SPARK ASSIGNMENT 23.2

Author--Chhaya Yadav

~~~~~~

Aviation data analysis

The U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics (BTS) tracks the on-time performance of domestic flights operated by large air carriers. Summary information on the number of on-time, delayed, cancelled, and diverted flights appears in DOT's monthly Air Travel Consumer Report, published about 30 days after the month's end, as well as in summary tables posted on this website. Summary statistics and raw data are made available to the public at the time the Air Travel Consumer Report is released.

You can download the datasets from the following links:

Delayed_Flights.csv

Delayed_Flights.csv Datasets

There are 29 columns in this dataset. Some of them have been mentioned below:

Year: 1987 - 2008

Month: 1 - 12

FlightNum: Flight number

Canceled: Was the flight canceled?

CancelleationCode: The reason for cancellation.

For complete details, refer to this link.

==

Problem Statement 1

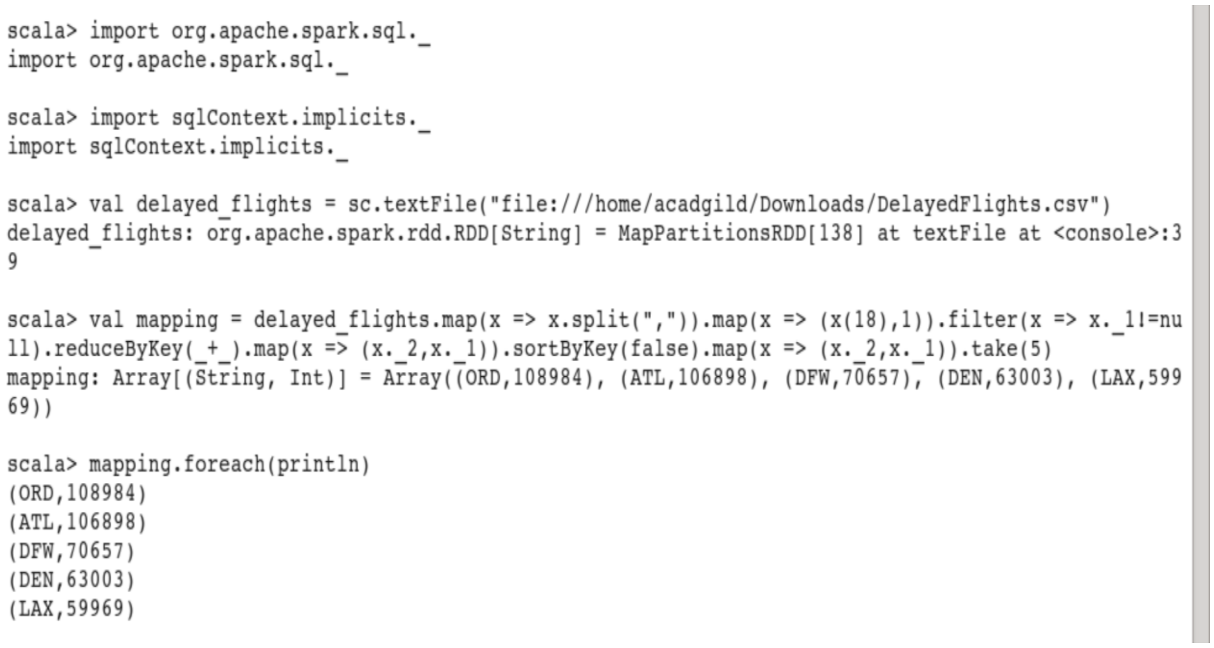
Find out the top 5 most visited destinations.

~~~~~~

```
//Importing the SPARK SQL Packages

import org.apache.spark.sql._

import sqlContext.implicits._

//Reading the DelayedFlights CSV file into an RDD
```

```
val delayed_flights =
sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")

val mapping = delayed_flights.map(x => x.split(",")).map(x =>
(x(18),1)).filter(x => x._1!=null).reduceByKey(_+_).map(x =>
(x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(5)

mapping.foreach(println)

mapping.collect()
```

SCREENSHOTS:



```
scala> import org.apache.spark.sql._
import org.apache.spark.sql._

scala> import sqlContext.implicits._
import sqlContext.implicits._

scala> val delayed_flights = sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[138] at textFile at <console>:3
9

scala> val mapping = delayed_flights.map(x => x.split(",")).map(x => (x(18),1)).filter(x => x._1!=nu
ll).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(5)
mapping: Array[(String, Int)] = Array((ORD,108984), (ATL,106898), (DFW,70657), (DEN,63003), (LAX,599
69))

scala> mapping.foreach(println)
(ORD,108984)
(ATL,106898)
(DFW,70657)
(DEN,63003)
(LAX,59969)
```

