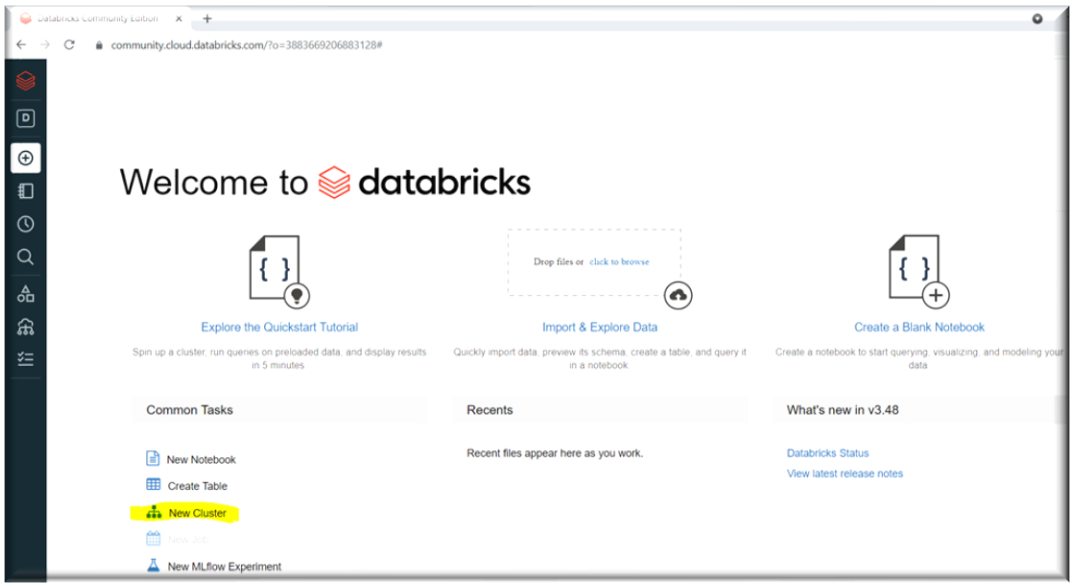
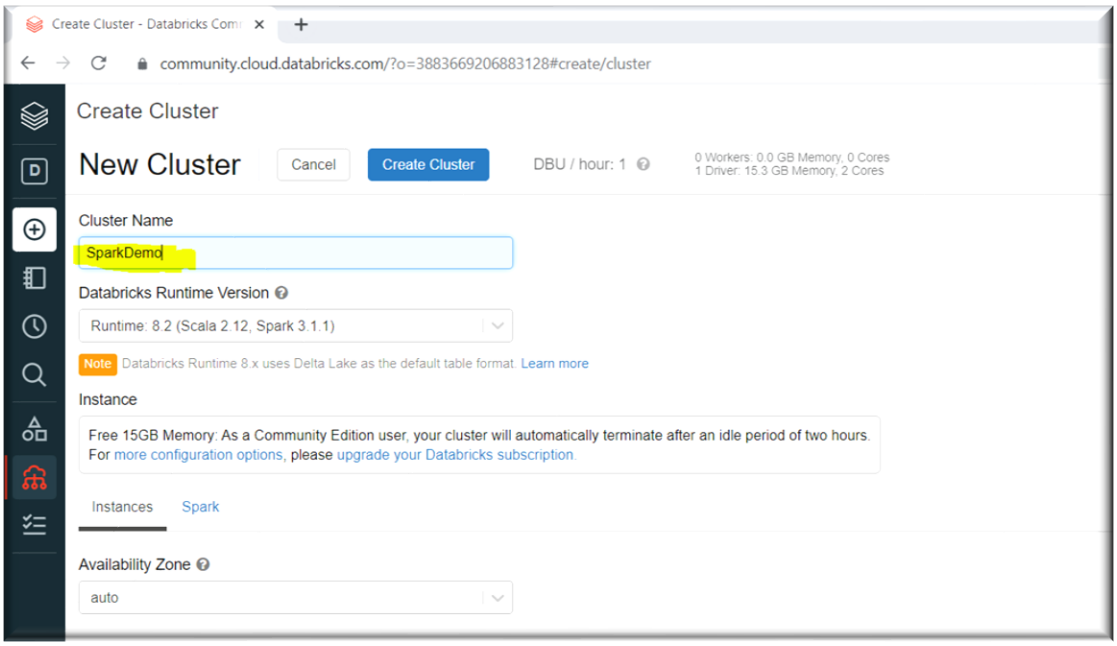
Spark RDD-Activity 1

