UNIVERSITÉ SIDI MOHAMED BEN ABDLLAH
FACULTÉ DES SCIENCES DHAR EL MAHRAZ DE FÈS
DÉPARTEMENT D'INFORMATIQUE

Département : Informatique

Master : Informatique Décisionnelle et Vision Intelligent

Titre :

TD / TP 1 With SPARK

Présenter par :
ABIBOU SOUKAYNA

Supervisé par :
Pr. Noura

Année universitaire : 2020/2021

Filière MIDVI

Exercice 1 :

- Méthode 1 :

1- Code :

```
public class WordCount
{
      public static void main(String[] args)
      {
                        SparkConf conf = new SparkConf();
                conf.setMaster("local").setAppName("WordCount");
               JavaSparkContext jsc = new JavaSparkContext(conf);
    JavaRDD<String> file=jsc.textFile("C:\\Users\\Lenovo\\Desktop\\MASTER-
              MIDVI\\S2\\Big Data\\Programme\\Word-Count\\input");
                file.foreach(line -> System.out.println(line));
   JavaRDD<String> words = file.flatMap(line -> Arrays.asList(line.split("
                                  ")).iterator());
                file.foreach(line -> System.out.println(line));
         JavaPairRDD<String,Integer> value=words.mapToPair(word -> new
                                Tuple2<>(word,1));
                value.foreach(line -> System.out.println(line));
  JavaPairRDD<String,Integer> wordCount=value.reduceByKey((first,second) ->
                                 first+second);
    wordCount.foreach(pair -> System.out.println(pair._1 + " "+ pair._2));
      }
}
```

2- Résultat d’exécution :

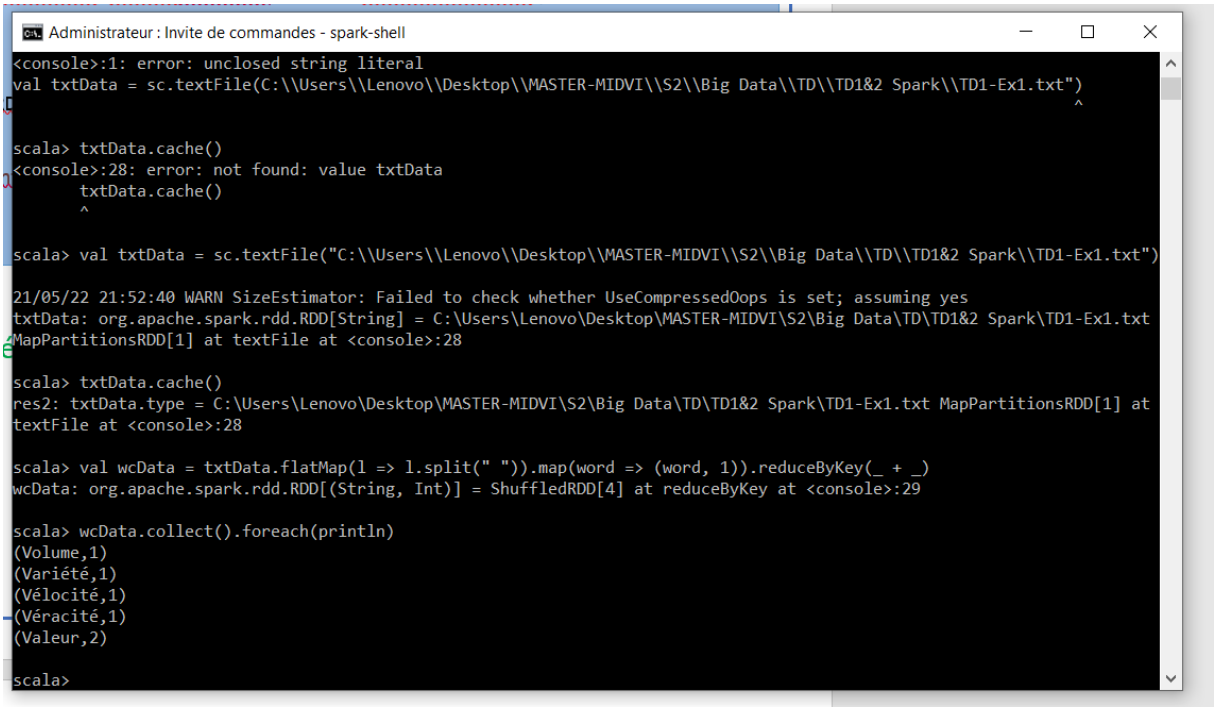
```
Velocite 1
Variete 1
Volume 1
Veracite 1
Valeur 2
21/05/10 22:34:53 INFO Executor: Finished task 0.0 in stage 4.0 (TID 4). 1095 bytes result sent to driver
21/05/10 22:34:53 INFO TaskSetManager: Finished task 0.0 in stage 4.0 (TID 4) in 68 ms on localhost (executor driver) (1/1)
21/05/10 22:34:53 INFO TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool
21/05/10 22:34:53 INFO DAGScheduler: ResultStage 4 (foreach at WordCount.java:26) finished in 0,086 s
21/05/10 22:34:53 INFO DAGScheduler: Job 3 finished: foreach at WordCount.java:26, took 0,184294 s
21/05/10 22:34:53 INFO SparkContext: Invoking stop() from shutdown hook
21/05/10 22:34:53 INFO SparkUI: Stopped Spark web UI at http://192.168.1.9:4041
21/05/10 22:34:53 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/05/10 22:34:53 INFO MemoryStore: MemoryStore cleared
21/05/10 22:34:53 INFO BlockManager: BlockManager stopped
21/05/10 22:34:53 INFO BlockManagerMaster: BlockManagerMaster stopped
21/05/10 22:34:53 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/05/10 22:34:53 INFO SparkContext: Successfully stopped SparkContext
21/05/10 22:34:53 INFO ShutdownHookManager: Shutdown hook called
21/05/10 22:34:53 INFO ShutdownHookManager: Deleting directory C:\Users\Lenovo\AppData\Local\Temp\spark-2bb03921-e4b0-4604-b99d-21bb4f9275a1
```

