Test2AssignmentchatGPTexpl.md

CS5104 Big Data Lab

Code

```
package in.iitpkd.scala
import org.apache.spark._
import org.apache.log4j._

object Test2Assignment {
  def main(args: Array[String]) {
    Logger.getLogger("org").setLevel(Level.ERROR)
    val sc = new SparkContext("local[*]", "RatingsCounter")


    // Read the input file and create an RDD
    val input = sc.textFile("/Users/ajaikumarmp/Downloads/2022.07_WAVES-A

    // Extract the header and remove it from the RDD
    val header = input.first()
    val data = input.filter(row => row != header)

    // (i) 10 most frequent visitors
    val visitors = data.map(row => {
      val tokens = row.split(",")
      ((tokens(0), tokens(1), tokens(2)), 1)
    })
    val mostFrequentVisitors = visitors.reduceByKey(_ + _).sortBy(_._2, a
    println("10 most frequent visitors:")
    mostFrequentVisitors.foreach(println)

    // (ii) 10 most frequently visited people
    val visitees = data.map(row => {
      val tokens = row.split(",")
      ((tokens(19), tokens(20)), 1)
    })
    val mostFrequentVisitees = visitees.reduceByKey(_ + _).sortBy(_._2, a
    println("10 most frequently visited people:")
    mostFrequentVisitees.foreach(println)

    // (iii) 10 most frequent visitor-visitee combinations
    val visitorVisitee = data.map(row => {
      val tokens = row.split(",")
      (((tokens(0), tokens(1), tokens(2)), (tokens(19), tokens(20))), 1)
    })
    val mostFrequentVisitorVisitee = visitorVisitee.reduceByKey(_ + _).so
    println("10 most frequent visitor-visitee combinations:")
    mostFrequentVisitorVisitee.foreach(println)

    // (iv) Highest appointments made according to APPT_MADE_DATE
    val appointments = data.map(row => {
      val tokens = row.split(",")
      (tokens(10), 1)
    })
    val highestAppointments = appointments.reduceByKey(_ + _).sortBy(_._2
    println(s"Highest appointments made on date: ${highestAppointments._1

    // (v) Percentage of tour visitors
    val totalVisitors = data.count()
    val tourVisitors = data.filter(row => row.split(",")(22) == "EW TOUR"
    val tourVisitorPercentage = (tourVisitors.toFloat / totalVisitors.toF
    println(s"Percentage of tour visitors: $tourVisitorPercentage%")

    // Stop the Spark context
    sc.stop()
  }
}
```

Question 1

Sure, here's a line-by-line explanation of the code snippet you provided:

```
// (i) 10 most frequent visitors
```

This is a comment that describes the purpose of the code that follows.

```
val visitors = data.map(row => {
  val tokens = row.split(",")
  ((tokens(0), tokens(1), tokens(2)), 1)
})
```

This declares a new visitors variable that is a transformation of a data variable (which we assume to be an RDD, Dataframe or Dataset). It uses the map method to apply a function to each element in data. The function takes each row and splits it into an array of tokens using a comma separator. It then creates a tuple that consists of the first three elements of tokens (presumably representing information about a visitor, such as their name or ID) and the integer value 1. This tuple represents a visitor and a count of 1 (since we haven't aggregated the counts yet).

```
val mostFrequentVisitors = visitors.reduceByKey(_ + _).sortBy(_._2, ascen
```

This declares a new mostFrequentVisitors variable by applying a series of transformations to the visitors RDD. First, it uses the reduceByKey method to aggregate the counts for each visitor tuple by key (i.e., by the visitor information). The _ + _ syntax is shorthand for a function that takes two integers and returns their sum. The resulting RDD has one element for each visitor tuple, with the count of visits as the value. Next, it uses the sortBy method to sort the RDD in descending order based on the second element of each tuple (i.e., the count). Finally, it uses the take method to return the first 10 elements of the RDD, which represents the 10 most frequent visitors.

```
println("10 most frequent visitors:")
mostFrequentVisitors.foreach(println)
```

This prints a message to the console indicating that the following lines will output the 10 most frequent visitors. Then it uses the foreach method to loop over the elements of the mostFrequentVisitors RDD and print each element to the console. Since each element is a tuple, it will be printed in the form (visitor information, count).

Question 2

Sure, here's a line-by-line explanation of the code snippet you provided:

```
// (ii) 10 most frequently visited people
```

This is a comment that describes the purpose of the code that follows.

```
val visitees = data.map(row => {
  val tokens = row.split(",")
  ((tokens(19), tokens(20)), 1)
})
```

This declares a new visitees variable that is a transformation of a data variable (which we assume to be an RDD, Dataframe or Dataset). It uses the map method to apply a function to each element in data. The function takes each row and splits it into an array of tokens using a comma separator. It then creates a tuple that consists of the 20th and 21st elements of tokens (presumably representing information about a person who was visited, such as their name or ID) and the integer value 1. This tuple represents a visitee and a count of 1 (since we haven't aggregated the counts yet).

```
val mostFrequentVisitees = visitees.reduceByKey(_ + _).sortBy(_._2, ascen
```

This declares a new mostFrequentVisitees variable by applying a series of transformations to the visitees RDD. First, it uses the reduceByKey method to aggregate the counts for each visitee tuple by key (i.e., by the visitee information). The _ + _ syntax is shorthand for a function that takes two integers and returns their sum. The resulting RDD has one element for each visitee tuple, with the count of visits as the value. Next, it uses the sortBy method to sort the RDD in descending order based on the second element of each tuple (i.e., the count). Finally, it uses the take method to return the first 10 elements of the RDD, which represents the 10 most frequently visited people.

```
println("10 most frequently visited people:")
mostFrequentVisitees.foreach(println)
```

This prints a message to the console indicating that the following lines will output the 10 most frequently visited people. Then it uses the foreach method to loop over the elements of the mostFrequentVisitees RDD and print each element to the console. Since each element is a tuple, it will be printed in the form (visitee information, count).

