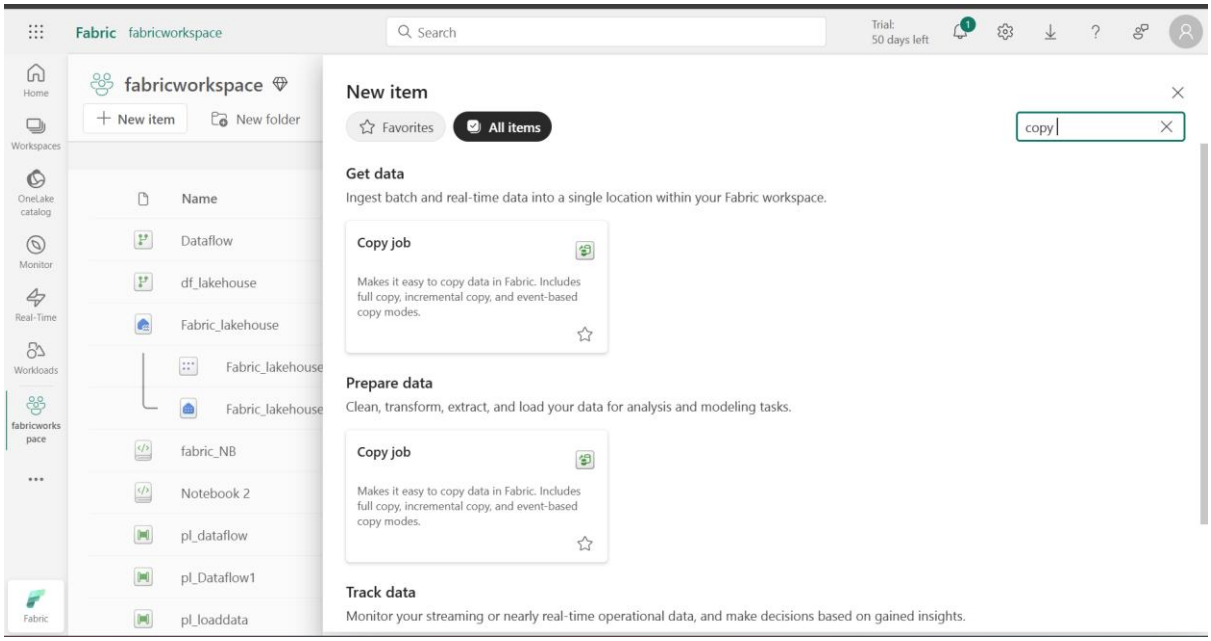
Copy Job

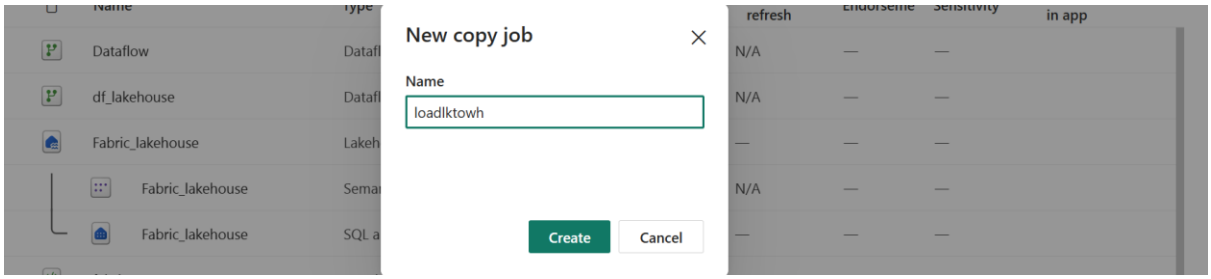
Use Copy job to load data using incremental copy from data warehouse to lake house tables

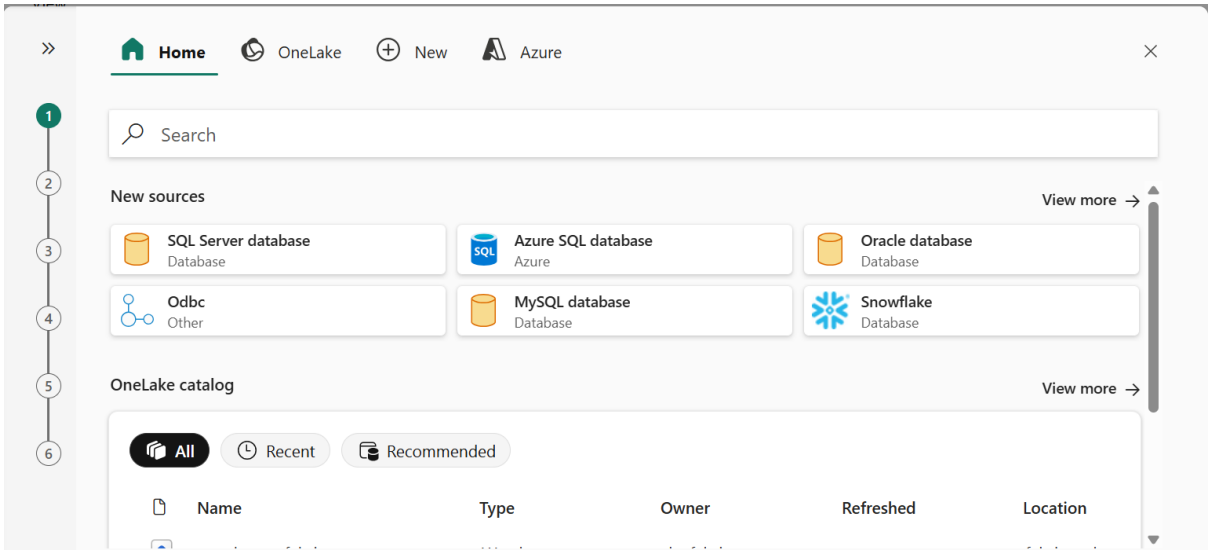
Copy Job:

A **Copy Job** in Microsoft Fabric refers to a data movement task that facilitates the transfer of data from one or more source systems to one or more destination systems. It is commonly used within **Data Pipelines** to support ETL (Extract, Transform, Load) or ELT (Extract, Load, Transform) scenarios.

