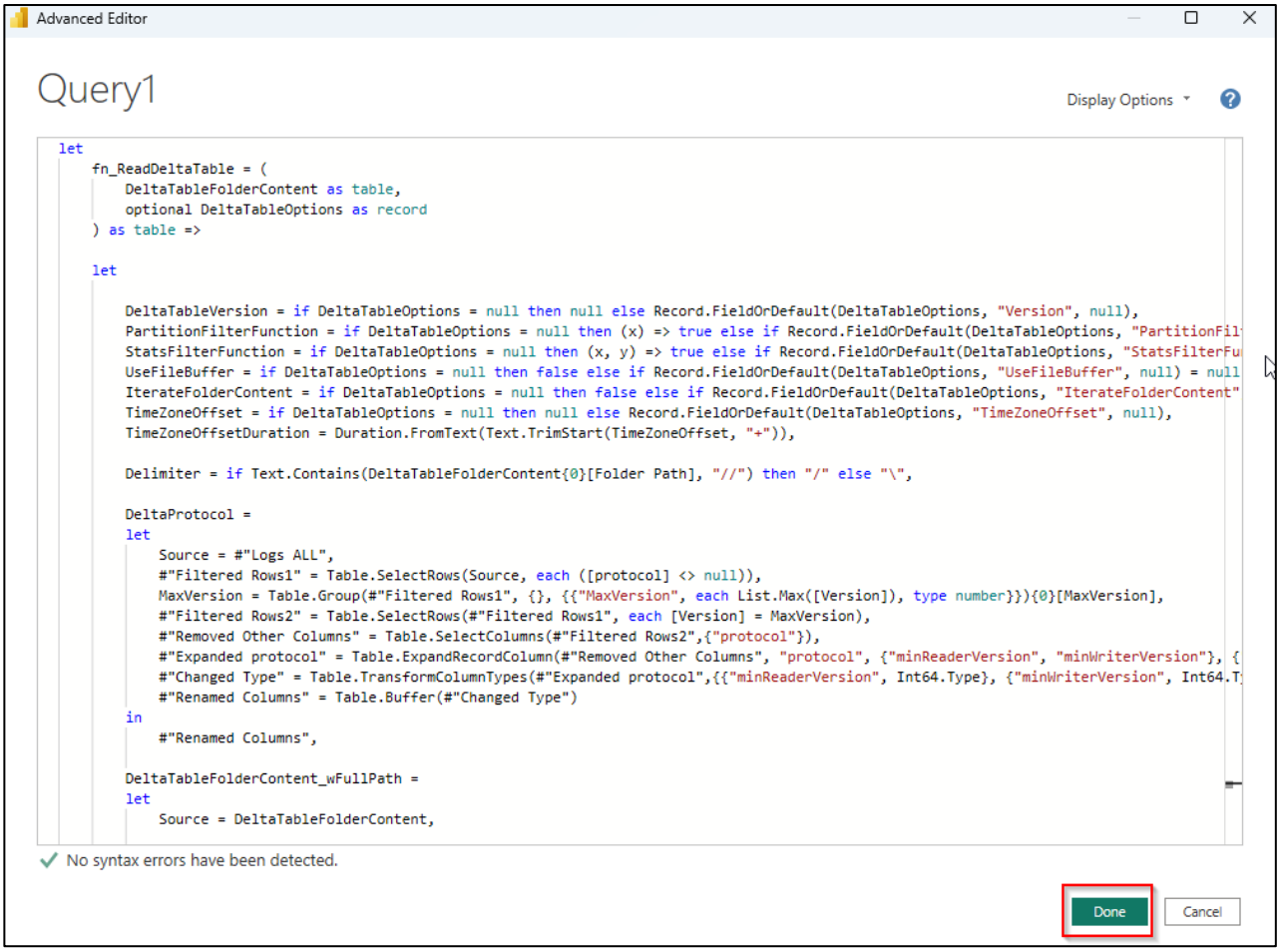


This will open the Power Query Editor.

