CENG 790 BIG DATA ANALYTICS
ASSIGNMENT I

İbrahim Can Gençel
2166429

Part 1: Processing data using the DataFrame API

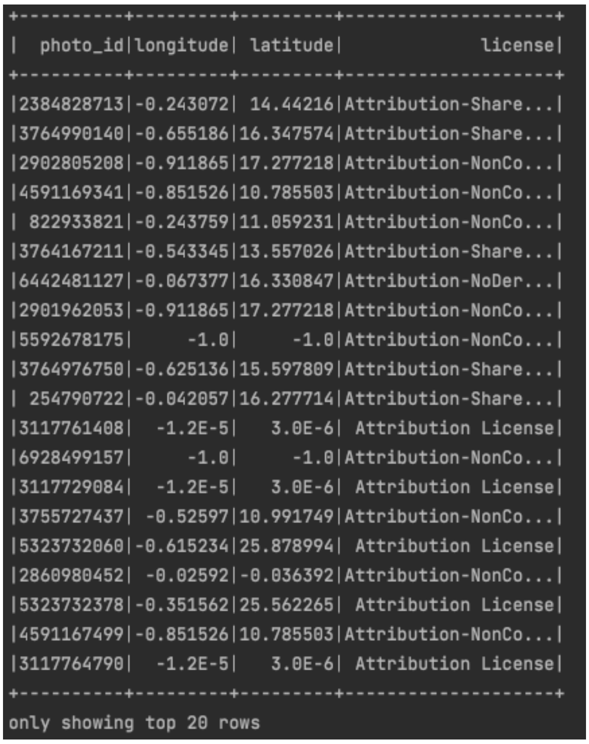
- I have added “Logger.getLogger("org").setLevel(Level.ERROR)” in order to log only errors in the terminal.

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

CODE =

```
// YOUR CODE HERE
println(originalFlickrMeta.count())

// Following line prints the schema so that I can see names of fields
originalFlickrMeta.printSchema()

// Following line craetes a view in order to use it with SQL
originalFlickrMeta.createOrReplaceTempView("flickrdata")

// In order to get photo_id, GPS coordinates and license type SQL is used
val all_data =spark.sql("SELECT distinct photo_id, longitude, latitude, license from flickrdata")
all_data.show()
```

OUTPUT =



```
+----------+---------+---------+--------------------+
|  photo_id|longitude| latitude|             license|
+----------+---------+---------+--------------------+
|2384828713|-0.243072| 14.44216|Attribution-Share...|
|3764990140|-0.655186|16.347574|Attribution-Share...|
|2902805208|-0.911865|17.277218|Attribution-NonCo...|
|4591169341|-0.851526|10.785503|Attribution-NonCo...|
| 822933821|-0.243759|11.059231|Attribution-NonCo...|
|3764167211|-0.543345|13.557026|Attribution-Share...|
|6442481127|-0.067377|16.330847|Attribution-NoDer...|
|2901962053|-0.911865|17.277218|Attribution-NonCo...|
|5592678175|     -1.0|     -1.0|Attribution-NonCo...|
|3764976750|-0.625136|15.597809|Attribution-Share...|
| 254790722|-0.042057|16.277714|Attribution-Share...|
|3117761408|  -1.2E-5|   3.0E-6| Attribution License|
|6928499157|     -1.0|     -1.0|Attribution-NonCo...|
|3117729084|  -1.2E-5|   3.0E-6| Attribution License|
|3755727437| -0.52597|10.991749|Attribution-NonCo...|
|5323732060|-0.615234|25.878994| Attribution License|
|2860980452| -0.02592|-0.036392|Attribution-NonCo...|
|5323732378|-0.351562|25.562265| Attribution License|
|4591167499|-0.851526|10.785503|Attribution-NonCo...|
|3117764790|  -1.2E-5|   3.0E-6| Attribution License|
+----------+---------+---------+--------------------+
only showing top 20 rows
```

