### Data Setup:

from pyspark.sql import SparkSession

from pyspark.sql import functions as F

from pyspark.sql.window import Window

# Initialize a Spark session

spark = SparkSession.builder \

.appName("Advanced DataFrame Operations - Different Dataset") \

.getOrCreate()

# Create two sample DataFrames for Product Sales

data1 = [

(1, 'Product A', 'Electronics', 1200, '2022-05-10'),

(2, 'Product B', 'Clothing', 500, '2022-07-15'),

(3, 'Product C', 'Electronics', 1800, '2021-11-05')

]

data2 = [

(4, 'Product D', 'Furniture', 3000, '2022-03-25'),

(5, 'Product E', 'Clothing', 800, '2022-09-12'),

(6, 'Product F', 'Electronics', 1500, '2021-10-19')

]

# Define schema (columns)

columns = ['ProductID', 'ProductName', 'Category', 'Price', 'SaleDate']

# Create DataFrames

sales\_df1 = spark.createDataFrame(data1, columns)

sales\_df2 = spark.createDataFrame(data2, columns)

#show the dataframes

sales\_df1.show()

sales\_df2.show()

### Tasks:

#1. \*\*Union of DataFrames (removing duplicates)\*\*:

union\_df = sales\_df1.union(sales\_df2).dropDuplicates()

print("Union of DataFrames (without duplicates):")

union\_df.show()

#2. \*\*Union of DataFrames (including duplicates)\*\*:

union\_all\_df = sales\_df1.union(sales\_df2)

print("Union of DataFrames (with duplicates):")

union\_all\_df.show()

#3. \*\*Rank products by price within their category\*\*:

window\_spec = Window.partitionBy("Category").orderBy(F.desc("Price"))

ranked\_df = union\_df.withColumn("Rank", F.row\_number().over(window\_spec))

print("DataFrame with ranks:")

ranked\_df.show()

#4. \*\*Calculate cumulative price per category\*\*:

cumulative\_df = union\_df.withColumn("CumulativePrice", F.sum("Price").over(window\_spec))

print("DataFrame with cumulative price:")

cumulative\_df.show()

#5. \*\*Convert `SaleDate` from string to date type\*\*:

converted\_df = union\_df.withColumn("SaleDate", F.to\_date("SaleDate", "yyyy-MM-dd"))

print("DataFrame with converted SaleDate:")

converted\_df.show()

#6. \*\*Calculate the number of days since each sale\*\*:

days\_since\_sale\_df = converted\_df.withColumn("DaysSinceSale", F.datediff(F.current\_date(), "SaleDate"))

print("DataFrame with days since sale:")

days\_since\_sale\_df.show()

#7. \*\*Add a column for the next sale deadline\*\*:

next\_sale\_deadline\_df = converted\_df.withColumn("NextSaleDeadline", F.date\_add("SaleDate", 30))

print("DataFrame with next sale deadline:")

next\_sale\_deadline\_df.show()

#8. \*\*Calculate total revenue and average price per category\*\*:

revenue\_df = union\_df.groupBy("Category").agg(

F.sum("Price").alias("TotalRevenue"),

F.avg("Price").alias("AveragePrice")

)

print("Total revenue and average price per category:")

revenue\_df.show()

#9. \*\*Convert all product names to lowercase\*\*:

lower\_case\_df = union\_df.withColumn("ProductNameLower", F.lower("ProductName"))

print("DataFrame with product names in lowercase:")

lower\_case\_df.show()