Assignment - 20.1

Below are the datasets which will be used in this assignment, right now placed in local file system as mentioned below.

```
-rw-rw-r--.  1 acadgild acadgild        932 May 25 23:13 S20_Dataset_Holidays.txt
-rw-rw-r--.  1 acadgild acadgild         45 May 25 23:13 S20_Dataset_Transport.txt
-rw-rw-r--.  1 acadgild acadgild        119 May 25 23:13 S20_Dataset_User_details.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

```
[acadgild@localhost ~]$ cat S20_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
```

```
[acadgild@localhost ~]$ cat S20_Dataset_Transport.txt
airplane,170
car,140
train,120
ship,200You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

```
[acadgild@localhost ~]$ cat S20_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[acadgild@localhost ~]$
```

Uploading the dataset into a RDD.

```
val baseRDD = sc.textFile("file:///home/acadgild/S20_Dataset_Holidays.txt")

import org.apache.spark.storage.StorageLevel

baseRDD.persist(StorageLevel.MEMORY_ONLY)
```

```
scala> val baseRDD = sc.textFile("/home/acadgild/S20_Dataset_Holidays.txt")
baseRDD: org.apache.spark.rdd.RDD[String] = /home/acadgild/S20_Dataset_Holidays.txt Ma

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

scala> baseRDD.persist(StorageLevel.MEMORY_ONLY)
res0: baseRDD.type = /home/acadgild/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at te

scala>
```

```
scala> baseRDD.foreach(println)
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
```

Task1: What is the distribution of the total number of air-travelers per year?

Answer:

```
val res1 = baseRDD.map(x => (x.split(",")(5).toInt,1))

val res2 = res1.reduceByKey((x,y) => (x + y))

res2.foreach(println)
```

Output:

```
scala> val res1 = baseRDD.map(x => (x.split(",")(5).toInt,1))
res1: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[6] at map at <console>:28

scala> res1.collect
res6: Array[(Int, Int)] = Array((1990,1), (1991,1), (1992,1), (1990,1), (1992,1), (1991,1), (1990,1), (1991,1), (1992,1), (1993,1), (1
993,1), (1993,1), (1993,1), (1991,1), (1992,1), (1993,1), (1990,1), (1990,1), (1991,1), (1992,1), (1993,1), (1991,1), (1991,1), (1990,
1), (1991,1), (1991,1), (1990,1), (1992,1), (1992,1), (1990,1), (1993,1), (1994,1))

scala> val res2 = res1.reduceByKey((x,y) => (x + y))
res2: org.apache.spark.rdd.RDD[(Int, Int)] = ShuffledRDD[7] at reduceByKey at <console>:30
```

```
scala> res2.collect
res7: Array[(Int, Int)] = Array((1994,1), (1992,7), (1990,8), (1991,9), (1993,7))

scala> res2.foreach(println)
(1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)

scala>
```

Task2: What is the total air distance covered by each user per year

Answer:

val splitRDD = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))

val distRDD = splitRDD.reduceByKey((x,y) => (x +y))

distRDD.foreach(println)

```
scala> val splitRDD = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
splitRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[8] at map at <console>:28

scala> val distRDD = splitRDD.reduceByKey((x,y) => (x +y))
distRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[9] at reduceByKey at <console>:30

scala> distRDD.foreach(println)
((3,1992),200)
((3,1993),200)
((5,1991),200)
((6,1991),400)
((10,1993),200)
((5,1992),400)
((8,1991),200)
((8,1990),200)
((1,1993),600)
((5,1994),200)
((2,1993),200)
((2,1991),400)
((4,1990),400)
((10,1992),200)
((3,1991),200)
((1,1990),200)
((10,1990),200)
((6,1993),200)
((9,1992),400)
((8,1992),200)
((7,1990),600)
((9,1991),200)
((4,1991),200)
```

Task3: Which user has travelled the largest distance till date

Answer:

val userRDD = baseRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))

val totaldistRDD = userRDD.reduceByKey((x,y) => (x+y))

val maxRDD = totaldistRDD.takeOrdered(1)

maxRDD.foreach(println)

```
scala> val userRDD = baseRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))
userRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[10] at map at <console>:28

scala> val totaldistRDD = userRDD.reduceByKey((x,y) => (x+y))
totaldistRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at reduceByKey at <console>:30

scala> val maxRDD = totaldistRDD.takeOrdered(1)
maxRDD: Array[(String, Int)] = Array((1,800))

scala> maxRDD.foreach(println)
(1,800)

scala>
```

Task4: What is the most preferred destination for all users.

