sales\_data = [

    ("ProductA", 100),

    ("ProductB", 150),

    ("ProductA", 200),

    ("ProductC", 300),

    ("ProductB", 250),

    ("ProductC", 100)

]

regional\_sales\_data = [

    ("ProductA", 50),

    ("ProductC", 150)

]

# 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.

rdd\_sales = sc.parallelize(sales\_data)

# print(rdd\_sales.collect())

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

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

#    - Print the grouped data to understand its structure.

group\_rdd = rdd\_sales.groupByKey()

print(group\_rdd.mapValues(list).collect())

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

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

#    - Print the total sales for each product.

reduce\_rdd = rdd\_sales.reduceByKey(lambda x,y: x+y)

print(reduce\_rdd.collect())

# 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.

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

print(sort\_rdd.collect())

# 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 = rdd\_sales.filter(lambda x: x[1] > 200)

print(high\_sales.collect())

# 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.

rdd\_regional = sc.parallelize(regional\_sales\_data)

combine\_rdd = rdd\_sales.union(rdd\_regional)

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

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

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

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

#    - Print the count of distinct products.

distinct\_count = rdd\_sales.map(lambda x: x[0]).distinct().count()

print(distinct\_count)

# 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 = reduce\_rdd.reduce(lambda x, y: x if x[1] > y[1] else y)

print(f"Product: {max\_sales\_product[0]}, Sales: {max\_sales\_product[1]}")

# Challenge Task:\*\*

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

#     - Print the average sales for each product.

def calculate\_average(data):

  total\_sum = sum(data)

  count = len(data)

  return total\_sum / count

average\_sales = rdd\_sales.groupByKey().mapValues(calculate\_average)

print(average\_sales.collect())