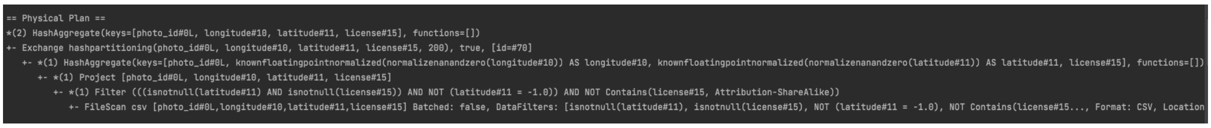
2. Create a DataFrame containing only data of interesting pictures, i.e. pictures for which the license information is not null, and GPS coordinates are valid (not -1.0).
CODE =

```
val filtered_data = all_data.filter("latitude != -1 and latitude !=
-1").filter("license not like '%Attribution-ShareAlike%'")
```

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

```
// Display the execution plan
filtered_data.explain()
```

OUTPUT =

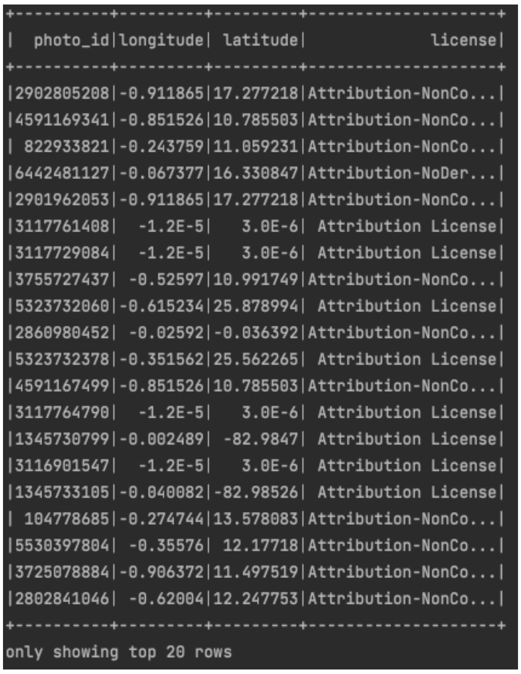


```
== Physical Plan ==
*(2) HashAggregate(keys=[photo_id#0L, longitude#10, latitude#11, license#15], functions=[])
+- Exchange hashpartitioning(photo_id#0L, longitude#10, latitude#11, license#15, 200), true, [id=#70]
   +- *(1) HashAggregate(keys=[photo_id#0L, knownfloatingpointnormalized(normalizenanandzero(longitude#10)) AS longitude#10, knownfloatingpointnormalized(normalizenanandzero(latitude#11)) AS latitude#11, license#15], functions=[])
      +- *(1) Project [photo_id#0L, longitude#10, latitude#11, license#15]
         +- *(1) Filter (((isnotnull(latitude#11) AND isnotnull(license#15)) AND NOT (latitude#11 = -1.0)) AND NOT Contains(license#15, Attribution-ShareAlike))
            +- FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15] Batched: false, DataFilters: [isnotnull(latitude#11), isnotnull(license#15), NOT (latitude#11 = -1.0), NOT Contains(license#15..., Format: CSV, Location
```

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.
CODE =

```
// In order to display the data
filtered_data.show()
```