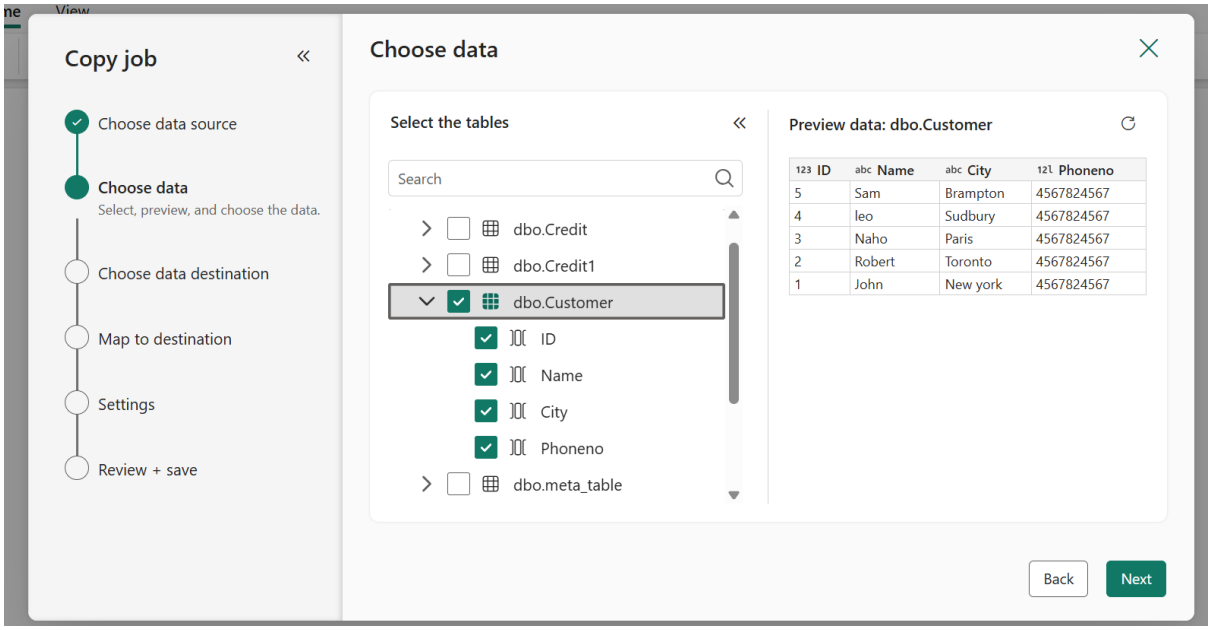
To create a copy job, go to workspace -> new item -> search for copy job



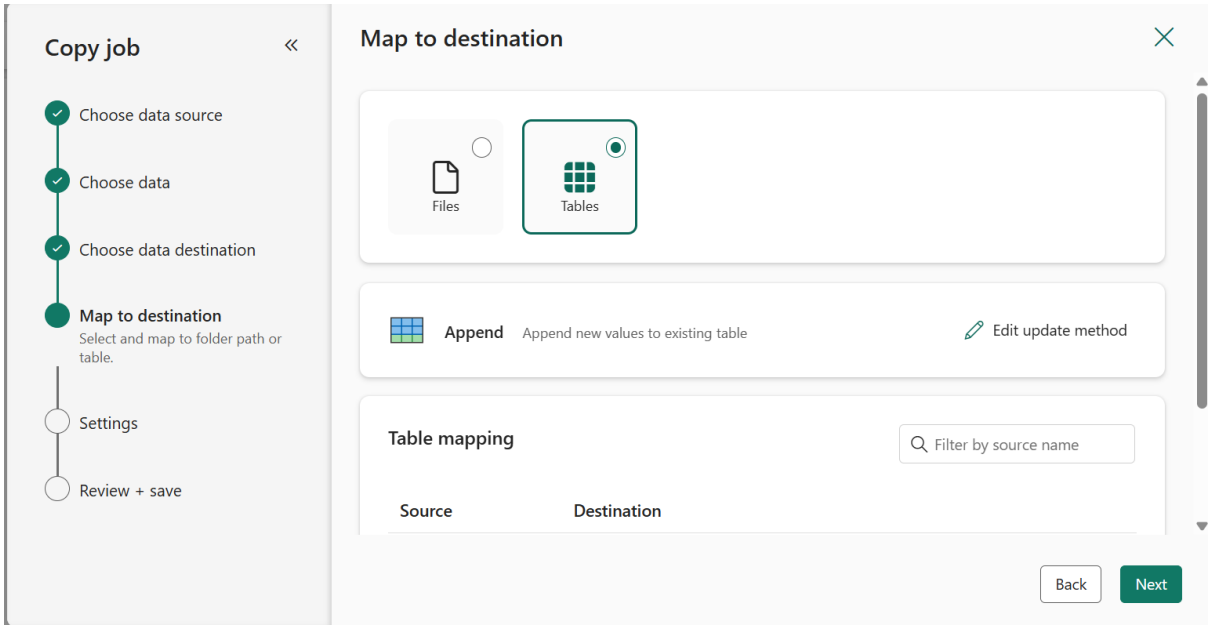




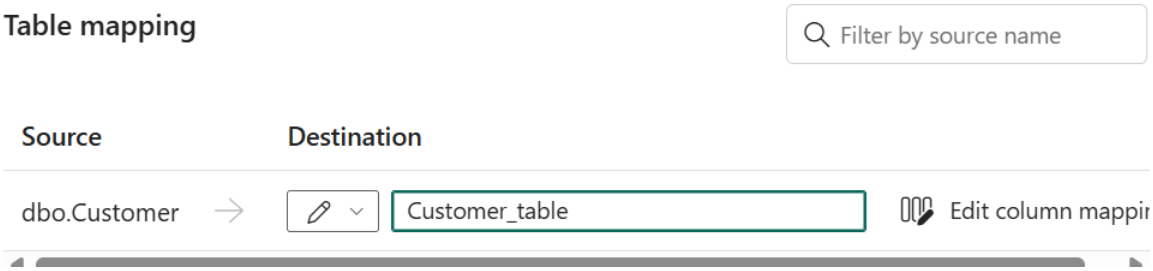
Copy job created, now we have to load source, destination and all other details.

