Big Data Systems

Semester Assignment

Aglaia Tsopelakou BAFT1716

Question 1: (25 points)

A. Firstly, we read the airport data from "/user/b-analytics/assignment/airports.text"

```
spark-shell --deploy-mode client --master yarn
val airports =sc.textFile("/user/b-
analytics/assignment/airports.text")
airports.take(5)
```

Then, we find all the airports which are located in Greece with the line.contains command and we selected the airport's name, the city's name and the airports IATA/FAA code as output columns. The contains command tests if some element is contained in this set in our case Greece. The command map builds a new collection (airports selection) by applying a function to all elements of this mutable set. Finally, with the last command we displayed our data in columns.

```
val greeceairports = airports.filter(line=>line.contains("Greece"))

val airports_selection
=greeceairports.map(_.split(",")).map(greeceairports=>(greeceairport
s(1), greeceairports(2), greeceairports(4)))

airports_selection.collect().foreach(println)
```

B. In second part, we found all the airports whose latitude are greater than 37 and lower than 39 using the command filter. Filter, selects all elements of this traversable collection which satisfy a predicate and it returns a new traversable collection consisting of all elements of this traversable collection that satisfy the given predicate.

```
val airportsb=
airports.filter(_.split(",")(6).toDouble>37).filter(_.split(",")(6).
toDouble<39)
```

Furthermore, we selected the columns that we wanted to display which are the Name of airport, Main city served by airport and the Latitude. In the last command we display our output only for the first 100 values because spark cannot display more values than that.

```
val airportsbmap=
airportsb.map(_.split(",")).map(airportsb =>(airportsb(1),
airportsb(2), airportsb(6).toString))
airportsbmap.take(100).foreach(println)
```

Question 2: (30 points)

A. The datasets of 1st of July and 1st of August were read.

```
val nasa_july  =sc.textFile("/user/b-
analytics/assignment/nasa_19950701.tsv")
val nasa_aug  =sc.textFile("/user/b-
analytics/assignment/nasa_19950801.tsv")
```

Furthermore, we split our data so as to have access in host column.

```
val first_hosts_july = nasa_july.map(line => line.split("\t")(0))
```

```
val first_hosts_aug = nasa_aug .map(line => line.split("\t")(0))
```

Finally, we find the intersection of the two datasets based on the Host and we remove the header. The intersection command computes the intersection between this set and another set.

```
val intersection_nasa= first_hosts_july
.intersection(first_hosts_aug)
intersection_nasa.collect.foreach(println)

val intersection_nasa_without_header= intersection_nasa.filter(line
=> line != "host")
intersection_nasa_without_header.collect.foreach(println)
```

B. We read an article from "/user/b-analytics/assignment/word_count.text".

```
val article =sc.textFile("/user/b-
analytics/assignment/word_count.text")
```

Below, we calculated the number of occurrence of each word using flat map. Flat map builds a new collection by applying a function to all elements of this mutable set and using the elements of the resulting collections.

```
val wordCounts = article.flatMap( line => line.split (" ")).map(word
=> (word , 1)).reduceByKey(( a,b) => a + b)
wordCounts.take(20).foreach(println)
```

Below, it is the number of occurrence of each word in descending order. Also, we did a count and we found 376 occurrences.

```
for (pair <-wordCounts.map(_.swap).top(20)) println(pair._2,
pair._1)
```

```
wordCounts.count()
```
So, we run again with top (376) to have them all in desc order.

```
for (pair <-wordCounts.map(_.swap).top(376)) println(pair._2,
pair._1)
```

Question 3: (45 points)

A. Firstly, the houses data were loaded. Then we kept all the columns that contained bedrooms. Further, we used map so as to split the data and keep only the columns with the price and the bedrooms. Finally, we used mapvalues and map so as to compute the average price for houses because we wanted to divide the value1 and the value2 which had the same key.

```
val houses =sc.textFile("/user/b-analytics/assignment/Real.csv")
val housescolumnremove = houses.filter(line=>
!line.contains("Bedrooms"))
val housescolumn=housescolumnremove.map(_.split(",")).map(houses
=>(houses(3).toDouble, houses(2).toDouble))
val map=housescolumn.mapValues(x => (x, 1)).reduceByKey((x, y) =>
(x._1 + y._1, x._2 + y._2))
val map2=map.map{ case (key,value) => (key, value._1/ value._2)}
```

B. In this question we used spark SQL. Firstly, we did some imports and we established a connection and then we read our data. Furthermore, we did filtering again and we put new headers in the dataset. Also, we split our data and we make it as a table. Finally, we execute the query.

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext.implicits._

val real = sc.textFile("/user/b-analytics/assignment/Real.csv")

val realremove = real.filter(line=> !line.contains("Bedrooms"))

case class Houses(MLS: Int, Location: String, Price: Double,
Bedrooms: Int, Bathrooms: Int, Size: Double, Price_SQ_Ft: Double,
Status: String)


val house = realremove.map(_.split(",")).map(p =>
Houses(p(0).trim.toInt, p(1), p(2).trim.toDouble, p(3).trim.toInt,
```

```
p(4).trim.toInt, p(5).trim.toDouble,p(6).trim.toDouble,
p(7))).toDF()

house.registerTempTable("house")

val query=sqlContext.sql("SELECT Location, avg(Price_SQ_Ft),
max(Price) FROM house GROUP BY Location ORDER BY avg(Price_SQ_Ft)")

query.collect().foreach(println)
```

References

1. https://www.scala-lang.org/api/current/scala/collection/immutable/Set.html
   https://www.scala-lang.org/api/current/scala/collection/mutable/Set.html