Session 18

Assignment 1

|  |  |
| --- | --- |
| **Prepared For:** | AcadGild |
|  |  |
| **Document Approval:** | **AcadGild** |
|  |  |
|  |  |
|  |  |
|  |  |
| **Project Title:** | Session 18 – Assignment 1 |
|  |  |
| **Prepared By:** | Duncan Burgess |
|  |  |
|  | dburgess@duncb.com |
|  |  |
| **Primary Engineer:** | Duncan Burgess |
|  |  |
| **Document Reference:** | **Session 18 – Assignment 1** |
|  |  |
| **Start Date:** | 17/10/2017 |
|  |  |
|  |  |



# 

# Contents

[Contents 2](#_Toc496185554)

[Change History 3](#_Toc496185555)

[1. Problem Statement 4](#_Toc496185556)

[2. Datasets 4](#_Toc496185557)

[3. Solution 5](#_Toc496185558)

[4. Results 8](#_Toc496185559)

# Change History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Revision** | **Date** | **Authored By** | **Authorised By** | **Sections Affected** | **Reason for Change** |
| Rev 01 | 19/10/2017 | Duncan Burgess |  | All | Initial release. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Problem Statement

* What is the distribution of the total number of air-travellers per year
* What is the total air distance covered by each user per year
* Which user has travelled the largest distance till date
* What is the most preferred destination for all users.

Using the datasets below:

# Datasets

**S18\_Dataset\_Holidays.txt**

1,CHN,IND,airplane,200,1990

2,IND,CHN,airplane,200,1991

3,IND,CHN,airplane,200,1992

4,RUS,IND,airplane,200,1990

5,CHN,RUS,airplane,200,1992

6,AUS,PAK,airplane,200,1991

7,RUS,AUS,airplane,200,1990

8,IND,RUS,airplane,200,1991

9,CHN,RUS,airplane,200,1992

10,AUS,CHN,airplane,200,1993

1,AUS,CHN,airplane,200,1993

2,CHN,IND,airplane,200,1993

3,CHN,IND,airplane,200,1993

4,IND,AUS,airplane,200,1991

5,AUS,IND,airplane,200,1992

6,RUS,CHN,airplane,200,1993

7,CHN,RUS,airplane,200,1990

8,AUS,CHN,airplane,200,1990

9,IND,AUS,airplane,200,1991

10,RUS,CHN,airplane,200,1992

1,PAK,IND,airplane,200,1993

2,IND,RUS,airplane,200,1991

3,CHN,PAK,airplane,200,1991

4,CHN,PAK,airplane,200,1990

5,IND,PAK,airplane,200,1991

6,PAK,RUS,airplane,200,1991

7,CHN,IND,airplane,200,1990

8,RUS,IND,airplane,200,1992

9,RUS,IND,airplane,200,1992

10,CHN,AUS,airplane,200,1990

1,PAK,AUS,airplane,200,1993

5,CHN,PAK,airplane,200,1994

**S18\_Dataset\_User\_Details.txt**

1,mark,15

2,john,16

3,luke,17

4,lisa,27

5,mark,25

6,peter,22

7,james,21

8,andrew,55

9,thomas,46

10,annie,44

**S18\_Dataset**\_Transport.txt

airplane,170

car,140

train,120

ship,200

# Solution

I have deliberately commented and printed out all steps for enhanced learning

**Code written**

**package** com.duncb.spark

**import** org.apache.spark.\_

**import** org.apache.spark.SparkContext.\_

**import** org.apache.log4j.\_

//\*\* Compute the the following

**object** holiday1 {

/\*\* A function that splits a line of input into (userID,src,dest,travelMode,distance,yearOfTravel) tuples. \*/

**def** travelInfo(line: *String*) = {

// Split by commas

**val** fields = line.split(",")

**val** userID = fields(0).toInt

**val** src = fields(1).toString

**val** dest = fields (2).toString

**val** travelMode = fields (3).toString

**val** distance = fields (4).toInt

**val** yearOfTravel =fields(5).toInt

// Create a tuple that is our result.

(userID,src,dest,travelMode,distance,yearOfTravel)

}

/\*\* A function that splits a line of input into (userID,name and age) tuples. \*/

**def** userInfo(line: *String*) = {

// Split by commas

**val** fields = line.split(",")

**val** userID = fields(0).toInt

**val** userName = fields(1).toString

**val** age = fields (2).toInt

// Create a tuple that is our result.

