ICP 8 REPORT

```
[1]
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder.master("local").appName("RDD Operations").getOrCreate()

# Create an RDD with first 15 natural numbers
rdd = spark.sparkContext.parallelize(range(1, 16))
```

```
# Show the elements in the RDD
print(rdd.collect())

# Show the number of partitions
print(f"Number of partitions: {rdd.getNumPartitions()}")
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Number of partitions: 1
```

```
[3] # Return the first element
first_element = rdd.first()
print(f"First element: {first_element}")
```

```
First element: 1
```

```
[4] # Filter even elements from the RDD
even_rdd = rdd.filter(lambda x: x % 2 == 0)
print(even_rdd.collect())
```

```
[2, 4, 6, 8, 10, 12, 14]
```

```
[5] # Map transformation to square each element
squared_rdd = rdd.map(lambda x: x ** 2)
print(squared_rdd.collect())
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225]
```

```
# Aggregate elements using reduce (sum in this case)
sum_rdd = rdd.reduce(lambda x, y: x + y)
print(f"Sum of all elements: {sum_rdd}")
```

```
Sum of all elements: 120
```

```
[7] # Save the RDD as a text file (output folder must not already exist)
rdd.saveAsTextFile("output_rdd.txt")
```

```
[8] # Create two RDDs
rdd1 = spark.sparkContext.parallelize([1, 2, 3])
rdd2 = spark.sparkContext.parallelize([4, 5, 6])

# Combine them using union
combined_rdd = rdd1.union(rdd2)
print(combined_rdd.collect())
```

```
[1, 2, 3, 4, 5, 6]
```

```
# Create two RDDs
rdd1 = spark.sparkContext.parallelize([1, 2])
rdd2 = spark.sparkContext.parallelize([3, 4])

# Perform cartesian transformation
cartesian_rdd = rdd1.cartesian(rdd2)
print(cartesian_rdd.collect())
```

```
[(1, 3), (1, 4), (2, 3), (2, 4)]
```

```
[10] # Create an RDD with a dictionary
     dict_rdd = spark.sparkContext.parallelize([{'a': 1}, {'b': 2}, {'c': 3}])
     print(dict_rdd.collect())
```

```
[{'a': 1}, {'b': 2}, {'c': 3}]
```

```
# Create an RDD with duplicate values
rdd = spark.sparkContext.parallelize([1, 2, 2, 3, 3, 3, 4])

# Get the count of unique values using `map` and `reduceByKey`
count_rdd = rdd.map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)
print(count_rdd.collect())
```

```
[(1, 1), (2, 2), (3, 3), (4, 1)]
```

```
[12] # Create sample text files
     with open("file1.txt", "w") as file:
         file.write("This is line 1 of file 1\n")
         file.write("This is line 2 of file 1\n")

     with open("file2.txt", "w") as file:
         file.write("This is line 1 of file 2\n")
         file.write("This is line 2 of file 2\n")
```

0s completed at 2:25 PM

```
[13] # Read the text files into an RDD
     rdd_from_files = spark.sparkContext.textFile("file1.txt,file2.txt")
     print(rdd_from_files.collect())
```

```
['This is line 1 of file 1', 'This is line 2 of file 1', 'This is line 1 of file 2', 'This is line 2 of file 2']
```

```
# Show the first 5 elements in the RDD
first_5_lines = rdd_from_files.take(5)
print(first_5_lines)
```

```
['This is line 1 of file 1', 'This is line 2 of file 1', 'This is line 1 of file 2', 'This is line 2 of file 2']
```

```
[15] # Create a DataFrame from an RDD
     rdd = spark.sparkContext.parallelize([(1, "Alice"), (2, "Bob"), (3, "Charlie")])
     df = spark.createDataFrame(rdd, ["id", "name"])
     df.show()

     # Create a Dataset (Spark Datasets are available in Scala and Java, not in Python)
     # In PySpark, we use DataFrames directly, which are equivalent to Datasets in terms of API usage
```

```
+---+-------+
| id|   name|
+---+-------+
|  1|  Alice|
|  2|    Bob|
|  3|Charlie|
+---+-------+
```

```
[16] rdd = spark.sparkContext.parallelize([1, 2, 3])
     print(rdd.collect())  # No schema
```

```
[1, 2, 3]
```

```
[17] # Creating DataFrame
     data = [(1, "Alice"), (2, "Bob"), (3, "Charlie")]
     df = spark.createDataFrame(data, ["id", "name"])
     df.show()
```

```
+---+-------+
| id|   name|
+---+-------+
|  1|  Alice|
|  2|    Bob|
|  3|Charlie|
+---+-------+
```

0s completed at 3:25 PM

```
# In PySpark, DataFrame represents Dataset in terms of functionality
df = spark.createDataFrame([(1, "Alice"), (2, "Bob")], ["id", "name"])
df.show()
```

```
+---+-----+
| id| name|
+---+-----+
|  1|Alice|
|  2|  Bob|
+---+-----+
```

My Github Link : https://github.com/Nitish300903/bda.git