

DAY-13 (Spark Assignment - 3)

Mitushi Vishwakarma

RDDs

Actions and transformations in PySpark

There are two types of operations that can be performed on RDDs, they are Actions and Transformations. We will initialize a SparkContext to perform the operations. Only one SparkContext can run in the environment.

Initializing SparkContext

```
Cmd 2

1  from pyspark import SparkContext
2  sc = SparkContext.getOrCreate()

Command took 0.06 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:48:34 AM on Test
```

1. Actions :

They are a kind of operation which are applied on an RDD to produce a single value. These methods are applied on a resultant RDD and produces a non-RDD value

.collect() : returns a list of all the elements of the RDD

```
Cmd 3

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

▶ (1) Spark Jobs

[1, 2, 3, 4, 5]

Command took 0.46 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:48:49 AM on Test
```

.count() : returns the number of elements of our RDD

```
Cmd 4

1  count_rdd = sc.parallelize([1,2,3,4,5])
2  print(count_rdd.count())

▶ (1) Spark Jobs

5

Command took 0.93 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:48:59 AM on Test
```

.first() : returns the first element from our RDD

```
Cmd 5

1  first_rdd = sc.parallelize([1,2,3,4,5])
2  print(first_rdd.first())

▶ (2) Spark Jobs

1

Command took 0.84 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:49:17 AM on Test
```

.take(n) : returns n number of elements from the RDD

```
Cmd 6

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

▶ (2) Spark Jobs

[1, 2, 3]

Command took 0.61 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:49:25 AM on Test
```

.reduce() : takes two elements from the given RDD and operates. This operation is performed using an anonymous function or lambda.

```
Cmd 7

1  reduce_rdd = sc.parallelize([1,2,3,4,5])
2  print(reduce_rdd.reduce(lambda x,y:x+y))

▶ (1) Spark Jobs

15

Command took 0.82 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:54:14 AM on Test
```

.saveAsTextFile() : is used to save the resultant RDD as a text file

```
Cmd 8

1  save_rdd = sc.parallelize([1,2,3,4,5])
2  print(save_rdd.saveAsTextFile('file.txt'))

▶ (1) Spark Jobs

None

Command took 3.87 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 11:56:56 AM on Test

Cmd 9

1  rdd2 = spark.sparkContext.textFile('file.txt')
2  rdd2.collect()

▶ (1) Spark Jobs

Out[1]: ['1', '2', '3', '4', '5']

Command took 7.83 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 2:46:02 PM on Test
```

2. Transformations : Transformations are the kind of operations that are performed on an RDD and return a new RDD

.map() : The **.map()** transformation takes in an anonymous function and applies this function to each of the elements in the RDD.

```
Cmd 10

1  map_rdd = sc.parallelize([1,2,3,4,5])
2  print(map_rdd.map(lambda x: x+10).collect())

▶ (1) Spark Jobs

[11, 12, 13, 14, 15]

Command took 0.74 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 2:57:33 PM on Test
```

.filter() : for filtering elements from a PySpark RDD. Takes in an anonymous function with a condition

```
Cmd 11

1  filter_rdd = sc.parallelize([1,2,3,4,5,6,7,8,9,10])
2  print(filter_rdd.filter(lambda x : x%2==0).collect())

▶ (1) Spark Jobs

[2, 4, 6, 8, 10]

Command took 1.06 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 3:20:54 PM on Test

Cmd 12

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

▶ (1) Spark Jobs

['Rahul', 'Rohan']

Command took 0.80 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 3:23:21 PM on Test
```

.union() : combines two RDDs and returns the union of the input two RDDs

```
Cmd 13

1  rdd1 = sc.parallelize([1,2,3])
2  rdd2 = sc.parallelize([3,4,5,6])
3  print(rdd1.union(rdd2).collect())

▶ (1) Spark Jobs

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

Command took 0.33 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 3:49:52 PM on Test

Cmd 14
```

