

Microsoft Fabric End-to-End Data Pipeline



Table Of Content

1	Introduction	2
2	Project Scope	2
3	Technologies used	3
4	Architecture Diagram	4
5	Dataset overview	4
6	Step 1: Data Ingestion	5
7	Step 2: Medallion Architecture Implementation	5
8	Step 3: Creating Notebook to copy data from raw to landing folder	8
9	Step 4: Transfer data from raw to landing folder using pipeline	10
10	Step 5: Transfer data from landing folder to Bronze layer(table)	18
11	Step 6: Transfer data from Bronze layer to Silver layer(table)	20
12	Step 7: Transfer data from Silver layer to Gold layer(table)	23
13	Step 8: Creating End to End Pipeline to transfer data from raw folder to Golden layer	27
14	Step 9: Creating relation between tables(Modelling)	30
15	Step 10: Creation of Power BI Report	33
16	Conclusion	35

Introduction

The goal of this project is to design and implement a fully functional end-to-end data engineering and analytics solution using Microsoft Fabric. The solution will ingest raw data from source systems, process and refine it through multiple transformation layers (such as bronze, silver, and gold in a medallion architecture), implement scalable data pipelines using Fabric components (e.g., pipelines, notebooks), and finally deliver actionable insights through reporting and analytics. The project will demonstrate how to build an integrated data platform that supports data cleansing, transformation, storage, and visualization at scale, ensuring data quality, governance, and performance throughout the pipeline.

Project Scope

1. Data Ingestion

- Collect data from various structured and unstructured data sources.
- Implement automated ingestion using Microsoft Fabric data pipelines.
- Ensure ingestion supports incremental and batch loads.

2. Medallion Architecture Implementation

- **Bronze Layer:** Store raw ingested data in its original form.
- **Silver Layer:** Clean, join, filter, and enrich the raw data.
- **Gold Layer:** Aggregate refined data to support analytics and reporting use cases.

3. Data Transformation and Processing

- Use PySpark notebooks within Microsoft Fabric to apply business logic to data.
- Manage transformation workflows and dependencies between pipeline stages.

4. Data Storage and Optimization

- Establish Lakehouse storage with structured tables optimized for querying.
- Implement partitioning, schema enforcement, and performance optimization practices.

5. Reporting & Visualization

- Build dashboards and reports (e.g., using Power BI connected to Fabric outputs).
- Ensure insights are consumable by business users.

Technologies used

🎯 Key Technologies

- Microsoft Fabric platform (Lakehouse, Data Pipelines, Notebooks)
- PySpark for data transformation and enrichment
- Power BI analytics
- Medallion architecture for data refinement

💡 What is Medallion Architecture?

Medallion Architecture is a **layered data design pattern** used in modern data platforms (like Microsoft Fabric, Databricks, Azure) to progressively refine data from **raw to business-ready** formats.

🔄 Core Idea

Data quality and value **increase** as data moves through layers.

🏅 Medallion Layers

🥉 Bronze Layer (Raw Data)

- Stores data **as-is** from source systems

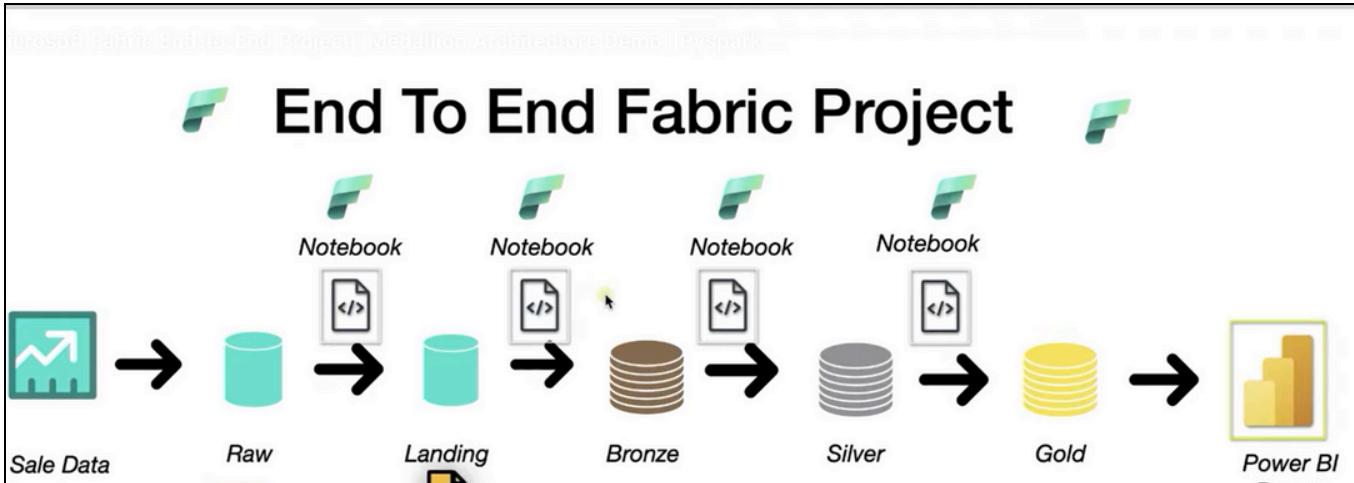
🥈 Silver Layer (Cleaned Data)

- Data is cleaned, validated, and enriched
- Duplicates removed, nulls handled
- Business rules applied

🥇 Gold Layer (Business Data)

- Aggregated and curated data
- Optimized for reporting and analytics
- Used by BI tools

Architecture Diagram



Datasets

Source Data				
	Name	Date modified	Type	Size
Desktop	2016 July	12/28/2025 2:41 PM	Microsoft Excel Work...	363 KB
Downloads	2016Apr	12/28/2025 2:41 PM	Microsoft Excel Work...	352 KB
Documents	2016Feb	12/28/2025 2:41 PM	Microsoft Excel Work...	294 KB
Pictures	2016Jun	12/28/2025 2:41 PM	Microsoft Excel Work...	363 KB
Music	2016Mar	12/28/2025 2:41 PM	Microsoft Excel Work...	352 KB
Videos	2016May	12/28/2025 2:41 PM	Microsoft Excel Work...	363 KB
resume				

2016Feb.xlsx — LibreOffice Calc																			
File Edit View Insert Format Styles Sheet Data Tools Window Help																			
Calibri 11 pt B I U A fx Row ID																			
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Postal Code	City	State	Country	Region	Market	Product ID	Category	Sub-Category	Product Name	
2	10562	ES-2015-2015-08-02	2015-08-04	Second Class	JK-1564045	Jim Kriz	Home Office	Lille		Nord-Pas-de-Calais	France	Western Europe	TEC-CO-5	Technological	Copiers	Shared			
3	15088	ES-2015-2015-08-23	2015-08-25	First Class	JL-1523545	Janet Lee	Consumer	Bordeaux		Aquitaine	France	Western Europe	TEC-CO-5	Technological	Copiers	Shared			
4	16084	IT-2015-2015-08-19	2015-08-25	Standard Class	CM-1244545	Chuck Magee	Consumer	Villeneuve-d'Ascq		Nord-Pas-de-Calais	France	Western Europe	TEC-PH-5	Technological	Phones	Samsung			
5	11639	IT-2015-2015-08-07	2015-08-09	Second Class	CA-1231045	Christine Abe	Corporate	Menton		Provence-Alpes-Côte d'Azur	France	Western Europe	TEC-PH-5	Technological	Phones	Nokia			
6	14392	ES-2015-2015-08-12	2015-08-16	Standard Class	ZC-2191045	Zuschuss Car	Consumer	Le Pontet		Provence-Alpes-Côte d'Azur	France	Western Europe	OFF-ST-5	Office Supply	Storage	Roger			
7	12588	ES-2015-2015-08-06	2015-08-10	Standard Class	BD-1150045	Bradley Druck	Consumer	Marseille		Provence-Alpes-Côte d'Azur	France	Western Europe	OFF-ST-6	Office Supply	Storage	Terence			
8	14391	ES-2015-2015-08-12	2015-08-16	Standard Class	ZC-2191045	Zuschuss Car	Consumer	Le Pontet		Provence-Alpes-Côte d'Azur	France	Western Europe	FUR-BQ-9	Furniture	Bookcase	Saf			
9	11640	IT-2015-2015-08-07	2015-08-09	Second Class	CA-1231045	Christine Abe	Corporate	Menton		Provence-Alpes-Côte d'Azur	France	Western Europe	FUR-CH-9	Furniture	Chairs	Saf			
10	12067	ES-2015-2015-08-21	2015-08-27	Standard Class	AH-1021045	Alan Hwang	Consumer	La Seyne-sur-Mer		Provence-Alpes-Côte d'Azur	France	Western Europe	OFF-AP-4	Office Supply	Appliance	Holiday			
11	10810	ES-2015-2015-08-06	2015-08-10	Standard Class	JF-1541545	Jennifer Ferg	Consumer	Maubeuge		Nord-Pas-de-Calais	France	Western Europe	OFF-ST-4	Office Supply	Storage	Elton			
12	17504	ES-2015-2015-08-02	2015-08-07	Standard Class	AC-1066045	Anna Chung	Consumer	Brive-la-Gaillarde		Auvergne-Rhône-Alpes	France	Western Europe	OFF-ST-6	Office Supply	Storage	Terence			
13	15782	ES-2015-2015-08-11	2015-08-17	Standard Class	HJ-1487545	Heather Jas	Home Office	Rillieux-la-Pape		Auvergne-Rhône-Alpes	France	Western Europe	OFF-ST-4	Office Supply	Storage	Fell			
14	13801	ES-2015-2015-08-05	2015-08-11	Standard Class	NM-1852045	Neoma Murr	Consumer	Chartres		Centre-Val de Loire	France	Western Europe	OFF-ST-6	Office Supply	Storage	Terence			
15	13265	ES-2015-2015-08-09	2015-08-15	Standard Class	EN-1378045	Edward Nazz	Consumer	Albi		Languedoc-Roussillon	France	Western Europe	FUR-CH-9	Furniture	Chairs	Off			
16	18344	ES-2015-2015-08-08	2015-08-14	Standard Class	CS-1249045	Cindy Schnell	Corporate	Lille		Nord-Pas-de-Calais	France	Western Europe	FUR-CH-9	Furniture	Chairs	Holiday			
17	12158	IT-2015-2015-08-02	2015-08-06	Standard Class	TH-2110045	Thea Hendri	Consumer	Tarbes		Languedoc-Roussillon	France	Western Europe	FUR-BQ-9	Furniture	Bookcase	Saf			
18	15783	ES-2015-2015-08-11	2015-08-17	Standard Class	HJ-1487545	Heather Jas	Home Office	Rillieux-la-Pape		Auvergne-Rhône-Alpes	France	Western Europe	OFF-AP-4	Office Supply	Appliance	Holiday			
19	15162	ES-2015-2015-08-02	2015-08-03	First Class	RA-1994545	Ryan Akin	Consumer	Le Petit-Quesnoy		Normandy	France	Western Europe	FUR-BO-3	Furniture	Bookcase	Bus			
20	48899	NI-2015-2015-08-12	2015-08-17	Standard Class	SF-1020095	Sarah Foster	Consumer	Lagos		Nigeria	Western Africa	OFF-AR-3	Office Supply	Art	BIC				
21	48900	NI-2015-2015-08-12	2015-08-17	Standard Class	SF-1020095	Sarah Foster	Consumer	Lagos		Nigeria	Western Africa	TEC-MA-9	Technological	Machines	Korean				

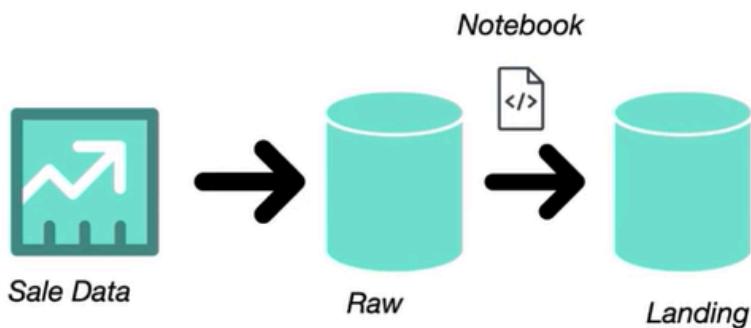
Step 1: Data Ingestion

Creating New Workspace:

The screenshot shows the Power BI Home interface. On the left, there's a sidebar with various icons for Home, Copilot, Create, Browse, OneLake catalog, Workspaces, and Power BI. The main area displays 'Recommended' workspaces: 'My workspace' (frequently open), 'Project_1_New' (frequently open), and 'project2_final2'. Below this is a 'Recent' section with 'Sales BI Report' (Report, opened a day ago) and 'Dev_Workspace' (Workspace, opened a day ago). On the right, a 'Create a workspace' dialog is open, prompting for a 'Name' (Dev_Workspace1, available), 'Description' (Microsoft Fabric End-to-End Project Workspace), 'Domain' (optional), 'Workspace image' (upload or reset), and 'Advanced' settings (Contact list: SupriyaBalajiGir (Owner)). Buttons for 'Apply' and 'Cancel' are at the bottom.

