Spark Assignment 18.1

Given the below 3 dataset

1) S18_Dataset_Holidays.txt as a text file (user_id, src, dest, travel_mode, distance, year_of_travel)
2) S18_Dataset_Transport.txt as a text file (travel_mode, cost_per_unit)
3) S18_Dataset_User_details.txt as a text file (user_id, src, dest, travel_mode, distance, year_of_travel)

Solve the below mentioned problem statement in spark rdd.

This assignment is done in the spark shell within the acadgildVM.

Steps Followed:

To Solve the problems first I've combined the datas from the above 3 files to one, using the common keys. Used travel_mode as a key to combine transport and holidays data. Used user_id to combine user_details and holidays.

1) Copied the dataset file in the path /home/acadgild/spark/S18_Dataset_Holidays.txt Then read the text file by using sc.textfile as below.

   ```
   val holidays = sc.textFile("/home/acadgild/spark/S18_Dataset_Holidays.txt")
   ```

2) Then created a holiday_tuple rdd by using map function over the holidays rdd with travel_mode as key. Then created a parallelized collection by using parallelize method over the mapped tuple.

   ```
   val holiday_tuple = sc.parallelize(holidays.map(line => line.split(",")).map(x => ((x(3)), (x(0).toInt,x(1), x(2),x(4).toInt,x(5).toInt))).collect)
   ```

Spark-shell output

```
scala> val holidays = sc.textFile("/home/acadgild/spark/S18_Dataset_Holidays.txt")
holidays: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/S18_Dataset_Holidays.txt MapPartitionsRDD[112] at textFi
le at <console>:24

scala> val holiday_tuple =  sc.parallelize(holidays.map(line => line.split(",")).map(x => ((x(3)), (x(0).toInt,x(1), x(2),
x(4).toInt,x(5).toInt))).collect)
holiday_tuple: org.apache.spark.rdd.RDD[(String, (Int, String, String, Int, Int))] = ParallelCollectionRDD[115] at paralle
lize at <console>:26
```

3) Copied the dataset file in the path /home/acadgild/spark/S18_Dataset_ Transport.txt Then read the text file by using sc.textfile as below.

val transport = sc.textFile("/home/acadgild/spark/S18_Dataset_Transport.txt")

4) Then created a transport_tuple rdd by using map function over the transport rdd with travel_mode as key. Then created a parallelized collection by using parallelize method over the mapped tuple.

val transport_tuple = sc.parallelize(transport.map(line => line.split(",")).map(x => ((x(0)), (x(1).toInt))).collect)

Spark-shell output

```
scala> val transport = sc.textFile("/home/acadgild/spark/S18_Dataset_Transport.txt")
transport: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/S18_Dataset_Transport.txt MapPartitionsRDD[117] at text
File at <console>:24

scala> val transport_tuple =  sc.parallelize(transport.map(line => line.split(",")).map(x => ((x(0)), (x(1).toInt))).colle
ct)
transport_tuple: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[120] at parallelize at <console>:26
```

---

5) Then using join function joined the holiday_tuple and transport_tuple as below as both had travel_mode as key.From the transport_tuple taken the cost_per unit and multiplied with distance, to calculate the expenses for each of the travel.

val holiday_trans = holiday_tuple.join(transport_tuple).map {

case ((travel_mode),((user_id,src,dest,distance,year_of_travel),cost_per_unit)) => (user_id,src,dest,travel_mode,distance,year_of_travel,cost_per_unit *distance)

}

Spark-shell output

```
scala> val holiday_trans = holiday_tuple.join(transport_tuple).map {
     |   case ((travel_mode),((user_id,src,dest,distance,year_of_travel),cost_per_unit)) => (user_id,src,dest,travel_mode,
distance,year_of_travel,cost_per_unit *distance  )
     | }
holiday_trans: org.apache.spark.rdd.RDD[(Int, String, String, String, Int, Int, Int)] = MapPartitionsRDD[124] at map at <c
onsole>:32

scala> val travel_details = sc.parallelize(holiday_trans.map(x =>(((x._1)), ((x._2),(x._3),(x._4),(x._5),(x._6),(x._7))))).
collect)
travel_details: org.apache.spark.rdd.RDD[(Int, (String, String, String, Int, Int, Int))] = ParallelCollectionRDD[126] at p
arallelize at <console>:34
```

6) Next,To combine the holiday_trans with user data,I've modified the holiday_trans and kept the key as user_id

```
val travel_details = sc.parallelize(holiday_trans.map(x =>((x._1)), ((x._2),(x._3),(x._4),(x._5),(x._6),(x._7)))).collect)
```

Spark-shell output

```
scala> val travel_details = sc.parallelize(holiday_trans.map(x =>(((x._1)), ((x._2),(x._3),(x._4),(x._5),(x._6),(x._7))))).
collect)
travel_details: org.apache.spark.rdd.RDD[(Int, (String, String, String, Int, Int, Int))] = ParallelCollectionRDD[126] at p
arallelize at <console>:34
```

