**PYSPARK EXERCISES 5 (4.9.2024)**

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)

sales\_df1.show()

sales\_df2.show()

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

**#Combine the two DataFrames (`sales\_df1` and `sales\_df2`) using `union` and remove any duplicate rows**.

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

union\_df.show()

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

**#Combine both DataFrames using `unionAll` (replaced by `union`) and include duplicate rows.**

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

union\_all\_df.show()

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

**#Use window functions to rank the products in each category by price in descending order.**

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

ranked\_df=union\_all\_df.withColumn("Rank",rank().over(window\_spec))

ranked\_df.show()

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

**#Use window functions to calculate the cumulative price of products within each category.**

window\_spec = Window.partitionBy("Category").orderBy("Price").rowsBetween(Window.unboundedPreceding, Window.currentRow)

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

union\_df\_with\_cumulative\_price.show()

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

**#Convert the `SaleDate` column from string format to a PySpark date type.**

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

union\_df\_with\_date.printSchema()

union\_df\_with\_date.show()

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

**#Calculate the number of days since each product was sold using the current date.**

union\_df\_with\_days = union\_df\_with\_date.withColumn("DaysSinceSale", F.datediff(F.current\_date(), F.col("SaleDate")))

union\_df\_with\_days.show()

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

**#Add a new column `NextSaleDeadline`, which should be 30 days after the `SaleDate`.**

union\_df\_with\_deadline = union\_df\_with\_date.withColumn("NextSaleDeadline", F.date\_add(F.col("SaleDate"), 30))

union\_df\_with\_deadline.show()

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

**#Find the total revenue (sum of prices) and the average price per category.**

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

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

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

)

revenue\_avg\_price\_df.show()

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

**#Create a new column with all product names in lowercase.**

union\_df\_with\_lowercase = union\_df.withColumn("ProductNameLower", F.lower(F.col("ProductName")))

union\_df\_with\_lowercase.show()