data3 = [

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

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

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

]

data4 =  [

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

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

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

]

new\_columns = ['ProductID', 'ProductName', 'Category', 'Price', 'SaleDate']

sales\_df1 = spark.createDataFrame(data3,new\_columns)

sales\_df2 = spark.createDataFrame(data4,new\_columns)

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

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

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

sales\_union\_df.show()

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

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

sales\_union\_all = sales\_df1.unionAll(sales\_df2)

sales\_union\_all.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\_sale = Window.partitionBy("Category").orderBy(col("Price").desc())

ranked\_sales\_df = sales\_union\_all.withColumn("Rank",rank().over(window\_spec\_sale))

ranked\_sales\_df.show()

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

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

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

cumulative\_total\_df = sales\_union\_all.withColumn("RunningTotal", sum(col("Price")).over(window\_spec\_total))

cumulative\_total\_df.show()

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

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

date\_df = sales\_union\_all.withColumn("SaleDate", F.to\_date(col("SaleDate"), "yyyy-MM-dd"))

date\_df.show()

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

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

days\_df = date\_df.withColumn("NoOfDays",F.datediff(F.current\_date(), col("SaleDate")))

days\_df.show()

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

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

sale\_deadline\_df = date\_df.withColumn("SaleDeadline", F.date\_add(col("SaleDate"),30))

sale\_deadline\_df.show()

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

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

revenue\_df = sales\_union\_all.groupBy("Category").agg(

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

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

)

revenue\_df.show()

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

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

lower\_name\_df = sales\_union\_all.withColumn("ProductNameLower",F.lower(col("ProductName")))

lower\_name\_df.show()