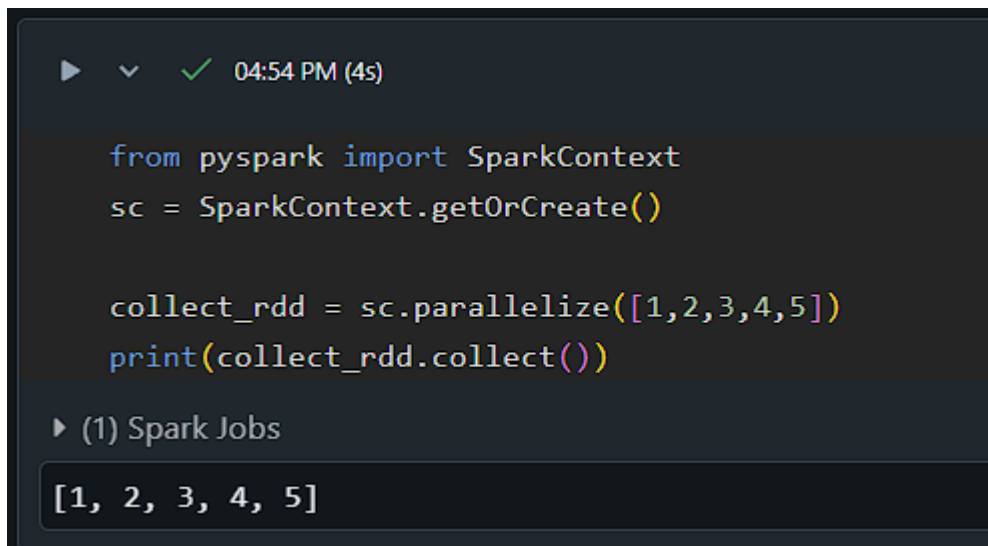


PySpark Hands-On Data-Engineering

1. The .collect() Action

The .collect() action on an RDD returns a list of all the elements of the RDD. It's a great asset for displaying all the contents of our RDD. Let's understand this with an example:

A screenshot of a PySpark console window. At the top, there is a play button icon, a green checkmark, and the text '04:54 PM (4s)'. Below this, the following code is entered:

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

collect_rdd = sc.parallelize([1,2,3,4,5])
print(collect_rdd.collect())
```

 Under the code, it says '▶ (1) Spark Jobs'. At the bottom, the output of the collect action is shown as a list:

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

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

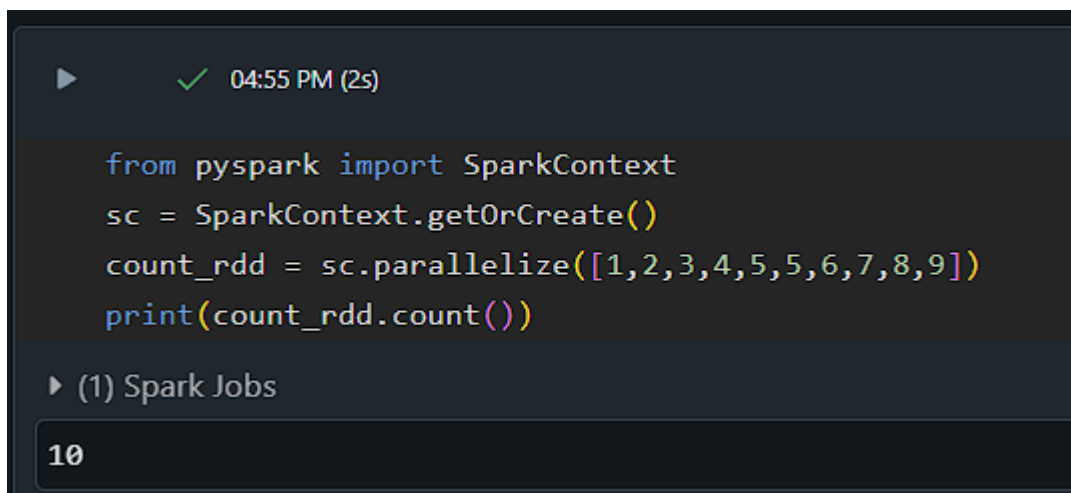
collect_rdd = sc.parallelize([1,2,3,4,5])
print(collect_rdd.collect())
```