OUTPUT =

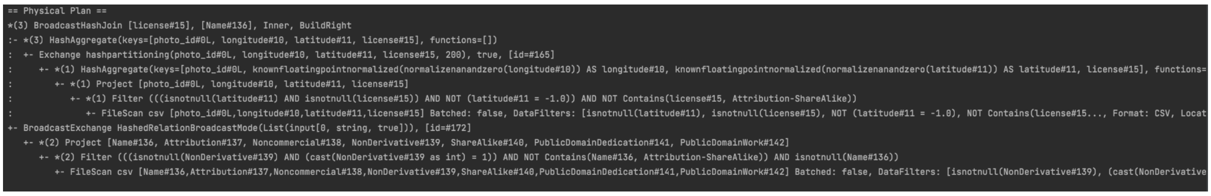


```
+----------+---------+---------+--------------------+
|  photo_id|longitude| latitude|             license|
+----------+---------+---------+--------------------+
|2902805208|-0.911865|17.277218|Attribution-NonCo...|
|4591169341|-0.851526|10.785503|Attribution-NonCo...|
| 822933821|-0.243759|11.059231|Attribution-NonCo...|
|6442481127|-0.067377|16.330847|Attribution-NoDer...|
|2901962053|-0.911865|17.277218|Attribution-NonCo...|
|3117761408|  -1.2E-5|   3.0E-6| Attribution License|
|3117729084|  -1.2E-5|   3.0E-6| Attribution License|
|3755727437| -0.52597|10.991749|Attribution-NonCo...|
|5323732060|-0.615234|25.878994| Attribution License|
|2860980452| -0.02592|-0.036392|Attribution-NonCo...|
|5323732378|-0.351562|25.562265| Attribution License|
|4591167499|-0.851526|10.785503|Attribution-NonCo...|
|3117764790|  -1.2E-5|   3.0E-6| Attribution License|
|1345730799|-0.002489| -82.9847| Attribution License|
|3116901547|  -1.2E-5|   3.0E-6| Attribution License|
|1345733105|-0.040082|-82.98526| Attribution License|
| 104778685|-0.274744|13.578083|Attribution-NonCo...|
|5530397804| -0.35576| 12.17718|Attribution-NonCo...|
|3725078884|-0.906372|11.497519|Attribution-NonCo...|
|2802841046| -0.62004|12.247753|Attribution-NonCo...|
+----------+---------+---------+--------------------+
only showing top 20 rows
```

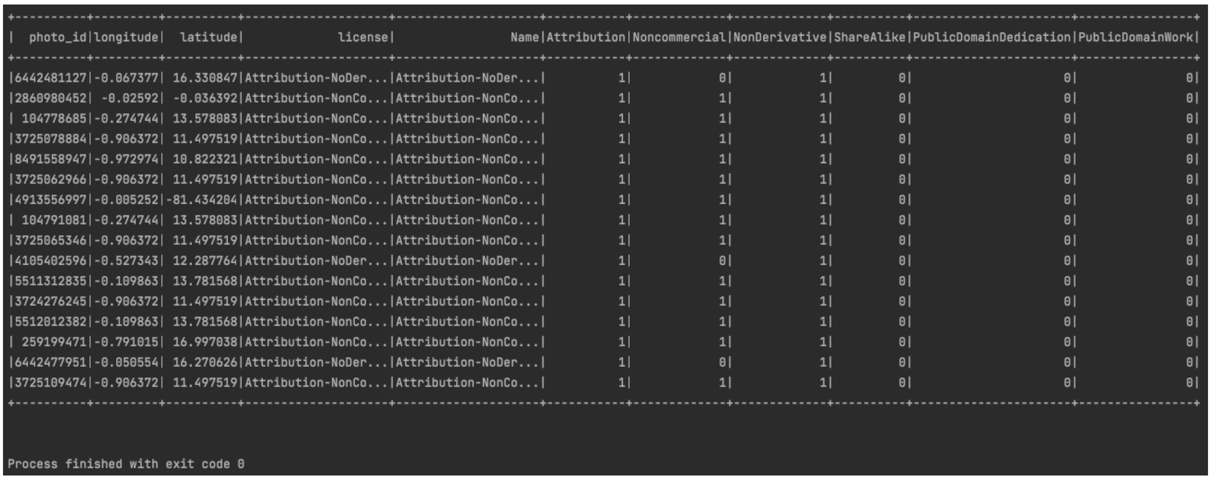
5. a) Display the execution plan used by Spark to compute the content of this DataFrame(explain()).
CODE =

```
// In order to read license file
val license_file = spark.sqlContext.read
 .format("csv")
 .option("delimiter", "\t")
 .option("header", "true")
 .load("flickrLicense.txt")

// Following line creates a view in order to use it with SQL
license_file.createOrReplaceTempView("license_data")
filtered_data.createOrReplaceTempView("filtered_data")

// Combining the data and select NonDerivatives
val filtered_DF = spark.sql("SELECT * from filtered_data")
val license_DF = spark.sql("SELECT * from license_data")
val combined_DF = filtered_DF.join(license_DF, filtered_data("license")
=== license_DF("Name"))
val final_DF = combined_DF.filter("NonDerivative == 1")

//Explain the combined data
final_DF.explain()

//Showing the combined data
final_DF.show()
```