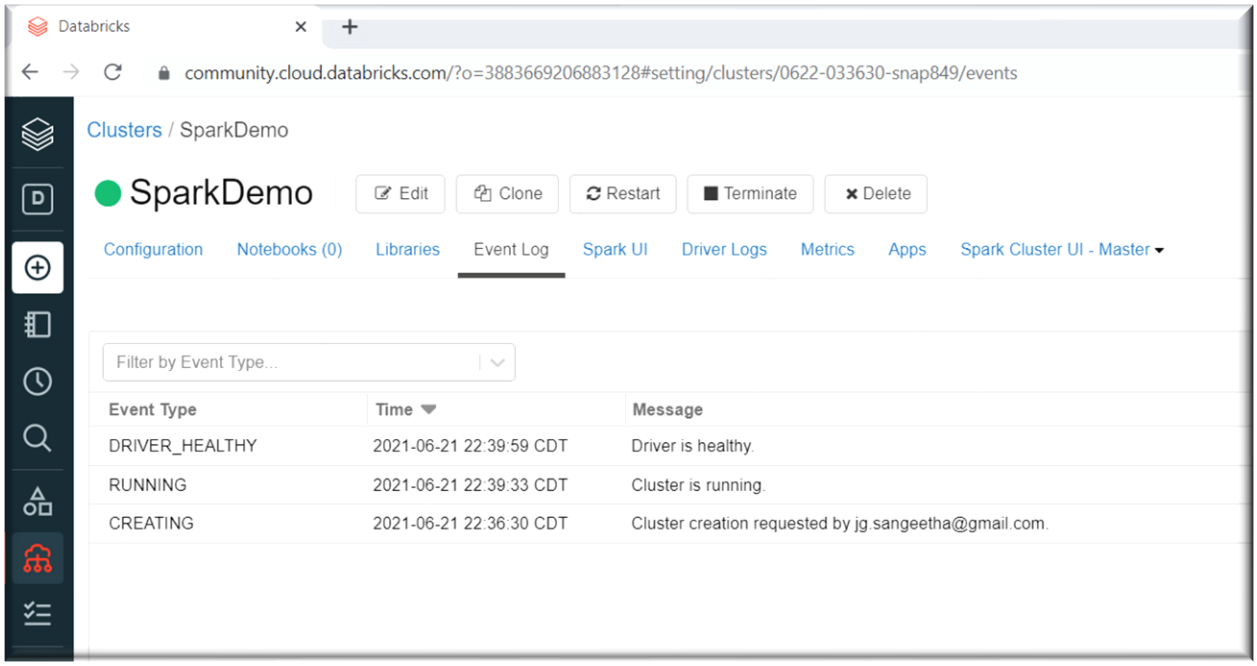
1)Click **New cluster** under Common Tasks