Step 2: Medallion Architecture Implementation

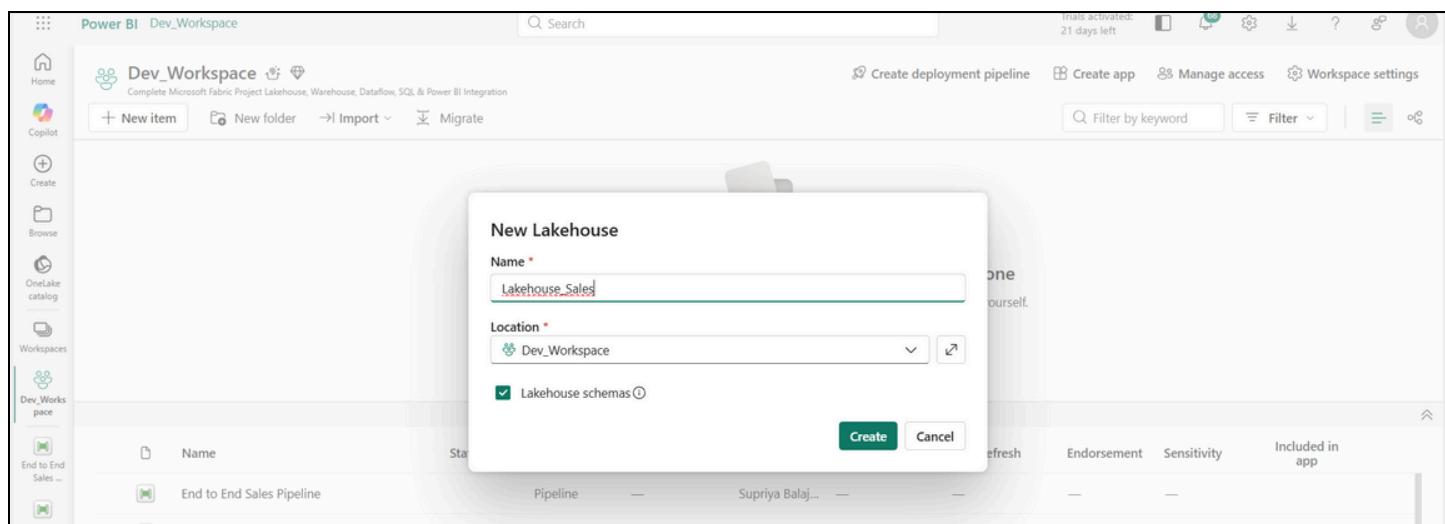
Raw to Landing Layer



Monthly only one file will be come from sales Department, and we will save it to raw folder.

Pipeline will pick up latest file from raw folder based on last modified date and will be copied to landing folder and Processing date will be added.

Creating Lakehouse in Fabric:

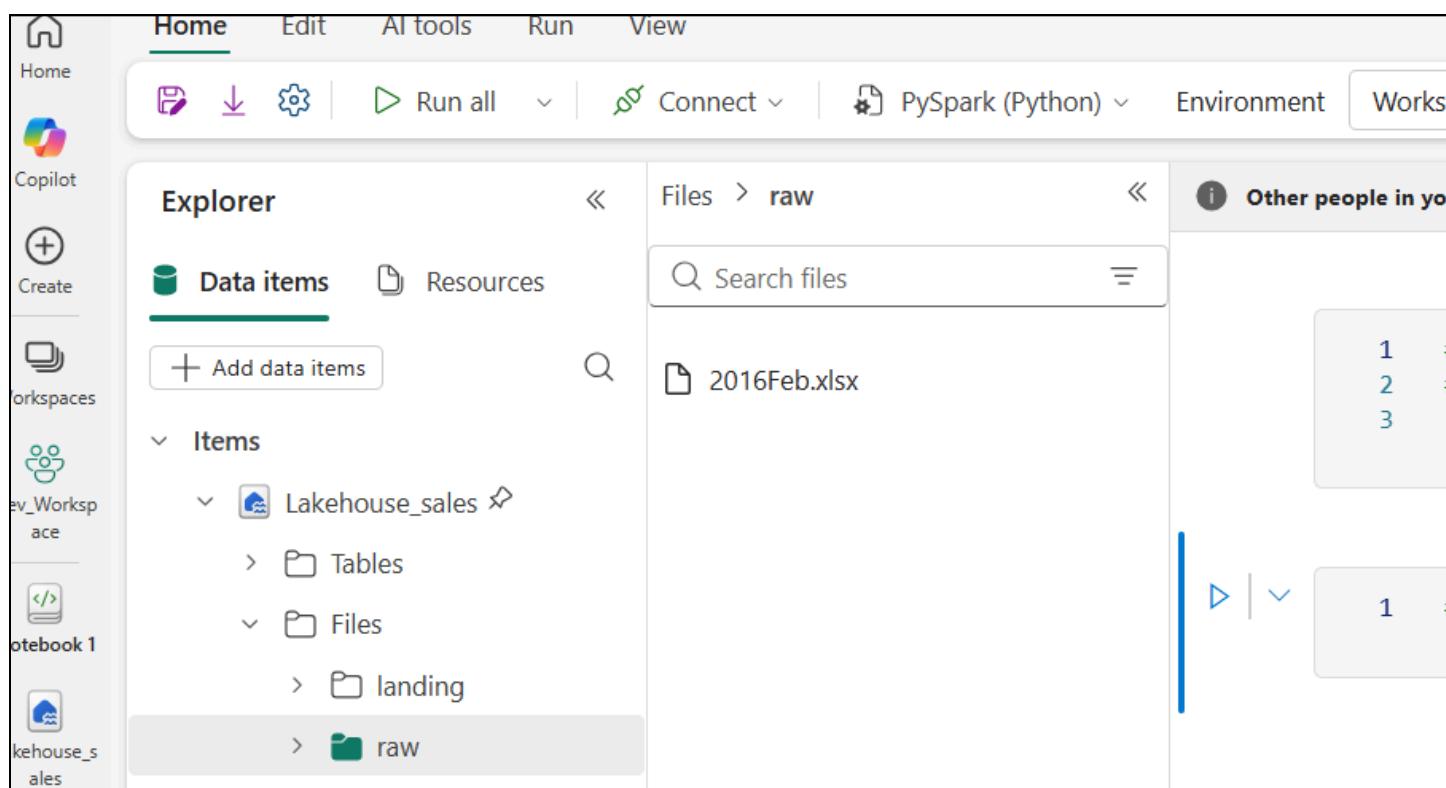
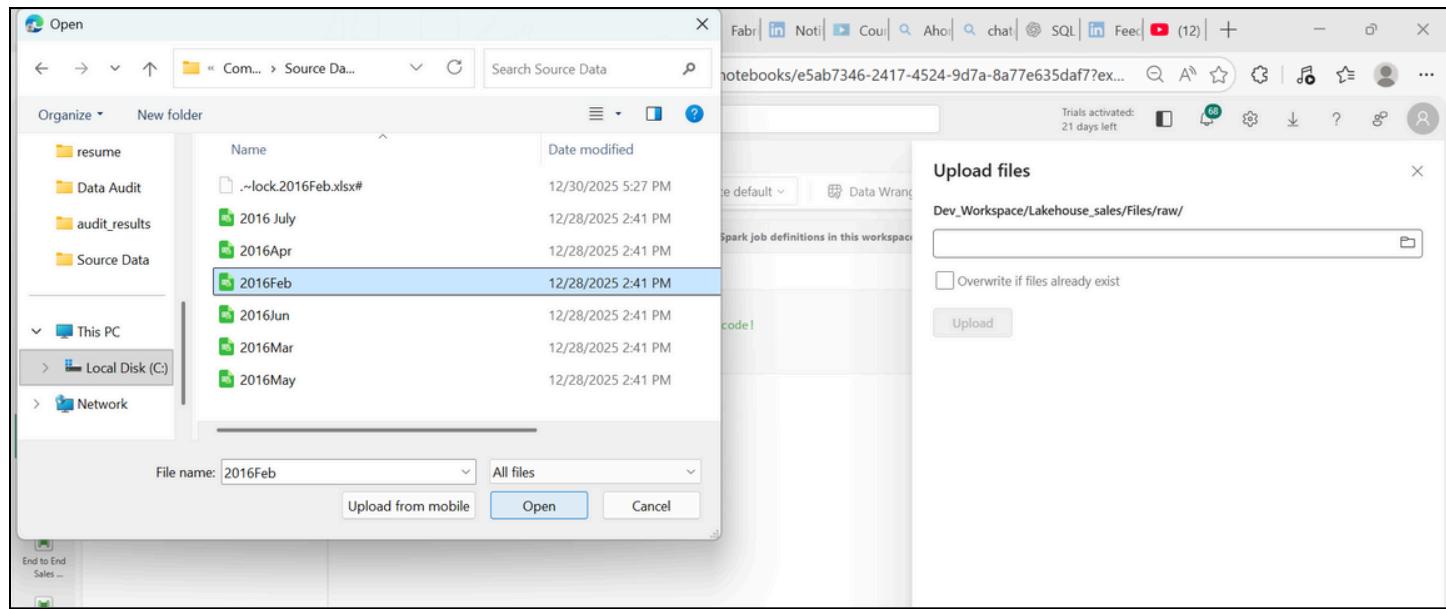


Create 2 folders under files section of lakehouse:

The screenshot shows a Microsoft Fabric Notebook 1 interface. The left sidebar lists workspaces: "Dev_Workspace" and "Lakehouse_sales". The main area has tabs for "Home", "Edit", "AI tools", "Run", and "View". The "Home" tab is selected. The "Explorer" panel on the left shows a tree structure under "Data items": "Items" > "Lakehouse_sales" > "Tables" > "Files". Inside "Files", there are two sub-folders: "landing" and "raw". The right side of the screen displays a code editor with the following content:

```
1 # Welcome to your new notebook
2 # Type here in the cell editor to add code!
3
```

Upload file to raw folder:



Step 3: Creating Notebook to copy data from raw to landing folder

In notebook first take two parameters, later these parameters will get data from pipeline. For timebeing we are hardcoding it:

```
1 file_name = '2016Feb.csv'
2 processed_date = '2025-12-28'

[4] ✓ - Command executed in 332 ms by Supriya Balaji Gir on 12/28/2025, 8:07:56 PM
```

Taking path of raw folder:

```
1 adfs_path = 'abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw'
2
3 full_path = f'{adfs_path}/{file_name}'
4 print(full_path)

[4] ✓ - Command executed in 268 ms by Supriya Balaji Gir on 12/28/2025, 8:07:58 PM
```

Read and display data to pandas Dataframe:

The screenshot shows a Jupyter Notebook interface. On the left, there's a sidebar with a tree view of data items, including 'raw_to_landing' and 'Lakehouse_sales'. The main area has a code cell and a preview pane. The code cell contains:1 df = spark.read.csv(path=full_path,header=True,inferSchema=True)
2 display(df)

The preview pane shows a table with 15 rows of data from the CSV file. The columns are: Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer Name, Segment, and Postal Code. The data includes various customer details like Jim Kriz, Janet Lee, Chuck Magee, etc., across different segments and postal codes.

Now load this Dataframe data to landing folder with todays' processing date:

```
1 from pyspark.sql.functions import lit
2
3 if df.count() > 1:
4     df_new = df.withColumn("Processing_date",lit(processed_date))
5
6     #display(df_new)
7     df_new.write.format("csv").option("header", "True").mode("append").partitionBy("Processing_date").save("abfss://Dev_Workspace@onelake."
8     print("Data loaded to Landing zone successfully")
9 else:
10     print("File contain no Data")

[7] ✓ - Command executed in 3 sec 246 ms by Supriya Balaji Gir on 12/28/2025, 8:08:31 PM
```

The output shows the command was executed successfully and the message "Data loaded to Landing zone successfully".

```

1 file_name = '2016Feb.csv'
2 processed_date = '2025-12-28'

[4] ✓ - Command executed in 332 ms by Supriya Balaji Gir on 12/28/2025, 8:07:56 PM
    abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw/2016Feb.csv

1 adfs_path = 'abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw/2016Feb.csv'
2
3 full_path = f'{adfs_path}/{file_name}'
4 print(full_path)

[5] ✓ - Command executed in 268 ms by Supriya Balaji Gir on 12/28/2025, 8:07:58 PM
    abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw/2016Feb.csv

1 df = spark.read.csv(path=full_path,header=True,inferSchema=True)
2 display(df)

[6] ✓ - Command executed in 19 sec 949 ms by Supriya Balaji Gir on 12/28/2025, 8:08:20 PM

```

Now for incremental data upload to landing folder for file of every month we have to make the processing_date filed and file name filed parameterized.

```

1 file_name = '2016Feb.csv'
2 processed_date = '2025-12-28'

[4] ✓ - Command executed in 332 ms by Supriya Balaji Gir on 12/28/2025, 8:07:56 PM
    abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw/2016Feb.csv

1 adfs_path = 'abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw/2016Feb.csv'
2
3 full_path = f'{adfs_path}/{file_name}'
4 print(full_path)

[5] ✓ - Command executed in 268 ms by Supriya Balaji Gir on 12/28/2025, 8:07:58 PM
    abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/raw/2016Feb.csv

```

Notebook should select current file of month and that days date as processing date:

Step 4: Transfer data from raw to landing folder using pipeline

Creating Data Pipeline for the same:

delete manually added file and we will add this file using pipeline.

The screenshot shows the Power BI Dev_Workspace interface. A modal window titled "New pipeline" is open, prompting the user to "Choose from one of these pre-built templates or start building one yourself." The "Name" field contains "Pipeline_raw_to_landing". Below the modal, a table lists existing pipelines: "End to End Sales Pipeline" (Pipeline) and "Lakehouse_sales" (Lakehouse). The workspace sidebar on the left includes items like "raw_to_landing", "Notebook 1", "Lakehouse_sales", "End to End Sales ...", and "Pipeline_raw_to_landing".