OUTPUT =



```
== Physical Plan ==
*(3) BroadcastHashJoin [license#15], [Name#136], Inner, BuildRight
:- *(3) HashAggregate(keys=[photo_id#0L, longitude#10, latitude#11, license#15], functions=[])
:  +- Exchange hashpartitioning(photo_id#0L, longitude#10, latitude#11, license#15, 200), true, [id=#165]
:     +- *(1) HashAggregate(keys=[photo_id#0L, knownfloatingpointnormalized(normalizenanandzero(longitude#10)) AS longitude#10, knownfloatingpointnormalized(normalizenanandzero(latitude#11)) AS latitude#11, license#15], functions=
:        +- *(1) Project [photo_id#0L, longitude#10, latitude#11, license#15]
:           +- *(1) Filter (((isnotnull(latitude#11) AND isnotnull(license#15)) AND NOT (latitude#11 = -1.0)) AND NOT Contains(license#15, Attribution-ShareAlike))
:              +- FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15] Batched: false, DataFilters: [isnotnull(latitude#11), isnotnull(license#15), NOT (latitude#11 = -1.0), NOT Contains(license#15..., Format: CSV, Locat
+- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true])), [id=#172]
   +- *(2) Project [Name#136, Attribution#137, Noncommercial#138, NonDerivative#139, ShareAlike#140, PublicDomainDedication#141, PublicDomainWork#142]
      +- *(2) Filter (((isnotnull(NonDerivative#139) AND (cast(NonDerivative#139 as int) = 1)) AND NOT Contains(Name#136, Attribution-ShareAlike)) AND isnotnull(Name#136))
         +- FileScan csv [Name#136,Attribution#137,Noncommercial#138,NonDerivative#139,ShareAlike#140,PublicDomainDedication#141,PublicDomainWork#142] Batched: false, DataFilters: [isnotnull(NonDerivative#139), (cast(NonDerivative
```



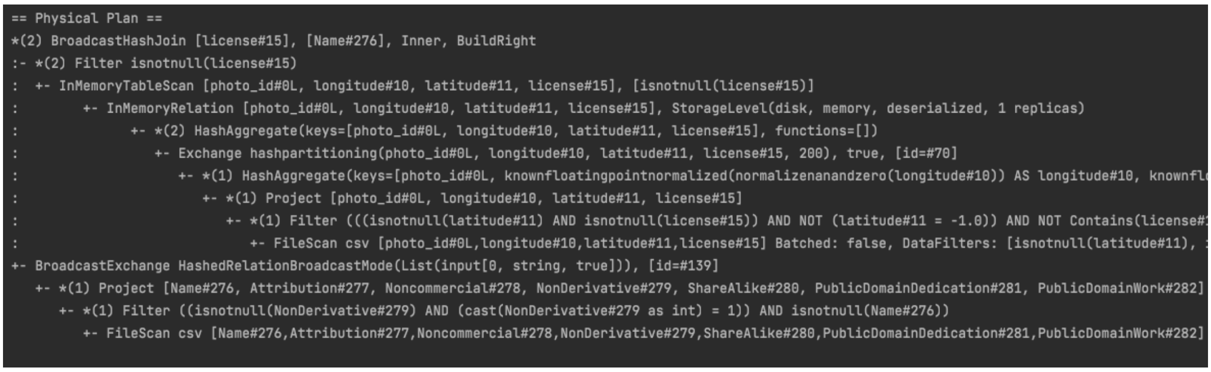
```
+----------+---------+----------+--------------------+--------------------+-----------+-------------+-------------+----------+----------------------+----------------+
|  photo_id|longitude|  latitude|             license|                Name|Attribution|Noncommercial|NonDerivative|ShareAlike|PublicDomainDedication|PublicDomainWork|
+----------+---------+----------+--------------------+--------------------+-----------+-------------+-------------+----------+----------------------+----------------+
|6442481127|-0.067377| 16.330847|Attribution-NoDer...|Attribution-NoDer...|          1|            0|            1|         0|                     0|               0|
|2860980452| -0.02592| -0.036392|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
| 104778685|-0.274744| 13.578083|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|3725078884|-0.906372| 11.497519|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|8491558947|-0.972974| 10.822321|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|3725062966|-0.906372| 11.497519|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|4913556997|-0.005252|-81.434204|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
| 104791081|-0.274744| 13.578083|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|3725065346|-0.906372| 11.497519|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|4105402596|-0.527343| 12.287764|Attribution-NoDer...|Attribution-NoDer...|          1|            0|            1|         0|                     0|               0|
|5511312835|-0.109863| 13.781568|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|3724276245|-0.906372| 11.497519|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|5512012382|-0.109863| 13.781568|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
| 259199471|-0.791015| 16.997038|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
|6442477951|-0.050554| 16.270626|Attribution-NoDer...|Attribution-NoDer...|          1|            0|            1|         0|                     0|               0|
|3725109474|-0.906372| 11.497519|Attribution-NonCo...|Attribution-NonCo...|          1|            1|            1|         0|                     0|               0|
+----------+---------+----------+--------------------+--------------------+-----------+-------------+-------------+----------+----------------------+----------------+


Process finished with exit code 0
```