- Méthode 2:

1- code :

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
val txtFile = "README.md"
val txtData = sc.textFile(txtFile)
txtData.cache()
val wcData = txtData.flatMap(l => l.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
wcData.collect().foreach(println)
```

2- Résultat d’exécution :

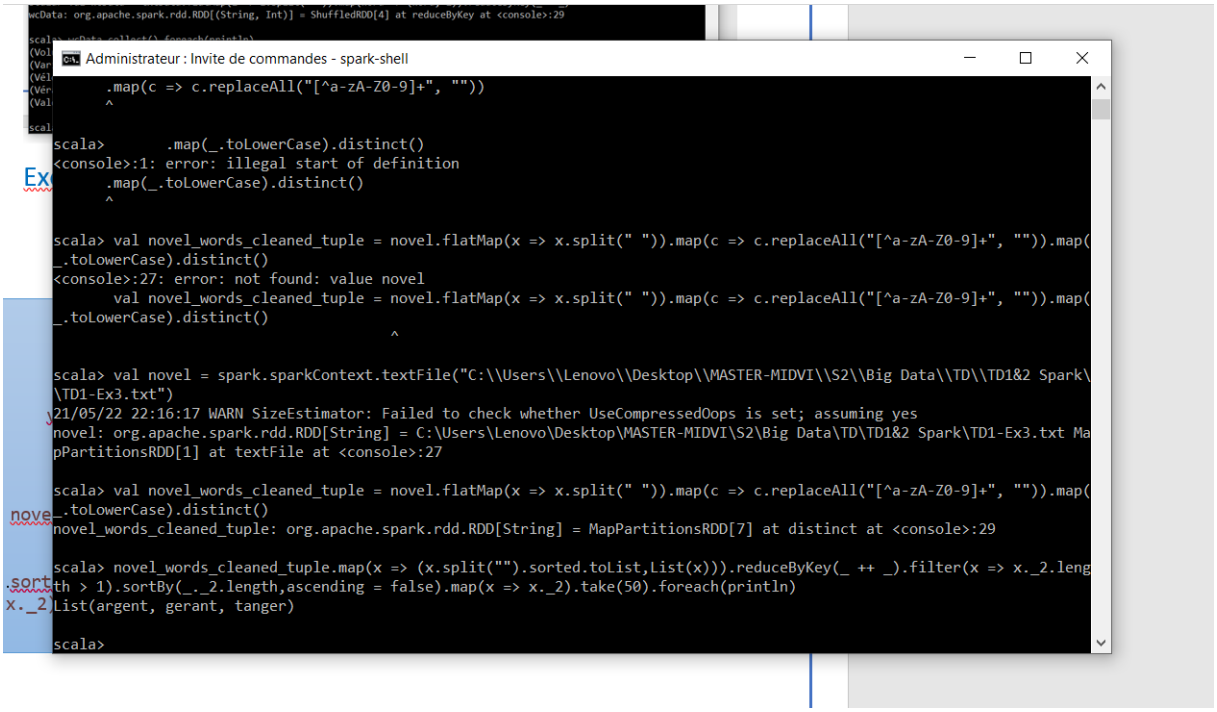


Exercice 3:

1- code :

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
val novel = spark.sparkContext.textFile("txtFile")
val novel_words_cleaned_tuple = novel.flatMap(x => x.split(" "))
.map(c => c.replaceAll("[^a-zA-Z0-9]+", ""))
.map(_.toLowerCase).distinct()
novel_words_cleaned_tuple.map(x => (x.split("").sorted.toList,List(x)))
.reduceByKey(_ ++ _).filter(x => x._2.length > 1)
.sortBy(_._2.length,ascending = false).map(x => x._2).take(50).foreach(println)
```

2- Résultat d’exécution :



Exercice 4:

1- Code :

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
var resultFile = ArrayBuffer[String]()
for ( i<- ArrayFile){
for (j <- 0 until i.length()-3){
var str = i.substring(j, j+4)
if((str.substring(0,1).equals(str.substring(1,2)) &&
str.substring(1,2).equals(str.substring(2,3))  &&
str.substring(2,3).equals(str.substring(3,4)) ) || (
str.substring(0,1).equals(str.substring(3,4)) &&
str.substring(1,2).equals(str.substring(2,3))  )  )
resultFile += str
}
}
var rdd = sc.parallelize(resultFile)
.flatMap(line => line.split(" "))
.map(word => (word , 1))
.reduceByKey(_+_)
.sortByKey(true, 1)
/* check and delete output directory if exists */
val outPath = new Path(outputPath);
val hadoopconf = new Configuration()
val fs = FileSystem.get(hadoopconf)

if (fs.exists(outPath) ){
fs.delete(outPath)
}
rdd.saveAsTextFile(outputPath)
```

2- Résultat d’exécution :

```
scala> val mots = t1.flatMap(line => line.split(" "))
mots: org.apache.spark.sql.Dataset[String] = [value: string]

scala> mots.show()
+------+
| value|
+------+
|  TOOT|
| PPTPP|
| PPOPP|
| PANAP|
| FFGFF|
|   TOT|
|   POP|
| FFRFF|
|FFGGFF|
|   KOK|
|  DEED|
|  FRRF|
| FDSDF|
|  TOOT|
+------+
```