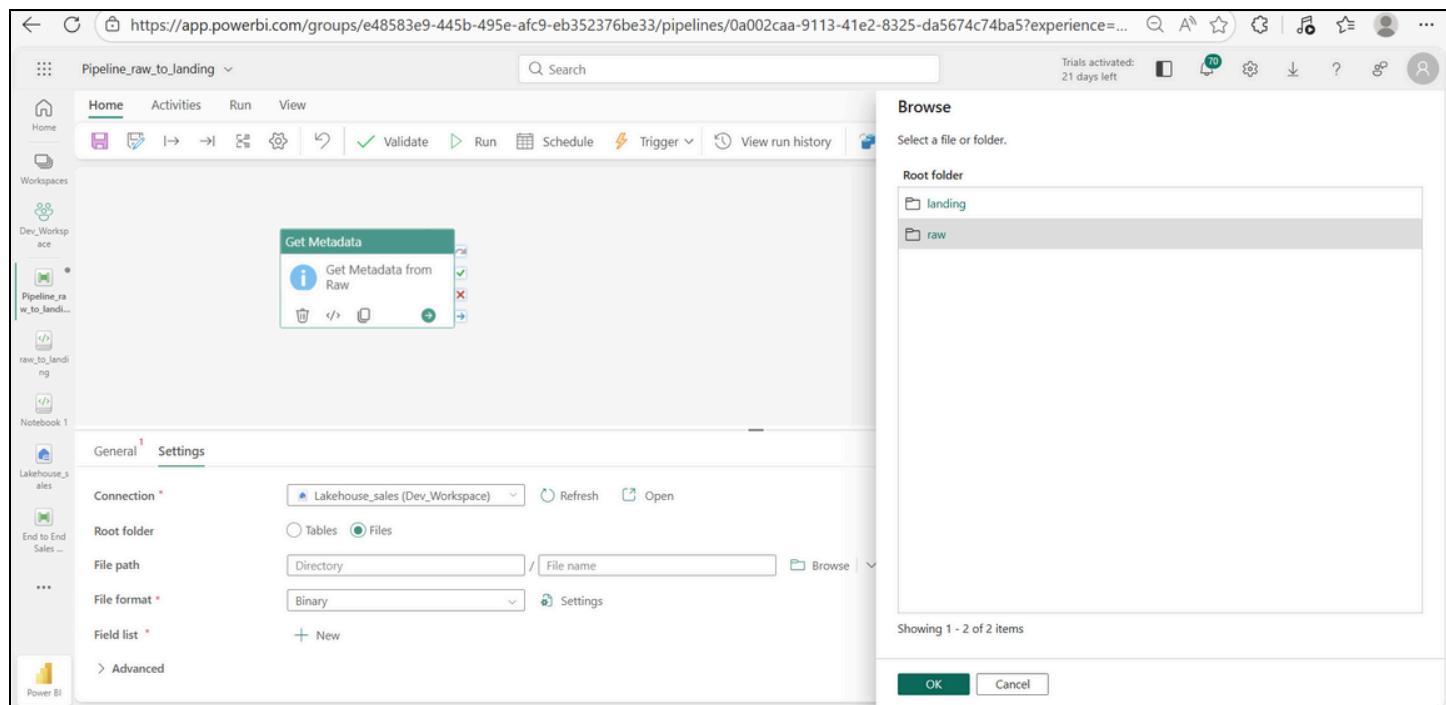
First add GET_Metadata activity to pipeline:

The screenshot shows the "Activities" tab for the "Pipeline_raw_to_landing" pipeline. It displays a single "Get Metadata" activity named "Get Metadata1". The workspace sidebar on the left shows "Pipeline_raw_to_landing" selected. The top navigation bar includes tabs for Home, Activities, Run, and View, along with various Power BI management and sharing icons.

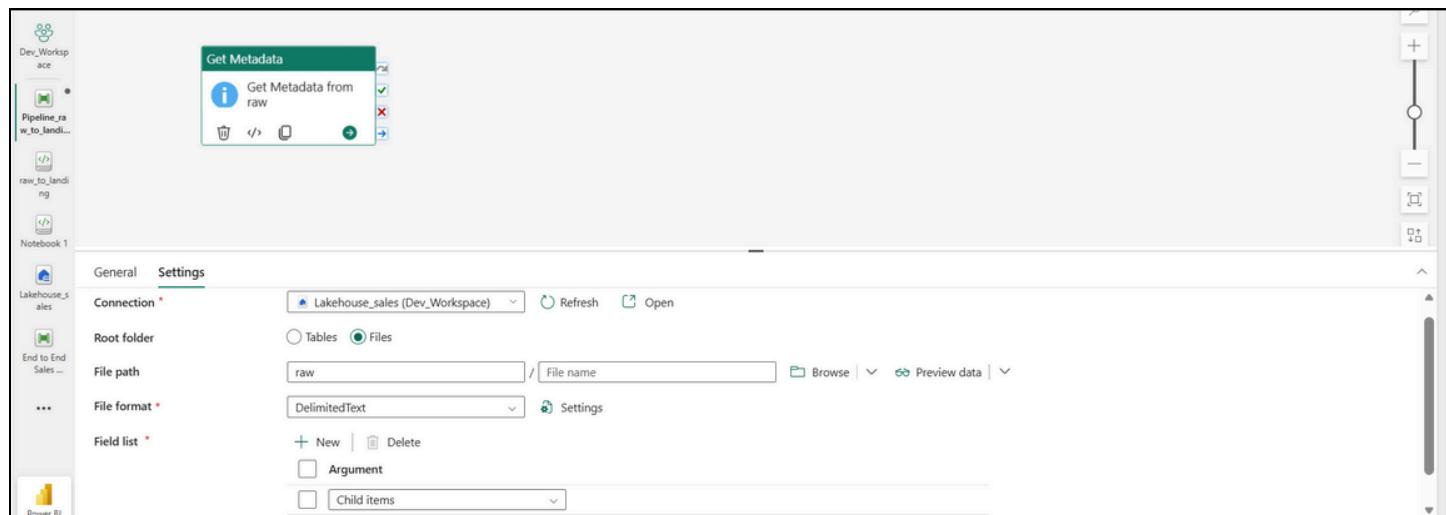
Select your lakehouse in setting tab of activity:

The screenshot shows the "Settings" tab for the "Get Metadata" activity. A modal window titled "Choose a data source to get started" is open, showing various data sources: SQL Server database, Azure SQL database, Folder, Azure Synapse Analytics (SQL DW), Azure Blobs, Azure Data Lake Storage Gen2, FTP, Hadoop Distributed File System, and OneLake catalog. Below this, the "OneLake catalog" section shows a list of lakehouses: "Lakehouse_sales", "Data_Audit_lakehouse", and "StadioLakehouseForDataflows 20...". The workspace sidebar on the left shows "Pipeline_raw_to_landing" selected. The top navigation bar includes tabs for Home, Activities, Run, and View, along with various Power BI management and sharing icons.

Select root folder as Files. Browse and select raw folder.



under file list select the child item, so that it will get the filename:



Select start and end time:

The screenshot shows the 'Get Metadata' activity configuration in the Azure Data Factory interface. At the top, there's a summary bar with a green 'Get Metadata from raw' status. Below it, the 'Settings' tab is selected. Under 'Field list *', there are buttons for '+ New' and 'Delete', and checkboxes for 'Argument' and 'Child items'. In the 'Advanced' section, there are fields for 'Start time (UTC)' set to '@startOfDay(utcNow())' and 'End time (UTC)' set to '@utcNow()'. A 'Skip line count' field is also present.

Now add file to raw folder and this file name will get in Get metadata activity of pipeline:

Save and run Activity:

The screenshot shows the 'Pipeline_raw_to_landing' run screen. A 'Save and run?' dialog box is centered over the pipeline activities. It contains the message: 'You have unsaved changes. If you continue, the pipeline will be saved and run.' with two buttons: 'Save and run' (highlighted in green) and 'Cancel'.

You get the data in output under childitems:

Run Succeeded
Successfully ran
Pipeline_raw_to_landing (Pipeline)

Output

```
{
  "childItems": [
    {
      "name": "2016Feb.xlsx",
      "type": "File"
    }
  ],
  "executionDuration": 2
}
```

Add Foreach Activity now to process this file through notebook previously created and load data to landing folder:

Get Metadata from childitems of Metadata Activity:

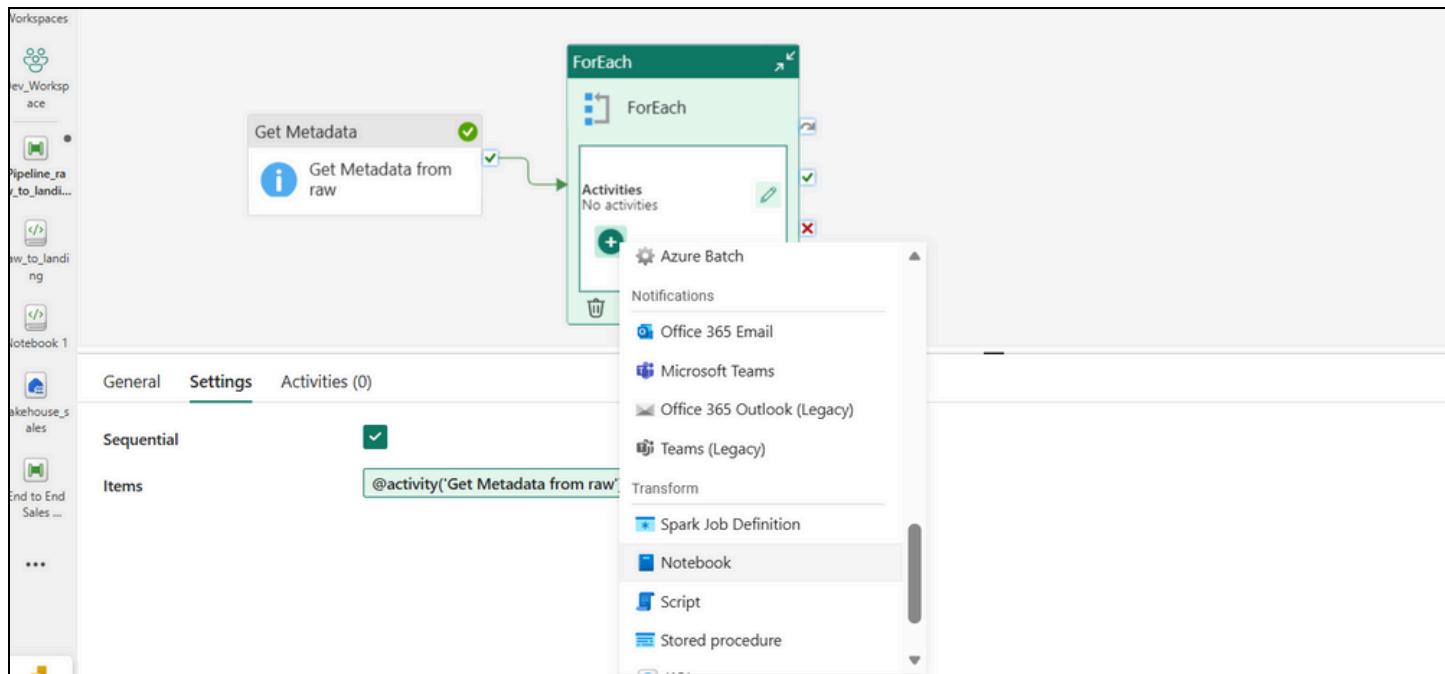
Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

Activity outputs Parameters System variables Trigger parameters ...

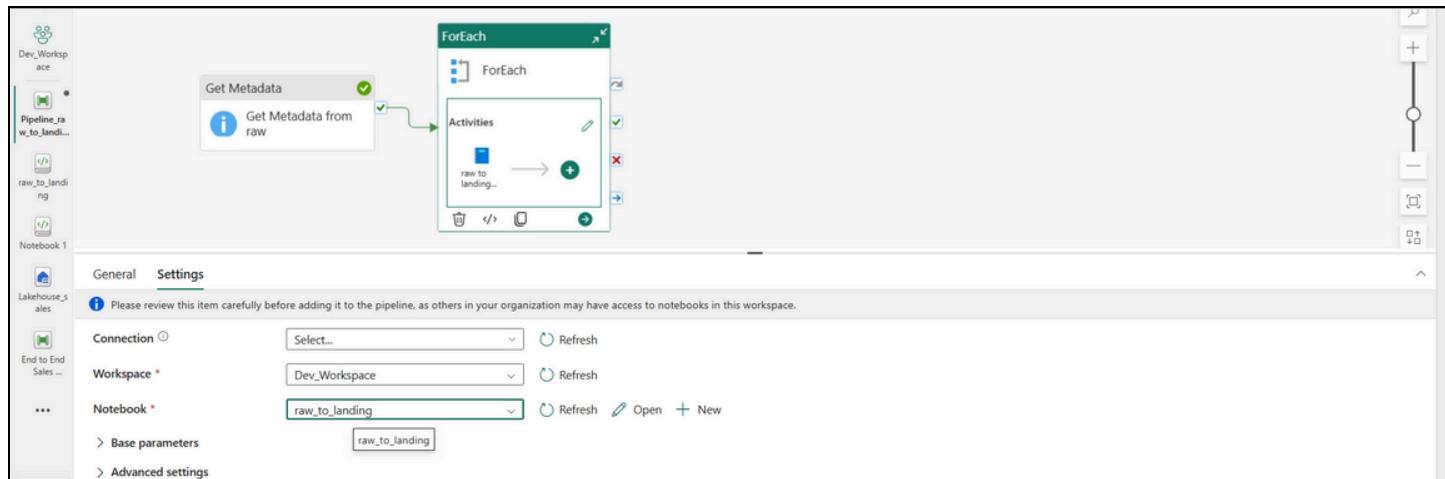
OK Cancel

Add Notebook inside this Foreach activity:



And this notebook will be linked to our raw_to_landing notebook which we have previously created:

Select Workspace and our Notebook:

