Hands on exercise on RDD:

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

#### \*\*Objective:\*\*

#In this exercise, you will work with key-value pair RDDs in PySpark. You will create RDDs, perform operations like grouping, aggregating, and sorting, and extract meaningful insights from the data.

### \*\*Dataset:\*\*

#You will be working with the following sales data. Each entry in the dataset represents a product and its corresponding sales amount.

sales\_data = [

("ProductA", 100),

("ProductB", 150),

("ProductA", 200),

("ProductC", 300),

("ProductB", 250),

("ProductC", 100)

]

#You will also be working with an additional dataset for regional sales:

regional\_sales\_data = [

("ProductA", 50),

("ProductC", 150)

]

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

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

# - Create a Spark session in PySpark and use the `spark.sparkContext` to create an RDD from the provided data.

from pyspark.sql import SparkSession

spark = SparkSession.builder \

.appName("Key-Value Pair RDDs") \

.getOrCreate()

sc = spark.sparkContext

print("SparkSession created successfully")

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

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

# - Create an RDD from the `sales\_data` list provided above.

# - Print the first few elements of the RDD.

sales\_data=[

("ProductA",100),

("ProductB", 150),

("ProductA", 200),

("ProductC", 300),

("ProductB", 250),

("ProductC", 100)

]

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

print(sales\_rdd.take(5))

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

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

# - Group the sales data by product name using `groupByKey()`.

# - Print the grouped data to understand its structure.

grouped\_data=sales\_rdd.groupByKey()

grouped\_sales=grouped\_data.mapValues(list)

print("Data by product name: ")

print(grouped\_sales.collect())

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

# - Use `reduceByKey()` to calculate the total sales for each product.

# - Print the total sales for each product.

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

print("Total sales by product: ")

print(total\_sales.collect())

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

# - Sort the products by their total sales in descending order.

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

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

print("Products by total sales after sorting: ")

print(sorted\_sales.collect())

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

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

# - Filter the products that have total sales greater than 200.

# - Print the products that meet this condition.

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

print("Products with high sales: ")

print(high\_sales.collect())

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

# - Create another RDD from the `regional\_sales\_data` list.

# - Combine this RDD with the original sales RDD using `union()`.

# - Calculate the new total sales for each product after combining the datasets.

# - Print the combined sales data.

regional\_sales\_data=[

("ProductA", 50),

("ProductB", 150)

]

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

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

new\_total\_sales=combined\_sales.reduceByKey(lambda x,y:x+y)

print("Combined sales data: ")

print(new\_total\_sales.collect())

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

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

# - Count the number of distinct products in the RDD.

# - Print the count of distinct products.

distinct\_products=sales\_rdd.keys().distinct().count()

print("No.of Distinct products: ")

print(distinct\_products)

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

# - Find the product with the maximum total sales using `reduce()`.

# - Print the product name and its total sales amount.

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

print("Product with maximum sales: ")

print(max\_sales\_product)

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

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

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

# - Print the average sales for each product.

product\_sales\_count=combined\_sales.mapValues(lambda x:(x,1))

product\_sales\_sum\_count=product\_sales\_count.reduceByKey(lambda x,y:(x[0]+y[0],x[1]+y[1]))

average\_sales\_rdd=product\_sales\_sum\_count.mapValues(lambda x: x[0] / x[1])

print("Average sales per product: ")

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