▶ (1) Spark Jobs

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

2. The .count() Action

The .count() action on an RDD is an operation that returns the number of elements of our RDD. This helps in verifying if a correct number of elements are being added in an RDD. Let's understand this with an example:

A screenshot of a PySpark console window. At the top, there is a play button icon, a green checkmark, and the text '04:55 PM (2s)'. Below this, the following code is entered:

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

count_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(count_rdd.count())
```

 Under the code, it says '▶ (1) Spark Jobs'. At the bottom, the output of the count action is shown as the integer 10:

```
10
```

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

count_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(count_rdd.count())
```

▶ (1) Spark Jobs

```
10
```

3. The .first() Action

The .first() action on an RDD returns the first element from our RDD. This can be helpful when we want to verify if the exact kind of data has been loaded in

our RDD as per the requirements. For example, if wanted an RDD with the first 10 natural numbers. We can verify this by checking the first element from our RDD i.e. 1. Let's understand this with an example:

```
▶ ▼ ✓ 04:55 PM (1s)

from pyspark import SparkContext
sc = SparkContext.getOrCreate()
count_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(count_rdd.count())
first_rdd = sc.parallelize([1,2,3,4,5,6,7,8,9,10])
print(first_rdd.first())

▶ (2) Spark Jobs

10
1
```

4. The .take() Action

The .take(n) action on an RDD returns n number of elements from the RDD. The 'n' argument takes an integer which refers to the number of elements we want to extract from the RDD. Let's understand this with an example:

```
▶ ▼ ✓ 04:56 PM (1s)

take_rdd = sc.parallelize([1,2,3,4,5])
print(take_rdd.take(3))

▶ (2) Spark Jobs

[1, 2, 3]
```

5. The .reduce() Action

The .reduce() Action takes two elements from the given RDD and operates. This operation is performed using an anonymous function or lambda. For example, if we want to add all the elements from the given RDD, we can use the .reduce() action.

```
▶ ✓ 04:56 PM (1s) 5

from pyspark import SparkContext
sc = SparkContext.getOrCreate()
reduce_rdd = sc.parallelize([1,3,4,6])
print(reduce_rdd.reduce(lambda x, y : x + y))

▶ (1) Spark Jobs

14
```

6. The .saveAsTextFile() Action

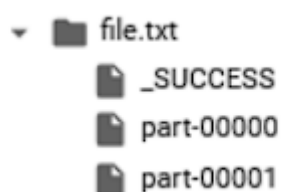
The .saveAsTextFile() Action is used to save the resultant RDD as a text file. We can also specify the path to which file needed to be saved. This helps in saving our results especially when we are working with a large amount of data.

```
▶ ✓ 04:57 PM (2s) 6

from pyspark import SparkContext
sc = SparkContext.getOrCreate()
save_rdd = sc.parallelize([1,2,3,4,5,6])
save_rdd.saveAsTextFile('file.txt')

▶ (1) Spark Jobs
```

On executing this code, we get:



7. The .map() Transformation

As the name suggests, the .map() transformation maps a value to the elements of an RDD. The .map() transformation takes in an anonymous function and applies this function to each of the elements in the RDD. For example, If we want to add 10 to each of the elements present in RDD, the .map() transformation would come in handy. This operation saves time and goes with the DRY policy. Let's understand this with an example:

```
▶ 04:58 PM (1s) 7

my_rdd = sc.parallelize([1,2,3,4])
print(my_rdd.map(lambda x: x+ 10).collect())

▶ (1) Spark Jobs

[11, 12, 13, 14]
```

