

ICP8 REPORT

+ Code + Text

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ] # Import required libraries
from pyspark import SparkContext
from pyspark.sql import SparkSession
```

```
[ ] # Initialize Spark Context
sc = SparkContext("local", "RDD Examples")
spark = SparkSession.builder.appName("DataFrame Examples").getOrCreate()
```

```
[ ] # 1. Produce RDD with List of first 15 natural numbers
rdd_list = sc.parallelize(range(1, 16))
print("RDD with first 15 natural numbers:", rdd_list.collect())
```

RDD with first 15 natural numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

+ Code + Text

```
[ ] # 2. Show the elements and number of partitions in RDD
print("Elements in RDD:", rdd_list.collect())
print("Number of partitions:", rdd_list.getNumPartitions())
```

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

```
[ ] # 3. Return the first element in the RDD
first_element = rdd_list.first()
print("First element:", first_element)
```

First element: 1

```
[ ] # 4. Use filter transformation to create a new RDD by selecting even elements
even_rdd = rdd_list.filter(lambda x: x % 2 == 0)
print("Even elements:", even_rdd.collect())
```

Even elements: [2, 4, 6, 8, 10, 12, 14]

+ Code + Text

```
[ ] # 5. Apply map transformation to square each element in the RDD
squared_rdd = rdd_list.map(lambda x: x ** 2)
print("Squared elements:", squared_rdd.collect())
```

↔ Squared elements: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225]

```
[ ] # 6. Aggregate all elements in the RDD using reduce action
sum_elements = rdd_list.reduce(lambda x, y: x + y)
print("Sum of elements in RDD:", sum_elements)
```

↔ Sum of elements in RDD: 120

```
[ ] # 7. Save the RDD data as a text file
rdd_list.saveAsTextFile("out_rdd_text")
```

```
[ ] # 8. Take two new list RDDs and combine them with union transformation
rdd_list2 = sc.parallelize(range(16, 21))
combined_rdd = rdd_list.union(rdd_list2)
print("Combined RDD:", combined_rdd.collect())
```

↔ Combined RDD: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

+ Code + Text

Reconnect

◆ Gemini

^

```
[ ] # 9. Use cartesian transformation to create a new list of ordered pairs
cartesian_rdd = rdd_list.cartesian(rdd_list2)
print("Cartesian product:", cartesian_rdd.collect())
```

↔ Cartesian product: [(1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (2, 16), (3, 16), (2, 17), (2, 18), (3, 17), (3, 18), (2, 19), (2, 20), (3, 19), (3, 20)]

```
[ ] # 10. Create an RDD with Dictionary
dict_rdd = sc.parallelize([{"name": "Niharika", "age": 22}, {"name": "Archana", "age": 21}, {"name": "Sahitha", "age": 20}])
print("RDD with dictionary:", dict_rdd.collect())
```

↔ RDD with dictionary: [{'name': 'Niharika', 'age': 22}, {'name': 'Archana', 'age': 21}, {'name': 'Sahitha', 'age': 20}]

```
[ ] # 11. Get unique value in the RDD as the key and its count as the value
rdd_flat = sc.parallelize(["apple", "banana", "apple", "orange", "banana", "apple"])
rdd_count = rdd_flat.map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)
print("Unique values and their counts:", rdd_count.collect())
```

↔ Unique values and their counts: [('apple', 3), ('banana', 2), ('orange', 1)]

+ Code + Text

Reconnect

◆ Gemini

^

```
[ ] # 12. Create RDD by combining multiple .text files
text_rdd = sc.textFile("/content/drive/My Drive/sample.txt")
```

```
[ ] # 13. Inspect the first 5 lines of an RDD
print("First 5 lines of text RDD:", text_rdd.take(5))
```

↔ First 5 lines of text RDD: ['Hello, this is line 1.', 'This is line 2 of the file.', 'Line 3 goes here.', 'More data in line 4.', 'And here is line 5']

```
[ ] # 14. Create DataFrame and Dataset
# Import the necessary class
from pyspark.sql import Row

# Creating DataFrame from RDD
data = [Row(name="Niharika", age=22), Row(name="Archana", age=21), Row(name="Sahitha", age=20)]
df = spark.createDataFrame(data)
print("DataFrame:")
df.show()

# Creating Dataset using toDF() function (PySpark doesn't support Dataset directly)
df_dataset = rdd_list.map(lambda x: Row(number=x)).toDF()
print("Dataset equivalent in PySpark (DataFrame of single column 'number'):")
df_dataset.show()
```

+ Code + Text

[] DataFrame:

↔ +-----+
| name|age|
+-----+
Niharika	22
Archana	21
Sahitha	20
+-----+

Dataset equivalent in PySpark (DataFrame of single column 'number'):

+-----+
| number|
+-----+
| 1|
| 2|
| 3|
| 4|
| 5|
| 6|
| 7|
| 8|
| 9|
| 10|
| 11|
| 12|
| 13|

+ Code + Text

Reconnect

[]
↔ +-----+
| 14|
| 15|
+-----+

```
[ ] # 15. Show difference between RDD, DataFrame, and Dataset
# RDD: Basic distributed data processing API, untyped, allows any type of data
print("RDD Example:", rdd_list.collect())

# DataFrame: Organized into named columns (structured data), similar to a table in SQL
print("DataFrame Example:")
df.show()

# Dataset: Only available in Scala and Java APIs in Spark, combines RDD and DataFrame features with compile-time safety
# In PySpark, DataFrames act as a replacement for Dataset
print("Dataset Example in PySpark is represented using DataFrame:")
df_dataset.show()
```

↔ RDD Example: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

DataFrame Example:

+-----+
| name|age|
+-----+
Niharika	22
Archana	21
Sahitha	20
+-----+

```
+ Code + Text

[ ] Dataset Example in PySpark is represented using DataFrame:
↕
+-----+
|number|
+-----+
|      1|
|      2|
|      3|
|      4|
|      5|
|      6|
|      7|
|      8|
|      9|
|     10|
|     11|
|     12|
|     13|
|     14|
|     15|
+-----+
```

My Github Repository Link:-

<https://github.com/niharika0912/BDA.git>