**#Databricks Delta Lake Exercise**

**#Tasks**

**#1. Create Delta Tables Using 3 Methods**

#1. Load the sales\_data.csv file into a DataFrame.

# Move the file from Workspace to DBFS

dbutils.fs.cp("file:/Workspace/Shared/Sales\_data.csv","dbfs:/FileStore/Sales\_data.csv")

# Load the CSV file from DBFS

sales\_df = spark.read.format("csv").option("header", "true").load("/FileStore/Sales\_data.csv")

sales\_df.show()

#2. Write the DataFrame as a Delta Table.

sales\_df.write.format("delta").mode("overwrite").save("/FileStore/delta/sales\_delta")

#3. Load the customer\_data.json file into a DataFrame.

# Move the file from Workspace to DBFS

dbutils.fs.cp("file:/Workspace/Shared/customer\_data.json", "dbfs:/FileStore/customer\_data.json")

# Load the JSON file from DBFS

customer\_df = spark.read.option("multiline", "true").json("/FileStore/customer\_data.json")

customer\_df.show()

#4. Write the DataFrame as a Delta Table.

customer\_df.write.format("delta").mode("overwrite").save("/FileStore/delta/customer\_data")

#5. Convert an existing Parquet file into a Delta Table

df.write.format("parquet").mode("overwrite").save("/FileStore/customer\_data.json")

parquet\_df = spark.read.format("parquet").load("/FileStore/customer\_data.json")

parquet\_df.write.format("delta").save("/delta/customer\_data\_parquet\_to\_delta")

**#2. Data Management**

#1. Load the new\_sales\_data.csv file into a DataFrame

# Move the file from Workspace to DBFS

dbutils.fs.cp("file:/Workspace/Shared/new\_sales\_data.csv","dbfs:/FileStore/new\_sales\_data.csv")

# Load the new\_sales\_data.csv file from DBFS

new\_sales\_df = spark.read.format("csv").option("header", "true").load("/FileStore/new\_sales\_data.csv")

new\_sales\_df.show()

#2. Write the new DataFrame as a Delta Table.

new\_sales\_df.write.format("delta").mode("overwrite").save("/FileStore/delta/new\_sales\_delta")

# Load the sales\_data.csv into a DataFrame

sales\_df = spark.read.format("csv").option("header", "true").load("/FileStore/Sales\_data.csv")

# Convert the DataFrame to a Delta Table

sales\_df.write.format("delta").save("/delta/sales\_data")

#3. Perform a MERGE INTO operation to update and insert records into the existing Delta table.

from delta.tables import \*

# Load the Delta Table

sales\_delta = DeltaTable.forPath(spark, "/delta/sales\_data")

# Load new sales data

new\_sales\_df = spark.read.format("csv").option("header", "true").load("/FileStore/new\_sales\_data.csv")

# Perform the MERGE INTO operation

sales\_delta.alias("old").merge(

new\_sales\_df.alias("new"),

"old.OrderID = new.OrderID"

).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()

# Create a new Delta Table

spark.sql("CREATE TABLE IF NOT EXISTS sales\_data\_delta USING DELTA LOCATION '/delta/sales\_data'")

#diaplay table

spark.sql("""

SELECT \* FROM sales\_data\_delta

""")

**#3. Optimize Delta Table**

#1. Apply the OPTIMIZE command on the Delta Table

spark.sql("""

OPTIMIZE delta.`/delta/sales\_data` ZORDER BY (CustomerID)

""")

**#4. Advanced Features**

#1. Use DESCRIBE HISTORY to inspect the history of changes for a Delta Table.

spark.sql("""

DESCRIBE HISTORY delta.`/delta/sales\_data`

""")

#2. Use VACUUM to remove old files from the Delta Table.

spark.sql("""

VACUUM delta.`/delta/sales\_data` RETAIN 168 HOURS

""")

**#5. Hands-on Exercises**

#1. Using Delta Lake for Data Versioning:

#Query historical versions of the Delta Table using Time Travel.

spark.sql("""

SELECT \* FROM delta.`/delta/sales\_data` VERSION AS OF 1

""")

#2. Building a Reliable Data Lake with Delta Lake:

#Implement schema enforcement and handle data updates with Delta Lake.

df.write.format("delta").mode("append").option("mergeSchema", "true").save("/delta/sales\_data")