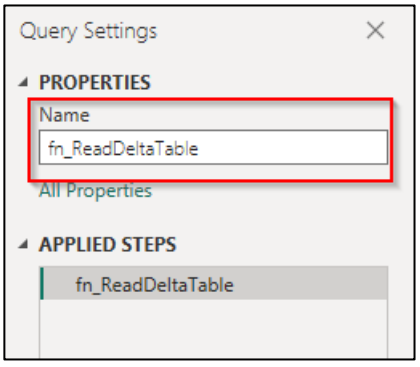
3. Within the Power Query Editor, in the **Home** tab, select **Advanced Editor**.



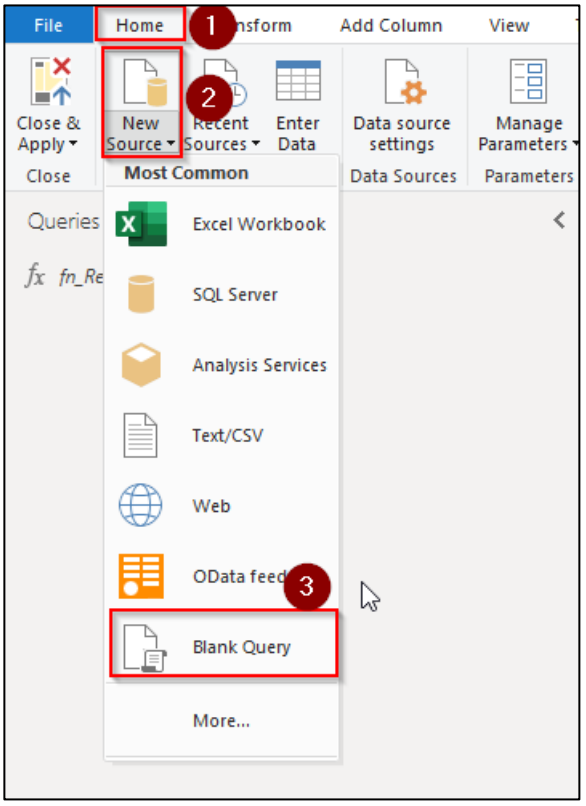
4. A new window will appear.
5. **Copy and paste** the custom M-Code function made available here into the **Advanced Editor** window.
6. Select **Done**.



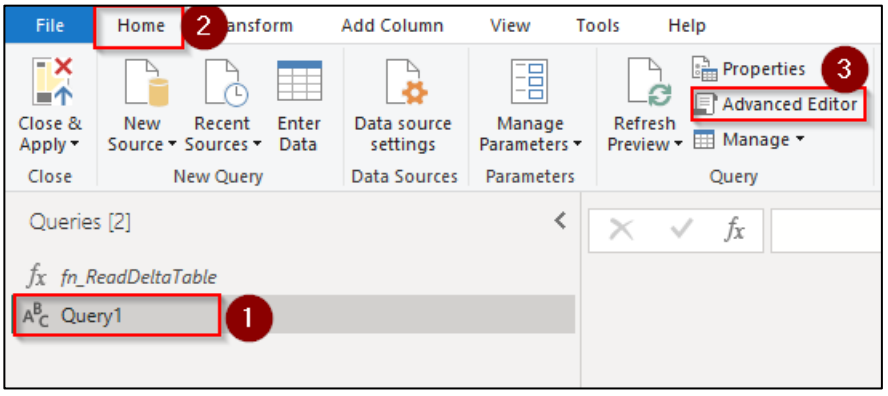
7. Under **Query Settings** located on the right panel, rename this Blank query to **fn_ReadDeltaTable.**