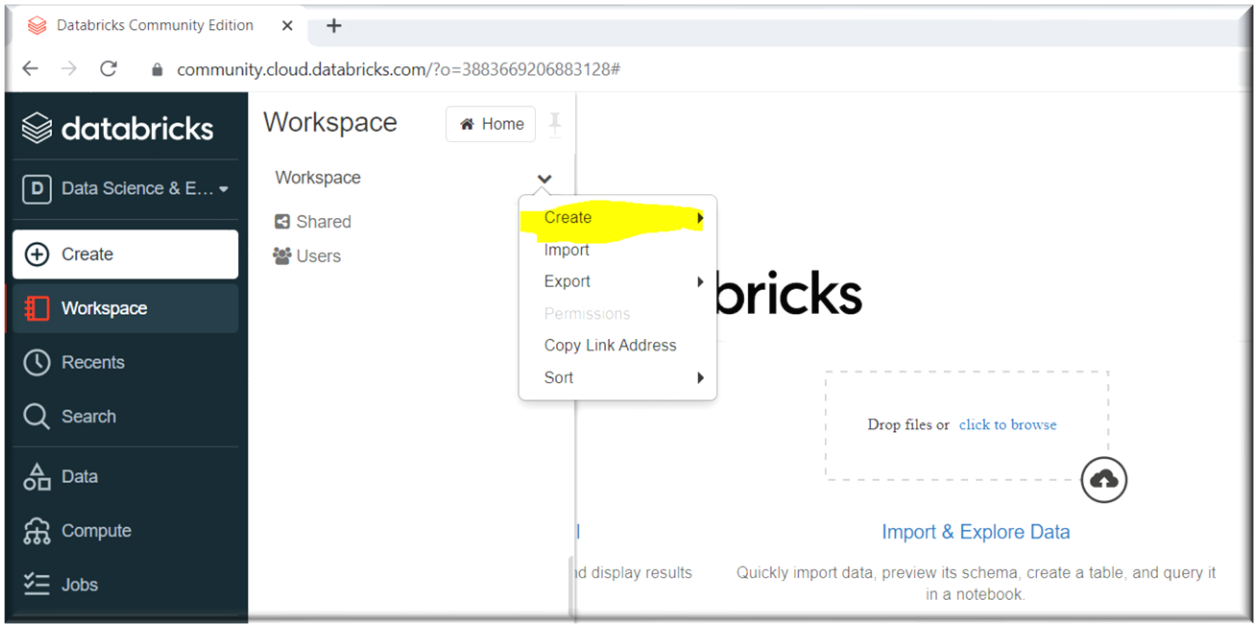
2)Enter the name of your cluster, select DataBricks run time version and click create cluster



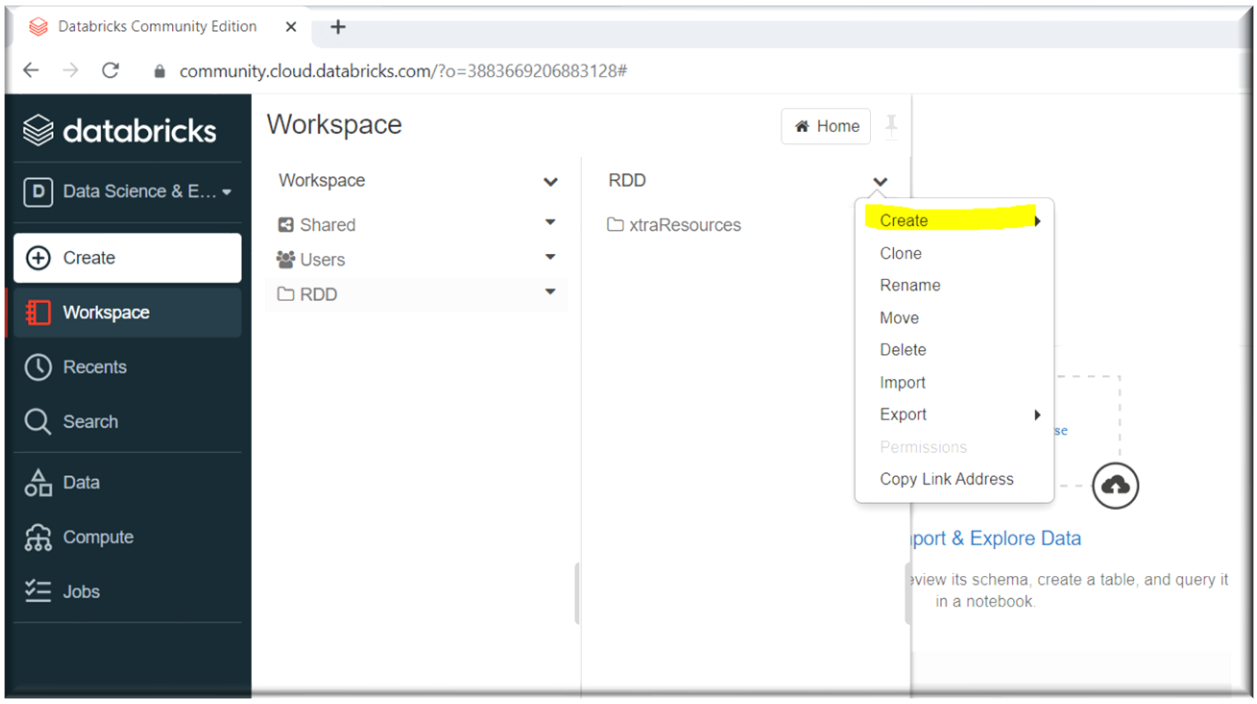
3)Click Event log and verify the cluster status

