1. Execute wordcount program in Spark Scala shell.

word\_count\_srcfile\_01.txt:

Hadoop is the Elephant King!

A yellow and elegant thing.

He never forgets

Useful data, or lets

An extraneous element cling!

word\_count\_srcfile\_02.txt:

A wonderful king is Hadoop.

The elephant plays well with Sqoop.

But what helps him to thrive

Are Impala, and Hive,

And HDFS in the group.

word\_count\_srcfile\_03.txt:

Hadoop is an elegant fellow.

An elephant gentle and mellow.

He never gets mad,

Or does anything bad,

Because, at his core, he is yellow.

scala> val wordcount\_01 = sc.textFile("file:///my-folder/src\_file/word\_count\_srcfile\_\*.txt")

wordcount\_01: org.apache.spark.rdd.RDD[String] = file:///my-folder/src\_file/word\_count\_srcfile\_\*.txt MapPartitionsRDD[30] at textFile at <console>:27

scala> wordcount\_01.flatMap(\_.split("\\W")).map((\_,1)).reduceByKey(\_+\_).sortByKey().foreach(println)

(,4)

(A,2)

(An,2)

(And,1)

(Are,1)

(Because,1)

(But,1)

(Elephant,1)

(HDFS,1)

(Hadoop,3)

(He,2)

(Hive,1)

(Impala,1)

(King,1)

(Or,1)

(Sqoop,1)

(The,1)

(Useful,1)

(an,1)

(and,3)

(anything,1)

(at,1)

(bad,1)

(cling,1)

(core,1)

(data,1)

(does,1)

(elegant,2)

(element,1)

(elephant,2)

(extraneous,1)

(fellow,1)

(forgets,1)

(gentle,1)

(gets,1)

(group,1)

(he,1)

(helps,1)

(him,1)

(his,1)

(in,1)

(is,4)

(king,1)

(lets,1)

(mad,1)

(mellow,1)

(never,2)

(or,1)

(plays,1)

(the,2)

(thing,1)

(thrive,1)

(to,1)

(well,1)

(what,1)

(with,1)

(wonderful,1)

(yellow,2)