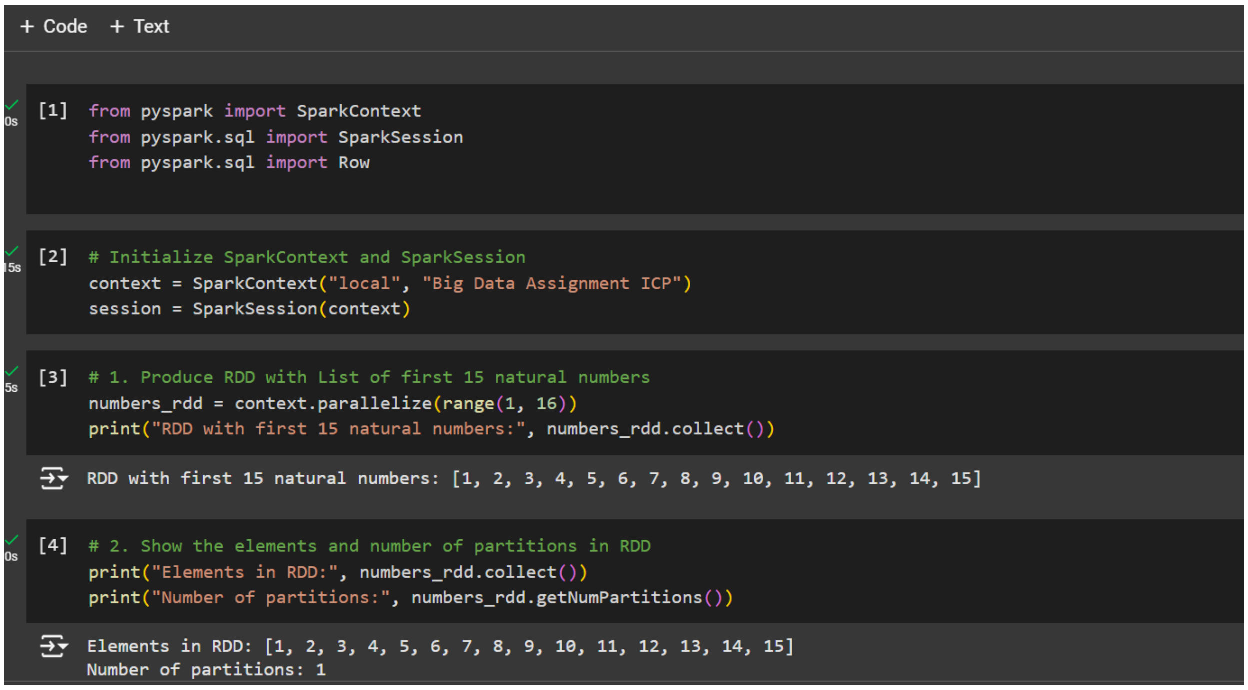
ICP8 REPORT



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

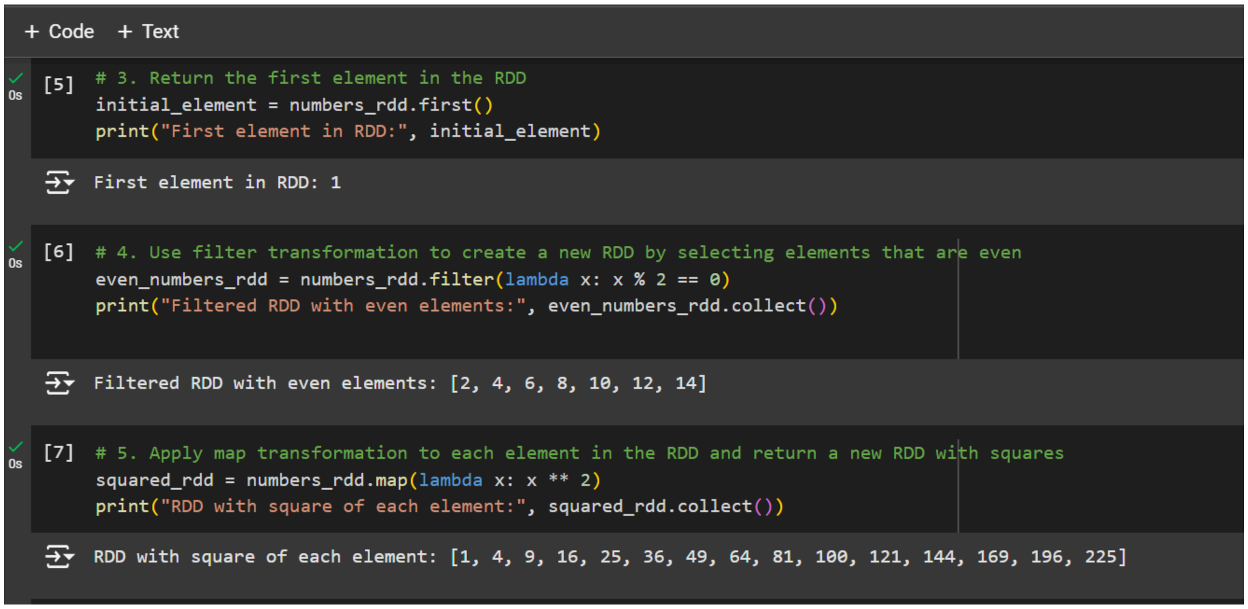
```
[2] # Initialize SparkContext and SparkSession
    context = SparkContext("local", "Big Data Assignment ICP")
    session = SparkSession(context)
```

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

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

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

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



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

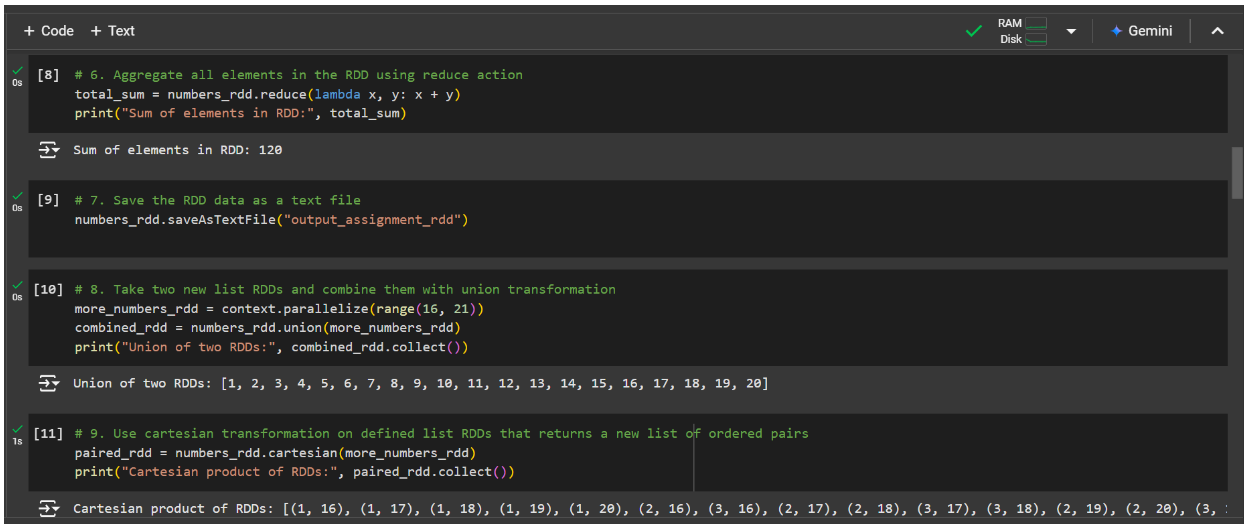
```
First element in RDD: 1
```

```
[6] # 4. Use filter transformation to create a new RDD by selecting elements that are even
    even_numbers_rdd = numbers_rdd.filter(lambda x: x % 2 == 0)
    print("Filtered RDD with even elements:", even_numbers_rdd.collect())
```

```
Filtered RDD with even elements: [2, 4, 6, 8, 10, 12, 14]
```

```
[7] # 5. Apply map transformation to each element in the RDD and return a new RDD with squares
    squared_rdd = numbers_rdd.map(lambda x: x ** 2)
    print("RDD with square of each element:", squared_rdd.collect())
```

```
RDD with square of each element: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225]
```