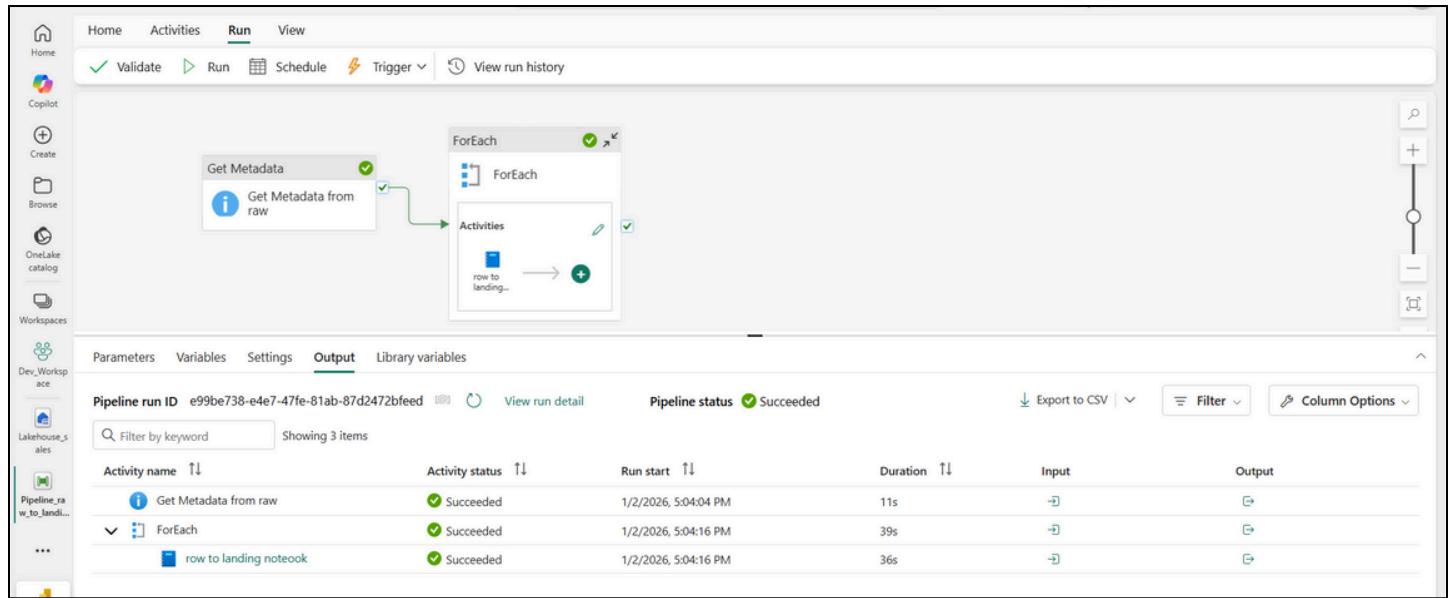


Now for this notebook we have to send parameters date and filename, so under base parameters add these parameters:

The screenshot shows the Microsoft Power BI Pipeline expression builder interface. On the left, a pipeline named "Pipeline_raw_to_landing" is displayed with a "Get Metadata from raw" activity followed by a "ForEach" loop containing an "Activities" activity. On the right, the "Pipeline expression builder" window is open, showing the expression "@item().name" in the main area. Below it are buttons for "Clear contents" and "Evaluate expression". A navigation bar at the bottom includes "ForEach iterator", "Activity outputs", "Parameters", "System variables", and "Functions".

This screenshot is similar to the one above, but the expression in the builder window has changed to "@formatDateTime(utcNow(), 'yyyy-MM-dd')". The right-hand panel now displays a detailed list of functions under the "Functions" tab, including "formatDateTime", "utcNow", "addTime", "addDays", and "addHours". The rest of the interface remains consistent with the first screenshot.

Save and run the pipeline:



In notebooks snapshot you can see current date:

The screenshot shows a Databricks notebook interface. The top navigation bar includes Refresh, Stop application, Monitor run series, and Spark History Server. Below the navigation, tabs for Jobs, Resources, Logs, Data, and Item snapshots are visible. The Item snapshots tab is selected.

The left sidebar shows a tree structure with Pipeline_raw_to_landing, raw_to_landing, and raw_to_landing selected. The main panel displays a cell titled 'raw_to_landing' with the following code and execution details:

```

1 file_name = '2016Feb.csv'
2 processed_date = '2025-12-28'

[1] ✓ - Command executed in 269 ms on 1/2/2026, 5:04:32 PM

```

Below the cell, a note states: '# This cell is generated from runtime parameters. Learn more: <https://go.microsoft.com/fwlink/?>'

```

1 # This cell is generated from runtime parameters. Learn more: https://go.microsoft.com/fwlink/?_
2 file_name = "2016Mar.xlsx"
3 processed_date = "2026-01-02"
4

[2] ✓ - Command executed in 319 ms on 1/2/2026, 5:04:33 PM

```

The Input parameter table shows the following values:

Content ↑	Type	Value
file_name	string	2016Mar.xlsx
processed_date	string	2026-01-02

Refresh Stop application Monitor run series Spark History Server

Jobs Resources Logs Data

Item snapshots

Item snapshots

- ✓ Pipeline_raw_to_landing
- ✓ raw_to_landing
- raw_to_landing

Snapshot: raw_to_landing

```

4     df_new = df.withColumn("Processing_date",lit(processed_date))
5
6     #display(df_new)
7     df_new.write.format('csv').option("header","True").mode("append").partitionBy("Processing_da
8     print("Data loaded to Landing zone successfully")
9 else:
10     print("File contain no Data")

```

[5] ✓ - Command executed in 2 sec 868 ms on 1/2/2026, 5:04:41 PM

> Diagnostics 1 |

Data loaded to Landing zone successfully

Input parameter

Content ↑	Type	Value
file_name	string	2016Mar.xlsx
processed_date	string	2026-01-02

Snapshot details

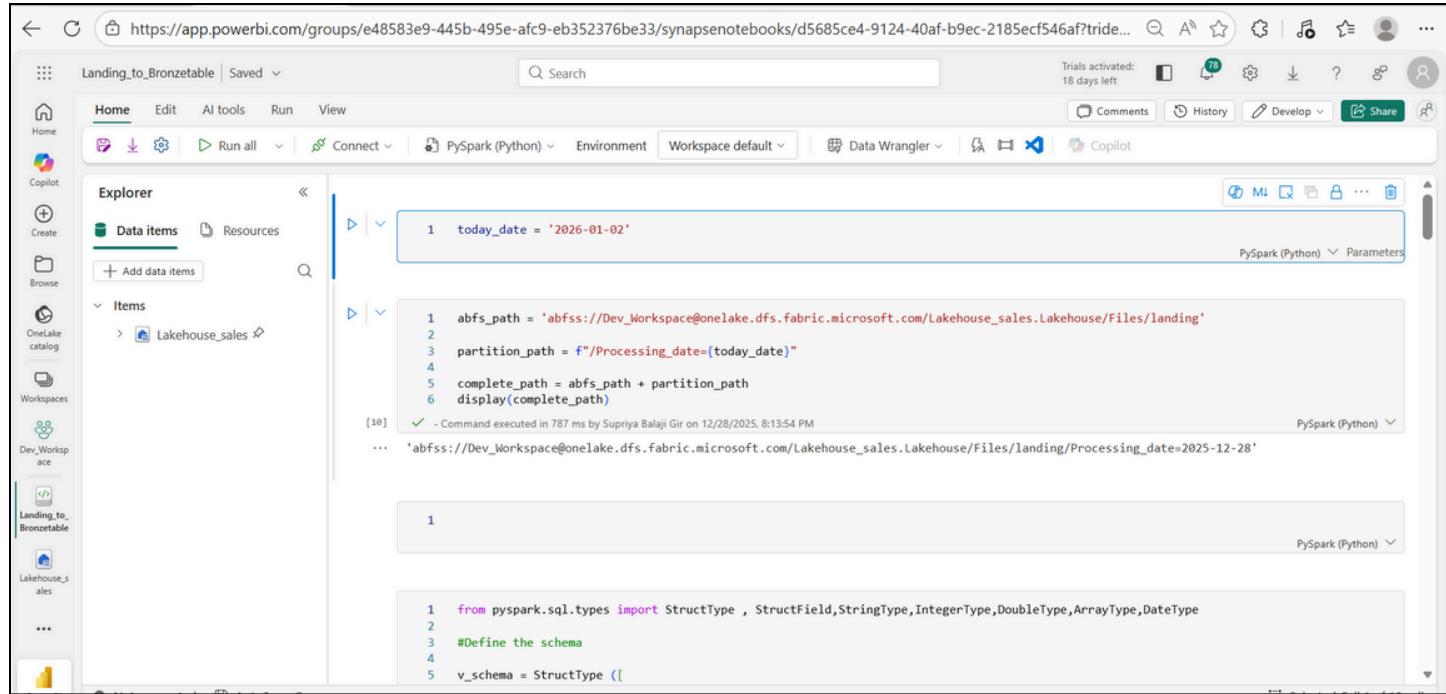
Data got loaded to landing folder:

The screenshot shows the Microsoft Power BI Lakehouse interface. The left sidebar displays the workspace structure: Dev_Workspace > Lakehouse_sales > Files > landing. The main content area shows a list of items in the landing folder:

Name	Date modified	Type	Size
Processing_date=2026-01-02	1/2/2026, 5:04:40 PM	Folder	-
_SUCCESS	1/2/2026, 5:04:41 PM	0 B	

Step 5: Transfer data from landing folder to Bronze layer(table)

Create a new Notebook: Landing to Bronze table notebook:



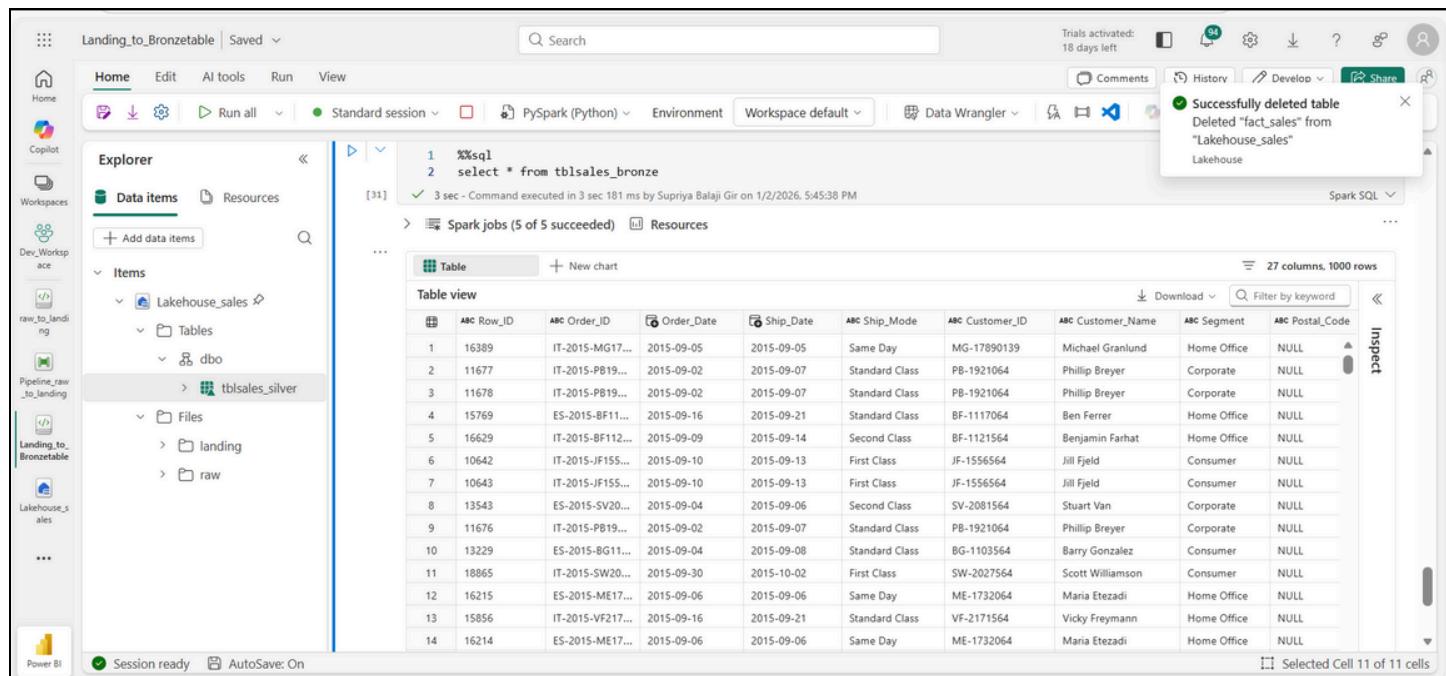
```
1 today_date = '2026-01-02'

1 abfs_path = 'abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/landing'
2
3 partition_path = f"/Processing_date={today_date}"
4
5 complete_path = abfs_path + partition_path
6 display(complete_path)

[10] ✓ - Command executed in 787 ms by Supriya Balaji Gir on 12/28/2025, 8:13:54 PM
.... 'abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Files/landing/Processing_date=2025-12-28'

1

1 from pyspark.sql.types import StructType , StructField, StringType, IntegerType, DoubleType, ArrayType, DateType
2
3 #Define the schema
4
5 v_schema = StructType ([
```



```
1 %%sql
2 select * from tblsales_bronze

[31] ✓ - Command executed in 3 sec 181 ms by Supriya Balaji Gir on 1/2/2026, 5:45:38 PM

> Spark jobs (5 of 5 succeeded) Resources
```

Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Postal_Code
1	IT-2015-MG17...	2015-09-05	2015-09-05	Same Day	MG-17890139	Michael Granlund	Home Office	NULL
2	IT-2015-PB19...	2015-09-02	2015-09-07	Standard Class	PB-1921064	Philip Breyer	Corporate	NULL
3	IT-2015-PB19...	2015-09-02	2015-09-07	Standard Class	PB-1921064	Philip Breyer	Corporate	NULL
4	ES-2015-BF11...	2015-09-16	2015-09-21	Standard Class	BF-1117064	Ben Ferrer	Home Office	NULL
5	IT-2015-BF12...	2015-09-09	2015-09-14	Second Class	BF-1121564	Benjamin Farhat	Home Office	NULL
6	IT-2015-JF155...	2015-09-10	2015-09-13	First Class	JF-1556564	Jill Fjeld	Consumer	NULL
7	IT-2015-JF155...	2015-09-10	2015-09-13	First Class	JF-1556564	Jill Fjeld	Consumer	NULL
8	ES-2015-SV20...	2015-09-04	2015-09-06	Second Class	SV-2081564	Stuart Van	Corporate	NULL
9	IT-2015-PB19...	2015-09-02	2015-09-07	Standard Class	PB-1921064	Philip Breyer	Corporate	NULL
10	ES-2015-BG11...	2015-09-04	2015-09-08	Standard Class	BG-1103564	Barry Gonzalez	Consumer	NULL
11	IT-2015-SW20...	2015-09-30	2015-10-02	First Class	SW-2027564	Scott Williamson	Consumer	NULL
12	ES-2015-ME17...	2015-09-06	2015-09-06	Same Day	ME-1732064	Maria Etezadi	Home Office	NULL
13	IT-2015-VF217...	2015-09-16	2015-09-21	Standard Class	VF-2171564	Vicky Freymann	Home Office	NULL
14	ES-2015-ME17...	2015-09-06	2015-09-06	Same Day	ME-1732064	Maria Etezadi	Home Office	NULL