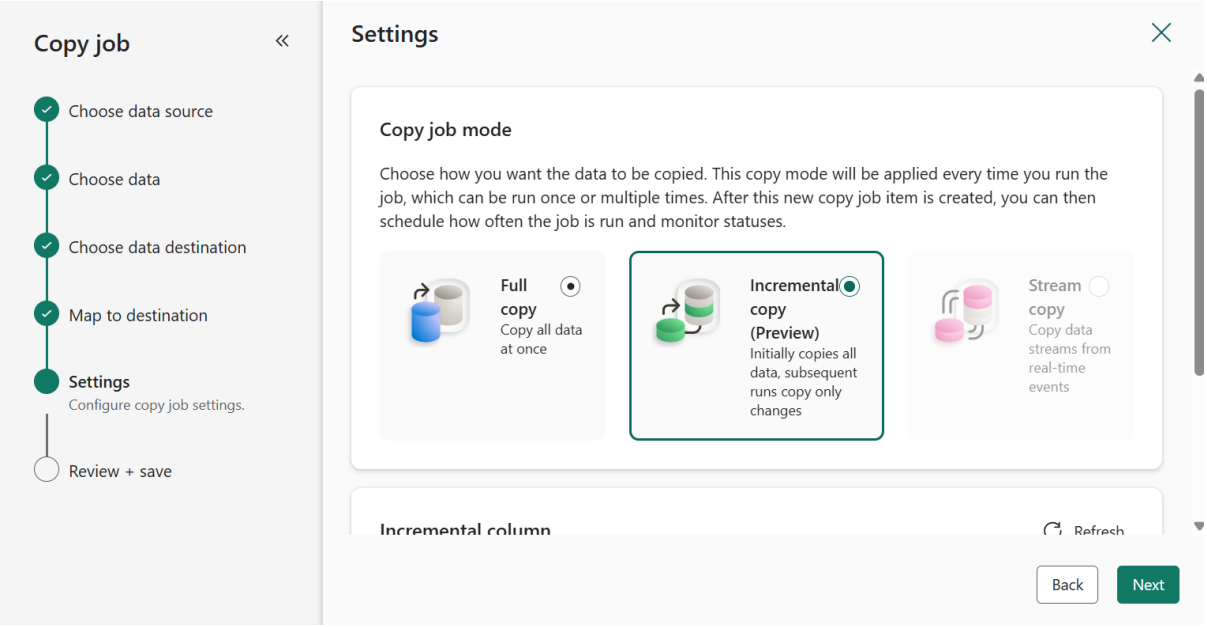


Choosing my Source as warehouse, and loading customer table.

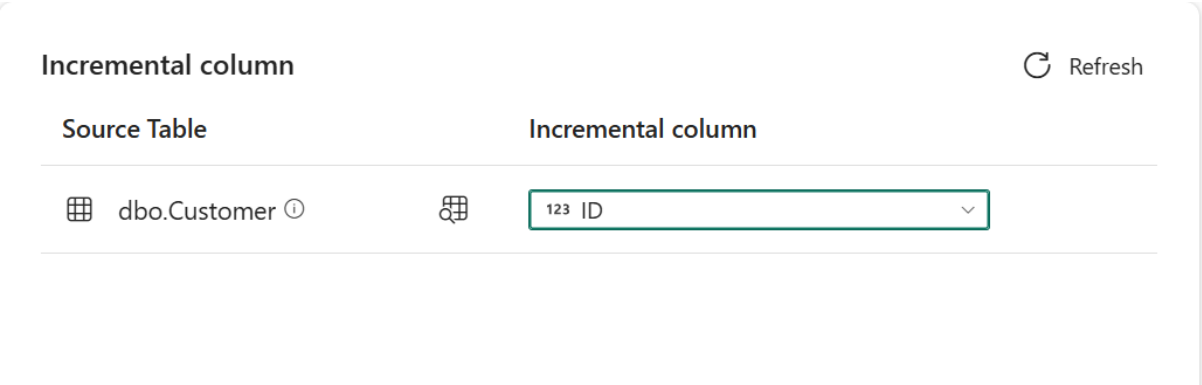


Selected my destination as lake house.

