```
List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

res1: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
// sum() :

//> It adds up the value in an RDD.
//> It is an package org.apache.spark.rdd.DoubleRDDFunctions.
//> Its return type is Double
val rdd1 = sc.parallelize(1 to 10)
rdd1.sum

rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[4] at parallelize at command-2760144754248951:6
res5: Double = 55.0
```

```
//max() :

//> It returns a max value from RDD element defined by implicit ordering (element order)
//> It is an package org.apache.spark.rdd

val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
rdd1.max

rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[6] at parallelize at command-2760144754248952:6
res6: Int = 10
```



```
//min() :

//> It returns a min value from RDD element defined by implicit ordering (element order)
//> It is an package org.apache.spark.rdd

val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
rdd1.min

rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[7] at parallelize at command-2760144754248954:6
res7: Int = 1
```

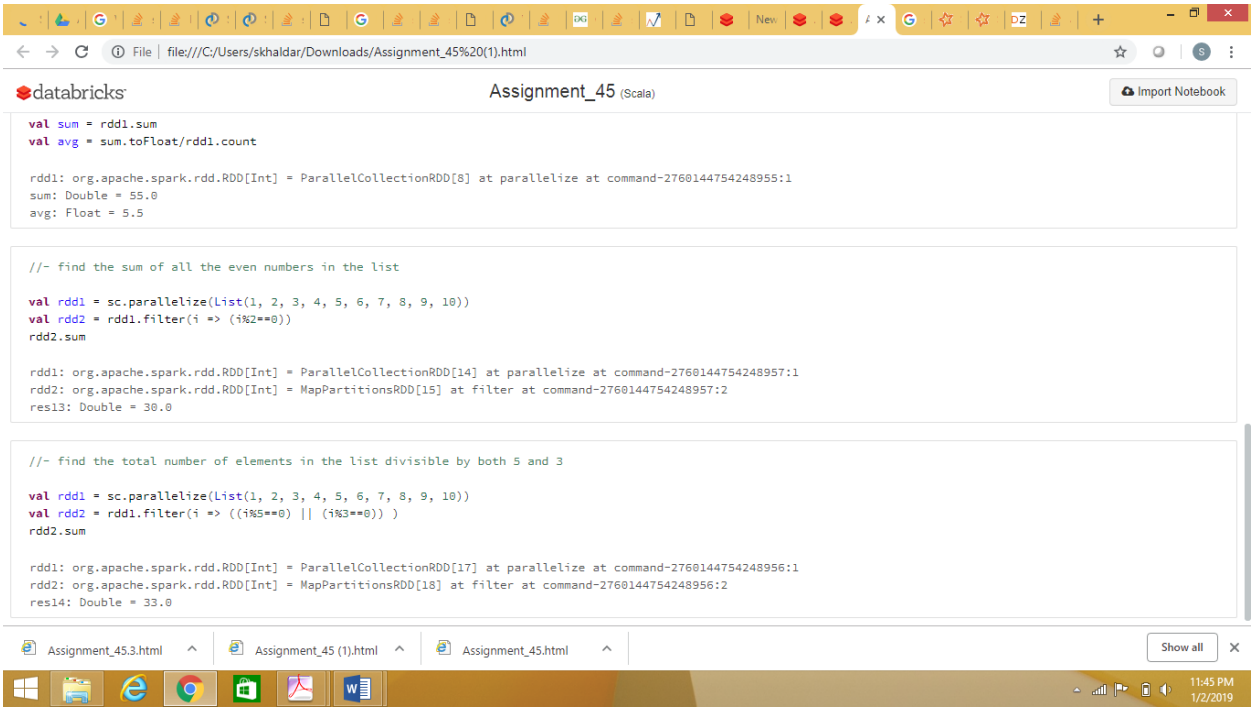
```
//- calculate the average of the numbers in the list

val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
val sum = rdd1.sum
val avg = sum.toFloat/rdd1.count

rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at command-2760144754248955:1
sum: Double = 55.0
avg: Float = 5.5
```

```
//- find the sum of all the even numbers in the list

val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
val rdd2 = rdd1.filter(i => (i%2==0))
rdd2.sum
```


File | file:///C:/Users/skhaldar/Downloads/Assignment_45%20(1).html
databricks
Assignment_45 (Scala)
Import Notebook
val sum = rdd1.sum
val avg = sum.toFloat/rdd1.count
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at command-2760144754248955:1
sum: Double = 55.0
avg: Float = 5.5
//- find the sum of all the even numbers in the list
val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
val rdd2 = rdd1.filter(i => (i%2==0))
rdd2.sum
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[14] at parallelize at command-2760144754248957:1
rdd2: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[15] at filter at command-2760144754248957:2
res13: Double = 30.0
//- find the total number of elements in the list divisible by both 5 and 3
val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
val rdd2 = rdd1.filter(i => ((i%5==0) || (i%3==0)))
rdd2.sum
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[17] at parallelize at command-2760144754248956:1
rdd2: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[18] at filter at command-2760144754248956:2
res14: Double = 33.0
Assignment_45.3.html
Assignment_45 (1).html
Assignment_45.html
Show all
11:45 PM
1/2/2019