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

```
Sum of elements in RDD: 120
```

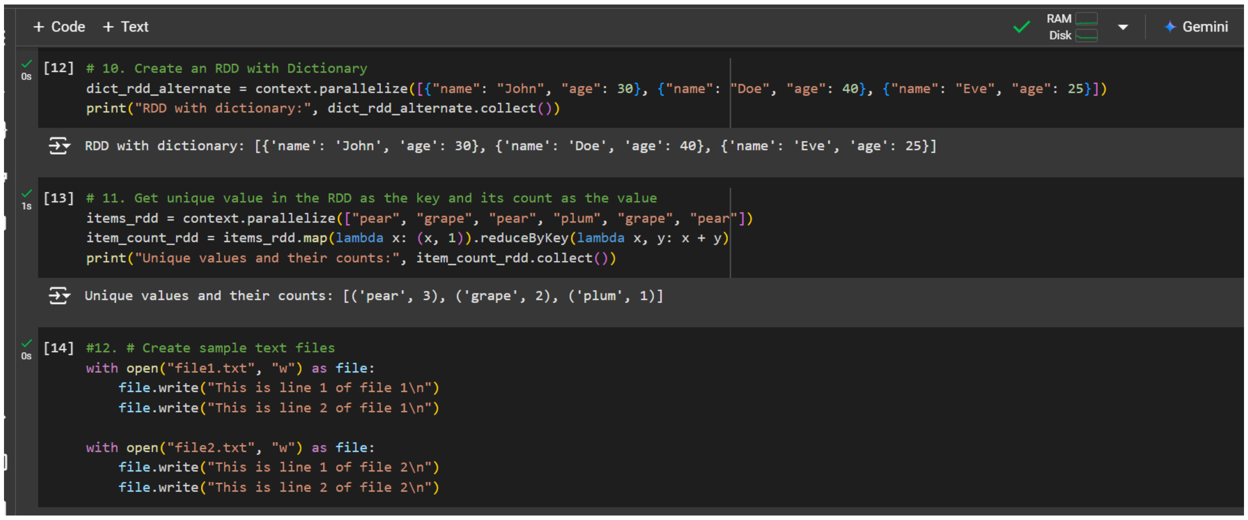
```
[9]  # 7. Save the RDD data as a text file
     numbers_rdd.saveAsTextFile("output_assignment_rdd")
```

```
[10] # 8. Take two new list RDDs and combine them with union transformation
     more_numbers_rdd = context.parallelize(range(16, 21))
     combined_rdd = numbers_rdd.union(more_numbers_rdd)
     print("Union of two RDDs:", combined_rdd.collect())
```

```
Union of two RDDs: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

```
[11] # 9. Use cartesian transformation on defined list RDDs that returns a new list of ordered pairs
     paired_rdd = numbers_rdd.cartesian(more_numbers_rdd)
     print("Cartesian product of RDDs:", paired_rdd.collect())
```