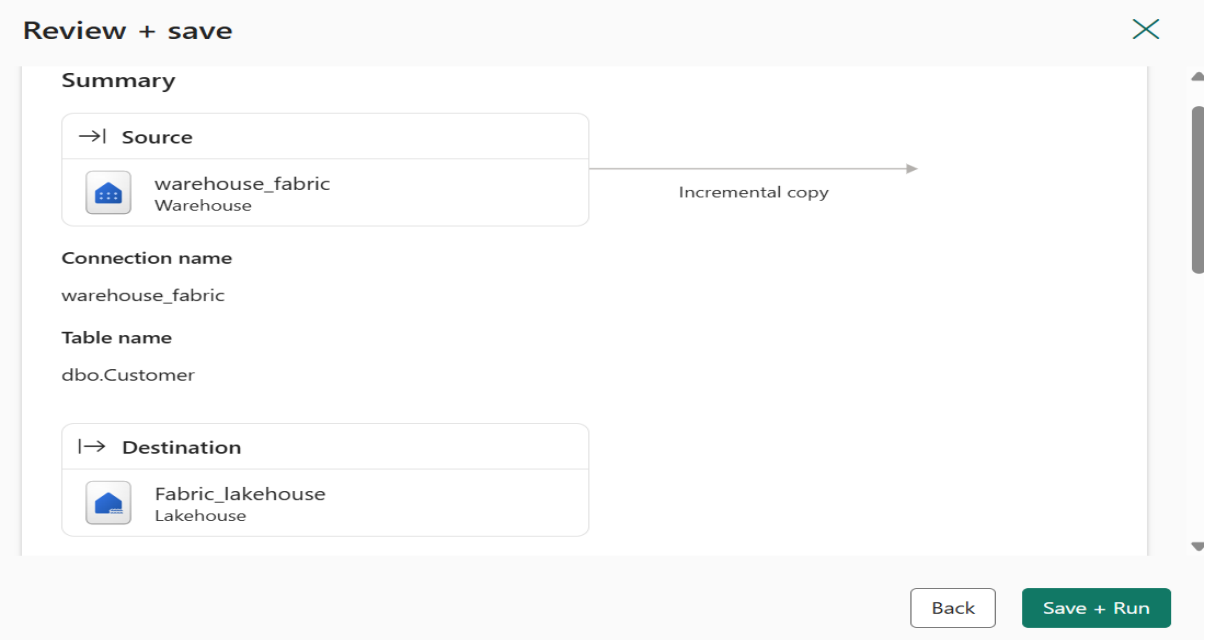


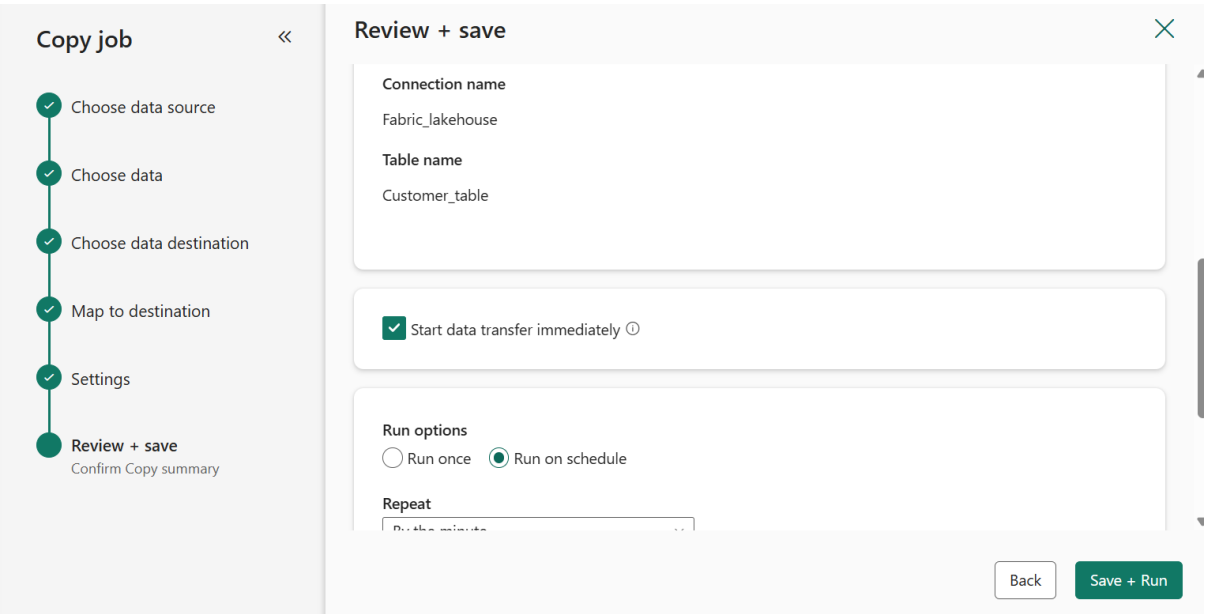


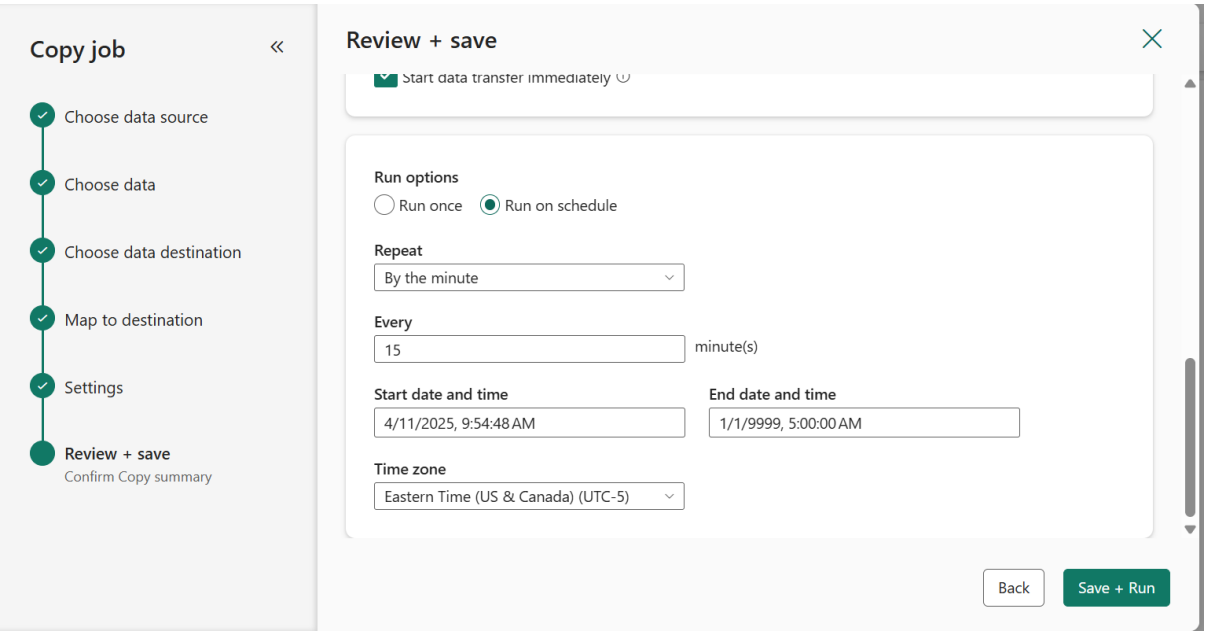
Choosing data to load in incremental load.