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

CODE =

```
val filtered_data = all_data.filter("latitude != -1 and latitude !=
-1").filter("license not like '%Attribution-ShareAlike%'").cache()
```

OUTPUT =



```
== Physical Plan ==
*(2) BroadcastHashJoin [license#15], [Name#276], Inner, BuildRight
:- *(2) Filter isnotnull(license#15)
:  +- InMemoryTableScan [photo_id#0L, longitude#10, latitude#11, license#15], [isnotnull(license#15)]
:        +- InMemoryRelation [photo_id#0L, longitude#10, latitude#11, license#15], StorageLevel(disk, memory, deserialized, 1 replicas)
:              +- *(2) HashAggregate(keys=[photo_id#0L, longitude#10, latitude#11, license#15], functions=[])
:                 +- Exchange hashpartitioning(photo_id#0L, longitude#10, latitude#11, license#15, 200), true, [id=#70]
:                    +- *(1) HashAggregate(keys=[photo_id#0L, knownfloatingpointnormalized(normalizenanandzero(longitude#10)) AS longitude#10, knownfl
:                       +- *(1) Project [photo_id#0L, longitude#10, latitude#11, license#15]
:                          +- *(1) Filter (((isnotnull(latitude#11) AND isnotnull(license#15)) AND NOT (latitude#11 = -1.0)) AND NOT Contains(license#
:                             +- FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15] Batched: false, DataFilters: [isnotnull(latitude#11), 
+- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true])), [id=#139]
   +- *(1) Project [Name#276, Attribution#277, Noncommercial#278, NonDerivative#279, ShareAlike#280, PublicDomainDedication#281, PublicDomainWork#282]
      +- *(1) Filter ((isnotnull(NonDerivative#279) AND (cast(NonDerivative#279 as int) = 1)) AND isnotnull(Name#276))
         +- FileScan csv [Name#276,Attribution#277,Noncommercial#278,NonDerivative#279,ShareAlike#280,PublicDomainDedication#281,PublicDomainWork#282]
```

As can be seen from .examine() result, when cache is used, filtered_data is stored inside cache. When filtered_data is called again inside scala, it is brought from in-memory. These tasks are listed with .examine() command.

5. c) Save the final result in a csv file (write). Don't forget to add a header to reuse it more easily. (Do not upload the csv just give the code)

CODE =

```
// Saving to a single CSV file with header
final_DF.coalesce(numPartitions = 1).write.option("header",
true).csv("final_output.csv")
```

Part 2: Processing data using RDDs

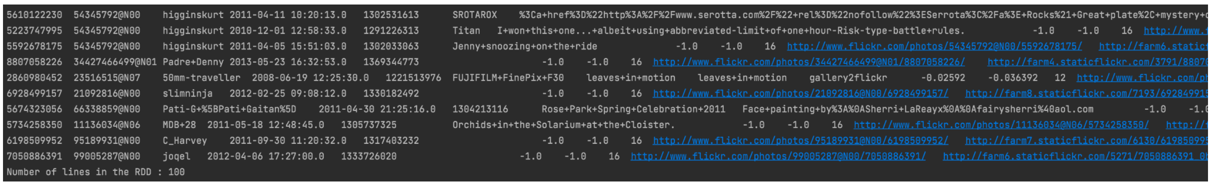
- I have added “Logger.getLogger("org").setLevel(Level.ERROR)” in order to log only errors in the terminal.