Answer:

val destRDD = baseRDD.map(x => (x.split(",")(2),1))

val destreduceRDD = destRDD.reduceByKey((x,y) => (x + y))

val maxRDD = destreduceRDD.takeOrdered(1)(Ordering[Int].reverse.on(_._2))

maxRDD.foreach(println)

```
scala> val destRDD = baseRDD.map(x => (x.split(",")(2),1))
destRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[13] at map at <console>:28

scala> val destreduceRDD = destRDD.reduceByKey((x,y) => (x + y))
destreduceRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[14] at reduceByKey at <console>:30

scala> val maxRDD = destreduceRDD.takeOrdered(1)(Ordering[Int].reverse.on(_._2))
maxRDD: Array[(String, Int)] = Array((IND,9))

scala> maxRDD.foreach(println)
(IND,9)

scala>
```

**********Loading Other files to Spark **********

val baseRDD1 = sc.textFile("file:///home/acadgild/S20_Dataset_Holidays.txt")

val baseRDD2 = sc.textFile("file:///home/acadgild/S20_Dataset_Transport.txt")

val baseRDD3 = sc.textFile("file:///home/acadgild/S20_Dataset_User_details.txt")

import org.apache.spark.storage.StorageLevel

baseRDD1.persist(StorageLevel.MEMORY_ONLY)

baseRDD2.persist(StorageLevel.MEMORY_ONLY)

baseRDD3.persist(StorageLevel.MEMORY_ONLY)

```
scala> val baseRDD1 = sc.textFile("file:///home/acadgild/S20_Dataset_Holidays.txt")
baseRDD1: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/S20_Dataset_Holidays.txt MapPartitionsRDD[13] at textFile at <conso
le>:24

scala> val baseRDD2 = sc.textFile("file:///home/acadgild/S20_Dataset_Transport.txt")
baseRDD2: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/S20_Dataset_Transport.txt MapPartitionsRDD[15] at textFile at <cons
ole>:24

scala> val baseRDD3 = sc.textFile("file:///home/acadgild/S20_Dataset_User_details.txt")
baseRDD3: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/S20_Dataset_User_details.txt MapPartitionsRDD[17] at textFile at <c
onsole>:24
```

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

scala> baseRDD1.persist(StorageLevel.MEMORY_ONLY)
res5: baseRDD1.type = file:///home/acadgild/S20_Dataset_Holidays.txt MapPartitionsRDD[13] at textFile at <console>:24

scala> baseRDD2.persist(StorageLevel.MEMORY_ONLY)
res6: baseRDD2.type = file:///home/acadgild/S20_Dataset_Transport.txt MapPartitionsRDD[15] at textFile at <console>:24

scala> baseRDD3.persist(StorageLevel.MEMORY_ONLY)
res7: baseRDD3.type = file:///home/acadgild/S20_Dataset_User_details.txt MapPartitionsRDD[17] at textFile at <console>:24
```

Task5: Which route is generating the most revenue per year?

Answer:

val travel = baseRDD1.map(x =>
(x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5).toInt))

val transport = baseRDD2.map(x => (x.split(",")(0),x.split(",")(1).toInt))

val user = baseRDD3.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))

val travelmap = travel.map(x=> x._4 -> (x._2,x._5,x._6))

val transportmap = transport.map(x=> x._1 -> x._2)

val join1 = travelmap.join(transportmap)

