Ceng790 Big Data Analytics
Assignment 1

Egemen Berk Galatali
2099018@ceng.metu.edu.tr

March 15, 2019

Part I

1. Using the Spark SQL API (accessible with spark.sql("...")), select fields containing the identifier, GPS coordinates, and type of license of each picture.

```
val sqlContext = neworg.apache.spark.sql.SQLContext(spark.sparkContext)
import sqlContext.implicits._

val first = sqlContext.sql("SELECT photo_id, longitude,latitude, license FROM
    flickrMeta")

// For printing DataFrame uncomment below
//first.map(f => "ID: " + f(0) + " - " + "Coord: " + f(1) + "- " + f(2) + " - " +
    "License: " +f(3)).collect().foreach(println)
```

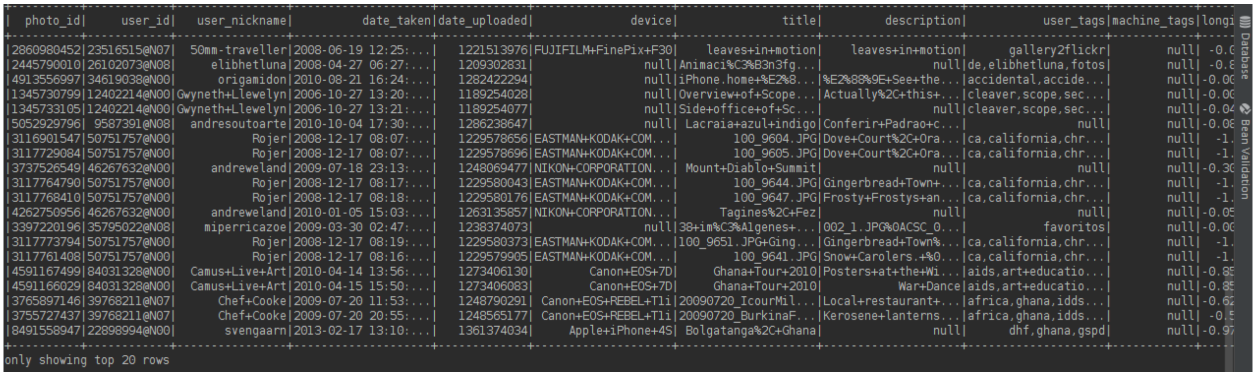
2. Create a DataFrame containing only data of interesting pictures, i.e. pictures for which the license information is not null, and GPS coordinates are valid (not -1.0).

```
val pictures = originalFlickrMeta
    .filter(l => l(15) !=null) // license
    .filter(l => (l.getFloat(10) != 1.0f)) // Coord
    .filter(l =>l.getFloat(11) != -1.0f) // Coord

```

3. Display the execution plan used by Spark to compute the content of this DataFrame(explain()).

```
== Physical Plan ==
*(1) Filter org.apache.spark.sql.catalyst.optimizer.
    CombineTypedFilters$$$Lambda$958/903794242@1e3f0aea.apply
+- *(1) FileScan csv [photo_id#0L,user_id#1,user_nickname#2,date_taken#3,
    date_uploaded#4,device#5,title#6,description#7,user_tags#8,machine_tags#9,
    longitude#10,latitude#11,accuracy#12,url#13,download_url#14,license#15,
    license_url#16,server_id#17,farm_id#18,secret#19,secret_original#20,
    extension_original#21,marker#22] Batched: false, Format: CSV, Location:
    InMemoryFileIndex[file:/home/egemen/IdeaProjects/untitled/flickrSample.txt],
    PartitionFilters: [], PushedFilters: [], ReadSchema: struct<photo_id:bigint,
    user_id:string,user_nickname:string,date_taken:string,date_uploaded:string...