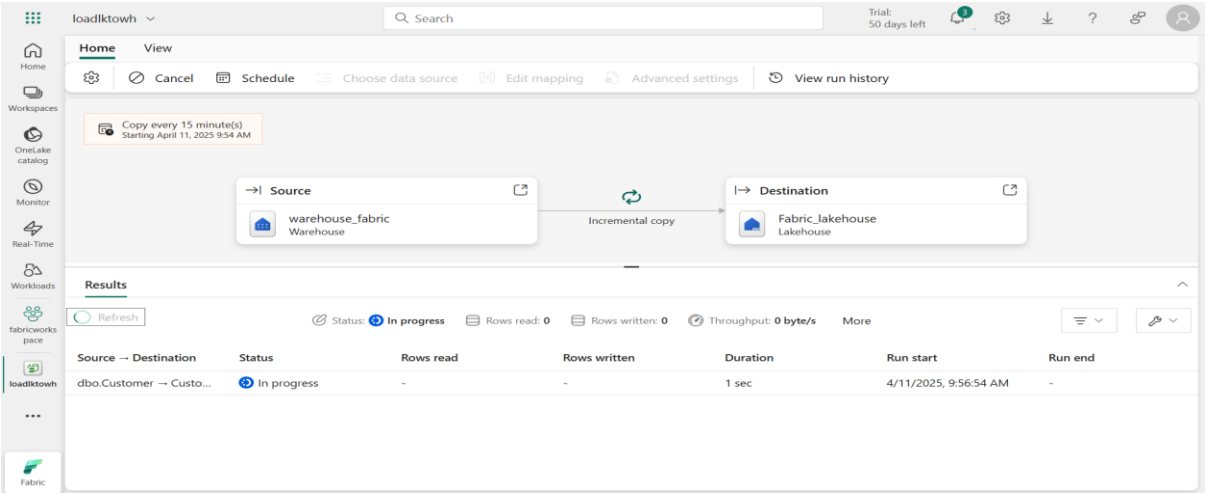
ID column is being used for incremental load.

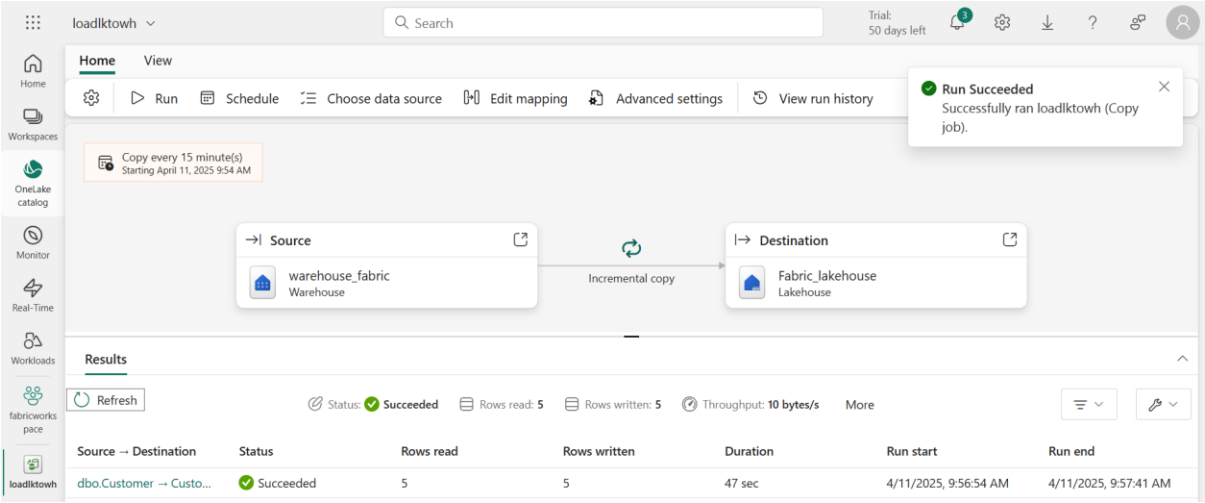






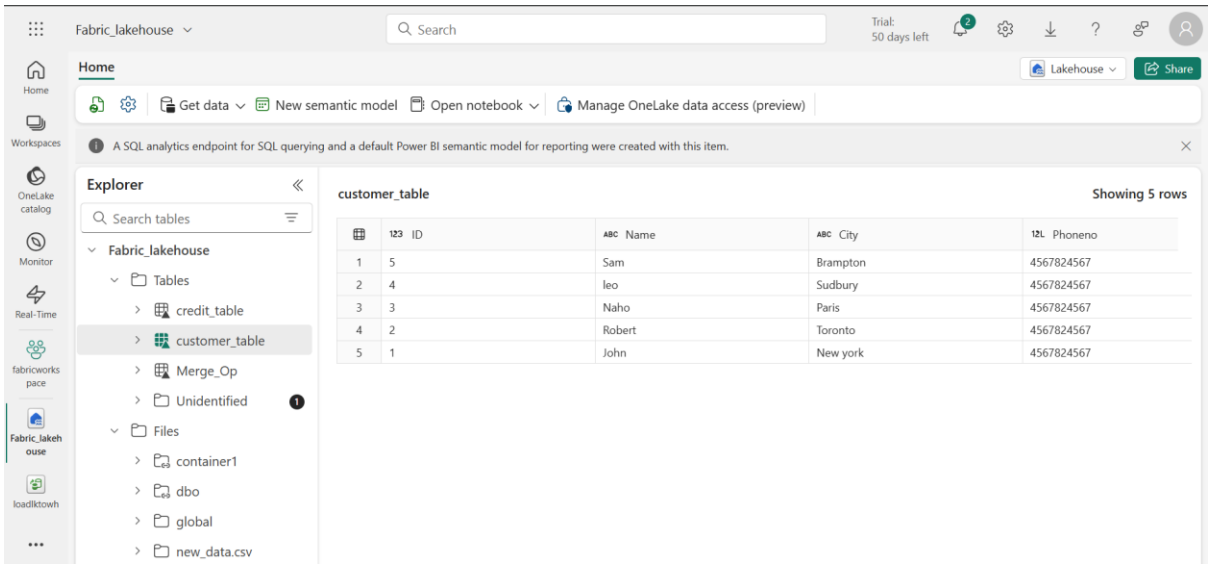
As we are loading data incrementally, we can run on schedule, click save and run.





Pipeline ran successfully.

Checking lake house if the data is loaded correctly or not.

