

DELTA TABLES IN FABRIC

Creating Delta
Tables in
Microsoft Fabric



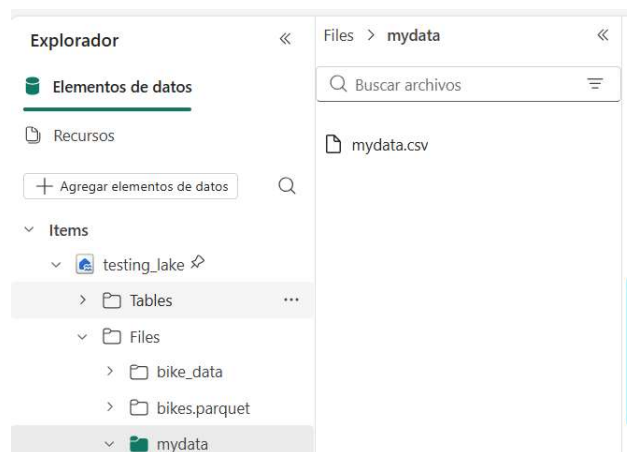
Creation of Delta Tables

When a table is created in a data lakehouse, a Delta Table is defined in the metastore, while the table's data is stored in Parquet files.

With Apache Spark, there is greater control over the data lakehouse, making it easier to create and manage tables.

Creating a Delta table from a dataframe

1. First a csv file is saved in a folder of the lakehouse:



2. Load and Save the data Frame as a delta Table

```
1 #load a file into a dataframe
2 df = spark.read.load('Files/mydata/mydata.csv', format='csv', header=True)
```

[2] ✓ 22 s - Command executed in 17 s 53 ms by Hernán Escriba Najarro on 5:29:22 PM, 9/20/25 PySpark (Python)

Trabajos de Spark (realizado correctamente: 1 de 1) Resources Registro

```
1 display(df)
```

[2] ✓ 2 s - Command executed in 2 s 311 ms by Hernán Escriba Najarro on 5:31:32 PM, 9/20/25 PySpark (Python)

Trabajos de Spark (realizado correctamente: 1 de 1) Resources Registro

Table + Nuevo gráfico Data Wrangler 3 columnas, 10 filas

Vista de tabla

	ABC ID	ABC Nombre	ABC Valor
1	1	Item1	10
2	2	Item2	20
3	3	Item3	30
4	4	Item4	40
5	5	Item5	50
6	6	Item6	60
7	7	Item7	70
8	8	Item8	80
9	9	Item9	90
10	10	Item10	100

Inspeccionar

```
1 # Save the data Frame as a delta Table
2 df.write.format("delta").saveAsTable("mytable")
```

[4] ✓ 30 s - Command executed in 30 s 846 ms by Hernán Escriba Najarro on 5:32:56 PM, 9/20/25 PySpark (Python)

Trabajos de Spark (realizado correctamente: 3 de 3) Resources Registro

- testing_lake
 - Tables
 - bike_data
 - mytable**
 - products
 - salesorders

Manage VS External Tables

Managed Tables: This means that both the table definition in the metastore and the data files are managed by the Spark environment. When the table is deleted, the associated files are also removed.

External Tables: In this case, the relational table definition points to a different storage location.

For example, in the figure below, the table is created in the metastore, but the Parquet files and JSON logs are stored in the *Files/* directory

Manage tables vs External Tables

The screenshot displays a PySpark command executed in a Databricks environment:

```
1 format("delta").saveAsTable("myexternaltable", path="Files/myexternaltable")
```

The command was executed successfully in 7 seconds. Below the command, the Databricks file explorer shows the directory structure of the `testing_lake` workspace:

- `testing_lake`
 - `Tables`
 - `bike_data`
 - `myexternaltable` (highlighted with a green box)
 - `mytable`
 - `products`
 - `salesorders`
 - `Files`
 - `bike_data`
 - `bikes.parquet`
 - `mydata`
 - `myexternaltable` (highlighted with a green box)
 - `_delta_log`
 - `_temporary`

A green arrow points from the `myexternaltable` entry under `Tables` to the `myexternaltable` folder under `Files`, illustrating that the table definition points to a different storage location.

If the table is deleted, the files are still stored.