```
Cartesian product of RDDs: [(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,
```



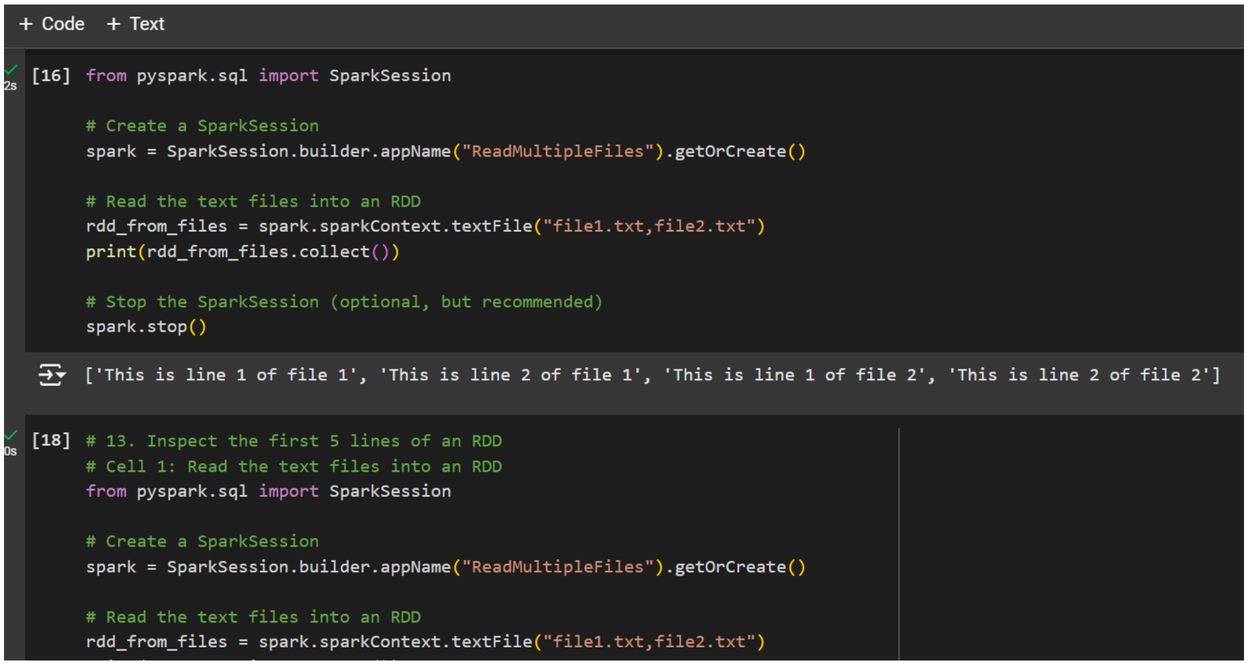
```
[12] # 10. Create an RDD with Dictionary
     dict_rdd_alternate = context.parallelize([{"name": "John", "age": 30}, {"name": "Doe", "age": 40}, {"name": "Eve", "age": 25}])
     print("RDD with dictionary:", dict_rdd_alternate.collect())
```

```
RDD with dictionary: [{'name': 'John', 'age': 30}, {'name': 'Doe', 'age': 40}, {'name': 'Eve', 'age': 25}]
```

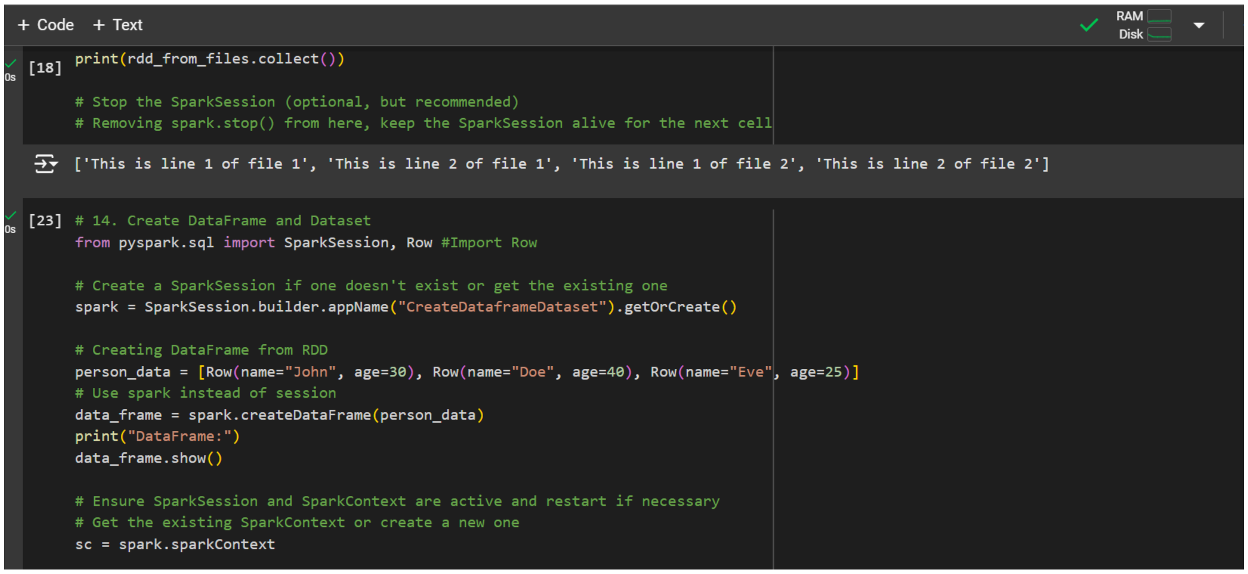
```
[13] # 11. Get unique value in the RDD as the key and its count as the value
     items_rdd = context.parallelize(["pear", "grape", "pear", "plum", "grape", "pear"])
     item_count_rdd = items_rdd.map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)
     print("Unique values and their counts:", item_count_rdd.collect())
```