8. The .filter() Transformation

A .filter() transformation is an operation in PySpark for filtering elements from a PySpark RDD. The .filter() transformation takes in an anonymous function with a condition. Again, since it's a transformation, it returns an RDD having elements that had passed the given condition. For example, we want to return only an even number of elements, we can use the .filter() transformation.

```
▶ 04:58 PM (<1s) 8

filter_rdd = sc.parallelize([2, 3, 4, 5, 6, 7])
print(filter_rdd.filter(lambda x: x%2 == 0).collect())

▶ (1) Spark Jobs

[2, 4, 6]
```

Here, we first created an RDD, filter_rdd using the .parallelize() method of SparkContext. Then we used the anonymous function lambda to filter the even numbers from our RDD filter_rdd. Since .filter() transformation returns a new RDD, we used the .collect() action to extract all the resultant elements in a list.

We can also filter strings from a certain text present in an RDD. For example, If we want to check the names of persons from a list of guests starting with a certain alphabet, we can use the .filter() for this operation as well. Let's understand this with an example:

```
▶ 04:59 PM (<1s) 9

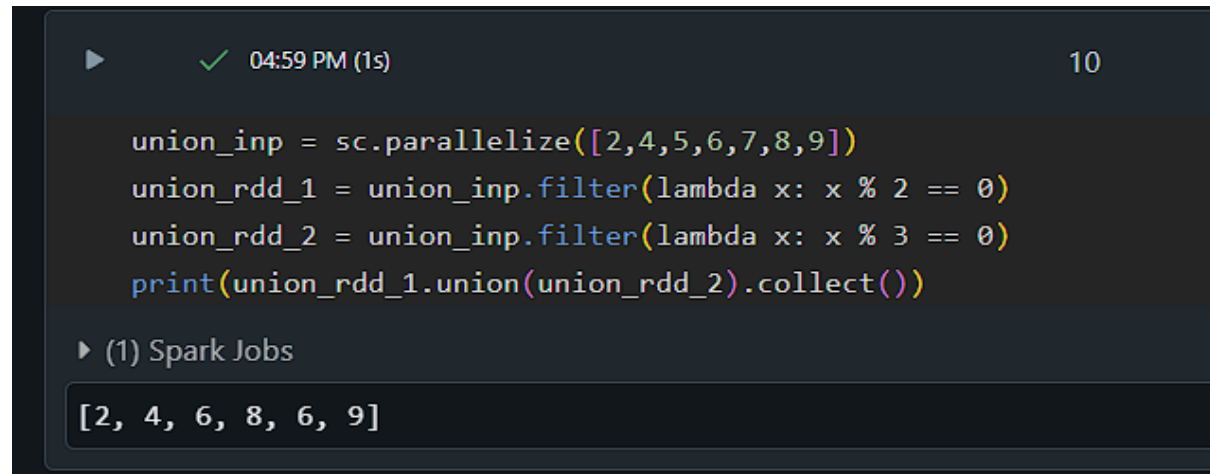
filter_rdd_2 = sc.parallelize(['Rahul', 'Swati', 'Rohan', 'Shreya', 'Priya'])
print(filter_rdd_2.filter(lambda x: x.startswith('R')).collect())

▶ (1) Spark Jobs

['Rahul', 'Rohan']
```

9. The .union() Transformation

The .union() transformation combines two RDDs and returns the union of the input two RDDs. This can be helpful to extract elements from similar characteristics from two RDDs into a single RDD. Let's understand this with an example:



```
04:59 PM (1s) 10

union_inp = sc.parallelize([2,4,5,6,7,8,9])
union_rdd_1 = union_inp.filter(lambda x: x % 2 == 0)
union_rdd_2 = union_inp.filter(lambda x: x % 3 == 0)
print(union_rdd_1.union(union_rdd_2).collect())

▶ (1) Spark Jobs

[2, 4, 6, 8, 6, 9]
```

10. The .flatMap() Transformation

The .flatMap() transformation performs same as the .map() transformation except the fact that .flatMap() transformation return separate values for each element from original RDD.

