

# CREATING A DELTA TABLE USING MICROSOFT FABRIC

- This below shows a simple way of creating a table and saving it as a Delta format

## Create Delta Tables

With apache spark, you can create a delta table which will be stored in underlining Parquet files for the table

### 1. Create Delta Table from Dataframe

One of the easiest way of creating a delta table in Spark is to save a dataframe in the **delta format**

```
In [2]: # First Let's load a file into a dataframe
df = spark.read.load('Files/measles.csv', format = 'csv', header = True)

# Then save the dataframe as a delta format
df.write.format("delta").saveAsTable("MyDeltaTable")
```

StatementMeta(, f2a86753-f390-439a-b815-2d9b325186ca, 4, Finished, Available)

```
In [ ]: """
You can also create a table as an external tables, where the relational table definition
in the metastore is mapped to an alternative file storage location
"""

df.write.format('delta').saveAsTable("myexternalsample", path = "Files/externaltable")

# The above can also be modified as a fully qualified path for astorage location
df.write.format("delta").saveAsTable("myexternalsample", path = "abffff://my_store_url")
#here deleting an external table from the lakehouse **does not** delete the associated data
```

## Creating table metadata

### 1. DeltaTableBuilder API Approach

This will enables spark to create a table based on one's specification

```
In [5]: from delta.tables import *

DeltaTable.create(spark) \
    .tableName("products") \
    .addColumn("ProductsId", "INT") \
    .addColumn("ProductName", "STRING") \
    .addColumn("Category", "STRING") \
```

```
.addColumn("Price", "FLOAT") \
.execute()
```

StatementMeta(, f2a86753-f390-439a-b815-2d9b325186ca, 7, Finished, Available)

## 2. Spark SQL Approach

This will enables spark **SQL CREATE TABLE STATEMENT** to create a table based on one's specification

```
In [ ]: %%sql

CREATE TABLE testsales
(
    Orderid INT NOT NULL,
    OrderDate TIMESTAMP NOT NULL,
    CustomerName STRING,
    SalesTotal FLOAT NOT NULL
)
USING DELTA
-- The above is a manage table
```

```
In [ ]: %%sql

-- To create an external table, use the below SQL statement

CREATE TABLE MyExternalTableTest
USING DELTA
LOCATION 'Files/mydataFolder'
```

```
In [ ]:
```

## Save Data in Delta Format

```
In [ ]: # To save a dataframe to a new folder location in delta format; Use the code below

delta_path = "Files/mydataTable"
df.write.format("delta").save(delta_path)
```

```
In [ ]: # Assuming you have created the delta_path = "Files/mydataTable" previously
# You can replace the content of an existing folder in a dataframe using the **Overwrite
latest_df.write.format("delta").mode("overwrite").save(delta_path)
```

```
In [ ]: # Adding rows from a dataframe to an existing folder is possible by using the append mode
new_rows_df.write.format("delta").mode("append").save(delta_path)
```

```
In [ ]:
```

# Working with Delta Tables in Spark

## 1. Using Spark SQL

```
In [ ]: spark.sql("INSERT INTO products VALUES (1, 'Widget', 'Accessories', 2.99)")

# Alternatively you can use the %%sql to call Spark SQL
```

## 2. Using the Delta API

```
In [ ]: from delta.tables import *
        from pyspark.sql.functions import *

        # Create a DeltaTable object
        delta_path = "Files/mytable"
        deltaTable = DeltaTable.forPath(spark, delta_path)

        # Update the table (reduce price of accessories by 10%)
        deltaTable.update(
            condition = "Category == 'Accessories'",
            set = { "Price": "Price * 0.9" })
```

```
In [ ]:
```

## 3. Use time travel to work with table versioning

```
In [4]: %%sql

DESCRIBE HISTORY sales

StatementMeta(, 830ddfff-3726-4f5e-b6a6-39a5bd6b4fd7, 6, Finished, Available)
Out[4]: <Spark SQL result set with 1 rows and 15 fields>
```

```
In [ ]: %%sql

DESCRIBE HISTORY 'Files/mytable'
-- use the above code for an external tables history
```

```
In [ ]: # You can also specify a specific delta file version into a dataframe using VersionAsOf

df = spark.read.format("delta").option("versionAsOf", 0).load(delta_path)
```

```
In [ ]: # Alternatively, use timestampAsOf option

df = spark.read.format("delta").option("timestampAsOf", '2022-01-01').load(delta_path)
```

PREPARED BY BAMIDELE AJAMU