```
scala> val join1 = travelmap.join(transportmap)
join1: org.apache.spark.rdd.RDD[(String, ((String, Int, Int), Int))] = MapPartitionsRDD[25] at join at <console>:37

scala> join1.foreach(println)
(airplane,((CHN,200,1990),170))
(airplane,((IND,200,1991),170))
(airplane,((IND,200,1992),170))
(airplane,((RUS,200,1990),170))
(airplane,((CHN,200,1992),170))
(airplane,((AUS,200,1991),170))
(airplane,((RUS,200,1990),170))
(airplane,((IND,200,1991),170))
(airplane,((CHN,200,1992),170))
(airplane,((AUS,200,1993),170))
(airplane,((AUS,200,1993),170))
(airplane,((CHN,200,1993),170))
(airplane,((CHN,200,1993),170))
(airplane,((IND,200,1991),170))
(airplane,((AUS,200,1992),170))
(airplane,((RUS,200,1993),170))
(airplane,((CHN,200,1990),170))
(airplane,((AUS,200,1990),170))
(airplane,((IND,200,1991),170))
(airplane,((RUS,200,1992),170))
(airplane,((PAK,200,1993),170))
```

val routeMap = join1.map(x => (x._2._1._1 -> x._2._1._3) -> (x._2._1._2 *x._2._2))

```
scala> val routeMap = join1.map(x => (x._2._1._1 -> x._2._1._3) -> (x._2._1._2 *x._2._2))
routeMap: org.apache.spark.rdd.RDD[((String, Int), Int)] = MapPartitionsRDD[26] at map at <console>:3

scala> routeMap.foreach(println)
((CHN,1990),34000)
((IND,1991),34000)
((IND,1992),34000)
((RUS,1990),34000)
((CHN,1992),34000)
((AUS,1991),34000)
((RUS,1990),34000)
((IND,1991),34000)
((CHN,1992),34000)
((AUS,1993),34000)
((AUS,1993),34000)
((CHN,1993),34000)
((CHN,1993),34000)
((IND,1991),34000)
((AUS,1992),34000)
((RUS,1993),34000)
((CHN,1990),34000)
((AUS,1990),34000)
((IND,1991),34000)
((RUS,1992),34000)
((PAK,1993),34000)
```

val costsum = routeMap.groupByKey().map(x => x._2.sum -> x._1)

```
scala> val costsum = routeMap.groupByKey().map(x => x._2.sum -> x._1)
costsum: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[28] at map at <console>:41

scala> costsum.foreach(println)
(102000,(RUS,1992))
(68000,(AUS,1993))
(170000,(CHN,1990))
(34000,(RUS,1993))
(34000,(AUS,1991))
(68000,(RUS,1990))
(34000,(IND,1992))
(204000,(IND,1991))
(34000,(AUS,1990))
(34000,(CHN,1994))
(34000,(CHN,1991))
(34000,(AUS,1992))
(68000,(CHN,1992))
(68000,(CHN,1993))
(34000,(PAK,1991))
(68000,(PAK,1993))
```

val sortRevenue = costsum.sortByKey(false).first()

```
scala> val sortRevenue = costsum.sortByKey(false).first()
sortRevenue: (Int, (String, Int)) = (204000,(IND,1991))

scala>
```

Task6: What is the total amount spent by every user on air-travel per year?

Answer:

val userMap = travel.map(x => x._4 -> (x._1,x._5,x._6))

```
scala> val userMap = travel.map(x => x._4 -> (x._1,x._5,x._6))
userMap: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[30] at map at <console>:29

scala> userMap.foreach(println)
(airplane,(1,200,1990))
(airplane,(2,200,1991))
(airplane,(3,200,1992))
(airplane,(4,200,1990))
(airplane,(5,200,1992))
(airplane,(6,200,1991))
(airplane,(7,200,1990))
(airplane,(8,200,1991))
(airplane,(9,200,1992))
```

val amtMap = userMap.join(transportmap)

