Assignment Solution

Week10: Apache Spark - in Depth

Spark-Assignment Solution

```
//Create Base RDD for chapters data


val chapterDataRDD =
sc.textFile("C:/Spark_Assignment/Dataset/chapters.csv").map(x => {

val chapterDataFields = x.split(",")

(chapterDataFields(0).toInt,chapterDataFields(1).toInt)


})



//Create Base RDD for views data


val viewDataRDD =
sc.textFile("C:/Spark_Assignment/Dataset/views-*.csv").map(x => {

(x.split(",")(0).toInt, x.split(",")(1).toInt)

})



//Create Base RDD for titles data
```

```
val titlesDataRDD =
sc.textFile("C:/Spark_Assignment/Dataset/titles.csv").map( x =>
(x.split(",")(0).toInt, x.split(",")(1)))
```

```
//Exercise1: Find Number of Chapters Per Course
```

```
val chapterCountRDD = chapterDataRDD.map(x =>
(x._2,1)).reduceByKey((x,y) => x + y)
```

```
//Output from the above Step
```

```
(Course,ChapterCount)
(4,10)
(16,7)
(14,28)
(6,9)
(8,32)
(12,5)
(18,17)
(10,6)
(2,30)
(13,7)
(15,12)
(11,8)
(1,34)
```

(17,7)

(3,14)

(7,15)

(9,6)

(5,10)

//Exercise 2:

//Step 1:Removing Duplicate Views from views RDD

```
val viewDataDistinctRDD = viewDataRDD.distinct()
```

//Sample Output

(75,200)

(378,184)

(490,132)

(52,236)

(18,93)

(276,53)

(52,40)

(524,214)

(486,137)

(101,171)

(576,15)

(72,161)

//Step 2: Joining chapterDataRDD with viewDataDistinctRDD, to get CourseID also.Join key is chapterID

//First flip the viewDataDistinctRDD to make chapterId as the key

```
val flippedviewDataRDD = viewDataDistinctRDD.map(x => (x._2,x._1))
```

Sample Output:

```
(110,323)
(127,303)
(60,343)
(199,156)
(84,556)
(125,250)
(234,187)
(96,239)
```


//JOIN flippedviewDataRDD with chapterDataRDD to get the courseIDs as well

```
val joinedRDD = flippedviewDataRDD.join (chapterDataRDD)
```

Sample Output:

(206,(182,14))

(206,(99,14))

(206,(510,14))

(206,(231,14))

(206,(271,14))

(206,(336,14))

(206,(487,14))

(209,(99,14))

(209,(144,14))

(209,(322,14))

(209,(456,14))

(209,(196,14))

(197,(99,14))

(197,(352,14))

(197,(103,14))

//Step 3: Dropping off the chapterIds and appending 1 as the value

```
val  pairRDD = joinedRDD.map( x => ((x._2._1, x._2._2),1))
```

Sample Output

((307,1),1)

((587,1),1)

((573,1),1)

((237,1),1)

((472,1),1)

((542,1),1)

((307,1),1)

((465,1),1)

((153,1),1)

((375,1),1)

((536,1),1)

((587,1),1)

((571,1),1)

((364,1),1)

//Step 4 - Count Views for User/Course -Finding out count of number of chapters a user has watched per course

```
val userPerCourseViewRDD = pairRDD.reduceByKey(_ + _)
```

Sample Output:

(UserId,CourseID),Views

((133,9),6)

((484,3),9)

((219,15),12)

((533,18),17)

((114,12),5)

((307,9),6)

((562,5),8)

((268,7),11)

//Step 5 Dropping the UserID going forward

```
 val courseViewsCountRDD = userPerCourseViewRDD.map( x =>
(x._1._2,x._2))
```

Sample Output:

(courseId, Views)

(13,7)

(7,15)

(1,6)

(16,7)

(15,12)

(6,5)

(13,7)

(13,7)

(10,6)

(4,7)

(1,25)

(14,28)

(17,7)

(13,7)

//Step-6 Join the chapterCountRDD with courseViewsCountRDD to integrate total chapters in a course

```
val  newJoinedRDD   =
courseViewsCountRDD.join(chapterCountRDD)
```