(userID,userName,age)

}

/\*\* A function that splits a line of input into (transport,cost) tuples. \*/

**def** transportInfo(line: *String*) = {

// Split by commas

**val** fields = line.split(",")

**val** transport = fields(0).toString

**val** cost = fields (1).toInt

// Create a tuple that is our result.

(transport,cost)

}

**def** main(args: Array[*String*]) {

// Set the log level to only print errors

Logger.getLogger("org").setLevel(Level.ERROR)

// Create a SparkContext using every core of the local machine

**val** sc = **new** SparkContext("local[1]", "holidayData")

// Load each line of the source data into an RDD

**val** lines = sc.textFile("file:///N:/Datasets/S18\_Dataset\_Holidays.txt")

**val** holidaysRDD = lines.map(travelInfo)

// Load each line of the source data into an RDD

**val** linesU = sc.textFile("file:///N:/Datasets/S18\_Dataset\_User\_Details.txt")

**val** usersRDD = lines.map(userInfo)

// Load each line of the source data into an RDD

**val** linesT = sc.textFile("file:///N:/Datasets/S18\_Dataset\_Transport.txt")

**val** TransportRDD = lines.map(transportInfo)

println(" Assignment 18.1 problem 1")

//val x = holidaysRDD.map(x => (x.\_1, x.\_6)).distinct.map(x=>(x.\_2,x.\_1)).groupByKey().map (x => x.\_1 -> x.\_2.sum) Broken down

**val** d1 = holidaysRDD.map(x=> (x.\_1,x.\_6))

d1.foreach (println)

println("-----------------------------")

**val** d2= d1.distinct.map(x=>(x.\_2,x.\_1))

d2.foreach (println)

println("-----------------------------")

**val** d3 = d2.groupByKey().map(x=> x.\_1 -> x.\_2.sum)

println(" Total number of travellers per year ")

d3.foreach(println)

println("-----------------------------")

//Key = UsedID Value = Year

//val totAirDist = holidaysRDD.map(x => (x.\_1 -> x.\_6 -> x.\_5)).groupByKey().map(x=>x.\_1 -> x.\_2.sum).sortByKey

println(" Assignment 18.1 problem 2")

**val** e1 = holidaysRDD.map(x=> (x.\_1 -> x.\_6 -> x.\_5))

e1.foreach (println)

println("-----------------------------")

**val** e2 = e1.groupByKey().map(x=>x.\_1 -> x.\_2.sum)

e2.foreach (println)

println("-----------------------------")

**val** e3 = e2.sortByKey()

e3.foreach (println)

println("-----------------------------")

println(" Assignment 18.1 problem 2")

**val** f1 = holidaysRDD.map(x=> ((x.\_1), x.\_5))

f1.foreach (println)

println("-----------------------------")

**val** f2 = f1.groupByKey().map(x=>(x.\_1 -> x.\_2.sum))

f2.foreach (println)

println("-----------------------------")

**val** f3 = f2.sortBy(x=> -x.\_2)

println(" Toatl air distance covered per user/year")

f3.foreach (println)

println("-----------------------------")

//This is to give top user or users if more than 1

**val** f4 = f3.sortBy(x=> -x.\_2)

**val** topx=f4.map(x=>(x.\_2)).max

println("The top distance is " + topx)

**val** mfly = f4.filter(x=> x.\_2==topx)

println("User traveled the largest distance till date")

mfly.foreach (println)

println("-----------------------------")

println(" Assignment 18.1 problem 4")

**val** g1 = holidaysRDD.map(x=> x.\_3 -> 1).groupByKey()

g1.foreach (println)

println("-----------------------------")

**val** g2 = g1.map(x=>x.\_1 -> x.\_2.sum)

g2.foreach (println)

println("-----------------------------")

**val** preferredDestination= g2.sortBy( x=> -x.\_2).first()

println(" The preferred destination is " + preferredDestination)

}

}

# Results

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

Assignment 18.1 problem 1

(1,1990)

(2,1991)

(3,1992)

(4,1990)

(5,1992)

(6,1991)

(7,1990)

(8,1991)

(9,1992)

(10,1993)

(1,1993)

(2,1993)

(3,1993)

(4,1991)

(5,1992)

(6,1993)