```
Unique values and their counts: [('pear', 3), ('grape', 2), ('plum', 1)]
```

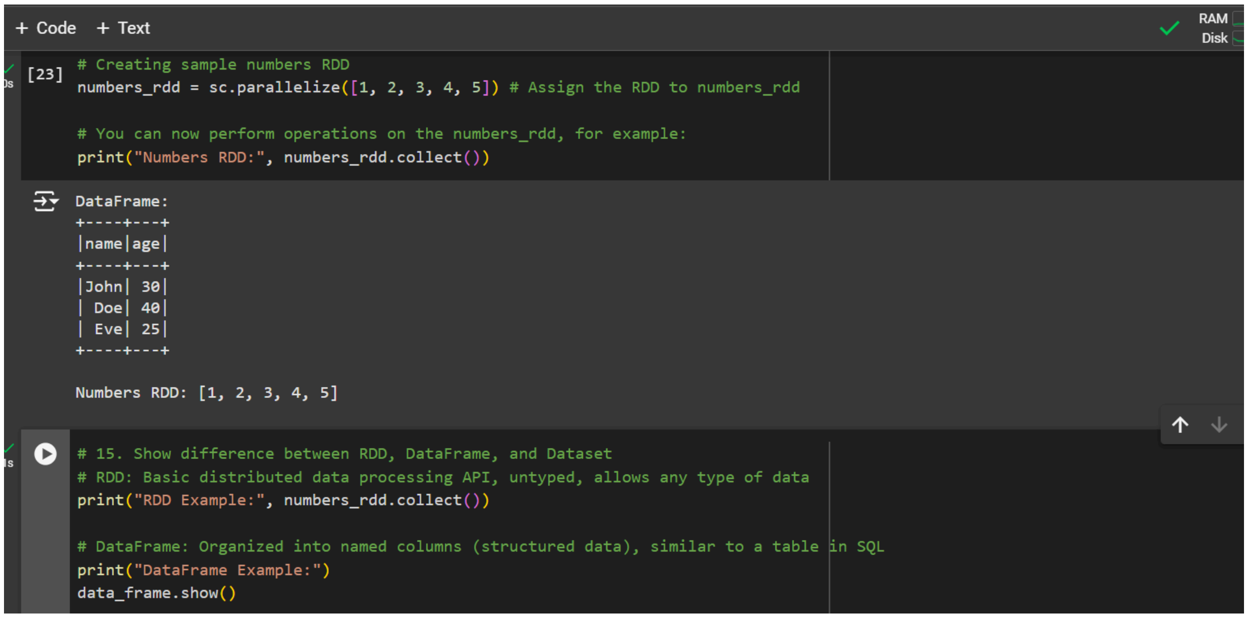
```
[14] #12. # Create sample text files
     with open("file1.txt", "w") as file:
         file.write("This is line 1 of file 1\n")
         file.write("This is line 2 of file 1\n")