7) Copied the dataset file in the path /home/acadgild/spark/ S18_Dataset_User_details .Then read the text file by using sc.textfile as below.

```
val user_details = sc.textFile("/home/acadgild/spark/S18_Dataset_User_details.txt")
```

8) Then created a user_tuple rdd by using map function over the user_details rdd with user_id as key. Then created a parallelized collection by using parallelize method over the mapped tuple.

```
val user_tuple = sc.parallelize(user_details.map(line => line.split(",")).map(x => ((x(0).toInt), (x(1), x(2).toInt))).collect)
```

Spark-shell output

```
scala> val user_details = sc.textFile("/home/acadgild/spark/S18_Dataset_User_details.txt")
user_details: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/S18_Dataset_User_details.txt MapPartitionsRDD[128] a
t textFile at <console>:24

scala> val user_tuple = sc.parallelize(user_details.map(line => line.split(",")).map(x => ((x(0).toInt), (x(1), x(2).toInt
))).collect)
user_tuple: org.apache.spark.rdd.RDD[(Int, (String, Int))] = ParallelCollectionRDD[131] at parallelize at <console>:26
```

---

9) Then using join function joined the user_tuple and travel_details as below as both had user_id as key.

```
val user_travel = user_tuple.join(travel_details).map {

 case (user_id,((name,age),(src,dest,travel_mode,distance,year_of_travel,charges)))
=> (user_id,name,age,src,dest,travel_mode,distance,year_of_travel,charges )

}
```

The user_travel tuple, will have items in the following order

user_id,name,age,src,dest,travel_mode,distance,year_of_travel,charges

user_id ->1,

name -> 2,

age ->3,

src ->4,

dest -> 5,

travel_mode -> 6,

distance -> 7,

year_of_travel -> 8,

charges -> 9

10) The above Rdd, user_travel is the final tuple, which is a combination of user_details, transport and holidays. It will be used to solve multiple problems, so we are storing it in the memory.

```
import org.apache.spark.storage.StorageLevel

user_travel.persist(StorageLevel.MEMORY_ONLY)
```

Spark-shell output

```
scala> val user_travel = user_tuple.join(travel_details).map {
     |   case (user_id,((name,age),(src,dest,travel_mode,distance,year_of_travel,charges))) => (user_id,name,age,src,dest,
travel_mode,distance,year_of_travel,charges )
     | }
user_travel: org.apache.spark.rdd.RDD[(Int, String, Int, String, String, String, Int, Int, Int)] = MapPartitionsRDD[135] a
t map at <console>:40

scala> import org.apache.spark.storage.StorageLevel
import org.apache.spark.storage.StorageLevel

scala> user_travel.persist(StorageLevel.MEMORY_ONLY)
res36: user_travel.type = MapPartitionsRDD[135] at map at <console>:40
```

Problem Statement:

1) What is the distribution of the total number of air-travelers per year.

To find the total number of air-travellers per year, we are calling the filter function over the user_travel, 6th element, travel_mode which equals airplane.

```
var air_traveller = user_travel.filter(x=> (x._6).equals("airplane"))
```

Once filtered, we are calling the map function over the air_traveller, creating a key with tuplerdds 8th element which refers to the year_of travel and setting 1 as value for each item.

Then calling the reduceBy key, where the values for each key are aggregated using the given reduce function. Here we are adding the values for each key and finally calling the collect action.

```
var total_traveller_per_year= air_traveller.map(x=> (((x._8)),1)).reduceByKey((x,y) => (x + y)).collect
```

Output

total_traveller_per_year: Array[(Int, Int)] = Array((1994,1), (1992,7), (1990,8), (1991,9), (1993,7))

Spark-shell output

```
scala> var air_traveller = user_travel.filter(x=> (x._6).equals("airplane"))
air_traveller: org.apache.spark.rdd.RDD[(Int, String, Int, String, String, String, Int, Int, Int)] = MapPartitionsRDD[140]
 at filter at <console>:43

scala> var total_traveller_per_year= air_traveller.map(x=> (((x._8)),1)).reduceByKey((x,y) => (x + y)).collect
total_traveller_per_year: Array[(Int, Int)] = Array((1994,1), (1992,7), (1990,8), (1991,9), (1993,7))
```

---

2) What is the total air distance covered by each user per year.,.

To find the the total number of air-travellers per year, we are calling the map function over the user_travel, creating a key with user_travel rdds 1st, 2nd and 8th element which refers to the, id, name, year_of travel, and setting user_travel rdds 7th element, distance as value for each item.