(7,1990)

(8,1990)

(9,1991)

(10,1992)

(1,1993)

(2,1991)

(3,1991)

(4,1990)

(5,1991)

(6,1991)

(7,1990)

(8,1992)

(9,1992)

(10,1990)

(1,1993)

(5,1994)

-----------------------------

(1993,2)

(1993,6)

(1993,10)

(1992,10)

(1991,2)

(1990,4)

(1990,10)

(1992,5)

(1991,4)

(1993,1)

(1992,9)

(1991,5)

(1993,3)

(1990,1)

(1990,8)

(1990,7)

(1991,6)

(1994,5)

(1991,3)

(1991,9)

(1992,3)

(1991,8)

(1992,8)

-----------------------------

Total number of travellers per year

(1994,5)

(1992,35)

(1990,30)

(1991,37)

(1993,22)

-----------------------------

Assignment 18.1 problem 2

((1,1990),200)

((2,1991),200)

((3,1992),200)

((4,1990),200)

((5,1992),200)

((6,1991),200)

((7,1990),200)

((8,1991),200)

((9,1992),200)

((10,1993),200)

((1,1993),200)

((2,1993),200)

((3,1993),200)

((4,1991),200)

((5,1992),200)

((6,1993),200)

((7,1990),200)

((8,1990),200)

((9,1991),200)

((10,1992),200)

((1,1993),200)

((2,1991),200)

((3,1991),200)

((4,1990),200)

((5,1991),200)

((6,1991),200)

((7,1990),200)

((8,1992),200)

((9,1992),200)

((10,1990),200)

((1,1993),200)

((5,1994),200)

-----------------------------

((2,1993),200)

((6,1993),200)

((10,1993),200)

((10,1992),200)

((2,1991),400)

((4,1990),400)

((10,1990),200)

((5,1992),400)

((4,1991),200)

((1,1993),600)

((9,1992),400)

((5,1991),200)

((3,1993),200)

((1,1990),200)

((8,1990),200)

((7,1990),600)

((6,1991),400)

((5,1994),200)

((3,1991),200)

((9,1991),200)

((3,1992),200)

((8,1991),200)

((8,1992),200)

-----------------------------

((1,1990),200)

((1,1993),600)

((2,1991),400)

((2,1993),200)

((3,1991),200)

((3,1992),200)

((3,1993),200)

((4,1990),400)

((4,1991),200)

((5,1991),200)

((5,1992),400)

((5,1994),200)

((6,1991),400)

((6,1993),200)

((7,1990),600)

((8,1990),200)

((8,1991),200)

((8,1992),200)

((9,1991),200)

((9,1992),400)

((10,1990),200)

((10,1992),200)

((10,1993),200)

-----------------------------

Assignment 18.1 problem 2

(1,200)

(2,200)

(3,200)

(4,200)

(5,200)

(6,200)

(7,200)

(8,200)

(9,200)

(10,200)

(1,200)

(2,200)

(3,200)

(4,200)

(5,200)

(6,200)

(7,200)

(8,200)

(9,200)

(10,200)

(1,200)

(2,200)

(3,200)

(4,200)

(5,200)

(6,200)

(7,200)

(8,200)

(9,200)

(10,200)

(1,200)

(5,200)

-----------------------------

(4,600)

(1,800)

(6,600)

(3,600)

(7,600)

(9,600)

(8,600)

(10,600)

(5,800)

(2,600)

-----------------------------

Toatl air distance covered per user/year

(1,800)

(5,800)

(4,600)

(6,600)

(3,600)

(7,600)

(9,600)

(8,600)

(10,600)

(2,600)

-----------------------------

The top distance is 800

User traveled the largest distance till date

(1,800)

(5,800)

-----------------------------

Assignment 18.1 problem 4

(CHN,CompactBuffer(1, 1, 1, 1, 1, 1, 1))

(IND,CompactBuffer(1, 1, 1, 1, 1, 1, 1, 1, 1))

(PAK,CompactBuffer(1, 1, 1, 1, 1))

(RUS,CompactBuffer(1, 1, 1, 1, 1, 1))

(AUS,CompactBuffer(1, 1, 1, 1, 1))

-----------------------------

(CHN,7)

(IND,9)

(PAK,5)

(RUS,6)

(AUS,5)

-----------------------------

The preferred destination is (IND,9)