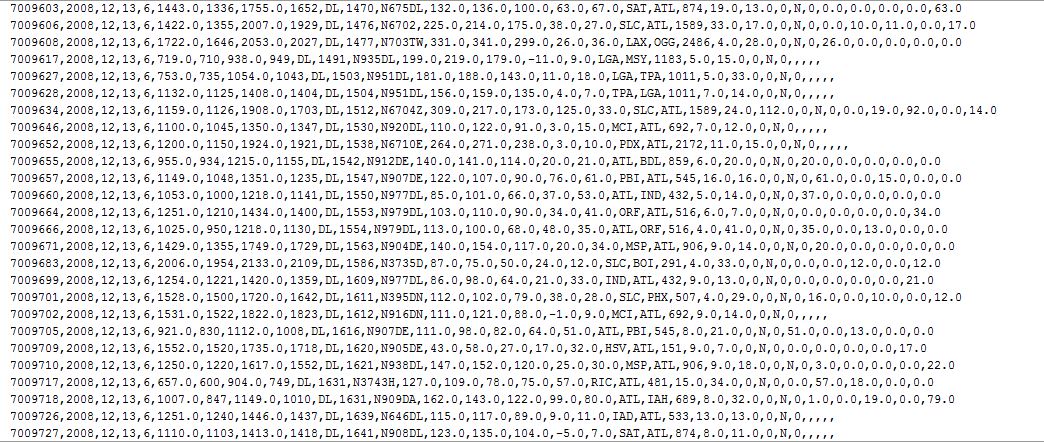
**Aviation data analysis**

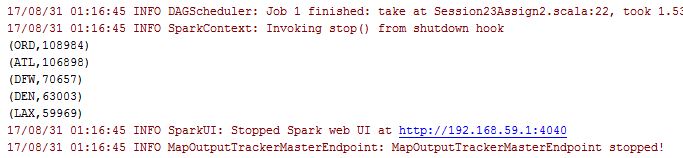
**Source Code:**

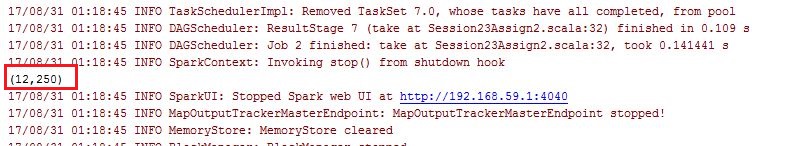
**import** org.apache.spark.sql.{Column, Row, SQLContext, SparkSession}

*//Explanation is already given in Assignment18.1***object** Session23Assign2 **extends** App {  
 **val** *spark* = SparkSession.*builder*()  
 .master(**"local"**)  
 .appName(**"Session23Assign2"**)  
 .config(**"spark.sql.warehouse.dir"**, **"file:///G:/ACADGILD/course material/Hadoop/Sessions/Session 23/Assignments/Assignment2"**)  
 .getOrCreate()  
 *//Explanation is already given in Assignment 18.1  
  
 //setting path of winutils.exe* System.*setProperty*(**"hadoop.home.dir"**, **"F:/Softwares/winutils"**)  
 *//winutils.exe needs to be present inside HADOOP\_HOME directory, else below error is returned:  
 //error: java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.  
  
 //<<<<<<<<<<-------- PROBLEM STATEMENT 1 ---------->>>>>>>>>>>  
 //1. Find out the top 5 most visited destinations.* **val** *delayed\_flights* = *spark*.sparkContext.textFile(**"file:///G:/ACADGILD/course material/Hadoop/Sessions/Session 23/Assignments/Assignment2/DelayedFlights.csv"**)  
  
 *//delayed\_flights.foreach(x => println(x))  
 //REFER Screenshot 1 for OUTPUT* **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)  
 *//sortByKey(false) -->> order by key desc  
  
 mapping*.foreach(*println*)  
 *//REFER Screenshot 2 for OUTPUT  
  
 //<<<<<<<<<<-------- PROBLEM STATEMENT 2 ---------->>>>>>>>>>>  
 //2. Which month has seen the most number of cancellations due to bad weather?* **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)  
  
 *cancelled*.foreach(*println*)  
 *//REFER Screenshot 3 for OUTPUT  
  
 //<<<<<<<<<<-------- PROBLEM STATEMENT 3 ---------->>>>>>>>>>>  
 //3. Top ten origins with the highest AVG departure delay  
 //skipping headers* **val** *header* = *delayed\_flights*.first()  
 **val** *filter\_delayed\_flights* = *delayed\_flights*.filter(row => row!= *header*)  
  
 **val** *avg* = *filter\_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)  
  
 *avg*.foreach(*println*)  
 *//REFER Screenshot 4 for OUTPUT  
  
 //<<<<<<<<<<-------- PROBLEM STATEMENT 4 ---------->>>>>>>>>>>  
 //4. Which route (origin & destination) has seen the maximum diversion* **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)  
  
 *diversion*.foreach(*println*)  
 *//REFER Screenshot 5 for OUTPUT*}

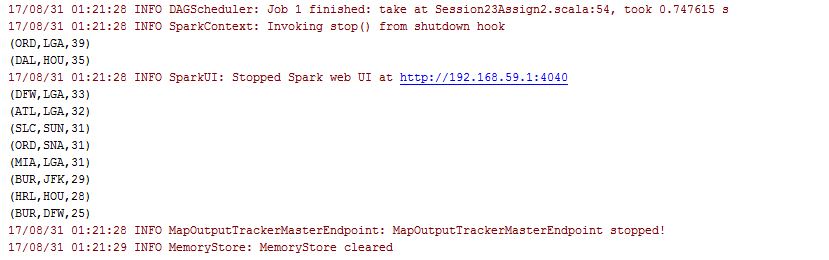
**OUTPUT:**

**Screenshot 1**

**Screenshot 2**

**Screenshot 3**

**Screenshot 4**

**Screenshot 5**