1. Display the 10 lines of the RDD (using take(..)) and display the number of elements in the RDD (count()).

CODE =

```
// Display the 10 lines of the RDD
originalFlickrMeta.take(10).foreach(println)

// Number of lines in RDD
println(f"Number of lines in the RDD : ${originalFlickrMeta.count()}")
```

OUTPUT =



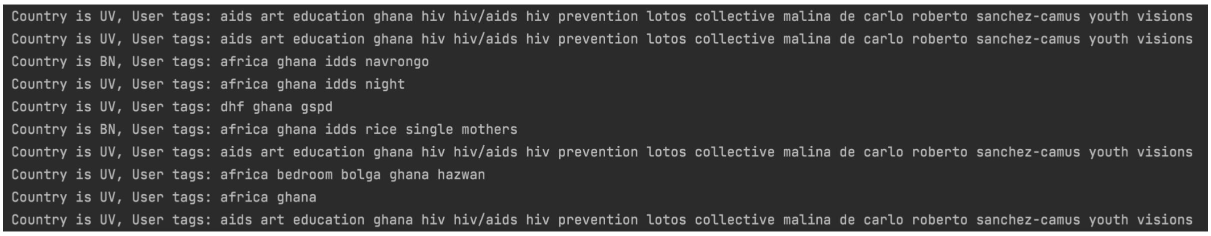
```
5610122230  54345792@N00    higginskurt 2011-04-11 10:20:13.0   1302531613      SROTAROX    %3Ca+href%3D%22http%3A%2F%2Fwww.serotta.com%2F%22+rel%3D%22nofollow%22%3ESerrota%3C%2Fa%3E+Rocks%21+Great+plate%2C+mystery+c
5223747995  54345792@N00    higginskurt 2010-12-01 12:58:33.0   1291226313      Titan   I+won+this+one...+albeit+using+abbreviated-limit+of+one+hour-Risk-type-battle+rules.          -1.0    -1.0    16  http://www.f
5592678175  54345792@N00    higginskurt 2011-04-05 15:51:03.0   1302033063      Jenny+snoozing+on+the+ride            -1.0    -1.0    16  http://www.flickr.com/photos/54345792@N00/5592678175/   http://farm6.static
8807058226  34427466499@N01 Padre+Denny 2013-05-23 16:32:53.0   1369344773                  -1.0    -1.0    16  http://www.flickr.com/photos/34427466499@N01/8807058226/    http://farm4.staticflickr.com/3791/8807
2860980452  23516515@N07    50mm-traveller  2008-06-19 12:25:30.0   1221513976  FUJIFILM+FinePix+F30    leaves+in+motion    leaves+in+motion    gallery2flickr     -0.02592    -0.036392   12  http://www.flickr.com/ph
6928499157  21092816@N00    slimninja   2012-02-25 09:08:12.0   1330182492                  -1.0    -1.0    16  http://www.flickr.com/photos/21092816@N00/6928499157/   http://farm8.staticflickr.com/7193/69284991
5674323056  66338859@N00    Pati-G+%5BPati+Gaitan%5D    2011-04-30 21:25:16.0   1304213116      Rose+Park+Spring+Celebration+2011   Face+painting+by%3A%0ASherri+LaReayx%0A%0Afairysherri%40aol.com        -1.0    -1.
5734258350  11136034@N06    MDB+28  2011-05-18 12:48:45.0   1305737325          Orchids+in+the+Solarium+at+the+Cloister.            -1.0    -1.0    16  http://www.flickr.com/photos/11136034@N06/5734258350/   http://f
6198509952  95189931@N00    C_Harvey    2011-09-30 11:20:32.0   1317403232                  -1.0    -1.0    16  http://www.flickr.com/photos/95189931@N00/6198509952/   http://farm7.staticflickr.com/6130/61985099
7050886391  99005287@N00    joqel   2012-04-06 17:27:00.0   1333726020              -1.0    -1.0    16  http://www.flickr.com/photos/99005287@N00/7050886391/   http://farm6.staticflickr.com/5271/7050886391_0
Number of lines in the RDD : 100
```

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 10 elements.

