

## Assignment 3: Working with Delta Tables

### Tasks:

#### 1. Convert CSV and JSON Data to Delta Format:

```
1 from pyspark.sql import SparkSession
2 import pyspark.sql.functions as F
3
4 # Load employee.csv file data
5 df_employee = spark.read.csv('/FileStore/Employees_data.csv', header=True, inferSchema=True).cache()
6 df_employee.show()
7 df_employee.printSchema()
```

Output Terminal Debug Console

EmployeeID	Name	Department	JoiningDate	Salary
1001	John Doe	HR	2021-01-15	55000
1002	Jane Smith	IT	2020-03-10	62000
1003	Emily Johnson	Finance	2019-07-01	70000
1004	Michael Brown	HR	2018-12-22	54000
1005	David Wilson	IT	2021-06-25	58000
1006	Linda Davis	Finance	2020-11-15	67000
1007	James Miller	IT	2019-08-14	65000
1008	Barbara Moore	HR	2021-03-29	53000

```
root
|-- EmployeeID: integer (nullable = true)
|-- Name: string (nullable = true)
|-- Department: string (nullable = true)
|-- JoiningDate: date (nullable = true)
|-- Salary: integer (nullable = true)
```

```
1 # Load product_data.json file
2 df = spark.read.option("multiline", "true").json("/FileStore/product_data.json")
3 df.show(10)
4 df.printSchema()
```

Output Terminal Debug Console

Category	Price	ProductID	ProductName	Stock
Electronics	1200	101	Laptop	35
Electronics	800	102	Smartphone	80
Furniture	150	103	Desk Chair	60
Electronics	300	104	Monitor	45
Furniture	350	105	Desk	25

```
root
|-- Category: string (nullable = true)
|-- Price: long (nullable = true)
|-- ProductID: long (nullable = true)
|-- ProductName: string (nullable = true)
|-- Stock: long (nullable = true)
```

## 2. Register Delta Tables:

- Register both the employee and product Delta tables as SQL tables.

## 3. Data Modifications with Delta Tables:

- Perform an update operation on the employee Delta table: Increase the salary by 5% for all employees in the IT department.
- Perform a delete operation on the product Delta table: Delete products where the stock is less than 40.

```
▶ ✓ Just now (6s) 6

# 2. Convert CSV and JSON Data to Delta Format
df_employee.write.format("delta").mode("overwrite").save("/dbfs/FileStore/delta/Employees_data")
df.write.format("delta").mode("overwrite").save("/dbfs/FileStore/delta/product_data")

# 3. Register Delta Tables as SQL Tables
spark.sql("CREATE TABLE IF NOT EXISTS Employees_delta USING DELTA LOCATION '/dbfs/FileStore/delta/Employees_data'")
spark.sql("CREATE TABLE IF NOT EXISTS product_delta USING DELTA LOCATION '/dbfs/FileStore/delta/product_data'")

# 3. Data Modifications with Delta Tables
# Increase salary by 5% for IT department employees
spark.sql("UPDATE Employees_delta SET Salary = Salary * 1.05 WHERE Department = 'IT'")
# Delete products where stock is less than 40
spark.sql("DELETE FROM product_delta WHERE Stock < 40")
```

EmployeeID	Name	Department	JoiningDate	Salary
1001	John Doe	HR	2021-01-15	55000
1002	Jane Smith	IT	2020-03-10	62000
1003	Emily Johnson	Finance	2019-07-01	70000
1004	Michael Brown	HR	2018-12-22	54000
1005	David Wilson	IT	2021-06-25	58000
1006	Linda Davis	Finance	2020-11-15	67000
1007	James Miller	IT	2019-08-14	65000
1008	Barbara Moore	HR	2021-03-29	53000

#### 4. Time Travel with Delta Tables:

- Query the product Delta table to show its state before the delete operation (use time travel).
- Retrieve the version of the employee Delta table before the salary update.

▶ 2 minutes ago (7s) 6

```
11 # 4. Time Travel with Delta Tables:
12 # Query the product Delta table to show its state before the delete operation (use time travel).
13 df_product_version_before_delete = spark.sql("SELECT * FROM product_delta VERSION AS OF 0")
14 df_product_version_before_delete.show()
15 # Retrieve the version of the employee Delta table before the salary update.
16 df_employee_version_before_update = spark.sql("SELECT * FROM Employees_delta VERSION AS OF 0")
17 df_employee_version_before_update.show()
```

Output Terminal Debug Console

Category	Price	ProductID	ProductName	Stock
Electronics	1200	101	Laptop	35
Electronics	800	102	Smartphone	80
Furniture	150	103	Desk Chair	60
Electronics	300	104	Monitor	45
Furniture	350	105	Desk	25

  

EmployeeID	Name	Department	JoiningDate	Salary
1001	John Doe	HR	2021-01-15	55000
1002	Jane Smith	IT	2020-03-10	62000
1003	Emily Johnson	Finance	2019-07-01	70000
1004	Michael Brown	HR	2018-12-22	54000
1005	David Wilson	IT	2021-06-25	58000
1006	Linda Davis	Finance	2020-11-15	67000
1007	James Miller	IT	2019-08-14	65000
1008	Barbara Moore	HR	2021-03-29	53000

## 5. Query Delta Tables:

- Query the employee Delta table to find the employees in the Finance department.
- Query the product Delta table to find all products in the Electronics category with a price greater than 500.

▶

✓ Just now (1s)

6

```
1 # 5. Query Delta Tables:
2 # Query the employee Delta table to find the employees in the Finance department.
3 df_finance_employees = spark.sql("SELECT * FROM Employees_delta WHERE Department = 'Finance'")
4 df_finance_employees.show()
5 # Query the product Delta table to find all products in the Electronics category with a price greater than 500.
6 df_expensive_electronics = spark.sql("SELECT * FROM product_delta WHERE Category = 'Electronics' AND Price > 500")
7 df_expensive_electronics.show()
```

Output Terminal Debug Console

▶ df\_finance\_employees: pyspark.sql.dataframe.DataFrame = [EmployeeID: integer, Name: string ... 3 more fields]

▶ df\_expensive\_electronics: pyspark.sql.dataframe.DataFrame = [Category: string, Price: long ... 3 more fields]

EmployeeID	Name	Department	JoiningDate	Salary
1003	Emily Johnson	Finance	2019-07-01	70000
1006	Linda Davis	Finance	2020-11-15	67000

  

Category	Price	ProductID	ProductName	Stock
Electronics	800	102	Smartphone	80