Session 20: MLIB AND GRAPHX

Assignment 21.2

Student Name: Abarajithan SA

Course: Big Data Hadoop & Spark Training

Start Date: 2017-09-09

End Date: 2017-11-26

Assignment 21.2– MLIB AND GRAPHX.

Contents

Introduction

In this assignment, we are going to analyze Aviation Data.

Problem Statement

Implement the below blog at your end and send the complete documentation.

https://drive.google.com/file/d/0B_Qjau8wv1KoUThzZ24tT1NsZGs/view?usp=sharing

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.

Delayed_Flights.csv Datasets

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

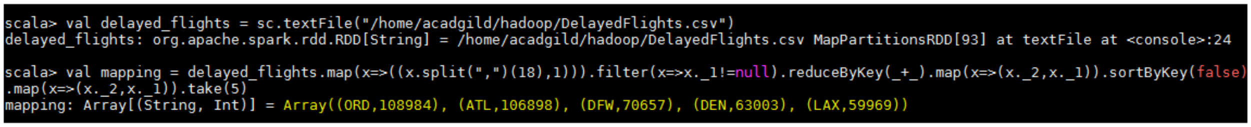
- Year: 1987 – 2008
- Month: 1 – 12
- FlightNum: Flight number
- Cancelled: Was the flight cancelled?
- CancelleationCode: The reason for cancellation.

Problem Statement 1 - Find out the top 5 most visited destinations.

Source code

```
val delayed_flights = sc.textFile("/home/acadgild/hadoop/DelayedFlights.csv")
```

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



```
scala> val delayed_flights = sc.textFile("/home/acadgild/hadoop/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /home/acadgild/hadoop/DelayedFlights.csv MapPartitionsRDD[93] at textFile at <console>:24

scala> val mapping = delayed_flights.map(x=>((x.split(",")(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: Array[(String, Int)] = Array((ORD,108984), (ATL,106898), (DFW,70657), (DEN,63003), (LAX,59969))
```

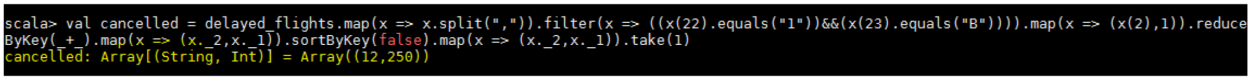
Expected Output

mapping: Array[(String, Int)] = Array((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?

Source code

```
val cancelled = delayed_flights.map(x => x.split(",")).filter(x =>
((x(22).equals("1"))&&(x(23).equals("B")))).map(x => (x(2),1)).reduceByKey(_+_).map(x =>
(x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(1)
```



```
scala> val cancelled = delayed_flights.map(x => x.split(",")).filter(x => ((x(22).equals("1"))&&(x(23).equals("B")))).map(x => (x(2),1)).reduce
ByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(1)
cancelled: Array[(String, Int)] = Array((12,250))
```

Expected Output

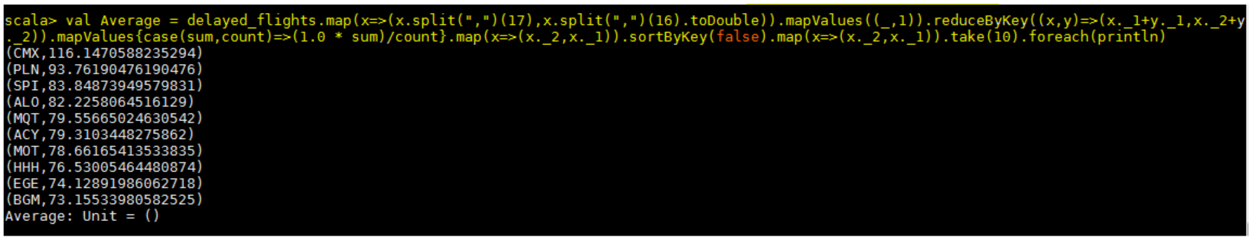
***cancelled**: Array[(String, Int)] = Array((12,250))*

Problem Statement 3 - Top ten origins with the highest AVG departure delay

Source Code

```
val Average =
delayed_flights.map(x=>(x.split(",")(17),x.split(",")(16).toDouble)).mapValues((_,1)).reduceByKey((x,y
)=>(x._1+y._1,x._2+y._2)).mapValues{case(sum,count)=>(1.0 *
sum)/count}.map(x=>(x._2,x._1)).sortByKey(false).map(x=>(x._2,x._1)).take(10)
```

Expected Output



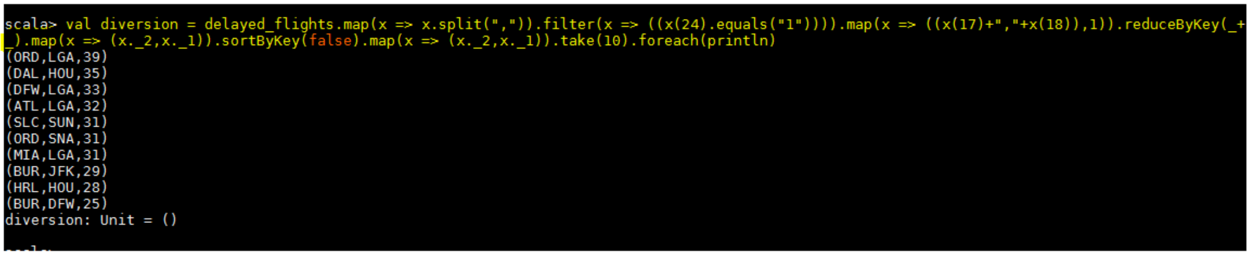
```
scala> val Average = delayed_flights.map(x=>(x.split(",")(17),x.split(",")(16).toDouble)).mapValues((_,1)).reduceByKey((x,y)=>(x._1+y._1,x._2+y
._2)).mapValues{case(sum,count)=>(1.0 * sum)/count}.map(x=>(x._2,x._1)).sortByKey(false).map(x=>(x._2,x._1)).take(10).foreach(println)
(CMX,116.1470588235294)
(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)
Average: Unit = ()
```

Problem Statement 4 - Which route (origin & destination) has seen the maximum diversion?

Source Code

```
val diversion = delayed_flights.map(x => x.split(",")).filter(x => ((x(24).equals("1")))).map(x =>
((x(17)+","+x(18)),1)).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x =>
(x._2,x._1)).take(10).foreach(println)
```

Expected Output



```
scala> val diversion = delayed_flights.map(x => x.split(",")).filter(x => ((x(24).equals("1")))).map(x => ((x(17)+","+x(18)),1)).reduceByKey(_+
_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(10).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)
diversion: Unit = ()
```