==

Problem Statement 2

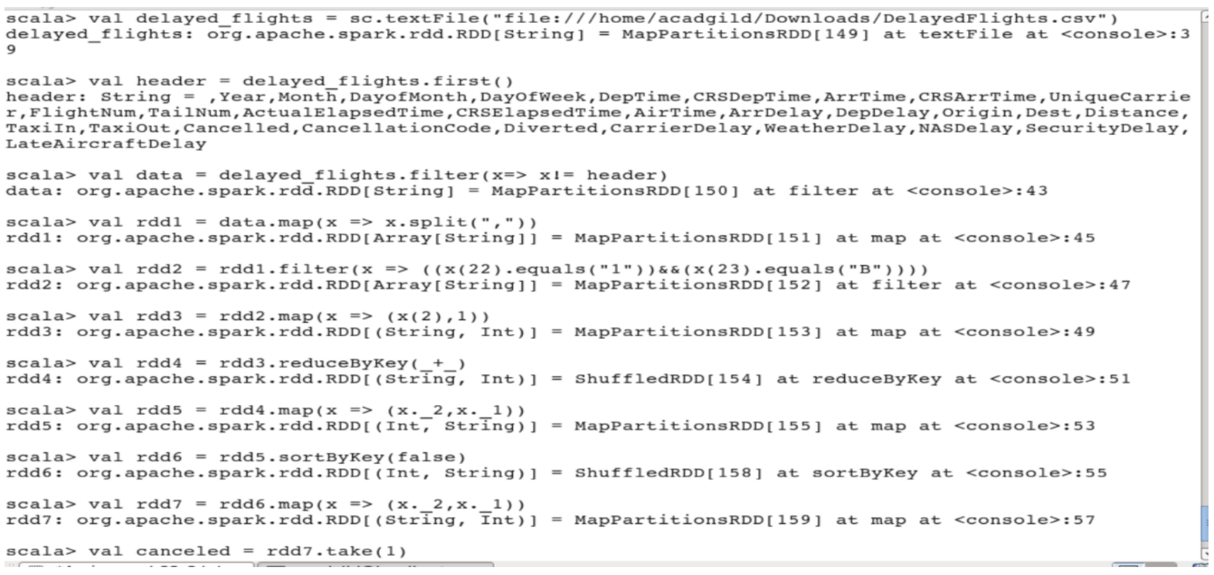
Which month has seen the most number of cancellations due to bad weather?

~~~~~~

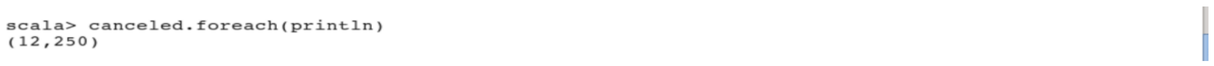
```
//Reading the DelayedFlights CSV file into an RDD

val delayed_flights =
sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")

//Fetching the Header into an RDD

val header = delayed_flights.first()

//Removing the header from the RDD with the filter operation

val data = delayed_flights.filter(x=> x!= header)

//Splitting the elements with comma delimiter
```
~~~~~~

```
val rdd1 = data.map(x => x.split(","))

//Filtering various columns with their values

val rdd2 = rdd1.filter(x =>
((x(22).equals("1"))&&(x(23).equals("B"))))

//Putting the third column as key and 1 as value

val rdd3 = rdd2.map(x => (x(2),1))

//Summing up the count of each key

val rdd4 = rdd3.reduceByKey(_+_)

val rdd5 = rdd4.map(x => (x._2,x._1))

val rdd6 = rdd5.sortByKey(false)

val rdd7 = rdd6.map(x => (x._2,x._1))

//Selecting first record from RDD

val canceled = rdd7.take(1)

//Displaying the expected output

canceled.foreach(println)
```