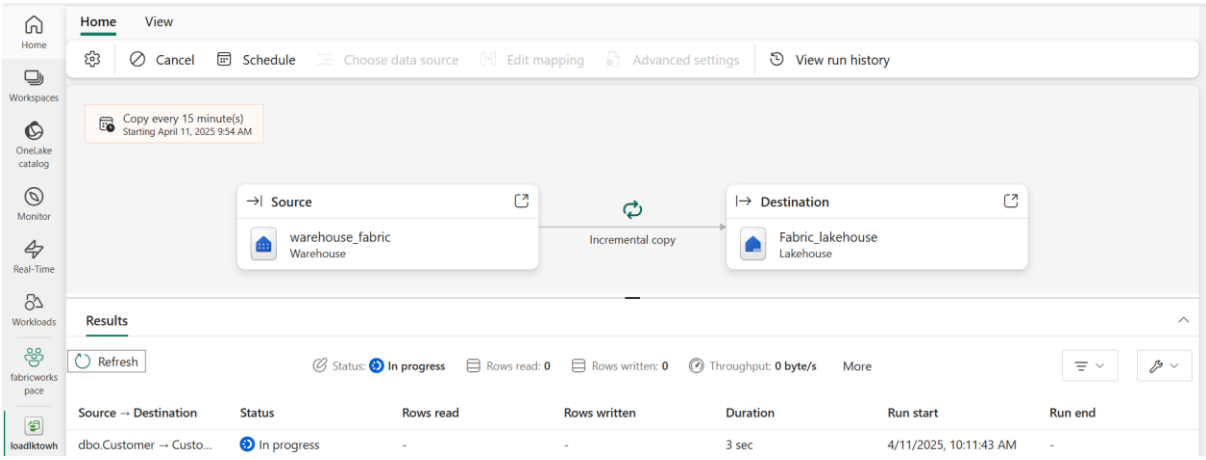


Data loaded successfully.

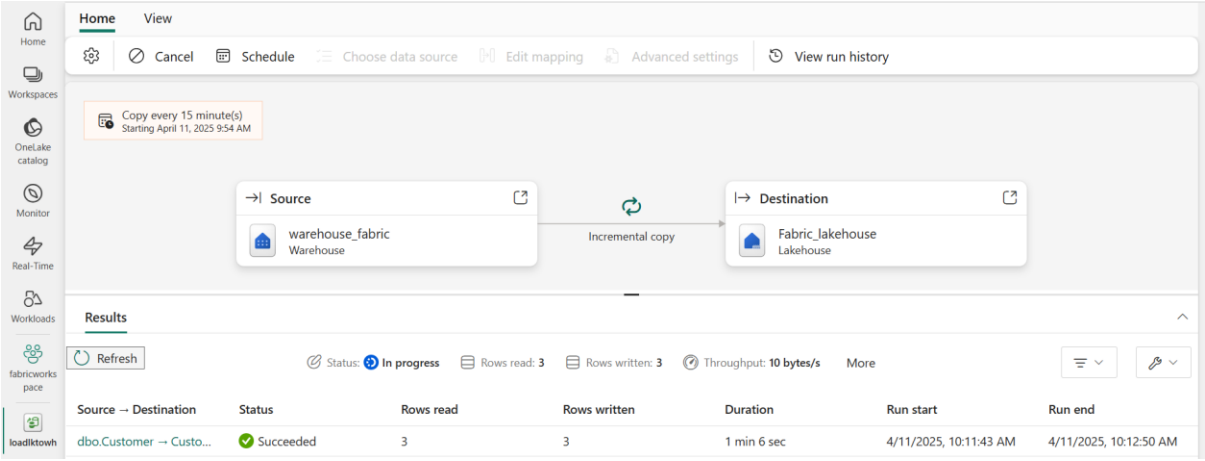
As we opted for incremental load, will add/insert some new rows to table in data warehouse to check if it will work or not.

Data inserted into table.

Pipeline will run after 15 minutes and then let's check.



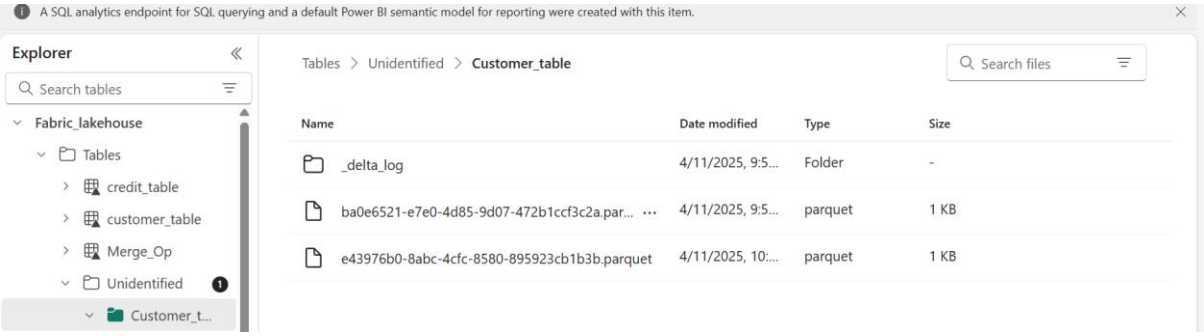
Pipeline started running.



Pipeline ran successfully.

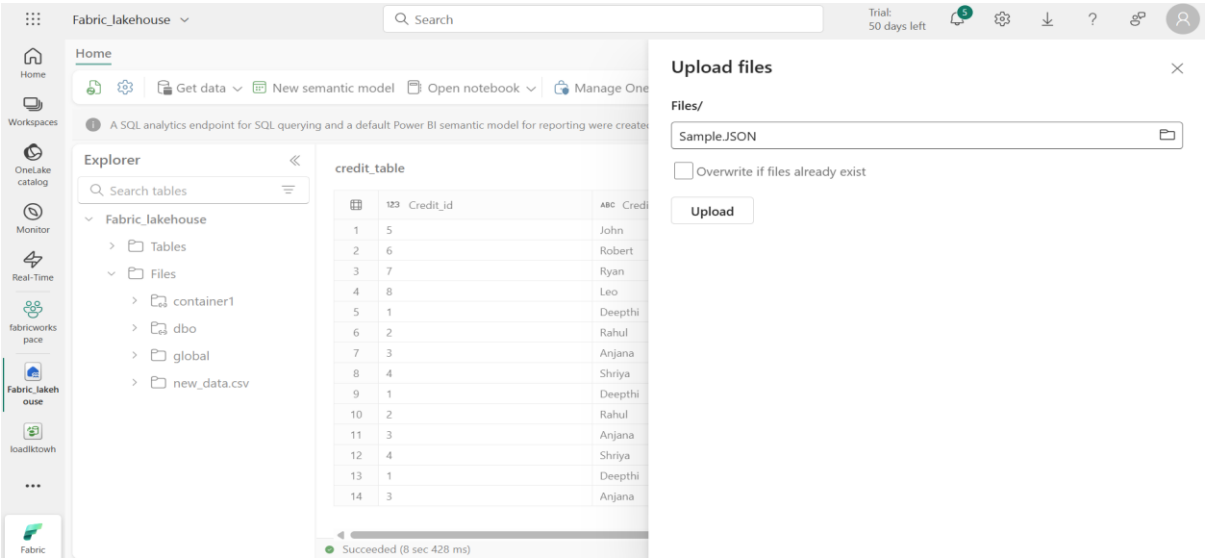
Lets check, if the data is loaded successfully.