DataFrames in PySpark : PySpark DataFrames are distributed collections of data organized into named columns. You can create PySpark DataFrames from various data sources such as CSV, JSON, Parquet, or even existing RDDs.

```
Cmd 1

1  import pyspark
2  from pyspark.sql import SparkSession
3
4  # Create a spark session
5  spark = SparkSession.builder.appName('DataFrame_operations').getOrCreate()
6
7  # Create data in dataframe
8  data = [ (('Ram'), '1991-04-01', 'M', 3000),
9           (('Mike'), '2000-05-19', 'M', 4000),
10          (('Rohini'), '1978-09-05', 'M', 4000),
11          (('Maria'), '1967-12-01', 'F', 4000),
12          (('Jenis'), '1980-02-17', 'F', 1200)]
13
14  # Column names in dataframe
15  columns = ["Name", "DOB", "Gender", "salary"]
16
17  # Create the spark dataframe
18  df = spark.createDataFrame(data=data , schema=columns)
19
20  # Print the dataframe
21  df.show()
```

```
▶ (3) Spark Jobs

▶ df: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]

+-----+-----+-----+-----+
|  Name|      DOB|Gender|salary|
+-----+-----+-----+-----+
|   Ram|1991-04-01|    M|  3000|
|  Mike|2000-05-19|    M|  4000|
|Rohini|1978-09-05|    M|  4000|
| Maria|1967-12-01|    F|  4000|
|  Jenis|1980-02-17|    F|  1200|
+-----+-----+-----+-----+

Command took 4.51 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 3:29:55 PM on Test
```

Renaming Column in dataframe :

There are various methods to rename data in dataframe.

1. **Using withColumnRenamed(existing,new):** Here withColumnRenamed() method takes two argument, existingstr and newstr. Returns dataframe with new column name

```
Cmd 2

1 df_columnRename = df.withColumnRenamed('DOB','DateOfBirth')
2 df_columnRename.printSchema()
3 df_columnRename.show()

▶ (3) Spark Jobs

▶ df_columnRename: pyspark.sql.dataframe.DataFrame = [Name: string, DateOfBirth: string ... 2 more fields]

root
 |-- Name: string (nullable = true)
 |-- DateOfBirth: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- salary: long (nullable = true)

+-----+-----+-----+-----+
|  Name|DateOfBirth|Gender|salary|
+-----+-----+-----+-----+
|   Ram| 1991-04-01|    M|  3000|
|  Mike| 2000-05-19|    M|  4000|
|Rohini| 1978-09-05|    M|  4000|
| Maria| 1967-12-01|    F|  4000|
|  Jenis| 1980-02-17|    F|  1200|
+-----+-----+-----+-----+

Command took 0.74 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:19:05 PM on Test
```

Multiple column renaming using withColumnRenamed :

```
Cmd 3

1 df.withColumnRenamed('DOB','DateOfBirth').withColumnRenamed('salary','Salary').show()

▶ (3) Spark Jobs

+-----+-----+-----+-----+
| Name|DateOfBirth|Gender|Salary|
+-----+-----+-----+-----+
| Ram|1991-04-01| M| 3000|
| Mike|2000-05-19| M| 4000|
| Rohini|1978-09-05| M| 4000|
| Maria|1967-12-01| F| 4000|
| Jenis|1980-02-17| F| 1200|
+-----+-----+-----+-----+

Command took 0.94 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:04:05 PM on Test
```

2. Using select(cols) : cols: List of column names as strings. And new column name can be given using .alias(newstr) with that particular col name.

```
Cmd 4

1 from pyspark.sql.functions import col
2 data = df.select(col("Name"),col("DOB"),col("Gender"),col("salary").alias('Payment'))
3 data.show()

▶ (3) Spark Jobs

▶ data: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]

+-----+-----+-----+-----+
| Name|DOB|Gender|Payment|
+-----+-----+-----+-----+
| Ram|1991-04-01| M| 3000|
| Mike|2000-05-19| M| 4000|
| Rohini|1978-09-05| M| 4000|
| Maria|1967-12-01| F| 4000|
| Jenis|1980-02-17| F| 1200|
+-----+-----+-----+-----+

Command took 0.69 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:21:52 PM on Test
```

