**PYSPARK EXERCISES 3 (3.9.2024)**

\*\*Working with Key-Value Pair RDDs in PySpark\*\*

\*Dataset\*

sales\_data = [

    ("ProductA", 100),

    ("ProductB", 150),

    ("ProductA", 200),

    ("ProductC", 300),

    ("ProductB", 250),

    ("ProductC", 100)

]

regional\_sales\_data = [

    ("ProductA", 50),

    ("ProductC", 150)

]

**#Initialize Sparksession**

spark = SparkSession.builder \

    .appName("RDD Key value pair") \

    .getOrCreate()

**#Get the SparkContext from the Sparksession**

sc= spark.sparkContext

print("Spark Session Created")

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

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

print(sales\_rdd.take(5))

**3.Grouping and Aggregating Data\*\***

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

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

print("Grouped Sales Data by Product Name:")

for product, sales in grouped\_sales\_rdd.collect():

    print(f"{product}: {sales}")

**4.Calculate Total Sales by Product\*\***

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

**# Print the total sales for each product**.

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

print("Total Sales by Product:")

for product, total\_sales in total\_sales\_rdd.collect():

    print(f"{product}: {total\_sales}")

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

print("Products Sorted by Total Sales (Descending):")

for product, total\_sales in sorted\_sales\_rdd.collect():

    print(f"{product}: {total\_sales}")

**6. Filter Products with High Sales\*\***

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

**#- Print the products that meet this condition.**

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

print("Products with Total Sales Greater Than 200:")

for product, total\_sales in high\_sales\_rdd.collect():

    print(f"{product}: {total\_sales}")

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

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

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

print("Combined Total Sales by Product:")

for product, total\_sales in combined\_total\_sales\_rdd.collect():

    print(f"{product}: {total\_sales}")

**8. Perform Actions on the RDD\*\***

**# Count the Number of Distinct Products\*\***

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

**# - Print the count of distinct products.**

distinct\_products\_count = combined\_sales\_rdd.keys().distinct().count()

print(f"Number of Distinct Products: {distinct\_products\_count}")

**9. 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.**

product\_with\_max\_sales = combined\_total\_sales\_rdd.reduce(

    lambda x, y: x if x[1] > y[1] else y

)

print(f"Product with Maximum Sales: {product\_with\_max\_sales[0]}")

print(f"Total Sales: {product\_with\_max\_sales[1]}")

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

**#- Print the average sales for each product.**

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

count\_rdd = combined\_sales\_rdd.mapValues(lambda x: 1).reduceByKey(lambda x, y: x + y)

average\_sales\_rdd = total\_sales\_rdd.join(count\_rdd).mapValues(lambda tc: tc[0] / tc[1])

print("Average Sales Amount Per Product:")

for product, average\_sales in average\_sales\_rdd.collect():

    print(f"{product}: {average\_sales:.2f}")