New table got created in bronze layer with additional processing_date column:

Landing_to_Bronzetable | Saved

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

27 columns, 1000 rows

Table view

duct_Name	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Priority	Month	Year	processing_date
Smart Phone...	327.12	1	0.5	-39.27	88.45	High	September	2015	2026-01-02
Folders, Blue	20.376	2	0.4	-7.824	1.44	Medium	September	2015	2026-01-02
I Folders, Sin...	42.048	4	0.4	-6.312	2	Medium	September	2015	2026-01-02
Trays, Wire F...	169.344	6	0.4	-2.9159999...	7.3	Medium	September	2015	2026-01-02
Shelving, Bl...	149.112	4	0.4	-87.048	12.29	Medium	September	2015	2026-01-02
Trays, Blue	130.104	4	0.4	-41.256	16.88	High	September	2015	2026-01-02
Box, Wire F...	55.296	4	0.4	-25.824	18.32	High	September	2015	2026-01-02
Lockers, Blue	237.492	2	0.4	-118.788	24.21	High	September	2015	2026-01-02
ng Audio Do...	100.152	1	0.4	-23.388	6.06	Medium	September	2015	2026-01-02
Office Telep...	199.71	5	0.4	13.26	13.87	Medium	September	2015	2026-01-02
ng Smart Ph...	1145.232	3	0.4	-706.248	236.69	Critical	September	2015	2026-01-02
Receipt Print...	351.09	5	0.4	-93.660000...	59.35	High	September	2015	2026-01-02
ur Creations ...	288.192	4	0.6	-180.168	18.21	Medium	September	2015	2026-01-02
ocking Chair...	153.828	3	0.6	-84.672	29.76	High	September	2015	2026-01-02
ur Creations ...	216.144	3	0.6	-135.126	30.58	High	September	2015	2026-01-02
es Trays, Blue	103.302	2	0.1	-3.498	7.18	Medium	September	2015	2026-01-02

Download Filter by keyword

Items

- Lakehouse_sales
 - Tables
 - dbo
 - tblsales_silver
 - Files
 - landing
 - raw

raw_to_landing

Pipeline_raw_to_landing

Landing_to_Bronzetable

Lakehouse_sales

Step 6: Transfer data from Bronze layer to Silver layer(table)

In this transformation the cleaning part is done, some business-related transformations are done:

The screenshot shows the Power BI Data Transformation workspace. The left sidebar lists workspaces: Dev_Workspace, Silver_Transformation, Lakehouse_sales, Pipeline_raw_to_landing, and Power BI. The main area is titled "Silver Transformation". It contains two sections: "Data Cleaning" and "Data Transformation". In the "Data Cleaning" section, there is a code editor with the following Python script:

```
1 today_date = '2025-12-28'
[2] ✓ - Command executed in 273 ms by Supriya Balaji Gir on 12/29/2025, 9:05:32 AM
PySpark (Python) Parameters
```

```
1 from pyspark.sql.functions import col
2
3 Fabric_bronze_path = 'abfss://Dev_Worksace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Tables/dbo/tblsales_bronze'
4
5 df = spark.read.format('delta').load(Fabric_bronze_path).filter(col('processing_date') == today_date)
6
7 display(df)
[3] ✓ - Command executed in 7 sec 600 ms by Supriya Balaji Gir on 12/29/2025, 9:05:42 AM
Output is hidden
```

Selected Cell 1 of 16 cells

The screenshot shows the Power BI Data Transformation workspace. The left sidebar lists workspaces: Dev_Workspace, Silver_Transformation, Lakehouse_sales, Pipeline_raw_to_landing, and Power BI. The main area is titled "Data Cleaning". It contains two sections: "handling Duplicates" and "Handle Missing Values". In the "handling Duplicates" section, there is a code editor with the following Python script:

```
1
2 print('Before removing Duplicate',df.count())
3 df_remove_duplicate = df.drop_duplicates()
4 print('after removing Duplicate',df.count())
[4] ✓ - Command executed in 1 sec 520 ms by Supriya Balaji Gir on 12/29/2025, 9:05:48 AM
PySpark (Python) Parameters
```

Before removing Duplicate 1675
after removing Duplicate 1675

In the "Handle Missing Values" section, there is a code editor with the following Python script:

```
1 df_dropped = df_remove_duplicate.dropna(subset=['Order_ID','Customer_ID'])
2
```

Selected Cell 1 of 16 cells

added Delivery Date column with the help of ship_date and order_Date:

The screenshot shows the Power BI Data Transformation workspace. The left sidebar lists workspaces: OneLake_catalog, Apps, Workspaces, Dev_Workspace, Silver_Transformation, and Power BI. The main area is titled "Business Transformations". It contains a "Table view" section showing a table with 28 columns and 1000 rows. The table includes columns such as Order_Priority, Month, Year, processing_date, and Delivery_days. The table view has a "Download" button and a "Filter by keyword" search bar. In the background, there is a code editor with the following Python script:

```
1 df_days = df_dropped.withColumn('Delivery_days',(col('Ship_Date')-col('Order_Date')).cast('int'))
2 display(df_days)
[6] ✓ - Command executed in 3 sec 336 ms by Supriya Balaji Gir on 12/29/2025, 9:06:00 AM
PySpark (Python) Parameters
```

Selected Cell 1 of 16 cells

add Profit margin column:

The screenshot shows the Copilot interface with the following details:

- Left Sidebar:** Includes icons for Copilot, Create, Browse, OneLake catalog, Apps, Workspaces, Dev_Workspace, and Silver_Transformation.
- Top Bar:** Shows various icons and tabs like Run all, Connect, PySpark (Python), Environment, Workspace default, Data Wrangler, and Copilot.
- Explorer:** Shows a tree structure under "Items" with "Lakehouse_sales" selected.
- Code Editor:** Displays PySpark code:


```
1 df_profit_margin = df_days.withColumn('Profit_Margin', col('Profit')/col('Sales'))
2 display(df_profit_margin)
```

[7] ✓ - Command executed in 2 sec 314 ms by Supriya Balaji Gir on 12/29/2025, 9:06:08 AM
- Table View:** Shows a table with 29 columns and 1000 rows. The columns include: Discount, Profit, Shipping_Cost, Order_Priority, Month, Year, processing_date, Delivery_days, Profit_Margin. The Profit_Margin column contains values like 0.3, 0.2999999999999999, and 0.2900000000000000.

The screenshot shows the Copilot interface with the following details:

- Left Sidebar:** Same as the first screenshot.
- Top Bar:** Shows various icons and tabs like PySpark (Python).
- Explorer:** Shows a tree structure under "Items" with "Lakehouse_sales" selected.
- Code Editor:** Displays Spark SQL code:


```
1 %%sql
2 select * from t_silver_new_data
```

[8] ✓ - Command executed in 265 ms by Supriya Balaji Gir on 12/29/2025, 9:06:13 AM
- Table View:** Shows a table with 29 columns and 1000 rows. The columns include: Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Postal_Code. The Customer_Name column includes entries like Jeremy Lonsdale, Brenda Bowman, Greg Matthias, and Lisa Ryan.

Create table in Silver layer:

The screenshot shows the Copilot interface with the following details:

- Left Sidebar:** Same as the previous screenshots.
- Top Bar:** Shows various icons and tabs.
- Explorer:** Shows a tree structure under "Items" with "Lakehouse_sales" selected.
- Code Editor:** Displays PySpark code for creating a table in the Silver layer:


```
1 Fabric_tblsales_silver = 'abfss://Dev_Workspace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Tables/dbt/tblsales_silver'
2 try:
3     spark.read.format('delta').load(Fabric_tblsales_silver).createOrReplaceTempView('t_tblsales_silver')
4 except:
5     v_create_table = f'''CREATE TABLE IF NOT EXISTS tbsales_silver (
6     Row_ID string,
7     Order_ID string,
8     Order_Date date,
9     Ship_Date date,
10    Ship_Mode string,
11    Customer_ID string,
12    Customer_Name string,
13    Segment string,
14    Postal_Code string,
15    City string,
16    State string,
17    Country string,
18    Region string,
19    Market string,
20    Product_ID string,
21    Category string,
22    Sub_Category string,
23    Product_Name string,
24    Sales DOUBLE,
25    Quantity int,
26    Discount DOUBLE,
27    Profit DOUBLE,
```

Load data:

```
1 Fabric_tblsales_silver = 'abfss://Dev_Worksace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Tables/dbo/tb sales_silver'
2 try:
3     spark.read.format('delta').load(Fabric_tblsales_silver).createOrReplaceTempView('t_tb sales_silver')
4 except:
5     v_create_table = f"""CREATE TABLE IF NOT EXISTS tb sales_silver (
6         Row_ID string ,
7         Order_ID string ,
8         Order_Date date ,
9         Ship Date date ,
10        Ship_Mode string ,
11        Customer_ID string ,
12        Customer_Name string ,
13        Segment string ,
14        Postal_Code string ,
15        City string ,
16        State string ,
17        Country string ,
18        Region string ,
19        Market string ,
20        Product_ID string ,
21        Category string ,
22        Sub_Category string ,
23        Product_Name string ,
24        Sales DOUBLE ,
25        Quantity int ,
26        Discount DOUBLE ,
27        Profit DOUBLE .
```

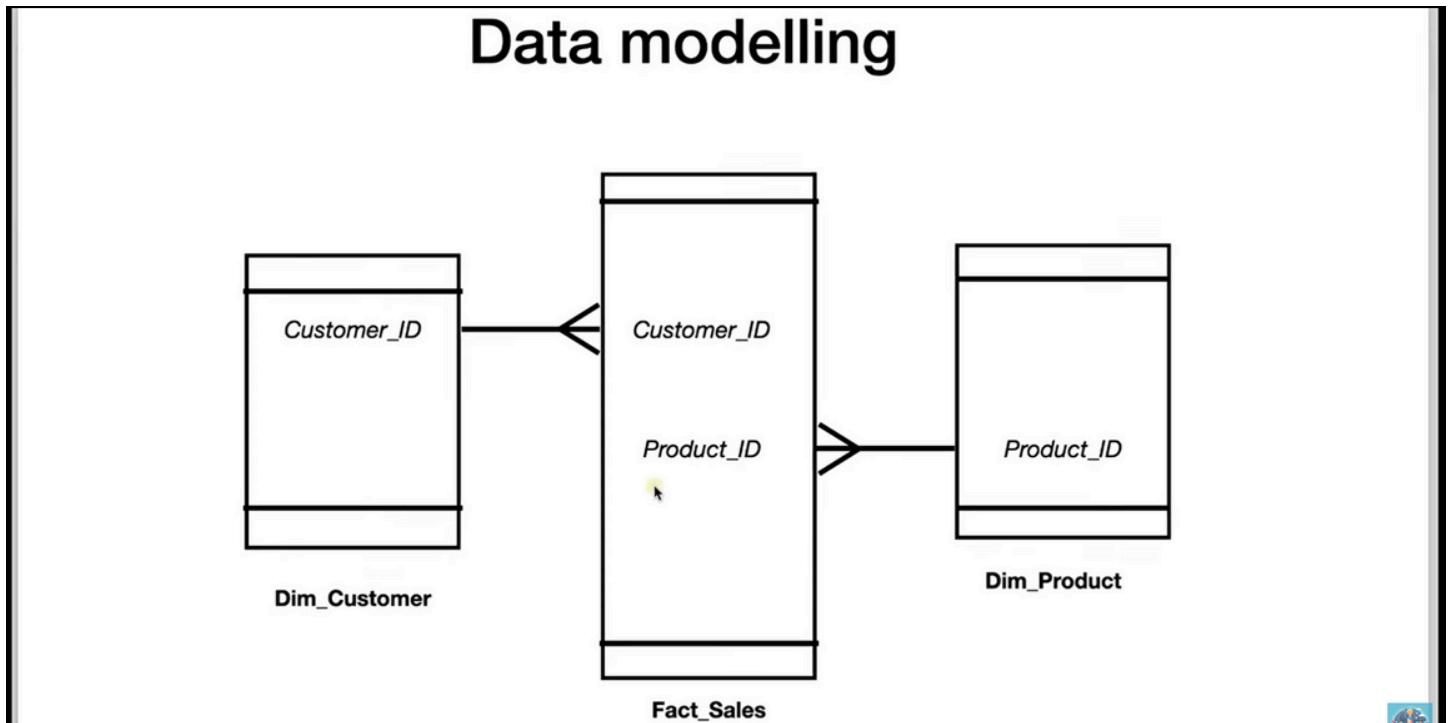
Silver table is seen now in lakehouse:

```
1 Fabric_tblsales_silver = 'abfss://Dev_Worksace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Tables/dbo/tb sales_silver'
2 try:
3     spark.read.format('delta').load(Fabric_tblsales_silver).createOrReplaceTempView('t_tb sales_silver')
4 except:
5     v_create_table = f"""CREATE TABLE IF NOT EXISTS tb sales_silver (
6         Row_ID string ,
7         Order_ID string ,
8         Order_Date date ,
9         Ship Date date ,
10        Ship_Mode string ,
11        Customer_ID string ,
12        Customer_Name string ,
13        Segment string ,
14        Postal_Code string ,
15        City string ,
16        State string ,
17        Country string ,
18        Region string ,
19        Market string ,
20        Product_ID string ,
21        Category string ,
22        Sub_Category string ,
23        Product_Name string ,
24        Sales DOUBLE ,
25        Quantity int ,
26        Discount DOUBLE ,
27        Profit DOUBLE .
```

At the end of Silver transformation data is cleaned and transformed. But the data is yet not ready for business consumption.

