quantexa
outcomes delivered

Coding Assignment

Introduction

The aim of this exercise is to test your programming ability.

The solution should be written in Scala, as this is the primary language used at Quantexa, and you should aim to write the solution in a functional style. If you find it helpful you may use any functionality from the standard library in your solution.

Please note we will be reviewing both the functionality of your code as well as the style so please ensure you read the Scala coding guidelines, found in the Technical Best Practice, section of the training, before attempting your answers.

Assignment

One technique to detect criminal behaviour is to consider the activity of groups of individuals, rather than only analysing each customer independently. To do this we must first define groups within our data. A commonly used definition of a group is simply the household - people who share the same address.

You arrive at work to find the following email from your boss:

> *Hi,*
>
> *Sorry I missed you, but I've had to run out to a client this morning.*
>
> *I have some issues with the household profiling exercise. There are 10 addresses with a large number of registered customers, and these addresses are confusing our analysis.*
>
> *For these 10 addresses, rather than grouping all the people ever registered to them together, instead I'd like to consider the dates customers were registered to the addresses. I only want to add a customer to a group when they lived at the address at the same time as at least one other member of that group.*
>
> *e.g. If person A is registered at address ADR1 from day 1 to day 100, person B from day 80 to 200 and person C from day 250 to 300; then results should be:*

Group	Address_ID	Start_date	End_date
Group1	ADD1	1	200
Group2	ADD1	250	300

> *I've sent you data giving the different customers and the addresses they are registered to. I've already formatted the data as described in the file description (NB dates are already formatted as integers, which should make it a little easier).*

Could you calculate all the different groups within the addresses, giving the start date and end date for each group. Please let me have:

- *The Address ID*
- *Customer IDs*
- *The start date for the group (same format as in the input data)*
- *The end date for the group (same format as in the input data)*

Also, calculate the total number of groups and any other important information that occurs to you.

Note: Each customer in the group only needs to overlap with at least one other customer in the group, therefore there may be pairs of customer in the group who never lived at the address at the same time.

Data

You have been provided with a CSV of 500 customers indicating the dates they are registered at addresses. This contains 4 fields:

Field	Description
Customer_ID	A unique ID for each individual customer in the data.
Address_ID	A unique ID for each address in the data.
From_date	An integer representing the first day the customer is registered at that address. Represents number of days after the start of the time window of our data, eg. date 1 indicates 1st January 2010.
To_date	An integer representing the last day the customer is registered at that address. Represents number of days after the start of the time window of our data, eg. date 1 indicates 1st January 2010.

Output

The expected output which sufficiently answers the assigned questions is described in the following tables. These contain both the final output format and any accompanying data structures required (e.g. case classes).

Please note that your results should adhere to the naming and Type conventions specified in the below tables.

Type	Name	Description
List	addressGroups	A List, which contains elements of the AddressGroupedData case class.
Int	numberOfGroups	The total number of output groups.

Table 1: Expected output format

Type	Name	Fields	Field Type	Description
Case class	AddressGroupedData	group addressId customerIds startDate endDate	Long String Seq[String] Int Int	A case class containing the relevant output information.

Table 2: Auxiliary case class

Code

The following code can be used to import the data and check the first few records. It also specifies the output format which must be used to solve the current problem.

Scala 2.11:

```
import scala.io.Source

object AddressAssignment{

//Define a case class AddressData which stores the occupancy data
case class AddressData(
      customerId: String,
      addressId: String,
      fromDate: Int,
      toDate: Int
)

//The full path to the file to import
val fileName = getClass.getResource("/address_data.csv").getPath

//The lines of the CSV file (dropping the first to remove the header)
val addressLines = Source.fromFile(fileName).getLines().drop(1)

val occupancyData: List[AddressData] = addressLines.map { line =>
  val split = line.split(',')
  AddressData(split(0), split(1), split(2).toInt, split(3).toInt)
}.toList

// Sorts ascending
val sortedOccupancyData = occupancyData.sortBy(r => (r.addressId, r.customerId))

println(sortedOccupancyData.take(20))

//END GIVEN CODE

}
```