Data is stored in delta format.

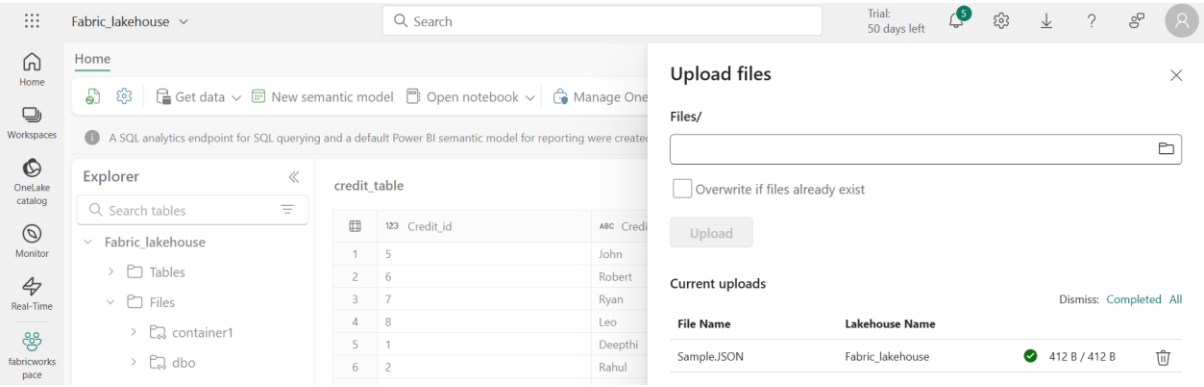


Load JSON data to csv file

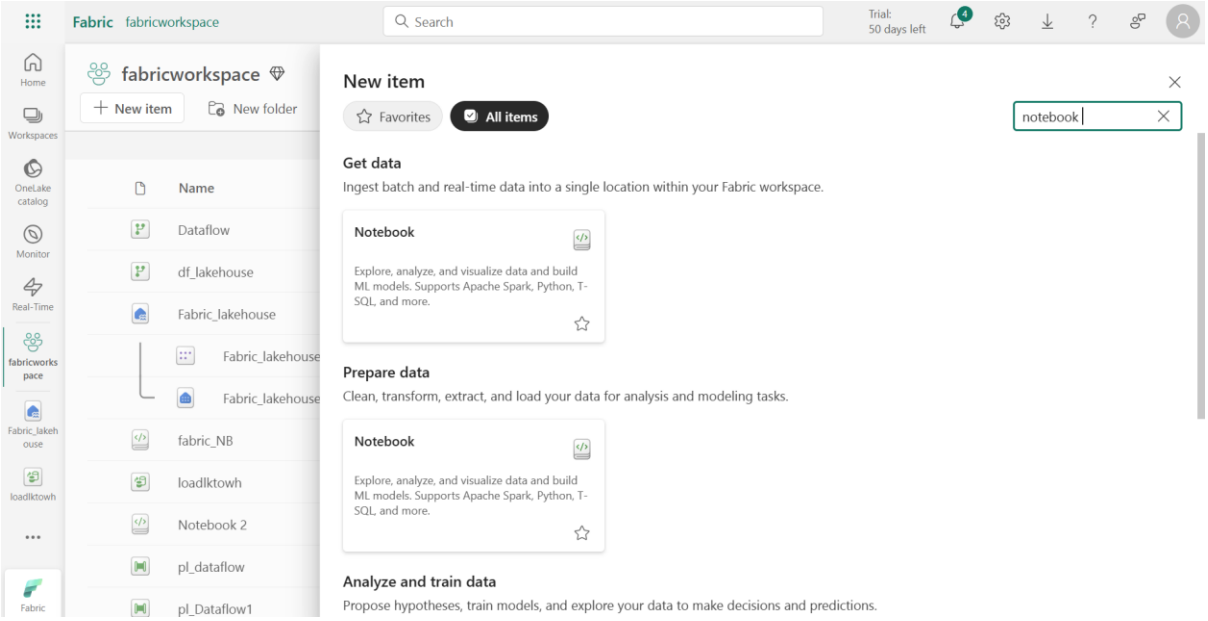
Upload JSON file into lake house.