Step 7: Transfer data from Silver layer to Gold layer(table)

Make data usable for Business by doing data modeling:



Creating new Notebook for Silver to gold transformation and attach lakehouse to it:

The screenshot shows a Databricks notebook interface. On the left, the **Explorer** sidebar lists several workspaces and a notebook named **Silver to Gold trans**. The main area displays a PySpark (Python) notebook with the following code and output:

```
1 today_date = '2026-01-05'  
2  
[4] ✓ <1 sec - Command executed in 353 ms by Supriya Balaji Gir on 1/5/2026, 2:54:35 PM  
PySpark (Python) Parameters  
  
[5] 1 from pyspark.sql.functions import col  
2  
3 df = spark.read.format('delta').load('abfss://Dev_Worksplace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Tables dbo/ti  
[5] ✓ 7 sec - Command executed in 7 sec 451 ms by Supriya Balaji Gir on 1/5/2026, 2:54:46 PM  
PySpark (Python)  
...  
  
[6] 1 display(df)  
[6] ✓ 11 sec - Command executed in 11 sec 427 ms by Supriya Balaji Gir on 1/5/2026, 2:55:04 PM  
PySpark (Python)  
...  
  
Table view  
ABC Row_ID ABC Order_ID Order_Date Ship_Date ABC Ship_Mode ABC Customer_ID ABC Customer_Name ABC Segment ABC Postal  
...  
Data Wrangler  
Download  
Filter by keyword
```

Creating dimension Tables:

The screenshot shows the Databricks workspace interface. On the left, the 'Items' sidebar is expanded, showing a folder 'Lakehouse_sales' containing 'Tables' and 'Files'. The 'Files' folder is further expanded to show 'landing' and 'raw' subfolders. In the main editor area, a Python code cell titled 'Creating Dimension Tables' is running. The code defines a schema for the 'Dim_Customers' table and creates it if it doesn't exist. A success message indicates the command was executed in 8 seconds.

```
1 #Dim_Customers Table
2 from pyspark.sql.types import StructType , StructField , StringType, IntegerType, DoubleType, DateType
3
4 from delta.tables import DeltaTable
5
6
7 #Define schemas from dimension tables
8
9 dim_customer_schema = StructType([
10     StructField('Customer_ID', StringType(), True),
11     StructField('Customer_Name', StringType(), True),
12     StructField('Segment', StringType(), True),
13     StructField('Postal_Code', StringType(), True),
14     StructField('City', StringType(), True),
15     StructField('State', StringType(), True),
16     StructField('Country', StringType(), True),
17     StructField('Region', StringType(), True),
18     StructField('Market', StringType(), True)
19 ])
20
21
22 DeltaTable.createIfNotExists(spark).tableName('Dim_Customer').addColumn(dim_customer_schema).execute()
[?]
✓ 8 sec - Command executed in 9 sec 269 ms by Supriya Balaji Gir on 1/5/2026, 2:55:41 PM
```

The screenshot shows the Databricks workspace interface. The 'Items' sidebar is expanded, showing the same 'Lakehouse_sales' structure as the previous screenshot. The main editor area contains a Python code cell titled 'Items' (number 8) defining a schema for the 'Dim_Product' table and creating it if it doesn't exist. A success message indicates the command was executed in 3 seconds.

```
1 # Dim_Product Table
2 dim_product_schema = StructType([
3     StructField('Product_ID', StringType(), True),
4     StructField('Category', StringType(), True),
5     StructField('Sub_Category', StringType(), True),
6     StructField('Product_Name', StringType(), True)
7 ])
8
9 DeltaTable.createIfNotExists(spark).tableName('Dim_Product').addColumn(dim_product_schema).execute()
[8]
✓ 3 sec - Command executed in 3 sec 236 ms by Supriya Balaji Gir on 1/5/2026, 2:56:26 PM
```

Creation of fact Table:

The screenshot shows the Databricks workspace interface. The 'Items' sidebar is expanded, showing the same 'Lakehouse_sales' structure. The main editor area contains a Python code cell titled 'Fact Table' (number 9) defining a schema for the 'Fact_Sales' table and creating it if it doesn't exist. A success message indicates the command was executed in 3 seconds.

```
1 #Sales Table
2 Fact_Sales_schema = StructType([
3     StructField('Row_ID', StringType(), True),
4     StructField('Order_ID', StringType(), True),
5     StructField('Customer_ID', StringType(), True),
6     StructField('Product_ID', StringType(), True),
7     StructField('Order_Date', DateType(), True),
8     StructField('Ship_Date', DateType(), True),
9     StructField('Ship_Mode', StringType(), True),
10    StructField('Sales', DoubleType(), True),
11    StructField('Quantity', IntegerType(), True),
12    StructField('Discount', DoubleType(), True),
13    StructField('Profit', DoubleType(), True),
14    StructField('Shipping_Cost', DoubleType(), True),
15    StructField('Order_Priority', StringType(), True),
16    StructField('Month', StringType(), True),
17    StructField('Year', StringType(), True),
18    StructField('processing_date', StringType(), True),
19    StructField('Delivery_Days', StringType(), True),
20    StructField('Profit_Margin', StringType(), True)
21 ])
22
23
24 DeltaTable.createIfNotExists(spark).tableName('Fact_Sales').addColumn(Fact_Sales_schema).execute()
[9]
✓ - Command executed in 3 sec 236 ms by Supriya Balaji Gir on 12/29/2025, 9:22:52 AM
```

insert data into newly created tables from table in silver transformation:

```

1  from delta.tables import *
2
3  dim_customer_table_path = 'abfss://Dev_Worksace@onelake.dfs.fabric.microsoft.com/Lakehouse_sales.Lakehouse/Tables/dbo/dim_customer'
4  dim_deltacustomer = DeltaTable.forPath(spark,dim_customer_table_path)
5
6  ## Perform the MERGE (UPINSERT)
7
8  dim_deltacustomer.alias('target').merge(
9      df_selected_dim_customer.alias('source'), 'target.Customer_ID=source.Customer_ID'
10 ).whenMatchedUpdate(set = {
11     'Customer_Name': 'source.Customer_Name',
12     'Segment': 'source.Segment',
13     'Postal_Code': 'source.Postal_Code',
14     'City': 'source.City',
15     'State': 'source.State',
16     'Country': 'source.Country',
17     'Region': 'source.Region',
18     'Market': 'source.Market'
19 }).whenNotMatchedInsert(values=(
20     'Customer_ID': 'source.Customer_ID',
21     'Customer_Name': 'source.Customer_Name',
22     'Segment': 'source.Segment',
23     'Postal_Code': 'source.Postal_Code',
24     'City': 'source.City',
25     'State': 'source.State',
26     'Country': 'source.Country',
27     'Region': 'source.Region',
28     'Market': 'source.Market'
29 )

```

Customer Table:

File 2016Mar.csv

```

1 %%sql
2 SELECT * FROM dim_customer

```

4 sec - Command executed in 4 sec 385 ms by Supriya Balaji Gir on 1/5/2026, 3:00:19 PM

Spark SQL

	ABC Customer_ID	ABC Customer_Name	ABC Segment	ABC Postal_Code	ABC City	ABC State	ABC Country
1	PP-189551408	Paul Prost	Home Office	33021	Hollywood	Florida	United S
2	PS-187601406	Pamela Stobb	Consumer	19140	Philadelphia	Pennsylvania	United S
3	VM-218351406	Vivian Mathis	Consumer	3060	Nashua	New Hamp...	United S
4	AB-100601406	Adam Bellavance	Home Office	19143	Philadelphia	Pennsylvania	United S
5	AP-109151408	Arthur Prichep	Consumer	32137	Palm Coast	Florida	United S
6	SS-204101408	Shahid Sharari	Consumer	30318	Atlanta	Georgia	United S
7	KC-165401406	Kelly Collister	Consumer	43055	Newark	Ohio	United S
8	AG-106751402	Anna Gayman	Consumer	53209	Milwaukee	Wisconsin	United S
9	CA-127751402	Cynthia Arntzen	Consumer	48205	Detroit	Michigan	United S
10	CS-124001404	Christopher Schild	Home Office	98115	Seattle	Washington	United S
11	RW-195401404	Rick Wilson	Corporate	98103	Seattle	Washington	United S
12	MH-181151406	Mick Hernandez	Home Office	43130	Lancaster	Ohio	United S
13	JF-154151404	Jennifer Ferguson	Consumer	98105	Seattle	Washington	United S

Product table:

File raw

```

1 %%sql
2 select * from dim_product

```

3 sec - Command executed in 4 sec 348 ms by Supriya Balaji Gir on 1/5/2026, 3:00:51 PM

Spark SQL

	ABC Product_ID	ABC Category	ABC Sub_Category	ABC Product_Name
1	OFF-PA-6538	Office Supplies	Paper	Xerox 1980
2	FUR-BO-3895	Furniture	Bookcases	Dania Corner Shelf...
3	OFF-PA-5847	Office Supplies	Paper	SandDisk Cards & E...
4	OFF-AP-4960	Office Supplies	Appliances	KitchenAid Refriger...
5	OFF-ST-4266	Office Supplies	Storage	Fellowes Lockers. B...
6	OFF-SU-4974	Office Supplies	Supplies	Kleencut Box Cutte...
7	TEC-CO-3606	Technology	Copiers	Brother Personal C...
8	FUR-TA-4944	Furniture	Tables	KI Conference Tables
9	OFF-PA-6591	Office Supplies	Paper	Xerox 227
10	OFF-AR-5458	Office Supplies	Art	OIC #2 Pencils. Me...
11	TEC-AC-3373	Technology	Accessories	Belkin F8E887 USB ...
12	OFF-SU-4326	Office Supplies	Supplies	Fiskars Trimmer, Hi...
13	OFF-ST-6247	Office Supplies	Storage	Tenex File Cart, Ind...

Fact_Sales table:

The screenshot shows a Databricks workspace interface. On the left, the sidebar displays various workspaces and a list of items under 'Lakehouse_sales'. In the center, a code editor window shows a Spark SQL command:

```
1 %%sql
2 select * from fact_sales
```

The command was executed in 4 seconds by Supriya Balaji Gir on 1/5/2026, 3:04:05 PM. To the right, a table view displays the results of the query:

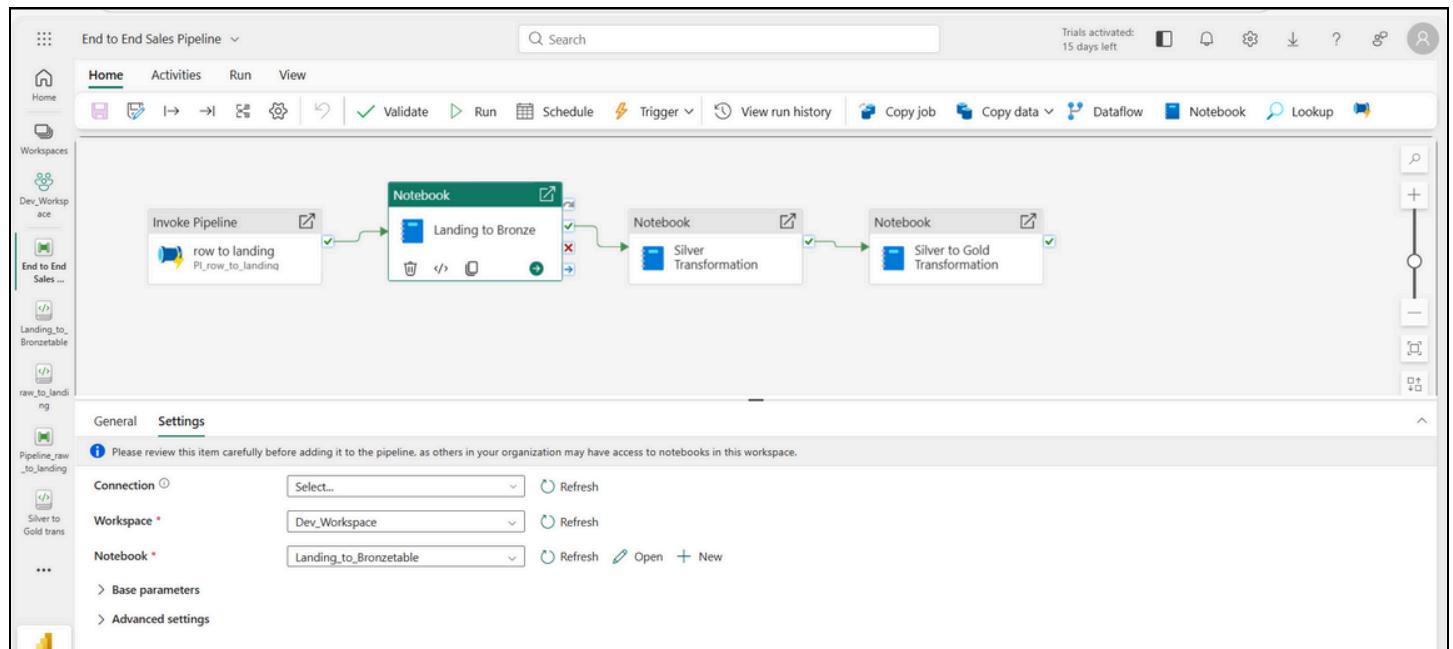
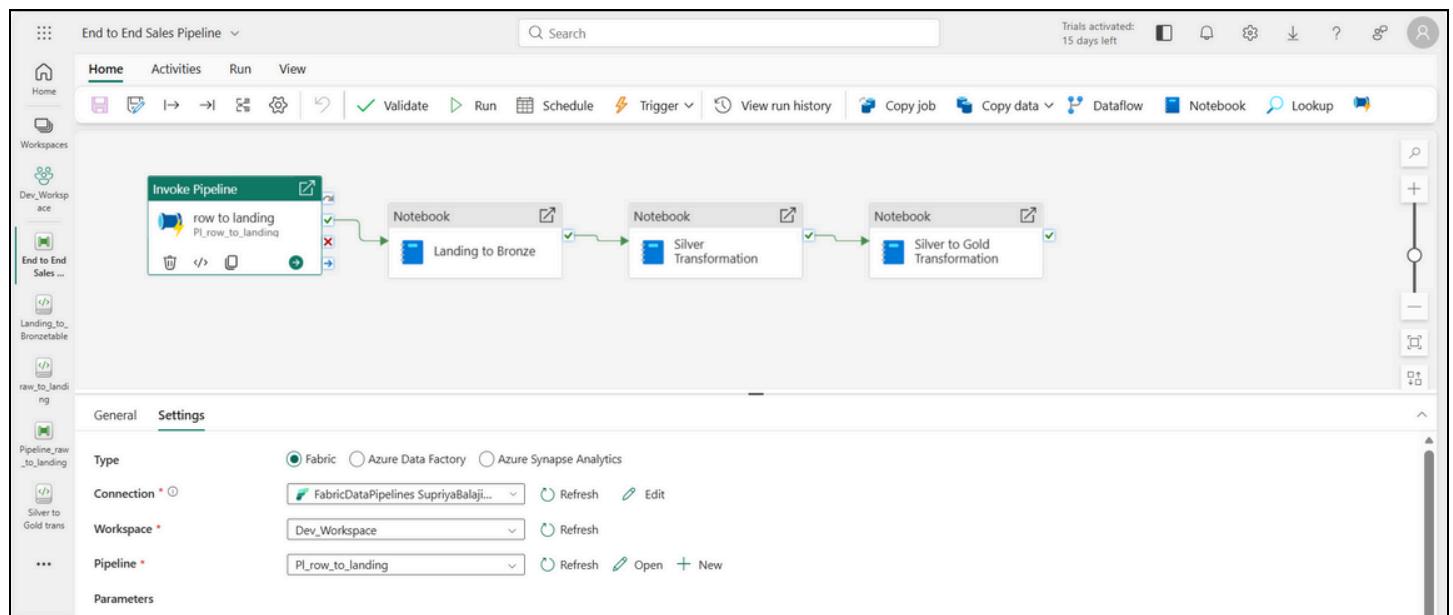
ABC Row_ID	ABC Order_ID	ABC Customer_ID	ABC Product_ID	Order_Date	Ship_Date	ABC Ship_Mode	12 Sales	123 Quantity	
1	40134	US-2015-KF16...	KF-162851408	OFF-LA-5978	2015-09-25	2015-09-29	Standard Class	15.75	5
2	35664	CA-2015-HA1...	HA-149201402	OFF-FA-2916	2015-09-12	2015-09-14	Second Class	10.528	4
3	26723	IN-2015-HG14...	HG-1484558	FUR-CH-5367	2015-09-17	2015-09-19	Second Class	87.06	2
4	18570	ES-2015-JH15...	JH-1582048	FUR-BO-5951	2015-09-12	2015-09-17	Second Class	1177.173	3
5	25172	IN-2015-LS17...	LS-1723078	OFF-BI-3734	2015-09-04	2015-09-08	Standard Class	20.52	3
6	8802	MX-2015-RM1...	RM-1967539	OFF-FA-3020	2015-09-18	2015-09-23	Standard Class	33.48	3
7	29591	IN-2015-BF11...	BF-1102059	TEC-CO-3703	2015-09-10	2015-09-17	Standard Class	268.1748	2
8	6514	MX-2015-CS1...	CS-1225093	FUR-CH-5408	2015-09-22	2015-09-29	Standard Class	109.64	2
9	3621	US-2015-KE16...	KE-1642082	FUR-FU-5729	2015-09-06	2015-09-08	First Class	42.768	1
10	41368	IR-2015-CA22...	CA-226560	OFF-EN-4442	2015-09-17	2015-09-20	First Class	51.36	1
11	21997	IN-2015-RS19...	RS-1942027	OFF-AR-3545	2015-09-23	2015-09-28	Standard Class	257.04	8
12	44882	MO-2015-MS...	MS-777086	TEC-MA-5568	2015-09-24	2015-09-28	Standard Class	81.99	1

