#Assignment 1

#Tasks:

#1. Load the CSV data:

# Move the file from Workspace to DBFS

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

# Load the file from DBFS

df = spark.read.format("csv").option("header", "true").load("/FileStore/employee\_data.csv")

df.show()

# Show the first 10 rows

print("Top 10 rows")

df.show(10)

# Inspect the schema

df.printSchema()

#2. Data Cleaning:

# Remove rows where Salary is less than 55000

df\_cleaned = df.filter(df['Salary'] > 55000)

# Filter employees who joined after the year 2020

df\_cleaned2 = df\_cleaned.filter(df\_cleaned['JoiningDate'] > '2020-12-31')

# Show the cleaned data

print("Cleaned Data:")

df\_cleaned.show()

df\_cleaned2.show()

#3. Data Aggregation:

# Find the average salary by Department

avg\_salary = df\_cleaned.groupBy('Department').agg({'Salary': 'avg'})

print("Avg salary")

avg\_salary.show()

# Count the number of employees in each Department

employee\_count = df\_cleaned.groupBy('Department').count()

print("Department count")

employee\_count.show()

#4. Write the Data to CSV:

df.write.format("csv").option("header","true").save("/workspace/Shared/employeecsv\_output")

print("Data written to CSV")

#Assignment 2

#Tasks:

#1. Load the JSON data:

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

# Load

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

df.show()

# Show the first 10 rows

print("Top 10 rows")

df.show(10)

# Inspect the schema

df.printSchema()

#2. Data Cleaning:

#Remove rows where Stock is less than 30.

df\_cleaned = df.filter(df['Stock'] >= 30)

#Filter the products that belong to the "Electronics" category.

df\_cleaned = df.filter(df['Category'] == 'Electronics')

#show

print("Cleaned Data:")

df\_cleaned.show()

#3. Data Aggregation:

#Calculate the total stock for products in the "Furniture" category.

total\_stock = df.filter(df['Category'] == 'Furniture').groupBy('Category').agg({'Stock': 'sum'}).withColumnRenamed('sum(Stock)', 'TotalStock')

print("Total Stock")

total\_stock.show()

#Find the average price of all products in the dataset.

avg\_price = df.groupBy('Category').agg({'Price': 'avg'}).withColumnRenamed('avg(Price)', 'AvgPrice')

print("Average Price")

avg\_price.show()

#4. Write the Data to JSON:

df\_cleaned.coalesce(1).write.json('/FileStore/product\_data\_cleaned.json')

print("Data written to JSON")

# Assignment 3:

# Tasks:

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

# Load employee.csv file data

df\_employee = spark.read.csv('/FileStore/employee\_data.csv', header=True, inferSchema=True).cache()

df\_employee.show()

# Load product\_data.json file

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

df.show()

#save data

df\_employee.write.format("delta").mode("overwrite").save("/dbfs/FileStore/delta/employee\_data")

df.write.format("delta").mode("overwrite").save("/dbfs/FileStore/delta/product\_data")

print("Data saved")

#2. Register Delta Tables:

print("Registered")

spark.sql("CREATE TABLE IF NOT EXISTS employee\_delta USING DELTA LOCATION '/dbfs/FileStore/delta/employee\_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 employee\_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")

print("Data modified")

#4. Time Travel with Delta Tables:

# Query the product Delta table to show its state before the delete

# operation (use time travel).

print("Data retrieved")

df\_product\_version\_before\_delete = spark.sql("SELECT \* FROM product\_delta VERSION AS OF 0")

df\_product\_version\_before\_delete.show()

# Retrieve the version of the employee Delta table before the salary update.

print("Data retrieved")

df\_employee\_version\_before\_update = spark.sql("SELECT \* FROM employee\_delta VERSION AS OF 0")

df\_employee\_version\_before\_update.show()

#5. Query Delta Tables:

# Query the employee Delta table to find the employees in the Finance department.

print("employees in Finance department")

df\_finance\_employees = spark.sql("SELECT \* FROM employee\_delta WHERE Department = 'Finance'")

df\_finance\_employees.show()

# Query the product Delta table to find all products in the Electronics category with a price greater than 500.

print("expensive electronics")

df\_expensive\_electronics = spark.sql("SELECT \* FROM product\_delta WHERE Category = 'Electronics' AND Price > 500")

df\_expensive\_electronics.show()