4 . Using selectExpr () : similar to SQL aliasing. Writing the new column name as expression.

```
Cmd 5

1 df_selectExpr = df.selectExpr("Name as names","DOB",'Gender','salary')
2 df_selectExpr.show()

▶ (3) Spark Jobs
▶ df_selectExpr: pyspark.sql.dataframe.DataFrame = [names: string, DOB: string ... 2 more fields]

+-----+-----+-----+-----+
| names|      DOB|Gender|salary|
+-----+-----+-----+
|  Ram|1991-04-01|  M|  3000|
|  Mike|2000-05-19|  M|  4000|
|Rohini|1978-09-05|  M|  4000|
| Maria|1967-12-01|  F|  4000|
|  Jenis|1980-02-17|  F|  1200|
+-----+-----+-----+

Command took 0.86 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:18:05 PM on Test
```

5. Using toDF(): This function returns a new DataFrame that with new specified column names

```
Cmd 6

1 datalist = ['First Name','DateOfBirth','M/F','Payment']
2 toDF = df.toDF(*datalist)
3 toDF.show()

▶ (3) Spark Jobs
▶ toDF: pyspark.sql.dataframe.DataFrame = [First Name: string, DateOfBirth: string ... 2 more fields]

+-----+-----+-----+-----+
|First Name|DateOfBirth|M/F|Payment|
+-----+-----+-----+
|    Ram| 1991-04-01|  M|  3000|
|    Mike| 2000-05-19|  M|  4000|
|  Rohini| 1978-09-05|  M|  4000|
|   Maria| 1967-12-01|  F|  4000|
|   Jenis| 1980-02-17|  F|  1200|
+-----+-----+-----+

Command took 0.93 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:25:59 PM on Test
```


Notes :

Day - 3

→ RDDs and Transformations

RDD is a core data structure of pyspark.

Transformations :-

- takes RDD as input.
- returns new RDD.
- original RDD remains same.
- DAG is created after transformation.

Action :-

- applied on RDD to produce single value (non-RDD)
- removing laziness of transformation.

initializing SparkContext

SC = SparkContext.getOrCreate()

Actions

→ collect() Action

Returns a list of all elements of RDD.

→ count()

returns the no. elements of RDD.

21-12-2023

21

2023
THURSDAY
WK 51 • DAY 355-010
DECEMBER

2024
JANUARY
M T W T F S S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

→ `first()`

- returns first element
- helpful in verifying

→ `take(n)` first

- returns n no. of elements
- n is no. of elements we want

→ `reduce()`

- take two elements and operates
- operation lambda

→ `saveAsTextFile()`

- to save RDD as text file.

Transformations

→ `map()`

- maps a value to elements of RDD.
- similar to map in python
- it returns a new RDD so we use `collect()` to extract all elements.

→ `filter()`

- filtering the elements
- takes anonymous func. with a condition

52	25	26	27	28	29	30	31
51	18	19	20	21	22	23	24
50	11	12	13	14	15	16	17
49	4	5	6	7	8	9	10
48	1	2	3				
WK	M	T	W	T	F	S	S
2023	DECEMBER						

2023
WEDNESDAY
WK 51 • DAY 354-011
DECEMBER

20-12-2023

20

→ .union()

→ combines two RDDs and

→ return union

→ ~~.format~~

Dataframe in spark

create dataframe :-

→ create spark session.

→ create data

→ column names

→ spark dataframe.

.createDataFrame(data = ...,
schema = ...)

→ .withColumnRenamed(existing, new)

→ to change column name

→ returns renamed column in df

→ df is immutable so original remains same

→ using select Expr(expr)

→ expr is SQL expression

→ using ~~toDF()~~ .toDF(*col)

Dataframes in pyspark can be created from
csv, json, RDD, SQL, python lists/dicts

19-12-2023

19

DECEMBER
WK 51 • DAY 35-012
TUESDAY
2023

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				
S	S	F	T	W	T	S