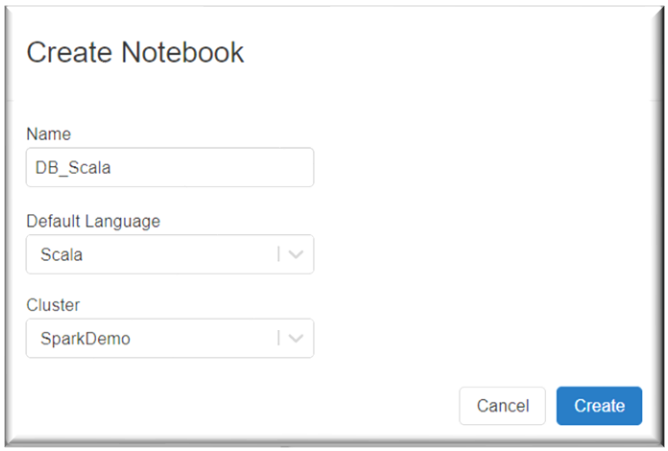


4)In your workspace, create new folder named RDD

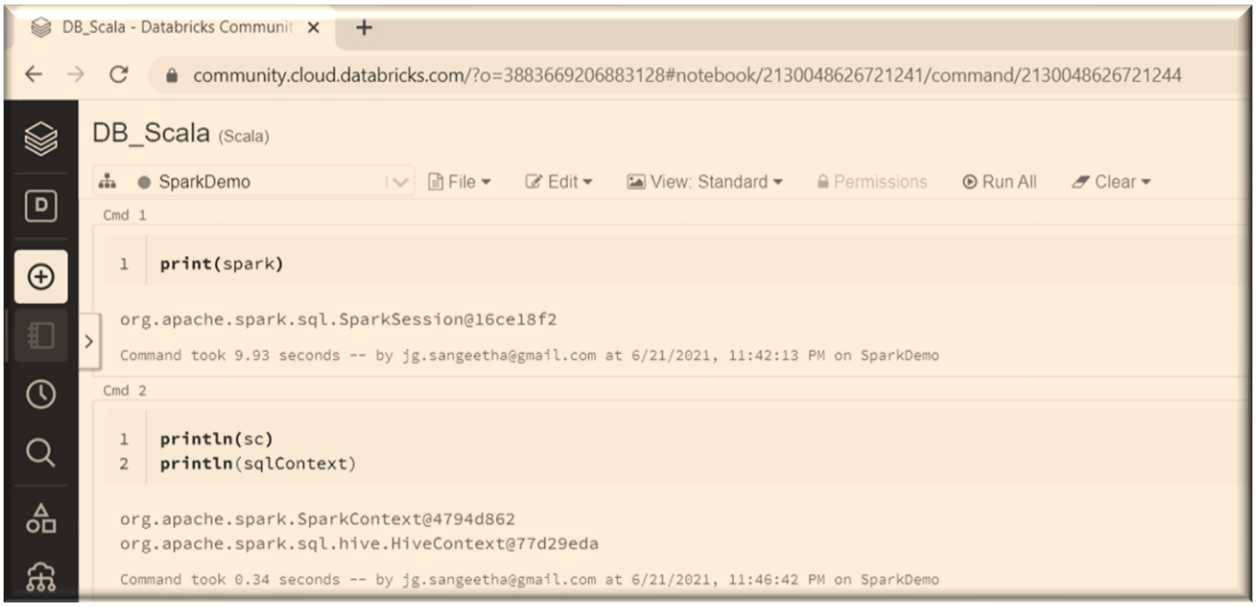


5) Create scala or python notebook in RDD folder.