```

4. Display the data of this pictures (show()). Keep in mind that Spark uses lazy execution, so as long as we do not perform any action, the transformations are not executed



```
+----------+------------+----------------+--------------------+-------------+--------------------+--------------------+--------------------+--------------------+------------+-----
|  photo_id|     user_id|   user_nickname|          date_taken|date_uploaded|              device|               title|         description|           user_tags|machine_tags|longi
+----------+------------+----------------+--------------------+-------------+--------------------+--------------------+--------------------+--------------------+------------+-----
|2860980452|23516515@N07|  50mm-traveller|2008-06-19 12:25:...|   1221513976|FUJIFILM+FinePix+F30|    leaves+in+motion|    leaves+in+motion|      gallery2flickr|        null| -0.0
|2445790010|26102073@N08|     elibhetluna|2008-04-27 06:27:...|   1209302831|                null|Animaci%C3%B3n3fg...|                null|de,elibhetluna,fotos|        null| -0.8
|4913556997|34619038@N00|      origamidon|2010-08-21 16:24:...|   1282422294|                null|iPhone.home+%E2%8...|%E2%88%9E+See+the...|accidental,accide...|        null|-0.00
|1345730799|12402214@N00|Gwyneth+Llewelyn|2006-10-27 13:20:...|   1189254028|                null|Overview+of+Scope...|Actually%2C+this+...|cleaver,scope,sec...|        null|-0.00
|1345733105|12402214@N00|Gwyneth+Llewelyn|2006-10-27 13:21:...|   1189254077|                null|Side+office+of+Sc...|                null|cleaver,scope,sec...|        null|-0.04
|5052929796| 9587391@N08|  andresoutoarte|2010-10-04 17:30:...|   1286238647|                null| Lacraia+azul+indigo|Conferir+Padrao+c...|                null|        null|-0.08
|3116901547|50751757@N00|           Rojer|2008-12-17 08:07:...|   1229578656|EASTMAN+KODAK+COM...|        100_9604.JPG|Dove+Court%2C+Ora...|ca,california,chr...|        null|  -1.
|3117729084|50751757@N00|           Rojer|2008-12-17 08:07:...|   1229578696|EASTMAN+KODAK+COM...|        100_9605.JPG|Dove+Court%2C+Ora...|ca,california,chr...|        null|  -1.
|3737526549|46267632@N00|     andreweland|2009-07-18 23:13:...|   1248069477|NIKON+CORPORATION...| Mount+Diablo+Summit|                null|                null|        null|-0.30
|3117764790|50751757@N00|           Rojer|2008-12-17 08:17:...|   1229580043|EASTMAN+KODAK+COM...|        100_9644.JPG|Gingerbread+Town+...|ca,california,chr...|        null|  -1.
|3117768410|50751757@N00|           Rojer|2008-12-17 08:18:...|   1229580176|EASTMAN+KODAK+COM...|        100_9647.JPG|Frosty+Frostys+an...|ca,california,chr...|        null|  -1.
|4262750956|46267632@N00|     andreweland|2010-01-05 15:03:...|   1263135857|NIKON+CORPORATION...|      Tagines%2C+Fez|                null|                null|        null|-0.05
|3397220196|35795022@N08|    miperricazoe|2009-03-30 02:47:...|   1238374073|                null|38+im%C3%A1genes+...|002_1.JPG%0ACSC_0...|           favoritos|        null|-0.00
|3117773794|50751757@N00|           Rojer|2008-12-17 08:19:...|   1229580373|EASTMAN+KODAK+COM...|100_9651.JPG+Ging...|Gingerbread+Town%...|ca,california,chr...|        null|  -1.
|3117761408|50751757@N00|           Rojer|2008-12-17 08:16:...|   1229579905|EASTMAN+KODAK+COM...|        100_9641.JPG|Snow+Carolers.+%0...|ca,california,chr...|        null|  -1.
|4591167499|84031328@N00|  Camus+Live+Art|2010-04-14 13:56:...|   1273406130|        Canon+EOS+7D|     Ghana+Tour+2010|Posters+at+the+Wi...|aids,art+educatio...|        null|-0.85
|4591166029|84031328@N00|  Camus+Live+Art|2010-04-15 15:50:...|   1273406083|        Canon+EOS+7D|     Ghana+Tour+2010|           War+Dance|aids,art+educatio...|        null|-0.85
|3765897146|39768211@N07|      Chef+Cooke|2009-07-20 11:53:...|   1248790291| Canon+EOS+REBEL+T1i|20090720_IcourMil...|Local+restaurant+...|africa,ghana,idds...|        null|-0.62
|3755727437|39768211@N07|      Chef+Cooke|2009-07-20 20:55:...|   1248565177| Canon+EOS+REBEL+T1i|20090720_BurkinaF...|Kerosene+lanterns...|africa,ghana,idds...|        null| -0.5
|8491558947|22898994@N00|       svengaarn|2013-02-17 13:10:...|   1361374034|     Apple+iPhone+4S| Bolgatanga%2C+Ghana|                null|      dhf,ghana,gspd|        null|-0.97
+----------+------------+----------------+--------------------+-------------+--------------------+--------------------+--------------------+--------------------+------------+-----
only showing top 20 rows
```

