Name:-Pooja Srinivasan

USN:-1BM18CS069

BDA Assignment-4

1) map()

```
lab@lab-VirtualBox: ~
Spark context available as 'sc' (master = local[*], app id = local-162161333714
2).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 11.0.11)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val input=sc.parallelize(List(1,2,3,4))
input: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize
at <console>:24

scala> val result=input.map(x=>x*x)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at map at <console>
:25

scala> println(result.collect().mkString(","))
[Stage 0:>                                                          (0 + 0) / 1
[Stage 0:>                                                          (0 + 1) / 1

1,4,9,16

scala>
```

2) flatmap()

```
scala> val lines=sc.parallelize(List("hello world","hi"))
lines: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[36] at parallel
ize at <console>:24

scala> val words=lines.flatMap(line=>line.split(" "))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[37] at flatMap at <c
onsole>:25

scala> words.first()
res11: String = hello
```

3) filter()

```
scala> val input=sc.parallelize(List(1,2,3,4))
input: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize
at <console>:24

scala> val result=input.filter(x=>x!=1)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[9] at filter at <conso
le>:25

scala> println(result.collect().mkString(","))
2,3,4
```

4) union()

```
scala> val input4=sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[14] at paralleliz
e at <console>:24

scala> val input5=sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[15] at paralleliz
e at <console>:24

scala> val result=input4.union(input5)
result: org.apache.spark.rdd.RDD[Int] = UnionRDD[16] at union at <console>:27

scala> println(result.collect().mkString(","))
1,2,3,3,4,5
```

5) intersection()

```
scala> val input4=sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[17] at parallel
e at <console>:24

scala> val input5=sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[18] at parallel
e at <console>:24

scala> val result=input4.intersection(input5)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[24] at intersection
 <console>:27

scala> println(result.collect().mkString(","))
3
```

6) distinct()

```
scala> val input4=sc.parallelize(List(1,2,2,3,3,4))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[10] at paralleliz
e at <console>:24

scala> val result=input4.distinct()
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[13] at distinct at <co
nsole>:25

scala> println(result.collect().mkString(","))
[Stage 5:>                                                          (0 + 1) / 1
[Stage 6:>                                                          (0 + 1) / 1

4,1,3,2
```

7) subtract()

```
scala> val input4=sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[25] at paralleliz
e at <console>:24

scala> val input5=sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[26] at paralleliz
e at <console>:24

scala> val result=input4.subtract(input5)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[30] at subtract at <co
nsole>:27

scala> println(result.collect().mkString(","))
1,2
```

8) cartisian()

```
scala> val input4=sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[31] at paralleliz
e at <console>:24

scala> val input5=sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[32] at paralleliz
e at <console>:24

scala> val result=input4.cartesian(input5)
result: org.apache.spark.rdd.RDD[(Int, Int)] = CartesianRDD[33] at cartesian at
 <console>:27

scala> println(result.collect().mkString(","))
(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5)
```