8. Create a new Blank Query within the Power Query Editor by selecting **New Source** followed by **Blank Query**.



9. Within the Power Query Editor, with the new Blank Query selected, go to the **Home** tab, select **Advanced Editor**.



10. A new window will appear.
11. Add the following M-Code into the **Advanced Editor** window.

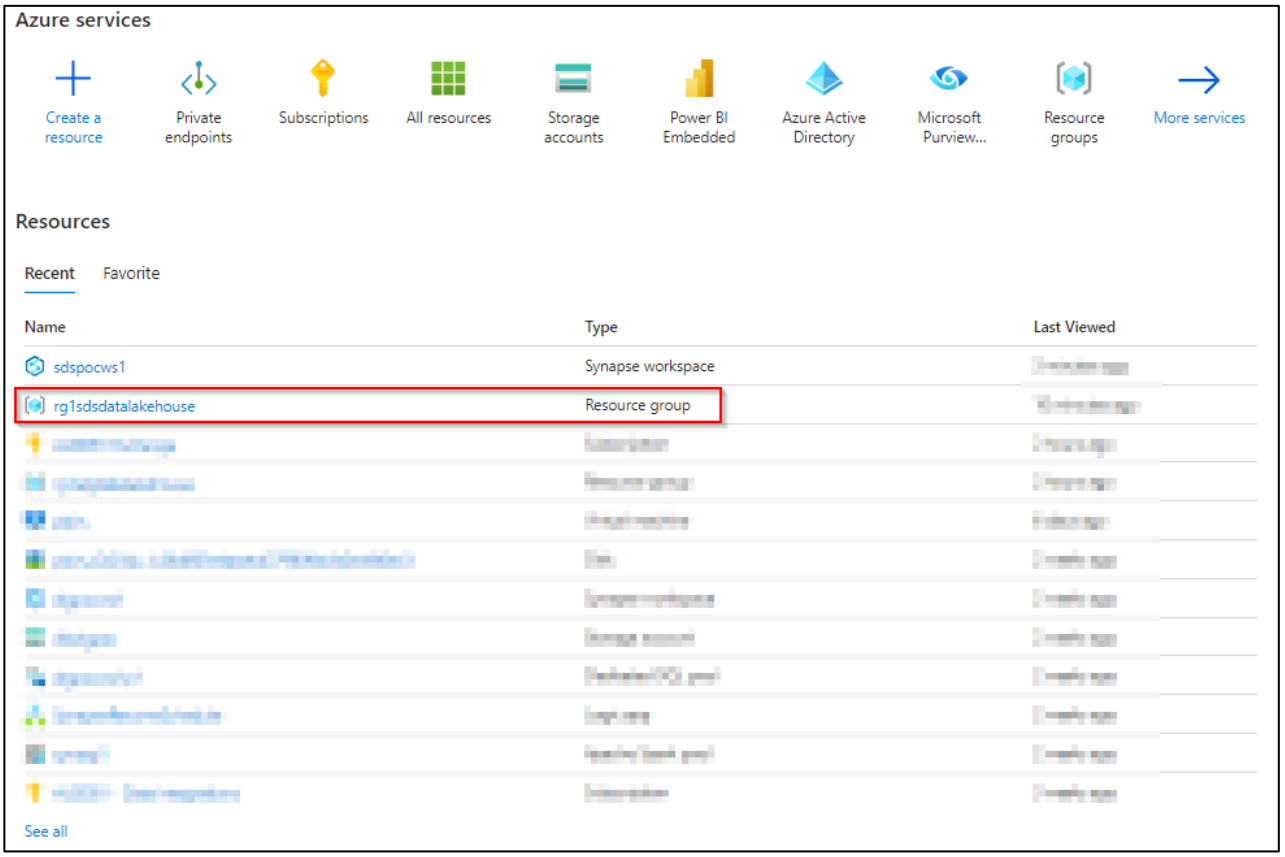
NOTE: You will need to change the AzureStorage.DataLake function below with your own data lake storage endpoint and container name. Follow the next steps in chapter Data Lake Storage endpoint and Container Name and add these missing values.

```
let
   #"Product Sales" = fn_ReadDeltaTable(
   AzureStorage.DataLake(
      "Your Data Lake Storage Endpoint/Your Container Name/gold-container/ProductSales",
      [HierarchicalNavigation = false]),
   [Version = 0])
in
   #"Product Sales"
```

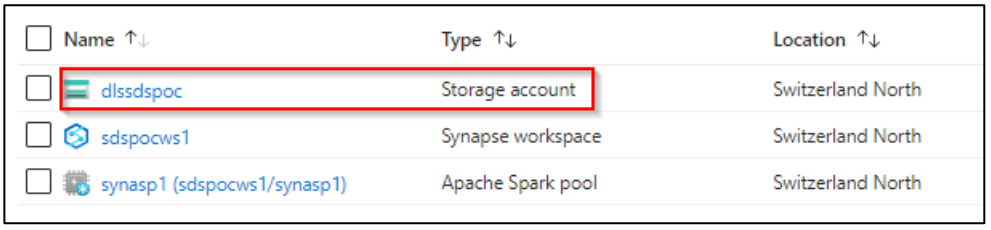
9.1.2.1 Data Lake Storage endpoint

The **Data Lake Storage endpoint** and **container name** can be found in our provisioned storage.