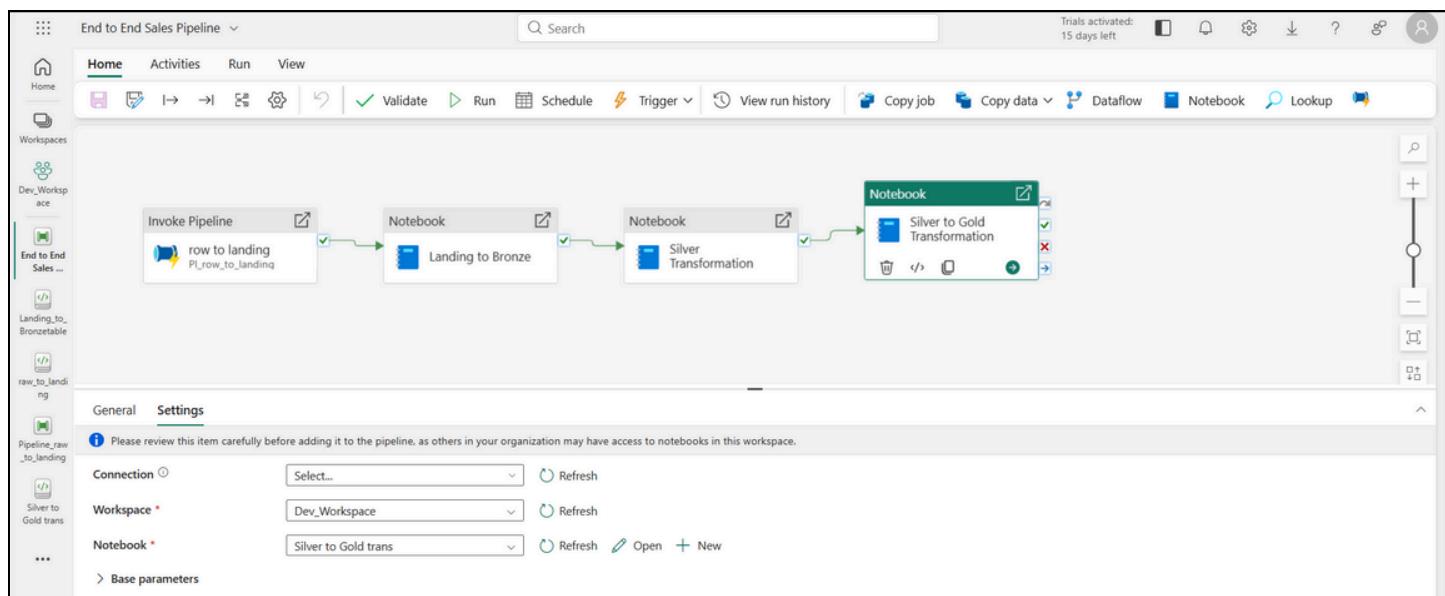
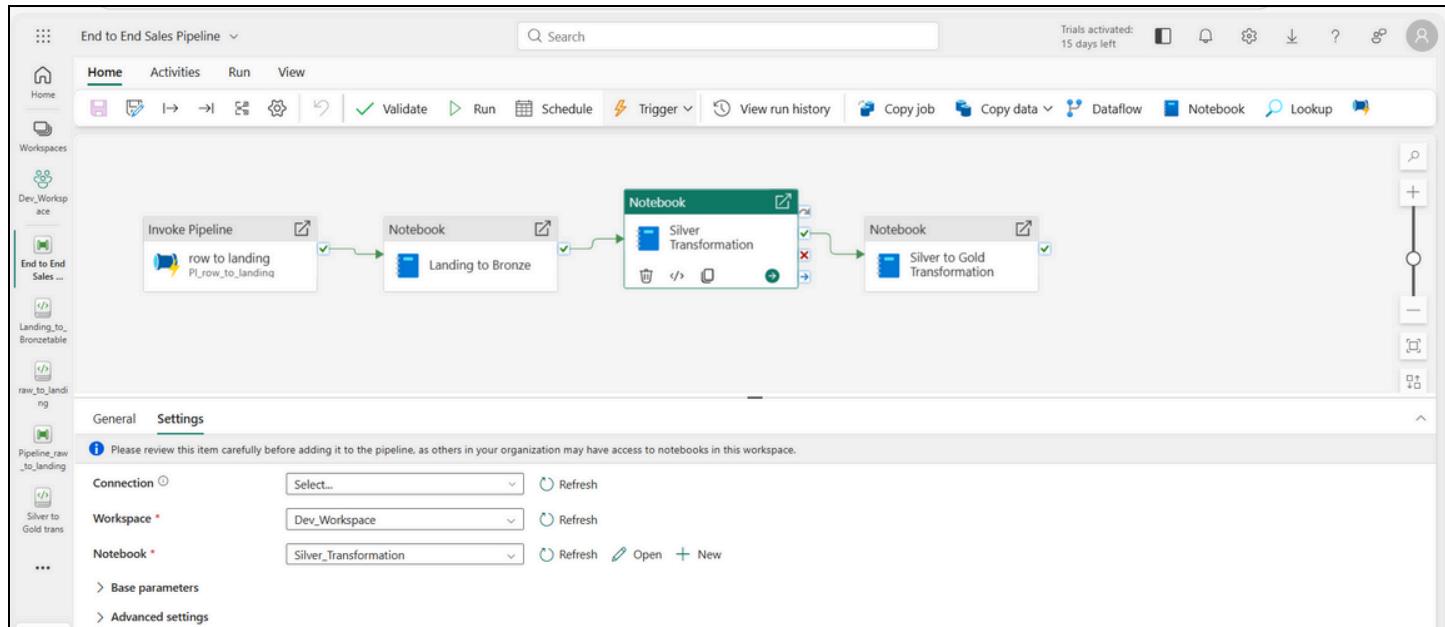
Step 8: Creating End to End Pipeline to transfer data from raw folder to Golden layer

Now creating again a full end-to-end pipeline for data movement from raw folder till Gold level:

Already the raw to landing pipeline is created so first in new pipeline we will invoke that pipeline as first activity of end-to-end pipeline:

And then attach 3 further notebooks as next activities:





Testing Final Pipeline:

In lakehouse file 2016Mar.csv is added. All other folders and tables are not created.

Landing folder is empty.

After running end-to-end pipeline, data file should move from:

1. raw to landing folder
2. from landing folder to bronze table
3. from bronze table to silver table
4. at last from silver table to gold table with creation of fact and dimension tables

Lakehouse_sales

Dev_Workspace

Dev_End_Sales

Landing_to_Bronzetable

raw_to_landing

Pipeline_raw_to_landing

2016Mar.csv

Showing 1 items

End to End Sales Pipeline

Invoke Pipeline

Notebook

Notebook

Notebook

Parameters

Variables

Settings

Output

Library variables

Pipeline run ID: 6b05932b-7d9c-43af-a038-bb39cd8133f7

Pipeline status: Succeeded

Activity name	Activity status	Run start	Duration	Input	Output
Silver to Gold Transformation	Succeeded	1/5/2026, 9:45:07 PM	1m 36s		
Silver Transformation	Succeeded	1/5/2026, 9:44:00 PM	1m 6s		
Landing to Bronze	Succeeded	1/5/2026, 9:42:52 PM	1m 7s		
row to landing	Succeeded	1/5/2026, 9:41:24 PM	1m 28s		

result:

Lakehouse_sales

Dev_Workspace

Dev_End_Sales

Landing_to_Bronzetable

raw_to_landing

Pipeline_raw_to_landing

dim_customer

Table view

	Customer_ID	Customer_Name	Segment	Postal_Code	City	State	Country	Region	Market
1	PP-189551408	Paul Prost	Home Office	33021	Hollywood	Florida	United States	Southern US	USCA
2	PS-187601406	Pamela Stobb	Consumer	19140	Philadelphia	Pennsylvania	United States	Eastern US	USCA
3	VM-218351406	Vivian Mathis	Consumer	3060	Nashua	New Hampshire	United States	Eastern US	USCA
4	AB-100601406	Adam Bellavance	Home Office	19143	Philadelphia	Pennsylvania	United States	Eastern US	USCA
5	AP-109151408	Arthur Pritchep	Consumer	32137	Palm Coast	Florida	United States	Southern US	USCA
6	SS-204101408	Shahid Sharari	Consumer	30318	Atlanta	Georgia	United States	Southern US	USCA
7	KC-165401406	Kelly Collister	Consumer	43055	Newark	Ohio	United States	Eastern US	USCA
8	AG-106751402	Anna Gayman	Consumer	53209	Milwaukee	Wisconsin	United States	Central US	USCA
9	CA-127751402	Cynthia Arntzen	Consumer	48205	Detroit	Michigan	United States	Central US	USCA
10	CS-124001404	Christopher Schild	Home Office	98115	Seattle	Washington	United States	Western US	USCA
11	RW-195401404	Rick Wilson	Corporate	98103	Seattle	Washington	United States	Western US	USCA
12	MH-181151406	Mick Hernandez	Home Office	43130	Lancaster	Ohio	United States	Eastern US	USCA
13	JF-154151404	Jennifer Ferguson	Consumer	98105	Seattle	Washington	United States	Western US	USCA
14	SP-208601408	Sung Pak	Corporate	32216	Jacksonville	Florida	United States	Southern US	USCA
15	GK-146201402	Grace Kelly	Corporate	75023	Plano	Texas	United States	Central US	USCA
16	PG-188201404	Patrick Gardner	Consumer	93309	Bakersfield	California	United States	Western US	USCA
17	RO-197801402	Rose O'Brian	Consumer	60623	Chicago	Illinois	United States	Central US	USCA

sales table with all new columns:

Lakehouse_sales

Search Trials activated: 15 days left

Get data New semantic model Open notebook Add to data agent Manage materialized lake views (preview) Manage OneLake security (preview) Update all variables

Home

Explorer

Add lakehouses

Lakehouse_sales

Tables

- dbo
 - dim_customer
 - dim_product
 - fact_sales**
 - tblsales_bronze
 - tblsales_silver
- Files
 - landing
 - Processing_da...
 - raw

Showing 1000 rows

Table view

fact_sales

Quantity	Discount	Profit	Shipping_C...	Order_Priority	Month	Year	processing_...	Delivery_Days	Profit_Margin
5	0	7.56	2.44	High	September	2015	2026-01-05	4	0.48
4	0.2	3.29	1.77	High	September	2015	2026-01-05	2	0.3125
2	0	0.84	15.09	High	September	2015	2026-01-05	2	0.009648518263...
3	0.1	405.423	131.28	High	September	2015	2026-01-05	5	0.344403923637...
3	0	6.75	2.47	High	September	2015	2026-01-05	4	0.328947368421...
3	0	2.64	2.399	Medium	September	2015	2026-01-05	5	0.078853046594...
2	0.07	17.2548	20.16	Low	September	2015	2026-01-05	7	0.064341615990...
2	0	0	15.967	Low	September	2015	2026-01-05	7	0.0
1	0.4	-27.812	12.976	Critical	September	2015	2026-01-05	2	-0.65029928918...
1	0	14.37	4.64	Critical	September	2015	2026-01-05	3	0.279789719626...
8	0	95.04	15.5	Medium	September	2015	2026-01-05	5	0.36974789159...
1	0	27.03	3.55	Medium	September	2015	2026-01-05	4	0.329674350530...
1	0.7	-5.052	1.29	Critical	September	2015	2026-01-05	3	-1.46945898778...
3	0	8.107199999999...	10.65	High	September	2015	2026-01-05	4	0.079999999999...
4	0	3.7128	2.08	High	September	2015	2026-01-05	4	0.26
4	0	8.2992	1.42	Medium	September	2015	2026-01-05	4	0.42
4	0.2	3.4496	1.34	Medium	September	2015	2026-01-05	6	0.350000000000...