6) Verify app Name, Spark session, Spark context, SQL context



7) Create an RDD using parallelize method available in spark context

```
val myData:List[String]=List("Alice","Carlos","Frank","Barbara")
val myRdd=sc.parallelize(myData)
myRdd.take(2)
```

(2) Spark Jobs

```
myData: List[String] = List(Alice, Carlos, Frank, Barbara)
myRdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[48] at parallelize at command-2130048626721281:3
res35: Array[String] = Array(Alice, Carlos)
```

Command took 0.43 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:49:01 AM on SparkDemo

Creating RDD of three elements from a Scala sequence with 2 partitions by using parallelize method

Cmd 3

```
/*Creating RDD of three elements from a scala sequence with 2 partitions by using parallelize method*/
val x=sc.parallelize(Array(4,5,6),2)
```

```
x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at command-2130048626721244:2
```

Command took 1.29 seconds -- by jg.sangeetha@gmail.com at 6/21/2021, 11:51:26 PM on SparkDemo

8) Perform collect action in RDD and finding the number of partitions using getNumPartitions action

Collect () -It returns all items in the RDD to the driver in a single list.

Glom ()-Return an RDD created by coalescing all elements within each partition into a list.

Cmd 4

```
/*Collect action in RDD
Glom()-Return an RDD created by coalescing all elements within each partition into a list*/
val x=sc.parallelize(Array(4,5,6),2)
val y=x.collect()
val xout=x.glom().collect()
println(y)
```

(1) Spark Jobs

```
[I@6856f1fa
x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at command-2130048626721246:3
y: Array[Int] = Array(4, 5, 6)
xout: Array[Array[Int]] = Array(Array(4), Array(5, 6))
```

Command took 2.51 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:02:18 AM on SparkDemo

Cmd 5

```
/*Finding number of partitions in RDD x */
x.getNumPartitions
```

```
res5: Int = 2
```

Command took 0.44 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:08:21 AM on SparkDemo

```
Cmd 1

x = sc.parallelize([4,5,6], 2)
y=x.collect()
print(x.glom().collect())
print(y)

▸ (2) Spark Jobs

[[4], [5, 6]]
[4, 5, 6]

Command took 4.01 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 11:32:01 PM on My Cluster
```

9) Perform **"take"** action in RDD

take(n)-It returns an array with first n elements of RDD.

```
Cmd 6

/*Performing take action in RDD*/
x.take(2)

▸ (2) Spark Jobs

res7: Array[Int] = Array(4, 5)

Command took 0.46 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:13:37 AM on SparkDemo
```

10) Transform RDD by map to make another RDD

Map transformation returns a new RDD that is formed by passing each element of the source RDD through a function.

```
Cmd 7

/* Tranform RDD using map function*/
val x=sc.parallelize(Array("m","n","o"))//RDD x:[m,n,o]
val y=x.map(z=>(z,1))//map x into RDD y:[(m,1),(n,1),(o,1)]

x: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[3] at parallelize at command-2130048626721248:2
y: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[4] at map at command-2130048626721248:3

Command took 0.88 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:30:05 AM on SparkDemo
```

```
Cmd 8

1  println(x.collect().mkString(","))
2  println(y.collect().mkString(","))

▸ (2) Spark Jobs

m,n,o
(m,1),(n,1),(o,1)

Command took 0.70 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:30:47 AM on SparkDemo
```

```
1  x=sc.parallelize(["m","n","o"])
2  y=x.map(lambda z:(z,1))
3  print(x.collect())
4  print(y.collect())

▸ (2) Spark Jobs

['m', 'n', 'o']
[('m', 1), ('n', 1), ('o', 1)]

Command took 0.70 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 11:35:05 PM on My Cluster
```