The screenshot shows the Databricks file explorer with the `myexternaltable` entry under `Tables` highlighted with a red box. A red arrow points to the `Eliminar` (Delete) button in the context menu. Below this, the file explorer shows the directory structure after the table has been deleted:

- `testing_lake`
 - `Tables`
 - `bike_data`
 - `mytable`
 - `products`
 - `salesorders`
 - `Files`
 - `bike_data`
 - `bikes.parquet`
 - `mydata`
 - `myexternaltable` (highlighted with a red box)
 - `_delta_log`

The `myexternaltable` folder under `Files` remains, demonstrating that the files are still stored even after the table definition is removed.

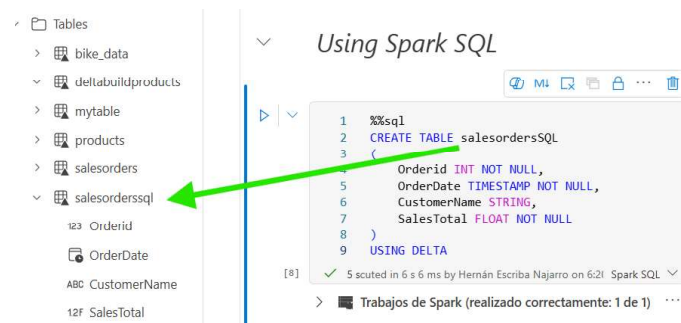
Creation of table Metadata

Usually, tables are created from an existing DataFrame, but sometimes there are scenarios where it is necessary to create a table definition in the metastore and then fill it in different ways. There are several ways to achieve this goal

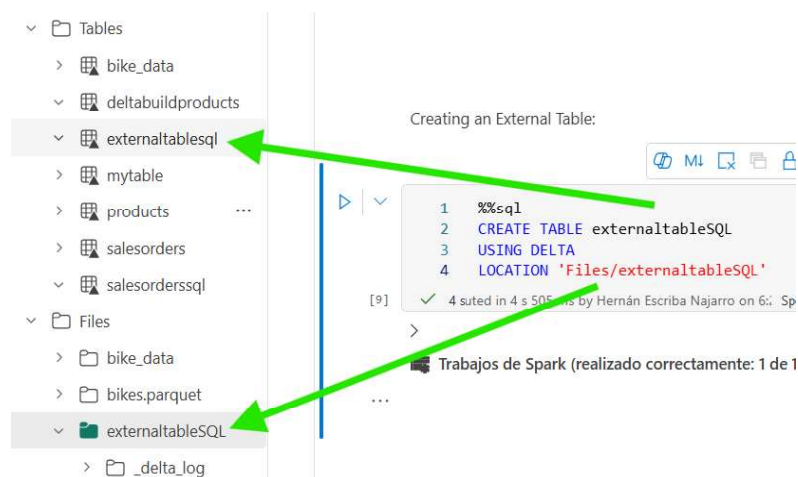
1. DeltaTableBuilder API



2. Using of Spark SQL:



It is also possible to create an external table:



Saving Data only in delta format

Without creating a table, it can be useful to save the results of transformations made with Spark. It is also possible to overwrite or append the data.


Saving only Data in delta format

Delta files are saved in Parquet format and it contains a `delta_log` folder that contains transactions.

```
1 delta_path = "Files/onlydatatable"
2 df.write.format("delta").save(delta_path)
```

[10] ✓ 2 s - Command executed in 2 s 239 ms by Hernán Escriba Najarro on 6:34:42 PM, 9/20/25

PySpark (Python) ▾

>  Trabajos de Spark (realizado correctamente: 3 de 3)  Resources ...

It is possible to replace contain of a existend folder with data from a different dataframe in overwrite mode:

```
1 new_df = df
2 new_df.write.format("delta").mode("overwrite").save(delta_path)
```

[11] ✓ 8 s - Command executed in 8 s 215 ms by Hernán Escriba Najarro on 6:41:36 PM, 9/20/25

PySpark (Python) ▾

>

  ... 

Append:

```
1 new_rows_df = new_df
2 new_rows_df.write.format("delta").mode("append").save(delta_path)
```

[12] ✓ 2 s - Command executed in 2 s 239 ms by Hernán Escriba Najarro on 6:44:20 PM, 9/20/25

PySpark (Python) ▾

>