ASSIGNMENT 1

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

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

df.show()

import pyspark.sql.functions as F

# Assuming the CSV file is in your Databricks workspace

employee\_df = spark.read.csv("/FileStore/exer.csv", header=True, inferSchema=True)

# Display the first 10 rows

employee\_df.show(10)

# Inspect the schema

employee\_df.printSchema()

# Remove rows with Salary less than 55,000

filtered\_df = employee\_df.filter(F.col("Salary") >= 55000)

# Filter employees who joined after 2020

filtered\_df = filtered\_df.filter(F.year(F.col("JoiningDate")) > 2020)

# Average salary by Department

avg\_salary\_by\_department = filtered\_df.groupBy("Department").agg(F.avg("Salary").alias("AverageSalary"))

# Count of employees in each Department

employee\_count\_by\_department = filtered\_df.groupBy("Department").count()

# Save the cleaned data to a new CSV file

filtered\_df.write.csv("cleaned\_employee\_data.csv", header=True)

ASSIGNMENT 2

dbutils.fs.cp("file:/Workspace/Shared/product\_data (1).json", "dbfs:/FileStore/product\_data(1).json")

df = spark.read.option("multiline", "true").json("/FileStore/product\_data(1).json")

df.show(10)

df.printSchema()

# Remove rows where Stock is less than 30.

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

df\_cleaned\_product = df.filter((df['Stock'] >= 30) & (df['Category'] == 'Electronics'))

df\_cleaned\_product.show()

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

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

df\_total\_furniture\_stock.show()

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

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

df\_avg\_price.show()

# Save the cleaned and aggregated data into a new JSON file.

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

ASSIGNMENT 3

# Load employee.csv file data

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

df\_employee.show()

df\_employee.printSchema()

# Load product\_data.json file

df = spark.read.option("multiline", "true").json("/FileStore/product\_data(1).json")

df.show(10)

df.printSchema()

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

df.write.format("delta").mode("overwrite").save("/dbfs/FileStore/delta/product\_data(1)")

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

spark.sql("CREATE TABLE IF NOT EXISTS product\_delta USING DELTA LOCATION '/dbfs/FileStore/delta/product\_data(1)'")

# 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")

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

# operation (use time travel).

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.

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

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

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

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

df\_expensive\_electronics.show()