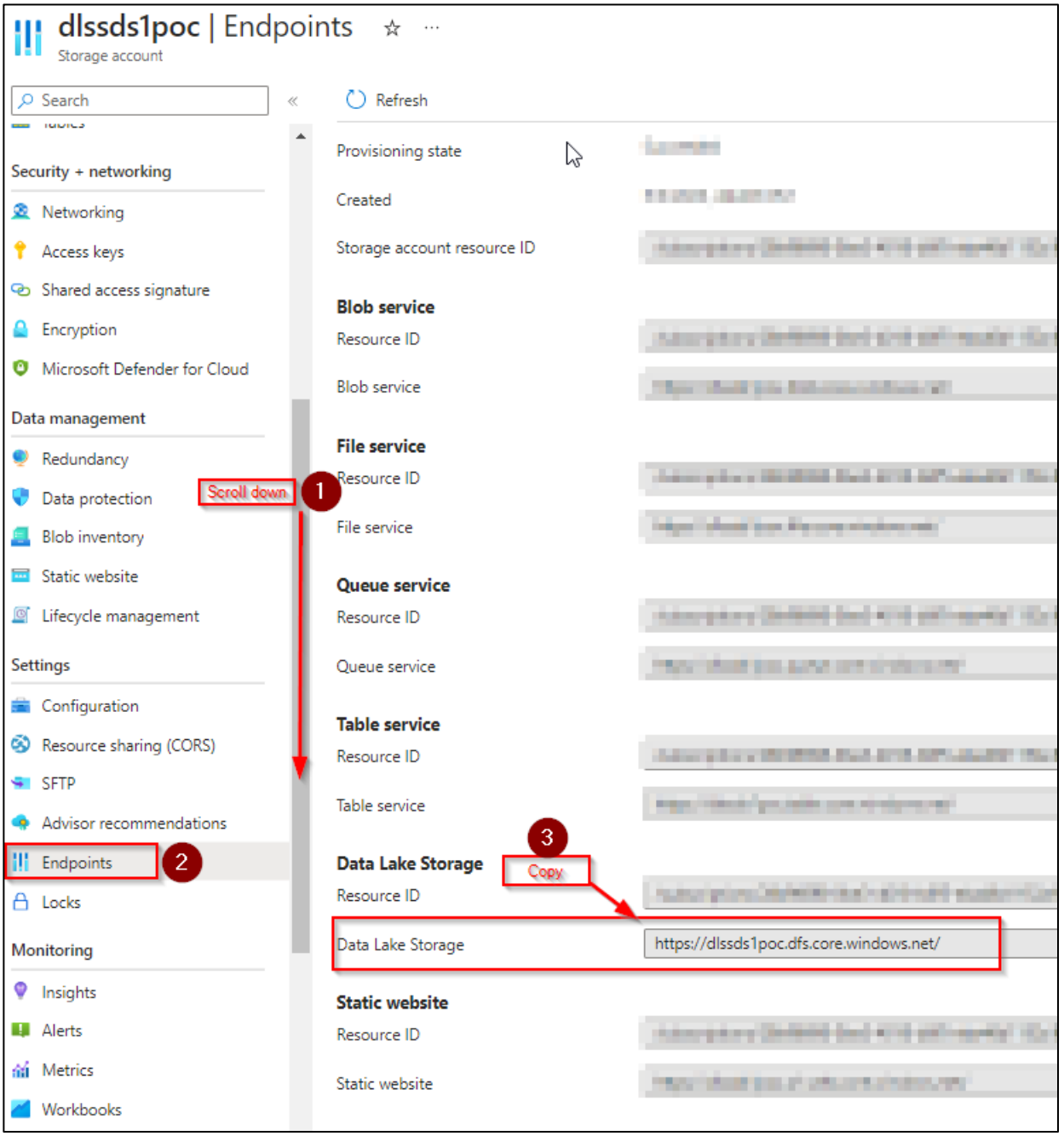
1. Go back to the Azure portal home screen and select the **Resource group** you provisioned.