```
scala> val amtMap = userMap.join(transportmap)
amtMap: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[33] at join at <console>:37

scala> amtMap.foreach(println)
(airplane,((1,200,1990),170))
(airplane,((2,200,1991),170))
(airplane,((3,200,1992),170))
(airplane,((4,200,1990),170))
(airplane,((5,200,1992),170))
(airplane,((6,200,1991),170))
(airplane,((7,200,1990),170))
(airplane,((8,200,1991),170))
(airplane,((9,200,1992),170))
(airplane,((10,200,1993),170))
(airplane,((1,200,1993),170))
```

val spendMap = amtMap.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))

```
scala> val spendMap = amtMap.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))
spendMap: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[34] at map at <console>:39

scala> spendMap.foreach(println)
((1,1990),34000)
((2,1991),34000)
((3,1992),34000)
((4,1990),34000)
((5,1992),34000)
((6,1991),34000)
((7,1990),34000)
((8,1991),34000)
((9,1992),34000)
((10,1993),34000)
((1,1993),34000)
((2,1993),34000)
((3,1993),34000)
((4,1991),34000)
((5,1992),34000)
((6,1993),34000)
((7,1990),34000)
((8,1990),34000)
((9,1991),34000)
((10,1992),34000)
((1,1993),34000)
```

val total = spendMap.groupByKey().map(x => x._1 -> x._2.sum)

```
scala> val total = spendMap.groupByKey().map(x => x._1 -> x._2.sum)
total: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[36] at map at <console>:41

scala> total.foreach(println)
((2,1993),34000)
((6,1993),34000)
((10,1993),34000)
((10,1992),34000)
((2,1991),68000)
((4,1990),68000)
((10,1990),34000)
((5,1992),68000)
((4,1991),34000)
((1,1993),102000)
((9,1992),68000)
((5,1991),34000)
((3,1993),34000)
((1,1990),34000)
((8,1990),34000)
((7,1990),102000)
((6,1991),68000)
((5,1994),34000)
((3,1991),34000)
((9,1991),34000)
((3,1992),34000)
((8,1991),34000)
((8,1992),34000)

scala>
```

Above output shows the total amount spent by every user on air-travel per year.

Task7: Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

Answer:

val AgeMap = user.map(x => x._1 -> {if(x._3<20) "20" else if(x._3>35) "35" else "20-35" })

```
scala> val AgeMap = user.map(x => x._1 -> {if(x._3<20) "20" else if(x._3>35) "35" else "20-35" })
AgeMap: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[37] at map at <console>:29

scala> AgeMap.foreach(println)
(1,20)
(2,20)
(3,20)
(4,20-35)
(5,20-35)
(6,20-35)
(7,20-35)
(8,35)
(9,35)
(10,35)
```

val UIDMap = travel.map(x => x._1 -> 1)

```
scala> val UIDMap = travel.map(x => x._1 -> 1)
UIDMap: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[39] at map at <console>:29

scala> UIDMap.foreach(println)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(6,1)
(7,1)
```

val joinMap = AgeMap.join(UIDMap)

```
scala> val joinMap = AgeMap.join(UIDMap)
joinMap: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[42] at join at <console>:37

scala> joinMap.foreach(println)
(4,(20-35,1))
(4,(20-35,1))
(4,(20-35,1))
(1,(20,1))
(1,(20,1))
(1,(20,1))
```

val joinMap2 = joinMap.map(x => x._2._1 -> x._2._2)

```
scala> val joinMap2 = joinMap.map(x => x._2._1 -> x._2._2)
joinMap2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[43] at map at <console>:39

scala> joinMap2.foreach(println)
(20-35,1)
(20-35,1)
(20-35,1)
(20,1)
(20,1)
(20,1)
(20,1)
(20-35,1)
```

val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)

```
scala> val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)
groupKey: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[45] at map at <console>:41

scala> groupKey.foreach(println)
(20,10)
(20-35,13)
(35,9)
```

val maxVal = groupKey.sortBy(x => -x._2).first()

```
scala> val maxVal = groupKey.sortBy(x => -x._2).first()
maxVal: (String, Int) = (20-35,13)

scala>
```

Hence we could see that the age group 20 – 35 is the one which travels the most through the year.