Question 3

Sure, here's a line-by-line explanation of the code snippet you provided:

```
// (iii) 10 most frequent visitor-visitee combinations
```

This is a comment that describes the purpose of the code that follows.

```
val visitorVisitee = data.map(row => {
  val tokens = row.split(",")
  (((tokens(0), tokens(1), tokens(2)), (tokens(19), tokens(20))), 1)
})
```

This declares a new visitorVisitee variable that is a transformation of a data variable (which we assume to be an RDD, Dataframe or Dataset). It uses the map method to apply a function to each element in data. The function takes each row and splits it into an array of tokens using a comma separator. It then creates a nested tuple that consists of two tuples: the first tuple consists of the 1st, 2nd and 3rd elements of tokens (presumably representing information about the visitor, such as their name or ID) and the second tuple consists of the 20th and 21st elements of tokens (presumably representing information about the visitee, such as their name or ID) and the integer value 1. This tuple represents a combination of a visitor and visitee and a count of 1 (since we haven't aggregated the counts yet).

```
val mostFrequentVisitorVisitee = visitorVisitee.reduceByKey(_ + _).sortBy
```

This declares a new mostFrequentVisitorVisitee variable by applying a series of transformations to the visitorVisitee RDD. First, it uses the reduceByKey method to aggregate the counts for each visitor-visitee combination tuple by key. The resulting RDD has one element for each visitor-visitee combination tuple, with the count of visits as the value. Next, it uses the sortBy method to sort the RDD in descending order based on the second element of each tuple (i.e., the count). Finally, it uses the take method to return the first 10 elements of the RDD, which represents the 10 most frequent visitor-visitee combinations.

```
println("10 most frequent visitor-visitee combinations:")
mostFrequentVisitorVisitee.foreach(println)
```

This prints a message to the console indicating that the following lines will output the 10 most frequent visitor-visitee combinations. Then it uses the foreach method to loop over the elements of the mostFrequentVisitorVisitee RDD and print each element to the console. Since each element is a tuple, it will be printed in the form ((visitor information), (visitee information), count).

Question 4

Sure, here's a line-by-line explanation of the code snippet you provided:

```
// (iv) Highest appointments made according to APPT_MADE_DATE
```

This is a comment that describes the purpose of the code that follows.

```
val appointments = data.map(row => {
  val tokens = row.split(",")
  (tokens(10), 1)
})
```

This declares a new appointments variable that is a transformation of a data variable (which we assume to be an RDD, Dataframe or Dataset). It uses the map method to apply a function to each element in data. The function takes each row and splits it into an array of tokens using a comma separator. It then creates a tuple that consists of the 11th element of tokens (presumably representing the date on which the appointment was made) and the integer value 1. This tuple represents an appointment made on a specific date and a count of 1 (since we haven't aggregated the counts yet).

```
val highestAppointments = appointments.reduceByKey(_ + _).sortBy(_._2, as
```

This declares a new highestAppointments variable by applying a series of transformations to the appointments RDD. First, it uses the reduceByKey method to aggregate the counts for each appointment date tuple by key. The resulting RDD has one element for each appointment date, with the count of appointments as the value. Next, it uses the sortBy method to sort the RDD in descending order based on the second element of each tuple (i.e., the count). Finally, it uses the first method to return the first element of the RDD, which represents the appointment date with the highest count of appointments.

```
println(s"Highest appointments made on date: ${highestAppointments._1) wi
```

This prints a message to the console indicating the date on which the highest number of appointments were made, and the count of appointments made on that date. The message is created using string interpolation, where the values of `highestAppointments._1` and `highestAppointments._2` are inserted into the string. The `_1` and `_2` are the index numbers of the elements of the tuple that highestAppointments points to.

Question 5

Sure, here's a line-by-line explanation of the code snippet you provided:

```
// (v) Percentage of tour visitors
```

This is a comment that describes the purpose of the code that follows.

```
val totalVisitors = data.count()
```

This declares a new totalVisitors variable that is the total number of visitors in the data RDD (or DataFrame/Dataset). It uses the count method to count the number of elements in the RDD.

```
val tourVisitors = data.filter(row => row.split(",")(22) == "EW TOUR").co
```

This declares a new tourVisitors variable that is the total number of visitors who went on a tour. It uses the filter method to keep only the rows where the 23rd element (presumably representing a visitor's tour status) equals "EW TOUR". Then, it uses the count method to count the number of elements in the resulting RDD, which represents the number of visitors who went on a tour.

```
val tourVisitorPercentage = (tourVisitors.toFloat / totalVisitors.toFloat
```

This declares a new tourVisitorPercentage variable that is the percentage of visitors who went on a tour. It divides the tourVisitors count by the totalVisitors count, converts both counts to Float values to ensure a floating point division, multiplies the result by 100 to get the percentage, and assigns the result to tourVisitorPercentage.

```
println(s"Percentage of tour visitors: $tourVisitorPercentage%")
```

This prints a message to the console indicating the percentage of visitors who went on a tour. The message is created using string interpolation, where the value of tourVisitorPercentage is inserted into the string. The %i sign is added to the end of the string to denote a percentage value.