### \*\*Exercise: Working with Key-Value Pair RDDs in PySpark\*\*

### \*\*Step 1: Initialize Spark Context\*\*

#1. \*\*Initialize SparkSession and SparkContext:\*\*

from pyspark.sql import SparkSession

spark = SparkSession.builder \

.appName("Key-Value Pair RDD Transformations") \

.getOrCreate()

sc = spark.sparkContext

print("Spark Session Created")

### \*\*Step 2: Create and Explore the RDD\*\*

#2. \*\*Task 1: Create an RDD from the Sales Data\*\*

sales\_data = [

("ProductA", 100),

("ProductB", 150),

("ProductA", 200),

("ProductC", 300),

("ProductB", 250),

("ProductC", 100)

]

sales\_rdd = sc.parallelize(sales\_data)

# Print the first few elements of the RDD

print(sales\_rdd.collect())

### \*\*Step 3: Grouping and Aggregating Data\*\*

#3. \*\*Task 2: Group Data by Product Name\*\*

grouped\_sales\_rdd = sales\_rdd.groupByKey()

# To see the grouped data

grouped\_sales = grouped\_sales\_rdd.mapValues(list).collect()

print("grouped sales:")

print(grouped\_sales)

#4. \*\*Task 3: Calculate Total Sales by Product\*\*

total\_sales\_rdd = sales\_rdd.reduceByKey(lambda a, b: a + b)

# Print the total sales for each product

print("Total Sales by Product:")

print(total\_sales\_rdd.collect())

#5. \*\*Task 4: Sort Products by Total Sales\*\*

sorted\_sales\_rdd = total\_sales\_rdd.sortBy(lambda x: x[1], ascending=False)

# Print the sorted list of products along with their sales amounts

print("Sorted Products by Total Sales:")

print(sorted\_sales\_rdd.collect())

### \*\*Step 4: Additional Transformations\*\*

#6. \*\*Task 5: Filter Products with High Sales\*\*

high\_sales\_rdd = total\_sales\_rdd.filter(lambda x: x[1] > 200)

# Print the products with high sales

print("Products with High Sales:")

print(high\_sales\_rdd.collect())

#7. \*\*Task 6: Combine Regional Sales Data\*\*

regional\_sales\_data = [

("ProductA", 50),

("ProductC", 150)

]

regional\_sales\_rdd = sc.parallelize(regional\_sales\_data)

combined\_rdd = sales\_rdd.union(regional\_sales\_rdd)

combined\_total\_sales\_rdd = combined\_rdd.reduceByKey(lambda a, b: a + b)

# Print the combined sales data

print("Combined Sales Data:")

print(combined\_total\_sales\_rdd.collect())

### \*\*Step 5: Perform Actions on the RDD\*\*

#8. \*\*Task 7: Count the Number of Distinct Products\*\*

distinct\_products\_count = sales\_rdd.map(lambda x: x[0]).distinct().count()

# Print the count of distinct products

print("Number of Distinct Products:", distinct\_products\_count)

#9. \*\*Task 8: Identify the Product with Maximum Sales\*\*

total\_sales\_rdd = sales\_rdd.reduceByKey(lambda x, y: x + y)

max\_sales\_product = total\_sales\_rdd.reduce(lambda a, b: a if a[1] > b[1] else b)

print(f"Product with maximum sales: {max\_sales\_product[0]} with sales amount: {max\_sales\_product[1]}")

### \*\*Challenge Task: Calculate the Average Sales per Product\*\*

#10. \*Challenge Task:\*\*

#Calculate the average sales amount per product using the key-value pair RDD.

average\_sales\_per\_product = total\_sales\_rdd.mapValues(lambda x: x / distinct\_products\_count).collect()

# Print the average sales per product

print("Average Sales per Product:")

print(average\_sales\_per\_product)