Sample Output:

```
(12,(3,5))
(12,(5,5))
(12,(4,5))
(12,(5,5))
(12,(5,5))
(12,(5,5))
(12,(5,5))
(12,(5,5))
(12,(5,5))
(12,(4,5))
(7,(11,15))
(7,(14,15))
(7,(15,15))
(7,(14,15))
(7,(11,15))
(7,(13,15))
(7,(14,15))
(5,(10,10))
(5,(9,10))
(5,(8,10))
(5,(9,10))
(5,(6,10))
(5,(10,10))
```

//Step-7 Calculating Percentage of course completion

```
 val CourseCompletionpercentRDD = newJoinedRDD.mapValues(
x => (x._1.toDouble/x._2))
```

Sample Output:

```
(10,1.0)
(10,0.6666666666666666)
(10,1.0)
(10,1.0)
(10,1.0)
(2,0.6333333333333333)
(2,0.8)
(2,0.7333333333333333)
(2,0.9)
(2,0.6333333333333333)
(2,0.9)
(2,0.9)
(2,0.6333333333333333)
(2,0.8)
```

formatting the RDD output :

```
val formattedpercentageRDD =
CourseCompletionpercentRDD.mapValues(x =>
f"$x%01.5f".toDouble)
```

Sample Output:

(courseID,percent)

(18,1.0)
(18,1.0)
(18,1.0)
(18,0.94118)
(18,1.0)
(8,1.0)
(8,0.90625)
(8,1.0)
(8,0.9375)
(8,1.0)
(8,0.96875)
(8,0.9375)

//Step-8 Map Percentages to Scores

```
  val scoresRDD = formattedpercentageRDD.mapValues (x => {

      if(x >= 0.9) 10l
      else if(x >= 0.5 && x < 0.9) 4l
      else if(x >= 0.25 && x < 0.5) 2l
      else 0l

    })
```

Sample Output:

(courseId,Score)

(4,4)
(4,10)
(4,10)
(4,4)
(4,10)
(4,10)
(4,10)
(4,4)
(4,4)
(2,4)
(2,4)
(2,4)
(2,4)
(2,4)
(2,4)
(2,4)
(2,4)
(2,4)
(2,0)
(2,10)

//Step -9 Adding up the total scores for a course

```
val totalScorePerCourseRDD =
scoresRDD.reduceByKey((V1,V2) => V1 + V2)
```

Actual Output:

(4,4176)

(16,5856)

(14,5940)

(6,4004)

(18,5940)

(8,5508)

(12,5616)

(10,5460)

(2,2202)

(13,5616)

(15,5964)

(11,5418)

(1,764)

(17,5856)

(7,4830)

(3,3376)

(9,5388)

(5,4580)

Exercise -3 Associate Titles with Courses and getting rid of courseIDs

```
val title_score_joinedRDD =
totalScorePerCourseRDD.join(titlesDataRDD).map( x =>
(x._2._1, x._2._2))
```

Output:

```
(4176,Wildfly 1)
(5856,Spring Security Module 1 )
(5940,Spring Remoting and Webservices)
(4004,Spring Security Module 3)
(5940,Microservice Deployment)
(5508,NoSQL)
(5616,Cloud Deployment)
(5460,Thymeleaf)
(2202,Java Fundamentals)
(5616,Spring Framework Fundmentals)
(5964,Spring Boot Microservices)
(5418,Android 1)
(764,HTML5)
(5856,SpringMVC)
(4830,Java Build Tools)
(3376,Wildfly 2)
(5388,Spring Boot)
(4580,Spring Security Module 2)
```

```
//Displaying courses starting with the most popular course

val popularCoursesRDD =
title_score_joinedRDD.sortByKey(false)

popularCoursesRDD.collect.foreach(println)
```

Final Output:

(5964,Spring Boot Microservices)

(5940,Spring Remoting and Webservices)

(5940,Microservice Deployment)

(5856,Spring Security Module 1)

(5856,SpringMVC)

(5616,Cloud Deployment)

(5616,Spring Framework Fundmentals)

(5508,NoSQL)

(5460,Thymeleaf)

(5418,Android 1)

(5388,Spring Boot)

(4830,Java Build Tools)

(4580,Spring Security Module 2)

(4176,Wildfly 1)

(4004,Spring Security Module 3)

(3376,Wildfly 2)

(2202,Java Fundamentals)

(764,HTML5)

****