2. Once the resource group is open, select the **Storage account**.

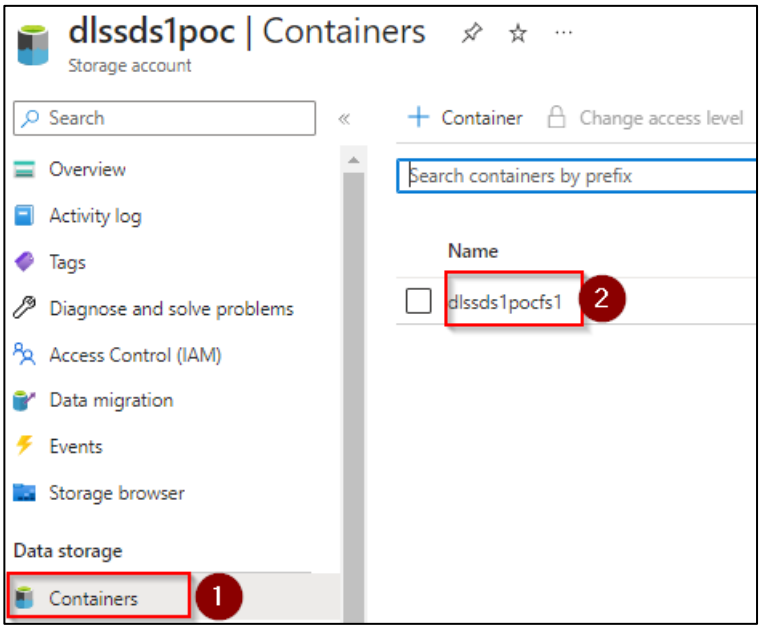


3. Within the storage account, select **Endpoints** on the left panel (You will need to scroll down).
4. **Copy** the Data Lake Storage Endpoint.



9.1.2.2 Container Name

5. Within the storage account, select **Container** on the left panel.
6. **Copy** the **Container** name (step 2 in the image).