11) Transform RDD by filter to make another RDD

Filter transformation returns a new RDD that is formed by selecting elements for which the function returns true.

```
Cmd 9

1  val x=sc.parallelize(Array(3,4,5))
2  val y=x.filter(n=>n%2==1)
3

x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[5] at parallelize at command-2130048626721250:1
y: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[6] at filter at command-2130048626721250:2

Command took 1.11 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:40:50 AM on SparkDemo
```

```
Cmd 10

1  println(x.collect().mkString(","))
2  println(y.collect().mkString(","))

▸ (2) Spark Jobs

3,4,5
3,5

Command took 0.60 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:42:16 AM on SparkDemo
```

```
1  x=sc.parallelize([3,4,5])
2  y=x.filter(lambda x:x%2==1)
3  print(x.collect())
4  print(y.collect())

▸ (2) Spark Jobs

[3, 4, 5]
[3, 5]

Command took 0.57 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 11:37:40 PM on My Cluster
```

12) Perform reduce action in RDD

Reduce aggregates all elements of RDD by applying user function pairwise to elements and partial results and returns result to driver.

```
Cmd 11

1  val x=sc.parallelize(Array(1,2,3,4))
2  val y=x.reduce((a,b)=>a+b)

▸ (1) Spark Jobs

x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[7] at parallelize at command-2130048626721252:1
y: Int = 10

Command took 0.76 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:46:14 AM on SparkDemo
```

```
Cmd 12

1  println(x.collect.mkString(","))
2  println(y)

▸ (1) Spark Jobs

1,2,3,4
10

Command took 0.48 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:47:30 AM on SparkDemo
```

```
1  x=sc.parallelize([1,2,3,4])
2  y=x.reduce(lambda a,b:a+b)
3  print(x.collect())
4  print(y)

▸ (2) Spark Jobs

[1, 2, 3, 4]
10

Command took 0.54 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 11:39:41 PM on My Cluster
```

13) Transform RDD by flat map to make another RDD

Return a new RDD by first applying function to all elements of this RDD and then flattening the result.

```
Cmd 13

1  val x=sc.parallelize(Array(1,2,3))
2  val y=x.flatMap(n=>Array(n,n*100,50))

x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at command-2130048626721254:1
y: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[9] at flatMap at command-2130048626721254:2

Command took 0.64 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:52:52 AM on SparkDemo
```

```
Cmd 14

1  println(x.collect().mkString(", "))
2  println(y.collect().mkString(", "))

▸ (1) Spark Jobs

1, 2, 3
1, 100, 50, 2, 200, 50, 3, 300, 50

Command took 0.63 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:53:35 AM on SparkDemo
```

```
1  x=sc.parallelize([1,2,3])
2  y=x.flatMap(lambda x:(x,x*100,50))
3  print(x.collect())
4  print(y.collect())

▸ (2) Spark Jobs

[1, 2, 3]
[1, 100, 50, 2, 200, 50, 3, 300, 50]

Command took 0.47 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 11:42:09 PM on My Cluster
```

14) Create a pair RDD

```
Cmd 15

1  val words = sc.parallelize(Array("a", "b", "a", "a", "b", "b", "a", "a", "a", "b", "b"))
2  words.collect()

▸ (1) Spark Jobs

words: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[10] at parallelize at command-2130048626721256:1
res14: Array[String] = Array(a, b, a, a, b, b, a, a, a, b, b)

Command took 0.64 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:54:50 AM on SparkDemo
```

```
Cmd 16

1  val wordCountPairRDD = words.map(s => (s, 1))
2  wordCountPairRDD.collect()

▸ (1) Spark Jobs

wordCountPairRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[11] at map at command-2130048626721257:1
res15: Array[(String, Int)] = Array((a,1), (b,1), (a,1), (a,1), (b,1), (b,1), (a,1), (a,1), (a,1), (b,1), (b,1))

Command took 0.51 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:55:39 AM on SparkDemo
```

15)Perform transformations in pair RDD

a. Reduce by key

It takes an RDD and returns new RDD of key value pairs. The values of each key are aggregated using the reduced function.