5. Our goal is now to select the pictures whose license is NonDerivative. To this end we will use a second file containing the properties of each license. Load this file in a DataFrame and do a join operation to identify pictures that are both interesting and NonDerivative. Examine the execution plan and display the results.

We first load the file and then filter this license frame so that we get only the nonDerivative Licenses as a tuple containing name of the license as first parameter and dumy integer as a second parameter so that join operation will perform. We rename the column of the first frame of nonDervLicense so that in the join procedure we ease our job. In the join procedure, we join interesting pictures, which were generated before, with nonDerivative Licenses.

```scala
val flickrLicense = spark.sqlContext
      .read
      .format("csv")
      .option("delimeter", " ")
      .option("sep", "\t")
      .option("inferSchema", "true")
      .option("header", "true")
      .load("/home/egemen/IdeaProjects/untitled/src/main/scla/edu/metu/ceng790/Assignment1/FlickrLicense.txt")

val nonDervLicense = flickrLicense
                          .filter(lis =>(lis.getInt(3) == 1))
                          .map(l => (l.getString(0), 1))

// We rename the column in order to use Seq("license") so that we remove duplicate columns.
val nonDervs = nonDervLicense.withColumnRenamed("_1","license")

println(flickrLicense.explain()) // For examining the execution
pictures.join(nonDervs, Seq("license")).show() // For showing the results.

```

6. During a work session, it is likely that we reuse multiple time the DataFrame of interesting pictures. It would be a good idea to cache it to avoid recomputing it fromthe file each time we use it. Do this, and examine the execution plan of the join operation again. What do you notice?

When I compared the cached execution plan(Physical Plan) with non-cached version, these two new lines were added to the cached version at the top of the plan.

```
InMemoryTableScan [license#15, photo_id#0L, user_id#1, user_nickname#2,
    date_taken#3, date_uploaded#4 ...]
    +- InMemoryRelation [license#15, photo_id#0L, user_id#1, user_nickname#2,
    date_taken#3, date_uploaded#4 ...], StorageLevel(disk, memory, deserialized,
    1 replicas)

```

7. Save the final result in a csv file (write). Dont forget to add a header to reuse it more easily.

```
val result = pictures.join(nonDervs, Seq("license"))

result.write.format("csv")
        .option("delimeter", " ")
        .option("sep", "\t")
        .option("header", true)
        .save("nonDervLicensedPictures.csv")

```

Results are attached to the submission folder.

Part II

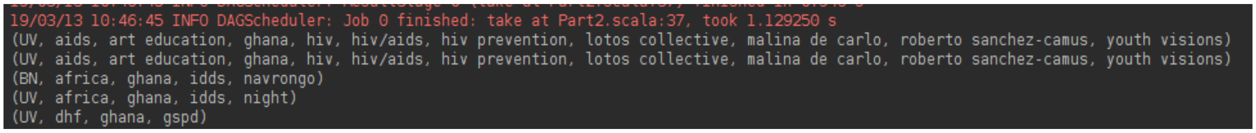
1. Display the 5 lines of the RDD (take(5)) and display the number of elements in the RDD (count())

```
originalFlickrMeta.take(5).foreach(println)
println("Count of the RDD: " + originalFlickrMeta.count())
```