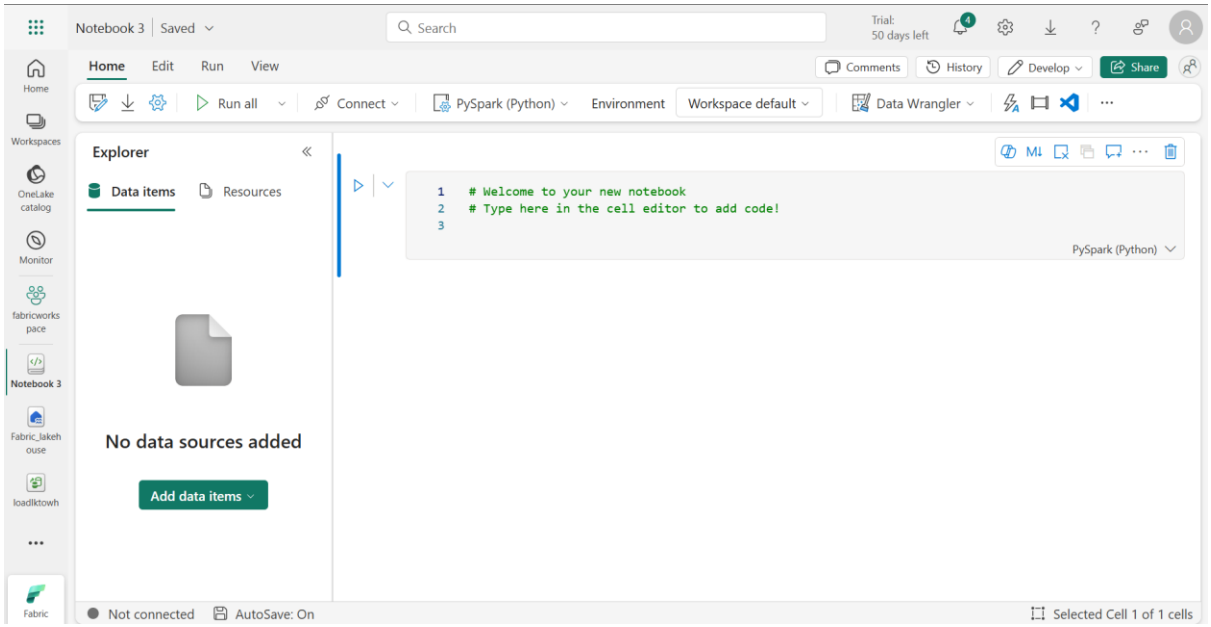


File uploaded successfully.



Create new notebook, go to workspace-> new item -> notebook





Loading data from lake house.

OneLake catalog

Discover data from your org and beyond and use it to create reports

All | My data | Endorsed in your org | Favorites | Filter by keyword

Explorer

Name	Owner	Refreshed	Location	Endorsement	Sensitivity
Fabric_lakehouse	deefabric	—	fabricworkspace	—	—
second_lh	deefabric	—	newws	—	—
DataflowsStagingLakehouse	deefabric	—	fabricworkspace	—	—

Explorer

```
from pyspark.sql.functions import explode, col

```

[1] ✓ 9 sec - Session ready in 9 sec 418 ms. Command executed in 483 ms by deefabric on 10:35:21 AM, 4/11/25 PySpark (Python)

```
json_path = "Files/Sample.JSON"

```

[2] ✓ <1 sec - Command executed in 408 ms by deefabric on 10:36:46 AM, 4/11/25 PySpark (Python)

```
df = spark.read.json(json_path, multiLine=True)
df.show(truncate=False)

```

[4] ✓ <1 sec - Command executed in 922 ms by deefabric on 10:38:01 AM, 4/11/25 PySpark (Python)

Spark jobs (2 of 2 succeeded) Resources Log

```
+---+-------------------------------------+-----+
|id |items                                |name |
+---+-------------------------------------+-----+
|1  |[{I1, Laptop, 1200}, {I2, Mouse, 25}]|Alice|
|2  |[{I3, Monitor, 300}]                 |Bob  |
+---+-------------------------------------+-----+
```

```
df_exploded = df.withColumn("item", explode(col("items")))
df_flat = df_exploded.select(
    col("id"),
    col("name"),
    col("item.id").alias("item_id"),
    col("item.name").alias("item_name"),
    col("item.price").alias("item_price")
)

df_flat.show(truncate=False)
```

[5] ✓ <1 sec - Command executed in 960 ms by deefabric on 10:38:49 AM, 4/11/25 PySpark (Python)

Spark jobs (1 of 1 succeeded) Resources Log

```
+---+-----+-------+---------+----------+
|id |name |item_id|item_name|item_price|
+---+-----+-------+---------+----------+
|1  |Alice|I1     |Laptop   |1200      |
|1  |Alice|I2     |Mouse    |25        |
|2  |Bob  |I3     |Monitor  |300       |
+---+-----+-------+---------+----------+
```

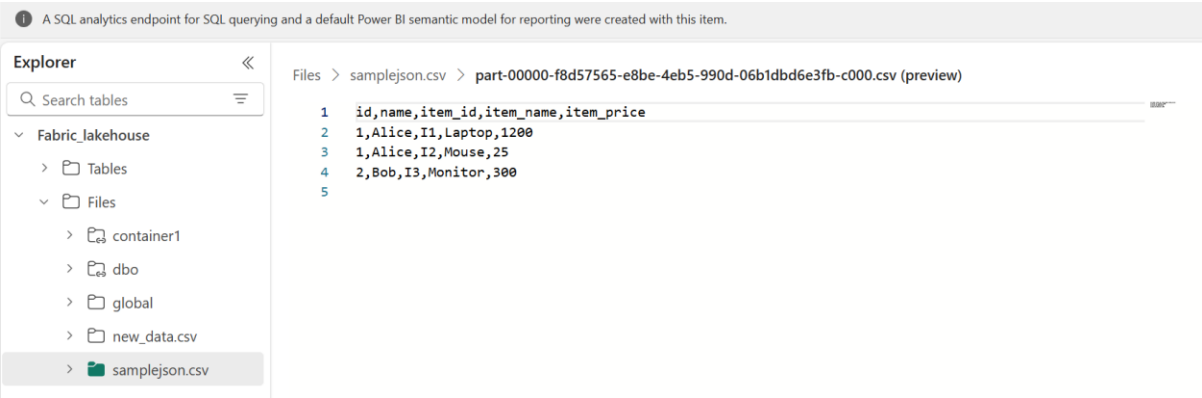
```
df_flat.write.mode("overwrite").option("header", "true").csv("Files/samplejson.csv")

```

[6] ✓ 2 sec - Command executed in 2 sec 485 ms by deefabric on 10:40:13 AM, 4/11/25 PySpark (Python)

Spark jobs (1 of 1 succeeded) Resources Log

Data loaded successfully as CSV file.



The **explode ()** function in PySpark is used to **flatten** an array or map column, turning each element into a separate row.

What It Does

- If a row contains an **array**, explode () creates a new row **for each element** in that array.
- It's commonly used to **flatten nested structures** such as arrays in JSON data.