Cmd 17

```
/*Includes intermediate RDD*/
val wordcounts = wordCountPairRDD.reduceByKey( (value1, value2) => value1 + value2 )
wordcounts.collect()
```

(1) Spark Jobs

```
wordcounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[12] at reduceByKey at command-2130048626721258:2
res16: Array[(String, Int)] = Array((a,6), (b,5))
```

Command took 1.56 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:58:08 AM on SparkDemo

Cmd 18

```
/*Avoid collecting intermediate RDD*/
val words = sc.parallelize(Array("a", "b", "a", "a", "b", "b", "a", "a", "a", "b", "b"))
val wordcounts = words
                    .map(s => (s, 1))
                    .reduceByKey(_ + _)
                    .collect()
```

(1) Spark Jobs

```
words: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[13] at parallelize at command-2130048626721259:2
wordcounts: Array[(String, Int)] = Array((a,6), (b,5))
```

Command took 0.92 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 12:59:09 AM on SparkDemo

b. Sort by key

This returns a new RDD of key-value pairs that's sorted by keys in ascending order.

```
/*Sort by key*/
val words = sc.parallelize(Array("a", "b", "a", "a", "b", "b", "a", "a", "a", "b", "b"))
val wordCountPairRDD = words.map(s => (s, 1))
val wordCountPairRDDSortedByKey = wordCountPairRDD.sortByKey()
```

(1) Spark Jobs

```
words: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[16] at parallelize at command-2130048626721260:1
wordCountPairRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[17] at map at command-2130048626721260:2
wordCountPairRDDSortedByKey: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[20] at sortByKey at command-2130048626721260:3
```

Command took 0.82 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:00:22 AM on SparkDemo

Cmd 20

```
wordCountPairRDD.collect()
```

(1) Spark Jobs

```
res18: Array[(String, Int)] = Array((a,1), (b,1), (a,1), (a,1), (b,1), (b,1), (a,1), (a,1), (a,1), (b,1), (b,1))
```

Command took 0.82 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:09:16 AM on SparkDemo

```
Cmd 21
1  wordCountPairRDDSortedByKey.collect()

▸ (1) Spark Jobs
res19: Array[(String, Int)] = Array((a,1), (a,1), (a,1), (a,1), (a,1), (a,1), (b,1), (b,1), (b,1), (b,1), (b,1))
Command took 0.37 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:10:00 AM on SparkDemo
```

c. Group by key

This returns a new RDD consisting of key and iterable-valued pairs.

```
Cmd 22
1  val wordCountPairRDDGroupByKey = wordCountPairRDD.groupByKey().collect()

▸ (1) Spark Jobs
wordCountPairRDDGroupByKey: Array[(String, Iterable[Int])] = Array((a,CompactBuffer(1, 1, 1, 1, 1, 1)), (b,CompactBuffer(1, 1, 1, 1, 1)))
Command took 0.64 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:11:36 AM on SparkDemo
```

16) Simple computation

```
1  val list = 1 to 10
2  var sum = 0
3  list.map(x => sum = sum + x)
4  print(sum)

55list: scala.collection.immutable.Range.Inclusive = Range 1 to 10
sum: Int = 55
Command took 0.50 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:12:55 AM on SparkDemo
```

```
md 24
1  val rdd = sc.parallelize(1 to 10)
2  var sum = 0

rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[23] at parallelize at command-2130048626721265:1
sum: Int = 0
Command took 0.34 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:13:12 AM on SparkDemo
md 25
1  val rdd1 = rdd.map(x => sum = sum + x).collect()

▸ (1) Spark Jobs
rdd1: Array[Unit] = Array((), (), (), (), (), (), (), (), (), ())
Command took 0.68 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:13:31 AM on SparkDemo
```

Cmd 26

```
val rdd1 = rdd.map(x =>
                   {var sum = 0;
                         sum = sum + x
                         sum}
                  ).collect()
```

(1) Spark Jobs

```
rdd1: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Command took 0.51 seconds -- by jg.sangeetha@gmail.com at 6/22/2021, 1:14:17 AM on SparkDemo