2. Transform the RDD[String] in RDD[Picture] using the Picture class. Only keep interesting pictures having a valid country and tags. To check your program, display 5 elements

```
// We first transform the RDD[String] to RDD[Pictures] using map
// Then filter out the picitures with valid country and tags
val pictures = originalFlickrMeta
                    .map(f => new Picture(f.split("\t")))
                    .filter(f => f.hasValidCountry)
                    .filter(f => f.hasTags)
                    // for printing results uncomment below
                    //.take(5)
                    //.foreach(println)
```

Results are below



```
19/03/13 10:46:45 INFO DAGScheduler: Job 0 finished: take at Part2.scala:37, took 1.129250 s
(UV, aids, art education, ghana, hiv, hiv/aids, hiv prevention, lotos collective, malina de carlo, roberto sanchez-camus, youth visions)
(UV, aids, art education, ghana, hiv, hiv/aids, hiv prevention, lotos collective, malina de carlo, roberto sanchez-camus, youth visions)
(BN, africa, ghana, idds, navrongo)
(UV, africa, ghana, idds, night)
(UV, dhf, ghana, gspd)
```

3. Now group these images by country (groupBy). Print the list of images corresponding to the first country. What is the type of this RDD?

```
pictures.groupBy(f => (f.c, f.toString))
        .take(1)
        .foreach(println)
```

It is actually a tuple contaning the first element as a Key(country) and the second element of it is the String

4. We now wish to process an RDD containing pairs in which the first element is a country, and the second element is the list of tags used on pictures taken in this country. When a tag is used on multiple pictures, it should appear multiple times in the list. As each image has its own list of tags, we need to concatenate these lists, and the flatten function could be useful.

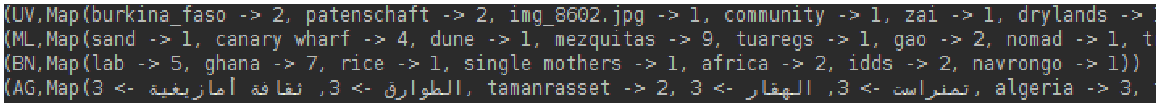
```
val flatted = pictures.map(f => (f.c.toString(), f.userTags))
                            .groupByKey()
                            .map(f => (f._1, f._2.flatten))
                            //.foreach(println)
```

   Mapping operation creates a country name and picture's tags for each picture. After that, We group according to the country name and then flatten the list of tags so that each pictures' tags will be equally distributed to the list.

5. We wish to avoid repetitions in the list of tags, and would rather like to have each tag associated to its frequency. Hence, we want to build a RDD of type RDD[(Country, Map[String, Int])]. The groupBy(identity) function, equivalent to groupBy(x=¿x) could be useful.

```
flatted.map(f => (f._1, f._2.groupBy(x=>x)
                                 .map(y => (y._1, y._2.size)))).foreach(println)
```

   Results are below



```
(UV,Map(burkina_faso -> 2, patenschaft -> 2, img_8602.jpg -> 1, community -> 1, zai -> 1, drylands ->
(ML,Map(sand -> 1, canary wharf -> 4, dune -> 1, mezquitas -> 9, tuaregs -> 1, gao -> 2, nomad -> 1, t
(BN,Map(lab -> 5, ghana -> 7, rice -> 1, single mothers -> 1, africa -> 2, idds -> 2, navrongo -> 1))
(AG,Map(ثقافة أمازيغية -> 3, الطوارق -> 3, tamanrasset -> 2, الهقار -> 3, تمنراست -> 3, algeria -> 3,
```

6. There are often several ways to obtain a result. The method we used to compute the frequency of tags in each country quickly reaches a state in which the size of the RDD is the number of countries. This can limit the parallelism of the execution as the number of countries is often quite small. Can you propose another way to reach the same result without reducing the size of the RDD until the very end?

   We can fasten the process by calculating the frequency of the tags for each picture before we gather the pictures according to their country.