**SESSION 21 ASSIGNMENT 2**

* **Importing API and loading the data**

scala> import org.apache.spark.sql.Row

import org.apache.spark.sql.Row

scala> import org.apache.spark.sql.DataFrame

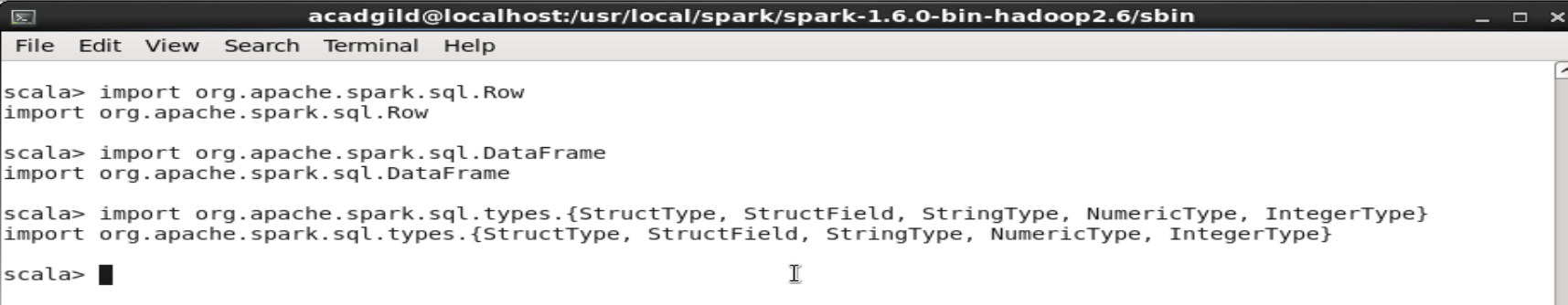
import org.apache.spark.sql.DataFrame

scala> import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType}

import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType}

scala> val delayed\_flights = sc.textFile("file:///home/acadgild/Downloads/DelayedFlights.csv")

delayed\_flights: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:30



* **PROBLEM 1 : Top 5 most visited destinations.**

scala> 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: Array[(String, Int)] = Array((ORD,108984), (ATL,106898), (DFW,70657), (DEN,63003), (LAX,59969))

scala> mapping.foreach(println)

(ORD,108984)

(ATL,106898)

(DFW,70657)

(DEN,63003)

(LAX,59969)



* **PROBLEM 2 : The month which has most number of cancellations due to bad weather**

scala> val canceled = delayed\_flights.map(x => x.split(",")).filter(x => ((x(22).equals("1"))&&

| (x(23).equals("B")))).map(x => (x(2),1)).reduceByKey(\_+\_).map(x =>

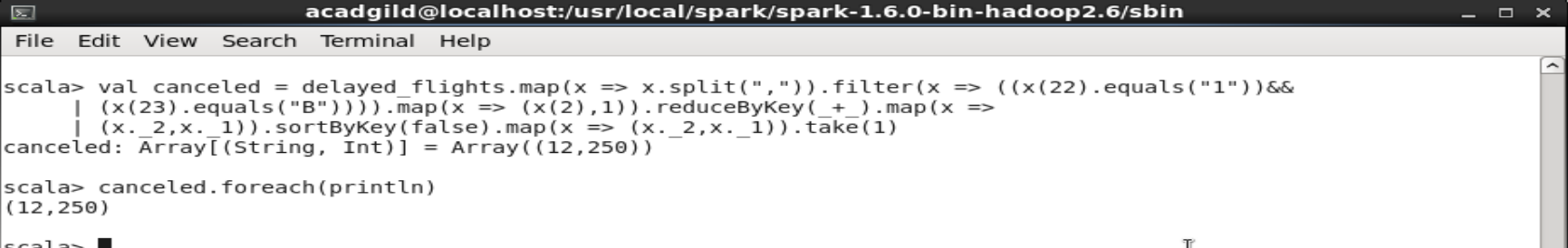
| (x.\_2,x.\_1)).sortByKey(false).map(x => (x.\_2,x.\_1)).take(1)

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

scala> canceled.foreach(println)

(12,250)

scala>

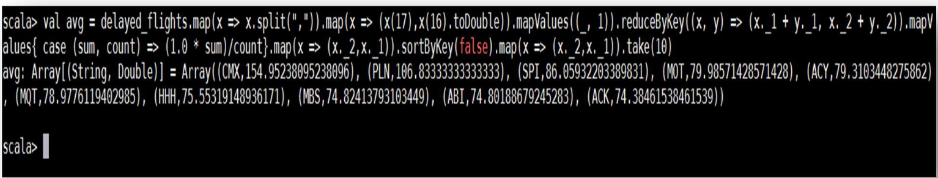


* **PROBLEM 3 : Top ten origins with the highest AVG departure delay**

val avg = delayed\_flights.map(x => x.split(",")).map(x => (x(17),x(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)



* **PROBLEM 4 : The route (origin & destination) which has seen the maximum diversion**

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 = ()

scala>