Then calling the reduceBy key, where the values for each key are aggregated using the given reduce function. Here we are adding the values for each key and finally calling the collect action.

```
var travel_each_user_per_year= user_travel.map(x=>
(((x._1),(x._2),(x._8)),(x._7))).reduceByKey((x,y) => (x + y)).collect
```

Output

travel_each_user_per_year: Array[((Int, String, Int), Int)] = Array(((8,andrew,1992),200), ((4,lisa,1991),200), ((10,annie,1992),200), ((10,annie,1993),200), ((3,luke,1991),200), ((1,mark,1990),200), ((3,luke,1992),200), ((5,mark,1992),400), ((7,james,1990),600), ((6,peter,1991),400), ((3,luke,1993),200), ((2,john,1993),200), ((4,lisa,1990),400), ((5,mark,1991),200), ((9,thomas,1991),200), ((5,mark,1994),200), ((8,andrew,1990),200), ((2,john,1991),400), ((8,andrew,1991),200), ((10,annie,1990),200), ((9,thomas,1992),400), ((6,peter,1993),200), ((1,mark,1993),600))

Spark-shell ouput

```
scala> var travel_each_user_per_year= user_travel.map(x=> (((x._1),(x._2),(x._8)),(x._7))).reduceByKey((x,y) => (x + y)).c
ollect
travel_each_user_per_year: Array[((Int, String, Int), Int)] = Array(((8,andrew,1992),200), ((4,lisa,1991),200), ((10,annie
,1992),200), ((10,annie,1993),200), ((3,luke,1991),200), ((1,mark,1990),200), ((3,luke,1992),200), ((5,mark,1992),400), ((
7,james,1990),600), ((6,peter,1991),400), ((3,luke,1993),200), ((2,john,1993),200), ((4,lisa,1990),400), ((5,mark,1991),20
0), ((9,thomas,1991),200), ((5,mark,1994),200), ((8,andrew,1990),200), ((2,john,1991),400), ((8,andrew,1991),200), ((10,an
nie,1990),200), ((9,thomas,1992),400), ((6,peter,1993),200), ((1,mark,1993),600))
```

3) *Which user has travelled the largest distance till date.*

To find which user has travelled the largest distance till date., we are calling the map function over the user_travel, creating a key with user_travel rdds 1st, 2nd element which refers to the, id, name and setting user_travel rdds 7th element, distance as value for each item.

Then calling the reduceBy key, where the values for each key are aggregated using the given reduce function. Here we are adding the values for each key and finally calling the collect action.

```
var most_travelled= user_travel.map(x=> (((x._1),(x._2)),(x._7))).reduceByKey((x,y) => (x +
y)).collect
```

The above rdd, will give the userid, name, total distance travelled. In order to find the user who travelled largest distance, I've called the maxBy function over the value item of the most_travelled.

```
var most_travelled_res = sc.parallelize(most_travelled).collect.maxBy(_._2)
```

Output

most_travelled_res: ((Int, String), Int) = ((1,mark),800)

Spark-shell ouput

```
scala> var most_travelled= user_travel.map(x=> (((x. 1),(x. 2)),(x. 7))).reduceByKey((x,y) => (x + y)).collect
most_travelled: Array[((Int, String), Int)] = Array(((3,luke),600), ((1,mark),800), ((7,james),600), ((6,peter),600), ((5,
mark),800), ((4,lisa),600), ((8,andrew),600), ((10,annie),600), ((9,thomas),600), ((2,john),600))

scala> var most_travelled_res = sc.parallelize(most_travelled).collect.maxBy(_._2)
most_travelled_res: ((Int, String), Int) = ((1,mark),800)
```

4) What is the most preferred destination for all users..

To find the most preferred destination for all users, we are calling the map function over the user_travel, creating a key with tuplerdds 5th element which refers to the destination and setting 1 as value for each item.

Then calling the reduceBy key, where the values for each key are aggregated using the given reduce function. Here we are adding the values for each key and finally calling the collect action.

```
var preferred_dest= user_travel.map(x=> (((x._5)),1)).reduceByKey((x,y) => (x + y)).collect
```

The above rdd, will give the destination name and number of times the destination travelled. In order to find the most travelled destination, I've called the maxBy function over the value item of the preferred_dest.

```
var preferred_dest_res = sc.parallelize(preferred_dest).collect.maxBy(_._2)
```

Output

preferred_dest_res: (String, Int) = (IND,9)

Spark-shell output

```
scala> var preferred_dest= user_travel.map(x=> (((x. 5)),1)).reduceByKey((x,y) => (x + y)).collect
preferred_dest: Array[(String, Int)] = Array((CHN,7), (IND,9), (PAK,5), (RUS,6), (AUS,5))

scala> var preferred_dest_res = sc.parallelize(preferred_dest).collect.maxBy(_._2)
preferred_dest_res: (String, Int) = (IND,9)
```