SCREENSHOTS:



```
scala> val delayed_flights = sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[149] at textFile at <console>:3
9

scala> val header = delayed_flights.first()
header: String = ,Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrie
r,FlightNum,TailNum,ActualElapsedTime,CRSElapsedTime,AirTime,ArrDelay,DepDelay,Origin,Dest,Distance,
TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,CarrierDelay,WeatherDelay,NASDelay,SecurityDelay,
LateAircraftDelay

scala> val data = delayed_flights.filter(x=> x!= header)
data: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[150] at filter at <console>:43

scala> val rdd1 = data.map(x => x.split(","))
rdd1: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[151] at map at <console>:45

scala> val rdd2 = rdd1.filter(x => ((x(22).equals("1"))&&(x(23).equals("B"))))
rdd2: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[152] at filter at <console>:47

scala> val rdd3 = rdd2.map(x => (x(2),1))
rdd3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[153] at map at <console>:49

scala> val rdd4 = rdd3.reduceByKey(_+_)
rdd4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[154] at reduceByKey at <console>:51

scala> val rdd5 = rdd4.map(x => (x._2,x._1))
rdd5: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[155] at map at <console>:53

scala> val rdd6 = rdd5.sortByKey(false)
rdd6: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[158] at sortByKey at <console>:55

scala> val rdd7 = rdd6.map(x => (x._2,x._1))
rdd7: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[159] at map at <console>:57

scala> val canceled = rdd7.take(1)
```



```
scala> canceled.foreach(println)
(12,250)
```

===

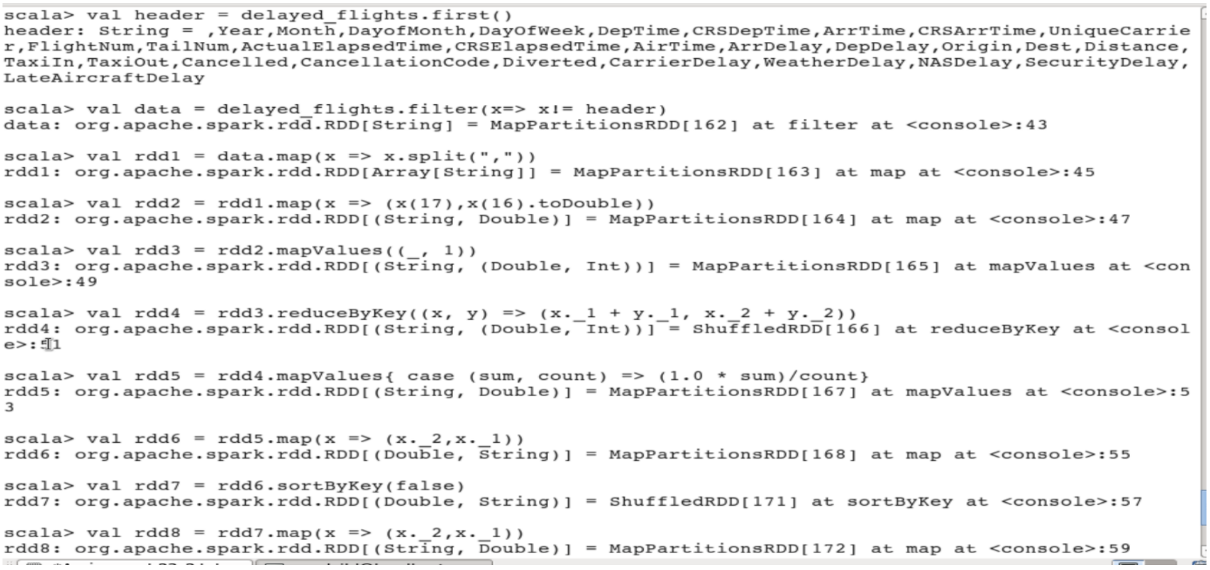
Problem Statement 3

Top ten origins with the highest AVG departure delay

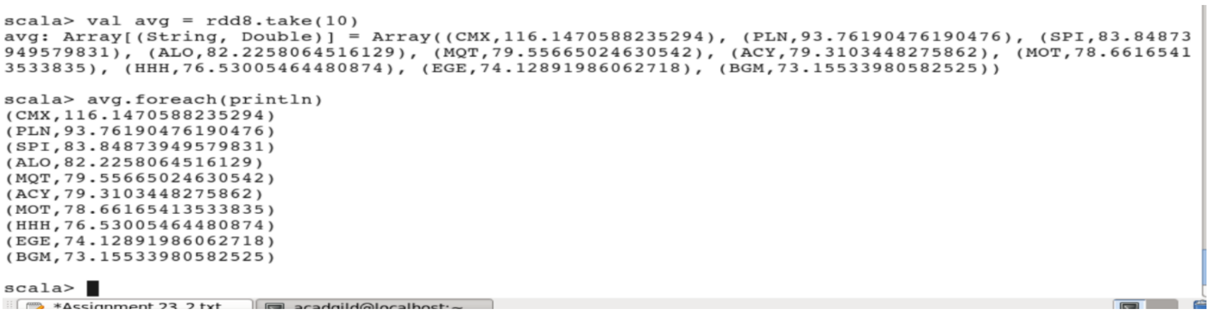
~~~~~~
~~~~~~

```
//Reading the DelayedFlights CSV file into an RDD

val delayed_flights =
sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")

//Selecting the first line of RDD into a new RDD

val header = delayed_flights.first()

//Removing the header from the RDD for data manipulation

val data = delayed_flights.filter(x=> x!= header)

//Splitting the elements with comma delimiter

val rdd1 = data.map(x => x.split(","))

//Selecting only column 17 and 18 in RDD and conversion of column 17
to Double

val rdd2 = rdd1.map(x => (x(17),x(16).toDouble))

//Mapping the data into Key, Value Pairs

val rdd3 = rdd2.mapValues((_, 1))

//Generating the total sum and count respectively

val rdd4 = rdd3.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))

//Calculation of Average with the help of Sum and Count calculated
above

val rdd5 = rdd4.mapValues{ case (sum, count) => (1.0 * sum)/count}

val rdd6 = rdd5.map(x => (x._2,x._1))

val rdd7 = rdd6.sortByKey(false)

val rdd8 = rdd7.map(x => (x._2,x._1))

//Selecting top 10 Values from RDD

val avg = rdd8.take(10)

//Displaying the expected output

avg.foreach(println)
```