CODE =

```
// Mapping - firstly line is splitted to Array with regex (" ")
val array_RDD = originalFlickrMeta.map( line => line.split("\t"))

// Creating Picture map
val picture_RDD = array_RDD.map(Picture)

// Filtering
val filtered_pictures = picture_RDD.filter(x => x.c != null).filter(x =>
x.hasTags == true)

// Displaying the top 10 country codes and user tags
filtered_pictures.take(10).foreach(picture => println(f"Country is
${picture.c}, User tags: ${picture.userTags.mkString(" ")}"))
```

OUTPUT =



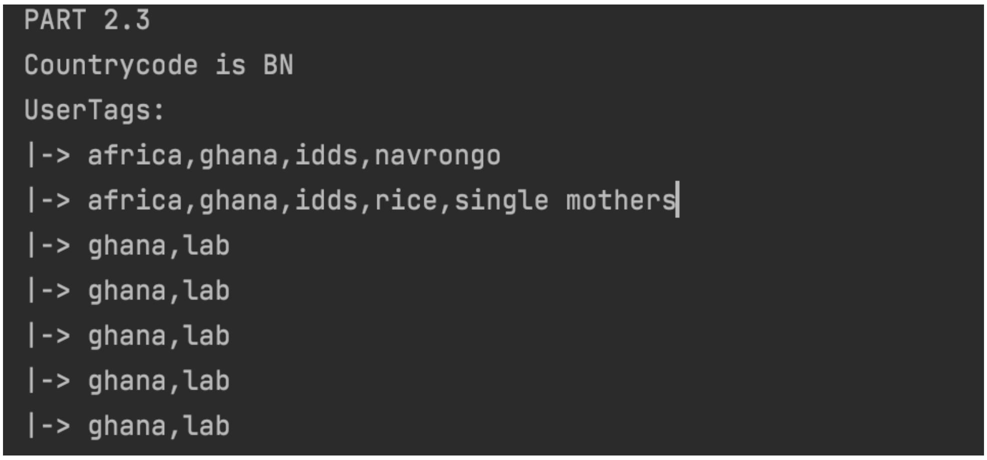
```
Country is UV, User tags: aids art education ghana hiv hiv/aids hiv prevention lotos collective malina de carlo roberto sanchez-camus youth visions
Country is UV, User tags: aids art education ghana hiv hiv/aids hiv prevention lotos collective malina de carlo roberto sanchez-camus youth visions
Country is BN, User tags: africa ghana idds navrongo
Country is UV, User tags: africa ghana idds night
Country is UV, User tags: dhf ghana gspd
Country is BN, User tags: africa ghana idds rice single mothers
Country is UV, User tags: aids art education ghana hiv hiv/aids hiv prevention lotos collective malina de carlo roberto sanchez-camus youth visions
Country is UV, User tags: africa bedroom bolga ghana hazwan
Country is UV, User tags: africa ghana
Country is UV, User tags: aids art education ghana hiv hiv/aids hiv prevention lotos collective malina de carlo roberto sanchez-camus youth visions
```

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?

CODE =

```
println("PART 2.3")
// Group by country and taking the first country and print countrycode and user tags
val pictures_g_country = filtered_pictures.map(x =>
(x.c,x.userTags.mkString(","))).groupByKey
val countrycode = pictures_g_country.take(1).map(_._1).mkString(" ")
val userTags = pictures_g_country.take(1).map(_._2).map(x => x.map(x => x.mkString))
println(f"Countrycode is ${countrycode}")
println("UserTags:")
userTags.foreach(x => x.foreach(x => println(f"|-> ${x}")))
```