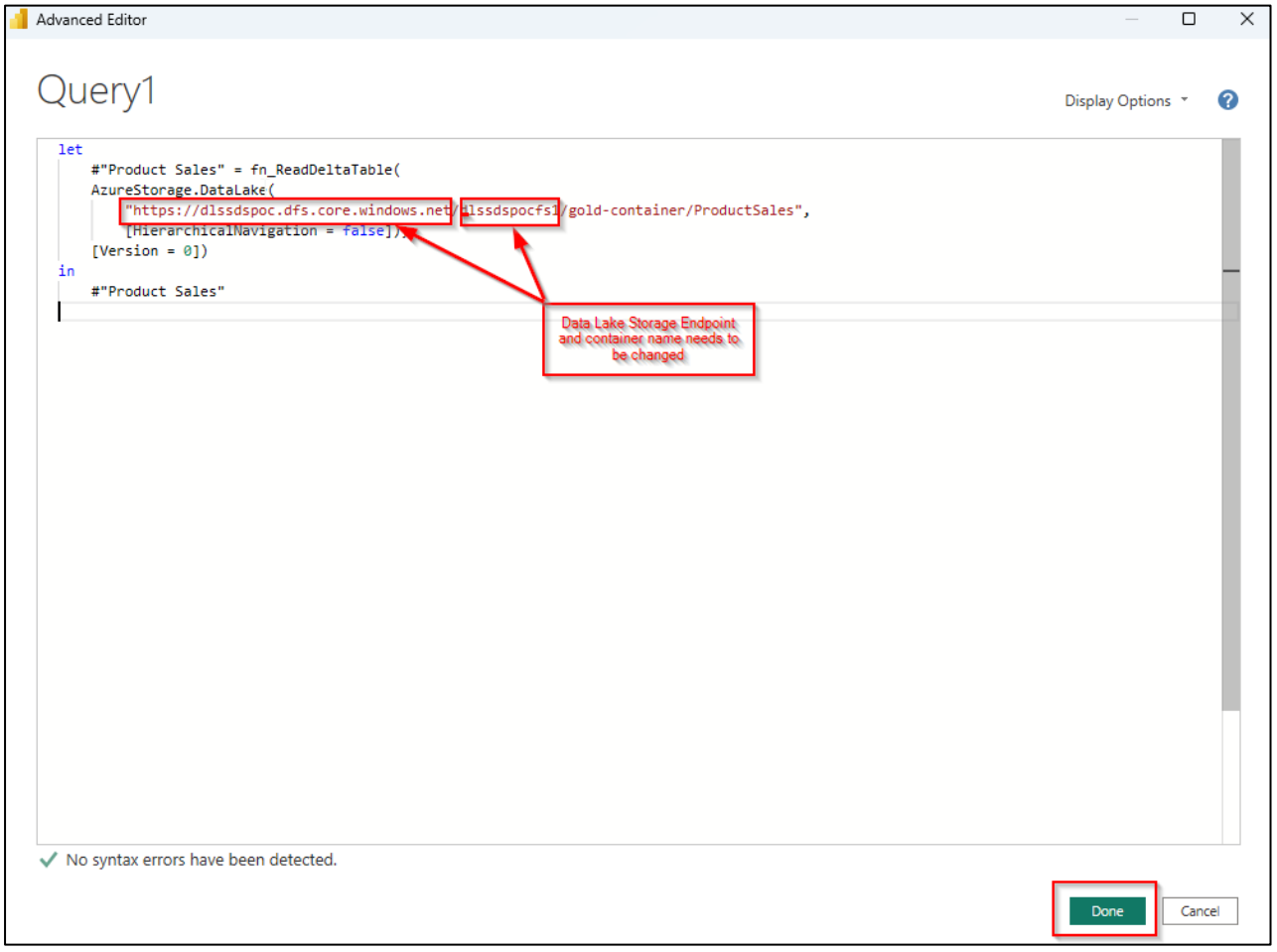


7. Go back to the **Power Query Editor.**
8. Adjust the M-Code in the **Advanced Editor** with the **Data Lake Storage Endpoint** and **container name** you copied before.

NOTE: Do NOT copy the same data lake storage endpoint and container name as the M-Code below (marked in red). This is only an example.

```
let
   #"Product Sales" = fn_ReadDeltaTable(
   AzureStorage.DataLake(
      "https://dlssdspoc.dfs.core.windows.net/dlssdspocfs1/gold-container/ProductSales",
      [HierarchicalNavigation = false]),
   [Version = 0])
in
   #"Product Sales"
```