Succeeded (2 sec 328 ms)

Columns 18 Rows 1,000

Step 9: Creating relation between tables(Modeling)

- Go to SQL Analytics Endpoint option of lakehouse:

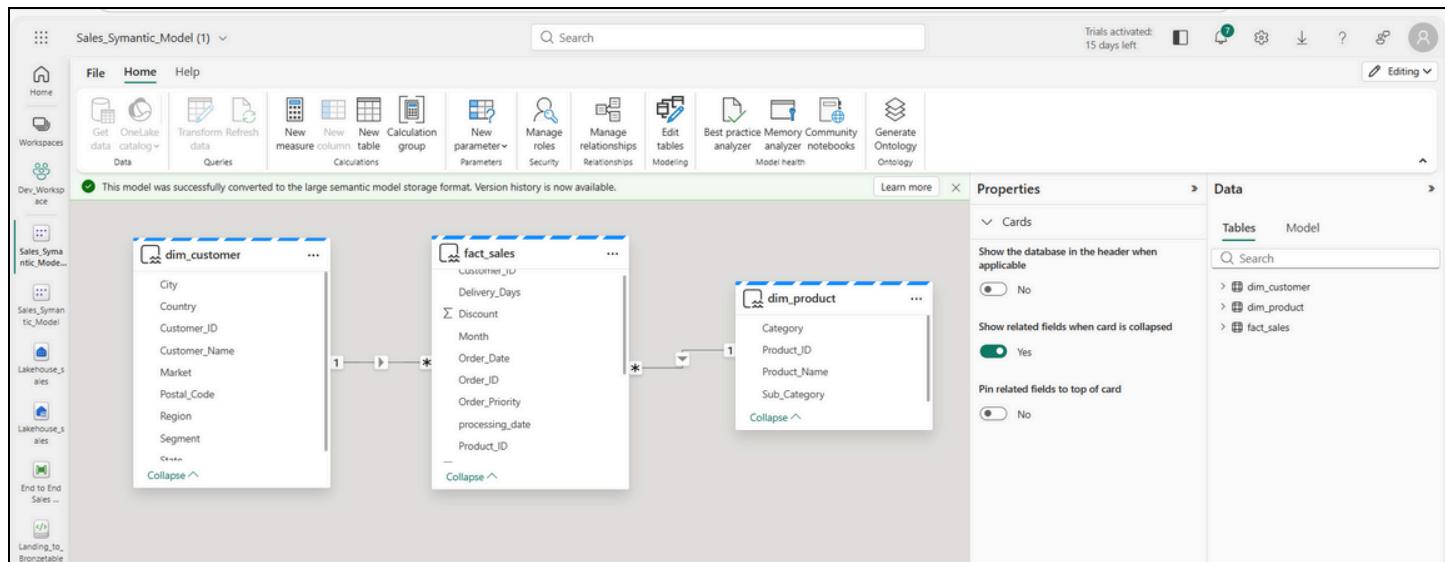
The screenshot shows the Azure Data Explorer interface. The left sidebar shows the workspace navigation with 'Lakehouse_sales' selected. The main area displays the 'fact_sales' table from the 'dbo' schema. The table has columns: Quantity, Discount, Profit, Shipping_Cost, Order_Priority, Month, Year, processing_date, Delivery_Days, and Profit_Margin. A preview of the first row is shown: 5, 0, 7.56, 2.44, High, September, 2015, 2026-01-05, 4, 0.48. On the right side, there is a 'SQL analytics endpoint' button with the sub-option 'Query data using SQL'.

- Go to Reporting tab and Select new Symantic Model and choose tables

The screenshot shows the 'New semantic model' dialog box. The 'Direct Lake semantic model name' field is set to 'Sales_Symantic_Model'. The 'Workspace' dropdown is set to 'Dev_Workspace'. Under 'Select or deselect tables for the semantic model', the 'dbo' schema is selected, and three tables are checked: 'dim_customer', 'dim_product', and 'fact_sales'. The 'Confirm' button is visible at the bottom right.

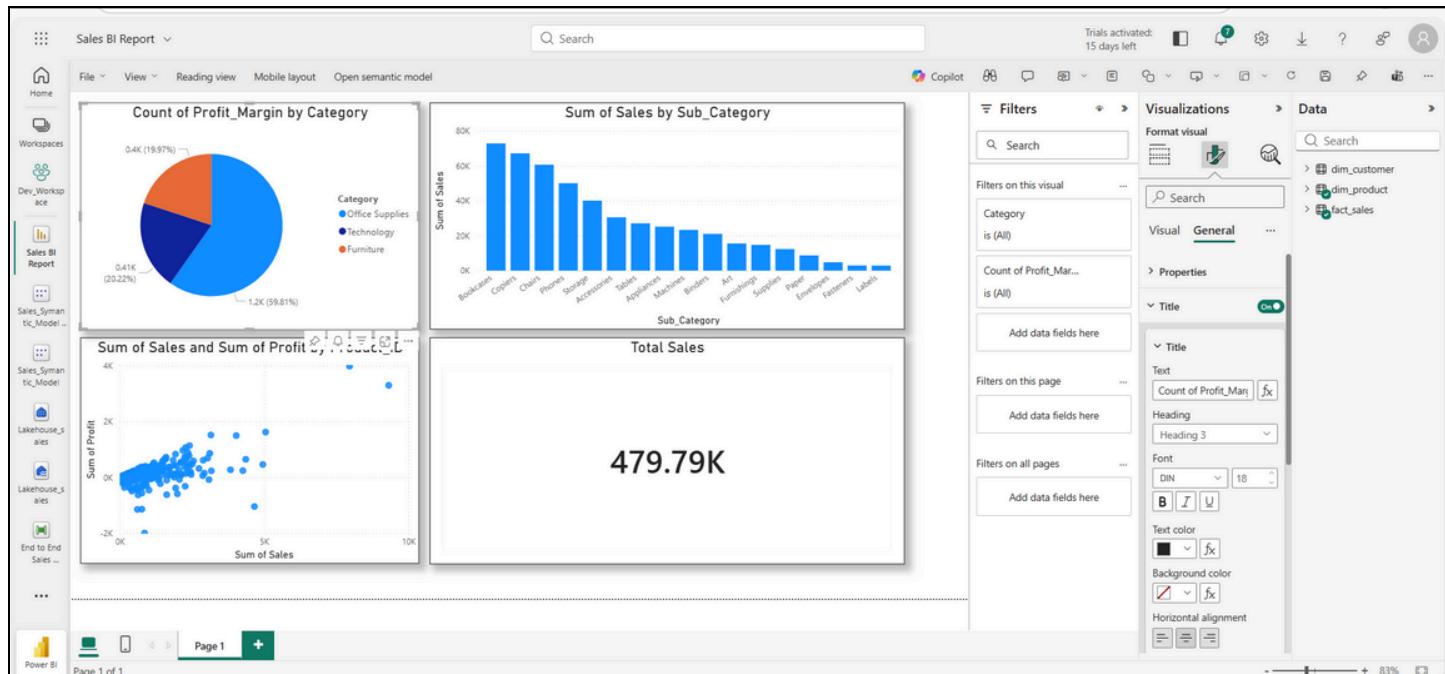
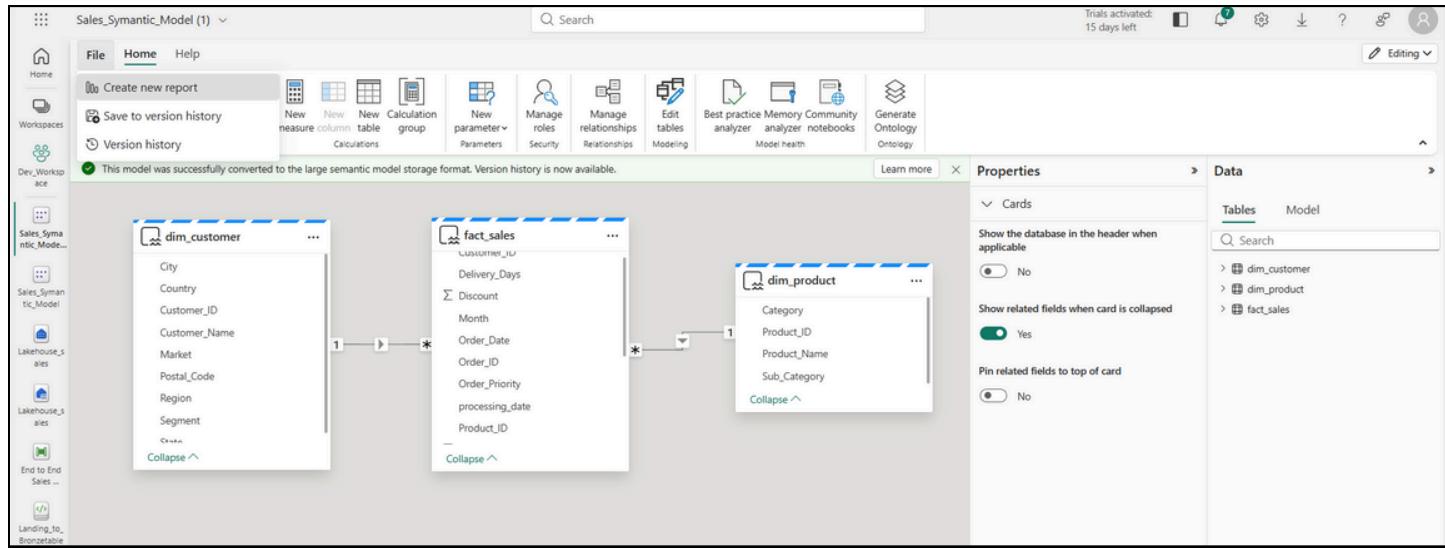
The screenshot shows the 'Sales_Symantic_Model (1)' workspace. The top ribbon has tabs for File, Home, Help, and various modeling tools like Get OneLake catalog, Transform data, and Modeling. The left sidebar shows workspace navigation. The main area displays three tables: 'dim_customer', 'dim_product', and 'fact_sales'. The 'Properties' pane on the right shows settings for cards, such as 'Show the database in the header when applicable' (set to No) and 'Show related fields when card is collapsed' (set to Yes). The 'Data' pane on the right lists the tables: 'dim_customer', 'dim_product', and 'fact_sales'. At the bottom, there are tabs for All tables, Model view, and DAX query view.

- join tables with primary and foreign keys



Step 10: Creation of Power BI Report

- now create Power BI report using this model:



Finally, the workspace contains following:

The screenshot shows the Power BI Dev_Workspace interface. The left sidebar lists various items under the 'Dev_Workspace' workspace, including 'Sales BI Report', 'Sales_Symantic_Model', 'Lakehouse_sales', 'Landing_to_Bronzetable', 'End to End Sales...', and 'Landing_to_Bronzetable'. The main area displays a table of items with the following columns: Name, Status, Type, Task, Owner, Refreshed, Next refresh, Endorsement, Sensitivity, and Included in app. The table contains the following data:

Name	Status	Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
End to End Sales Pipeline	Pipeline	—	Supriya Balaji ...	—	—	—	—	—	No
Lakehouse_sales	Lakehouse	—	Supriya Balaji ...	—	—	—	—	—	
Lakehouse_sales	SQL analytics ...	—	Supriya Balaji ...	—	—	—	—	—	
Landing_to_Bronzetable	Notebook	—	Supriya Balaji ...	—	—	—	—	—	
Pipeline_raw_to_landing	Pipeline	—	Supriya Balaji ...	—	—	—	—	—	
PL_row_to_landing	Pipeline	—	Supriya Balaji ...	—	—	—	—	—	
raw_to_landing	Notebook	—	Supriya Balaji ...	—	—	—	—	—	
Sales BI Report	Report	—	Dev_Workspace	1/5/2026, 10:41:0...	—	—	—	—	
Sales_Symantic_Model	Semantic model	—	Dev_Workspace	1/5/2026, 10:41:0...	N/A	—	—	—	
Silver to Gold trans	Notebook	—	Supriya Balaji ...	—	—	—	—	—	
Silver_Transformation	Notebook	—	Supriya Balaji ...	—	—	—	—	—	

Conclusion

This project demonstrates the successful design and implementation of a fully functional, end-to-end data engineering and analytics solution using Microsoft Fabric. By employing a medallion architecture (Bronze → Silver → Gold), the solution ensures that raw data is progressively refined, cleansed, and transformed into business-ready insights. The automated data pipelines, combined with PySpark-based transformation notebooks, enable scalable and reliable processing of structured and unstructured data.

The implementation provides the following benefits:

- **Improved Data Quality:** Data is validated, cleansed, and enriched at multiple stages.
- **Scalability:** Pipelines and PySpark transformations can handle growing volumes of data efficiently.
- **Business-Ready Insights:** Gold-layer tables support actionable reporting and analytics via Power BI.
- **Governance & Monitoring:** Data lineage, quality checks, and monitoring ensure reliability and compliance.