SCREENSHOTS:



```
scala> val header = delayed_flights.first()
header: String = ,Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrie
r,FlightNum,TailNum,ActualElapsedTime,CRSElapsedTime,AirTime,ArrDelay,DepDelay,Origin,Dest,Distance,
TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,CarrierDelay,WeatherDelay,NASDelay,SecurityDelay,
LateAircraftDelay

scala> val data = delayed_flights.filter(x=> x!= header)
data: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[162] at filter at <console>:43

scala> val rdd1 = data.map(x => x.split(","))
rdd1: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[163] at map at <console>:45

scala> val rdd2 = rdd1.map(x => (x(17),x(16).toDouble))
rdd2: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[164] at map at <console>:47

scala> val rdd3 = rdd2.mapValues((_, 1))
rdd3: org.apache.spark.rdd.RDD[(String, (Double, Int))] = MapPartitionsRDD[165] at mapValues at <con
sole>:49

scala> val rdd4 = rdd3.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
rdd4: org.apache.spark.rdd.RDD[(String, (Double, Int))] = ShuffledRDD[166] at reduceByKey at <consol
e>:51

scala> val rdd5 = rdd4.mapValues{ case (sum, count) => (1.0 * sum)/count}
rdd5: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[167] at mapValues at <console>:5
3

scala> val rdd6 = rdd5.map(x => (x._2,x._1))
rdd6: org.apache.spark.rdd.RDD[(Double, String)] = MapPartitionsRDD[168] at map at <console>:55

scala> val rdd7 = rdd6.sortByKey(false)
rdd7: org.apache.spark.rdd.RDD[(Double, String)] = ShuffledRDD[171] at sortByKey at <console>:57

scala> val rdd8 = rdd7.map(x => (x._2,x._1))
rdd8: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[172] at map at <console>:59
```



```
scala> val avg = rdd8.take(10)
avg: Array[(String, Double)] = Array((CMX,116.1470588235294), (PLN,93.76190476190476), (SPI,83.84873
949579831), (ALO,82.2258064516129), (MQT,79.55665024630542), (ACY,79.3103448275862), (MOT,78.6616541
3533835), (HHH,76.53005464480874), (EGE,74.12891986062718), (BGM,73.15533980582525))

scala> avg.foreach(println)
(CMX,116.1470588235294)
(PLN,93.76190476190476)
(SPI,83.84873949579831)
(ALO,82.2258064516129)
(MQT,79.55665024630542)
(ACY,79.3103448275862)
(MOT,78.66165413533835)
(HHH,76.53005464480874)
(EGE,74.12891986062718)
(BGM,73.15533980582525)

scala> █
```

