Upload your files:

1. Go to Data Ingestion on the left menu
2. In Legacy products selectUpload files to DBFS
3. Upload the files here

File to be uploaded: https://raw.githubusercontent.com/deepanshuMeteor/Databricks-AWS/refs/heads/main/products.csv

In the first cell of the notebook, enter the following code, which uses shell commands to download data files from GitHub into the file system used by your cluster.

code

df = spark.read.load('/FileStore/tables/products.csv', format='csv', header=True)

display(df.limit(10))

**Load the file data into a delta table**

The data has been loaded into a dataframe. Let’s persist it into a delta table.

1. Add a new code cell and use it to run the following code:

code

delta\_table\_path = "/delta/products-delta"

df.write.format("delta").save(delta\_table\_path)

The data for a delta lake table is stored in Parquet format. A log file is also created to track modifications made to the data.

1. The file data in Delta format can be loaded into a **DeltaTable** object, which you can use to view and update the data in the table. Run the following code in a new cell to update the data; reducing the price of product 771 by 10%.

code

from delta.tables import \*

from pyspark.sql.functions import \*

# Create a deltaTable object

deltaTable = DeltaTable.forPath(spark, delta\_table\_path)

# Update the table (reduce price of product 771 by 10%)

deltaTable.update(

condition = "ProductID == 771",

set = { "ListPrice": "ListPrice \* 0.9" })

# View the updated data as a dataframe

deltaTable.toDF().show(10)

The update is persisted to the data in the delta folder, and will be reflected in any new dataframe loaded from that location.

1. Run the following code to create a new dataframe from the delta table data:

code

new\_df = spark.read.format("delta").load(delta\_table\_path)

new\_df.show(10)

Explore logging and *time-travel*

Data modifications are logged, enabling you to use the *time-travel* capabilities of Delta Lake to view previous versions of the data.

1. In a new code cell, use the following code to view the original version of the product data:

code

new\_df = spark.read.format("delta").option("versionAsOf", 0).load(delta\_table\_path)

new\_df.show(10)

1. The log contains a full history of modifications to the data. Use the following code to see a record of the last 10 changes:

code

deltaTable.history(10).show(10, False, True)

**Create catalog tables**

So far you’ve worked with delta tables by loading data from the folder containing the parquet files on which the table is based. You can define *catalog tables* that encapsulate the data and provide a named table entity that you can reference in SQL code.

* *Managed* tables, that are defined in the metastore.

**Create a managed table**

1. Run the following code to create (and then describe) a managed table named **ProductsManaged** based on the dataframe you originally loaded from the **products.csv** file (before you updated the price of product 771).

code

df.write.format("delta").saveAsTable("AdventureWorks.ProductsManaged")

spark.sql("DESCRIBE EXTENDED AdventureWorks.ProductsManaged").show(truncate=False)

You did not specify a path for the parquet files used by the table - this is managed for you in the Hive metastore, and shown in the **Location** property in the table description.

1. Use the following code to query the managed table, noting that the syntax is just the same as for a managed table:

sql

%sql

USE AdventureWorks;

SELECT \* FROM ProductsManaged;

Show managed tables

1. Use the following code to list the tables in the **AdventureWorks** database:

sql

%sql

USE AdventureWorks;

SHOW TABLES;