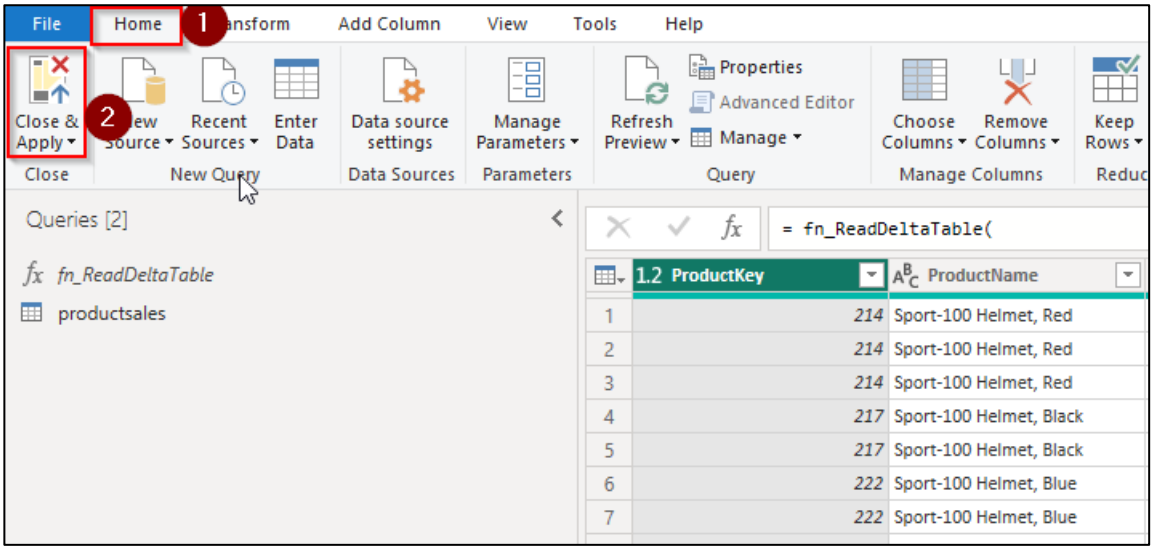
9. Select **Done.**



Notice that you can adjust the Version of the Delta table you are querying, by adjusting the Version parameter within the M-Code. This is identical to the time traveling we did in chapter Time Traveling . Unfortunately, time traveling using a Timestamp is not currently supported by this connector.

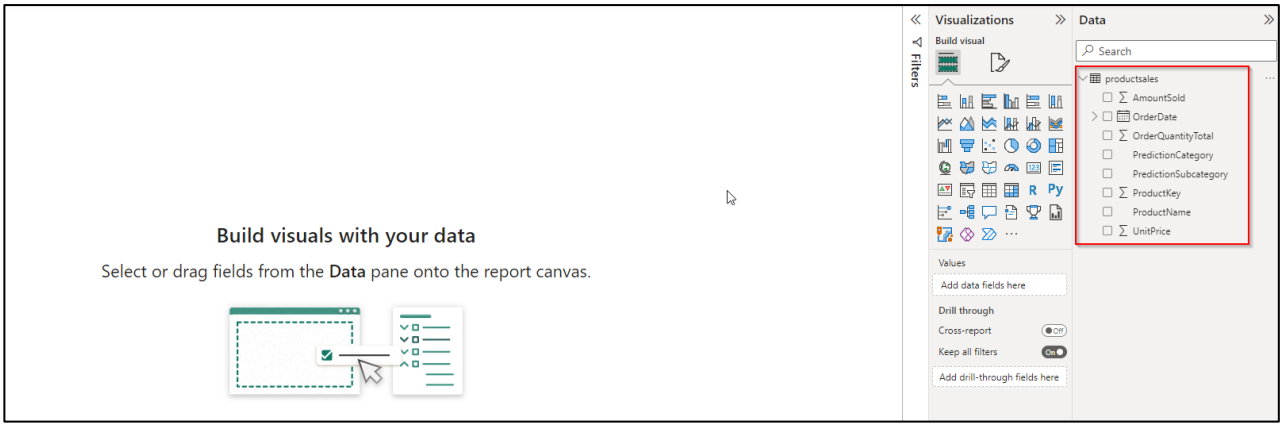
You can now import your data into Power BI.

10. Select **Close & Apply** button within the Power Query Editor to load the data into Power BI



14. A loading screen will appear.

15. The data is now imported and available to be used for visualizations in Power BI.



10 References

Browse Azure Architectures - Azure Architecture Center | Microsoft Learn
What is the medallion lakehouse architecture? - Azure Databricks | Microsoft Learn
Medallion architecture: best practices for managing Bronze, Silver and Gold | by Piethein Strengholt | Medium
The Fundamentals of Delta Lake | Medium
Delta lake ETL with data flows - Azure Data Factory | Microsoft Learn
Building the Lakehouse - Implementing a Data Lake Strategy with Azure Synapse - Microsoft Community Hub
What is a Lake Database in Azure Synapse Analytics? (endjin.com)
Microsoft BI Tools: Synapse - Add existing Delta Table to Lake Database (microsoft-bitools.blogspot.com)
What is the difference between "SQL Database" and "Lake Database" in Synapse - Microsoft Q&A
Azure Synapse Analytics | Spark pool | Delta Lake - Part 1 - YouTube
Microsoft BI Tools: Synapse - Using Spark SQL to time travel Delta Tables (microsoft-bitools.blogspot.com)
Synapse Espresso: Timetravel with Delta tables in Azure Synapse Spark! - YouTube
Azure Synapse Analytics | Spark pool | Delta Lake - Part 1 - YouTube
Catching Up With Delta Lake in Azure Synapse - Iteration Insights