Let's understand this with an example:



```
05:00 PM (1s) 11

flatmap_rdd = sc.parallelize(["Hey there", "This is PySpark RDD Transformations"])
(flatmap_rdd.flatMap(lambda x: x.split(" ")).collect())

▶ (1) Spark Jobs

Out[11]: ['Hey', 'there', 'This', 'is', 'PySpark', 'RDD', 'Transformations']
```

11. PySpark Pair RDD Operations

PySpark has a dedicated set of operations for Pair RDDs. Pair RDDs are a special kind of data structure in PySpark in the form of key-value pairs, and that's how it got its name. Practically, the Pair RDDs are used more widely because of the reason that most of the real-world data is in the form of Key/Value pairs. The Pair RDDs use different terminology for key and value. The key is known as the identifier while the value is known as data.

Now, let's see how to create Pair RDDs in PySpark.

```
▶ 05:00 PM (<1s) 12

marks = [('Rahul', 88), ('Swati', 92), ('Shreya', 83), ('Abhay', 93), ('Rohan', 78)]
sc.parallelize(marks).collect()

▶ (1) Spark Jobs

Out[12]: [('Rahul', 88), ('Swati', 92), ('Shreya', 83), ('Abhay', 93), ('Rohan', 78)]
```

12. The .reduceByKey() Transformation

The .reduceByKey() transformation performs multiple parallel processes for each key in the data and combines the values for the same keys. It uses an anonymous function or lambda to perform the task. Since it's a transformation, it returns an RDD as a result.

```
▶ 05:01 PM (2s) 13

marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29), ('Rohan', 22),
                           ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
print(marks_rdd.reduceByKey(lambda x, y: x + y).collect())

▶ (1) Spark Jobs

[('Shreya', 50), ('Swati', 45), ('Rahul', 48), ('Abhay', 55), ('Rohan', 44)]
```

13. The .sortByKey() Transformation

The .sortByKey() transformation sorts the input data by keys from key-value pairs either in ascending or descending order. It returns a unique RDD as a result.

```
▶ 05:01 PM (1s) 14

marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29), ('Rohan', 22),
                           ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
print(marks_rdd.sortByKey('ascending').collect())

▶ (3) Spark Jobs

[('Abhay', 29), ('Abhay', 26), ('Rahul', 25), ('Rahul', 23), ('Rohan', 22), ('Rohan', 22), ('Shreya', 22), ('Shreya', 28), ('Swati', 26), ('Swati', 19)]
```

14. The .groupByKey() Transformation

The .groupByKey() transformation groups all the values in the given data with the same key together. It returns a new RDD as a result. For example, if we want to extract all the Cultural Members from a list of committee members, the .groupByKey() will come in handy to perform the necessary task.

```
▶ 05:01 PM (1s) 15

marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29), ('Rohan', 22),
('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
dict_rdd = marks_rdd.groupByKey().collect()
for key, value in dict_rdd:
    print(key, list(value))

▶ (1) Spark Jobs

Shreya [22, 28]
Swati [26, 19]
Rahul [25, 23]
Abhay [29, 26]
Rohan [22, 22]
```

15. The countByKey() Action

The `.countByKey()` option is used to count the number of values for each key in the given data. This action returns a dictionary and one can extract the keys and values by iterating over the extracted dictionary using loops. Since we are getting a dictionary as a result, we can also use the dictionary methods such as `.keys()`, `.values()` and `.items()`.

```
▶ 05:02 PM (1s) 16 Python

marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Rohan', 22), ('Rahul', 23), ('Swati', 19),
('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
dict_rdd = marks_rdd.countByKey().items()
for key, value in dict_rdd:
    print(key, value)

▶ (1) Spark Jobs

Rahul 2
Swati 2
Rohan 2
Shreya 1
Abhay 1
```