==

Problem Statement 4

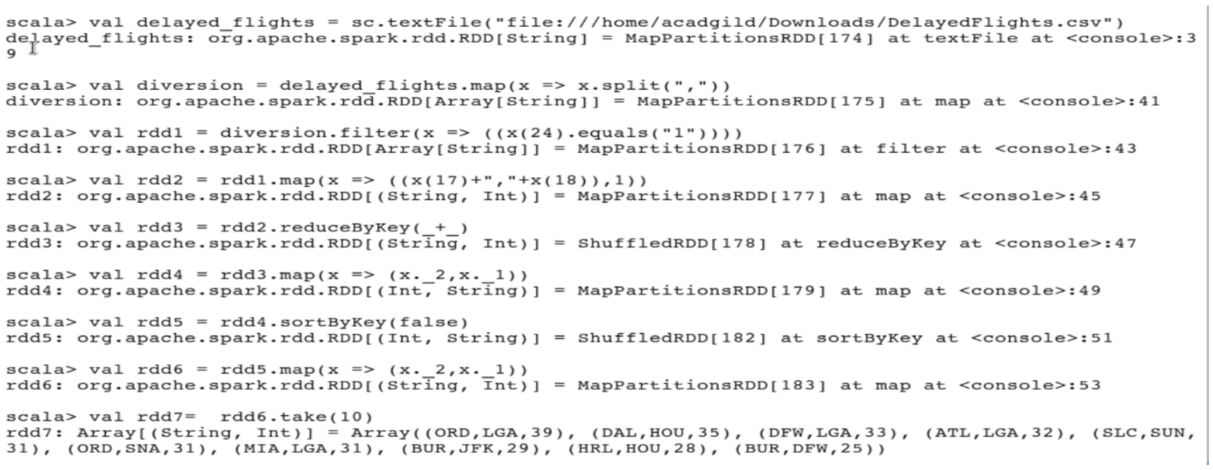
Which route (origin & destination) has seen the maximum diversion?

~~~~~~

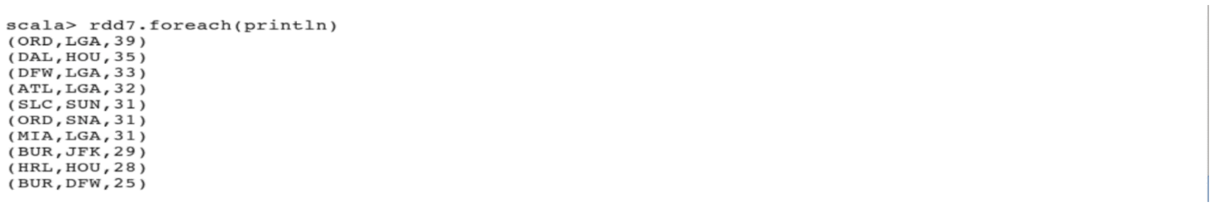
```
//Reading the DelayedFlights CSV file into an RDD

val delayed_flights =
sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")

//Splitting the data into individual elements with comma delimiter

val diversion = delayed_flights.map(x => x.split(","))

//Filtering the filter where 25th Column diversion is equal to 1.

val rdd1 = diversion.filter(x => ((x(24).equals("1"))))

//Mapping the origin and destination as key and 1 as value

val rdd2 = rdd1.map(x => ((x(17)+","+x(18)),1))

//Grouping the keys and summing up the values

val rdd3 = rdd2.reduceByKey(_+_)

//Changing the order of elements in each key

val rdd4 = rdd3.map(x => (x._2,x._1))
```
~~~~~~

```
val rdd5 = rdd4.sortByKey(false)

val rdd6 = rdd5.map(x => (x._2,x._1))

//Selecting first 10 values from rdd

val rdd7=  rdd6.take(10)

//Displaying the expected output

rdd7.foreach(println)
```

SCREENSHOTS:



```
scala> val delayed_flights = sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[174] at textFile at <console>:3
9

scala> val diversion = delayed_flights.map(x => x.split(","))
diversion: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[175] at map at <console>:41

scala> val rdd1 = diversion.filter(x => ((x(24).equals("1"))))
rdd1: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[176] at filter at <console>:43

scala> val rdd2 = rdd1.map(x => ((x(17)+","+x(18)),1))
rdd2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[177] at map at <console>:45

scala> val rdd3 = rdd2.reduceByKey(_+_)
rdd3: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[178] at reduceByKey at <console>:47

scala> val rdd4 = rdd3.map(x => (x._2,x._1))
rdd4: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[179] at map at <console>:49

scala> val rdd5 = rdd4.sortByKey(false)
rdd5: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[182] at sortByKey at <console>:51

scala> val rdd6 = rdd5.map(x => (x._2,x._1))
rdd6: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[183] at map at <console>:53

scala> val rdd7=  rdd6.take(10)
rdd7: Array[(String, Int)] = Array((ORD,LGA,39), (DAL,HOU,35), (DFW,LGA,33), (ATL,LGA,32), (SLC,SUN,
31), (ORD,SNA,31), (MIA,LGA,31), (BUR,JFK,29), (HRL,HOU,28), (BUR,DFW,25))
```



```
scala> rdd7.foreach(println)
(ORD,LGA,39)
(DAL,HOU,35)
(DFW,LGA,33)
(ATL,LGA,32)
(SLC,SUN,31)
(ORD,SNA,31)
(MIA,LGA,31)
(BUR,JFK,29)
(HRL,HOU,28)
(BUR,DFW,25)
```

==