     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")
```



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

     # Create a SparkSession
     spark = SparkSession.builder.appName("ReadMultipleFiles").getOrCreate()

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

     # Stop the SparkSession (optional, but recommended)
     spark.stop()

['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']

[18] # 13. Inspect the first 5 lines of an RDD
     # Cell 1: Read the text files into an RDD
     from pyspark.sql import SparkSession

     # Create a SparkSession
     spark = SparkSession.builder.appName("ReadMultipleFiles").getOrCreate()

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



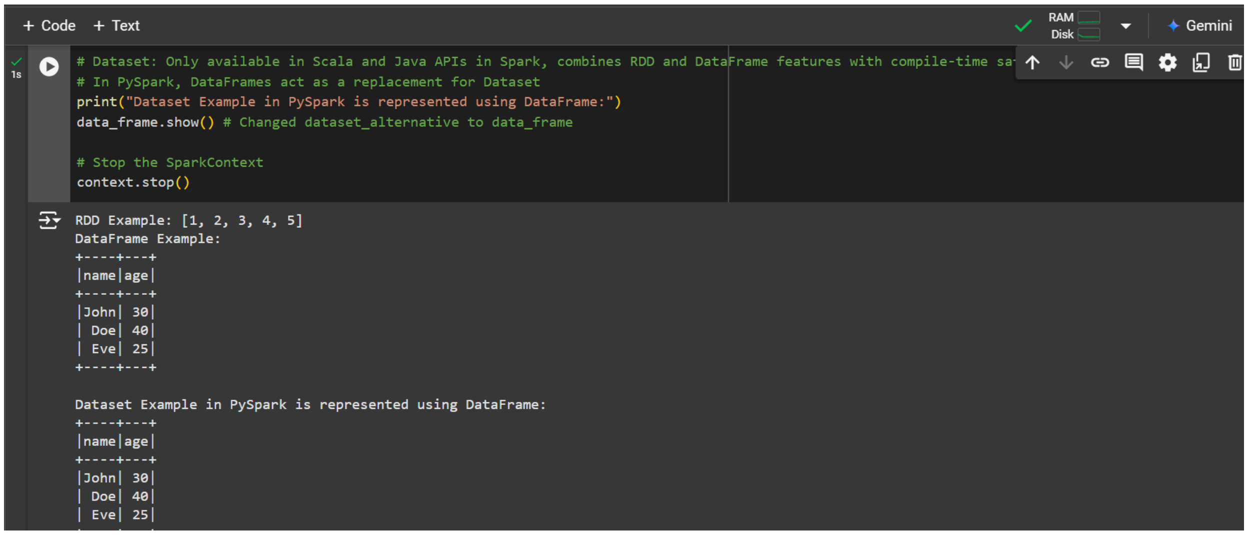
```
[18] print(rdd_from_files.collect())

     # Stop the SparkSession (optional, but recommended)
     # Removing spark.stop() from here, keep the SparkSession alive for the next cell

['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']

[23] # 14. Create DataFrame and Dataset
     from pyspark.sql import SparkSession, Row #Import Row

     # Create a SparkSession if one doesn't exist or get the existing one
     spark = SparkSession.builder.appName("CreateDataframeDataset").getOrCreate()

     # Creating DataFrame from RDD
     person_data = [Row(name="John", age=30), Row(name="Doe", age=40), Row(name="Eve", age=25)]
     # Use spark instead of session
     data_frame = spark.createDataFrame(person_data)
     print("DataFrame:")
     data_frame.show()

     # Ensure SparkSession and SparkContext are active and restart if necessary
     # Get the existing SparkContext or create a new one
     sc = spark.sparkContext
```



```
[23]
# Creating sample numbers RDD
numbers_rdd = sc.parallelize([1, 2, 3, 4, 5]) # Assign the RDD to numbers_rdd

# You can now perform operations on the numbers_rdd, for example:
print("Numbers RDD:", numbers_rdd.collect())
```

```
DataFrame:
+----+---+
|name|age|
+----+---+
|John| 30|
| Doe| 40|
| Eve| 25|
+----+---+

Numbers RDD: [1, 2, 3, 4, 5]
```

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

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



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

# Stop the SparkContext
context.stop()
```

```
RDD Example: [1, 2, 3, 4, 5]
DataFrame Example:
+----+---+
|name|age|
+----+---+
|John| 30|
| Doe| 40|
| Eve| 25|
+----+---+

Dataset Example in PySpark is represented using DataFrame:
+----+---+
|name|age|
+----+---+
|John| 30|
| Doe| 40|
| Eve| 25|
```

My Github Repository Link:-

https://github.com/akshaykumarpathem/bda.git