OUTPUT =



```
PART 2.3
Countrycode is BN
UserTags:
|-> africa,ghana,idds,navrongo
|-> africa,ghana,idds,rice,single mothers
|-> ghana,lab
|-> ghana,lab
|-> ghana,lab
|-> ghana,lab
|-> ghana,lab
```

COMMENT =

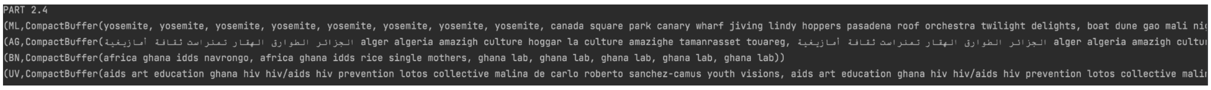
RDD type is **RDD [Country, Iterable[String]]**

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.

CODE =

```
println("PART 2.4")
// Firstly grouped by key and then userTags are flatted, finally they are printed
val country_w_Array_userTags = filtered_pictures.map(x => (x.c,x.userTags)).groupByKey()
val country_w_userTags = country_w_Array_userTags.map(x => (x._1,x._2.flatten))
country_w_userTags.foreach(println)
```

OUTPUT =



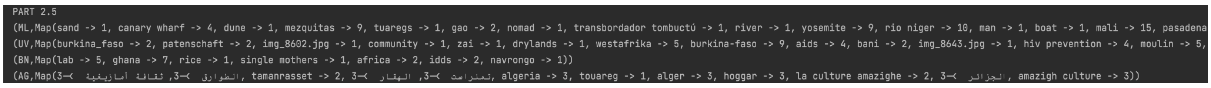
```
PART 2.4
(ML,CompactBuffer(yosemite, yosemite, yosemite, yosemite, yosemite, yosemite, yosemite, yosemite, canada square park canary wharf jiving lindy hoppers pasadena roof orchestra twilight delights, boat dune gao mali nig
(AG,CompactBuffer(الجزائر الطوارق الهقار تمنراست ثقافة أمازيغية alger algeria amazigh culture hoggar la culture amazighe tamanrasset touareg, الجزائر الطوارق الهقار تمنراست ثقافة أمازيغية alger algeria amazigh cultu
(BN,CompactBuffer(africa ghana idds navrongo, africa ghana idds rice single mothers, ghana lab, ghana lab, ghana lab, ghana lab, ghana lab))
(UV,CompactBuffer(aids art education ghana hiv hiv/aids hiv prevention lotos collective malina de carlo roberto sanchez-camus youth visions, aids art education ghana hiv hiv/aids hiv prevention lotos collective malin
```

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.

CODE =

```
println("PART 2.5")
// In this part, I have used groupby(identity) as indicated in the
question, also
// I have used
https://www.java-success.com/10-%E2%99%A5-coding-scala-way-groupby-mapva
lues-identity/ -> for .size
country_w_userTags.map(f=> (f._1, f._2.groupBy(identity).map(y => (y._1,
y._2.size)))).foreach(println)
```

OUTPUT =



```
PART 2.5
(ML,Map(sand -> 1, canary wharf -> 4, dune -> 1, mezquitas -> 9, tuaregs -> 1, gao -> 2, nomad -> 1, transbordador tombuctú -> 1, river -> 1, yosemite -> 9, rio niger -> 10, man -> 1, boat -> 1, mali -> 15, pasadena
(UV,Map(burkina_faso -> 2, patenschaft -> 2, img_8602.jpg -> 1, community -> 1, zai -> 1, drylands -> 1, westafrika -> 5, burkina-faso -> 9, aids -> 4, bani -> 2, img_8643.jpg -> 1, hiv prevention -> 4, moulin -> 5,
(BN,Map(lab -> 5, ghana -> 7, rice -> 1, single mothers -> 1, africa -> 2, idds -> 2, navrongo -> 1))
(AG,Map(3-< ثقافة أمازيغية ,3-< الطوارق, tamanrasset -> 2, 3-< الهقار ,3-< تمنراست, algeria -> 3, touareg -> 1, alger -> 3, hoggar -> 3, la culture amazighe -> 2, 3-< الجزائر, amazigh culture -> 3))
```