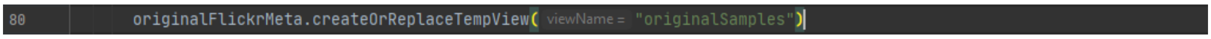
CENG 790: Big Data Analytics, Fall 2020

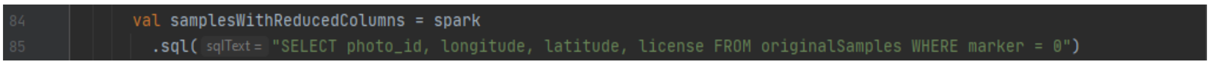
Report of Assignment 1: Introduction to Spark

Part 1: Processing data using the DataFrame API

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



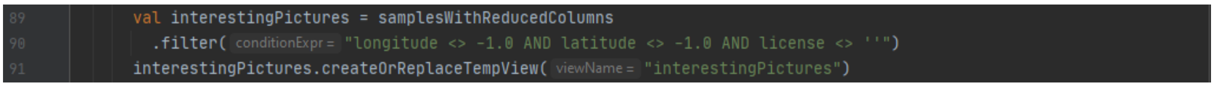
```
originalFlickrMeta.createOrReplaceTempView( viewName = "originalSamples")
```



```
val samplesWithReducedColumns = spark
  .sql( sqlText = "SELECT photo_id, longitude, latitude, license FROM originalSamples WHERE marker = 0")
```

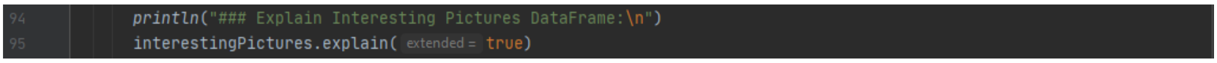
In order to get only pictures, *marker* field must be 0.

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).

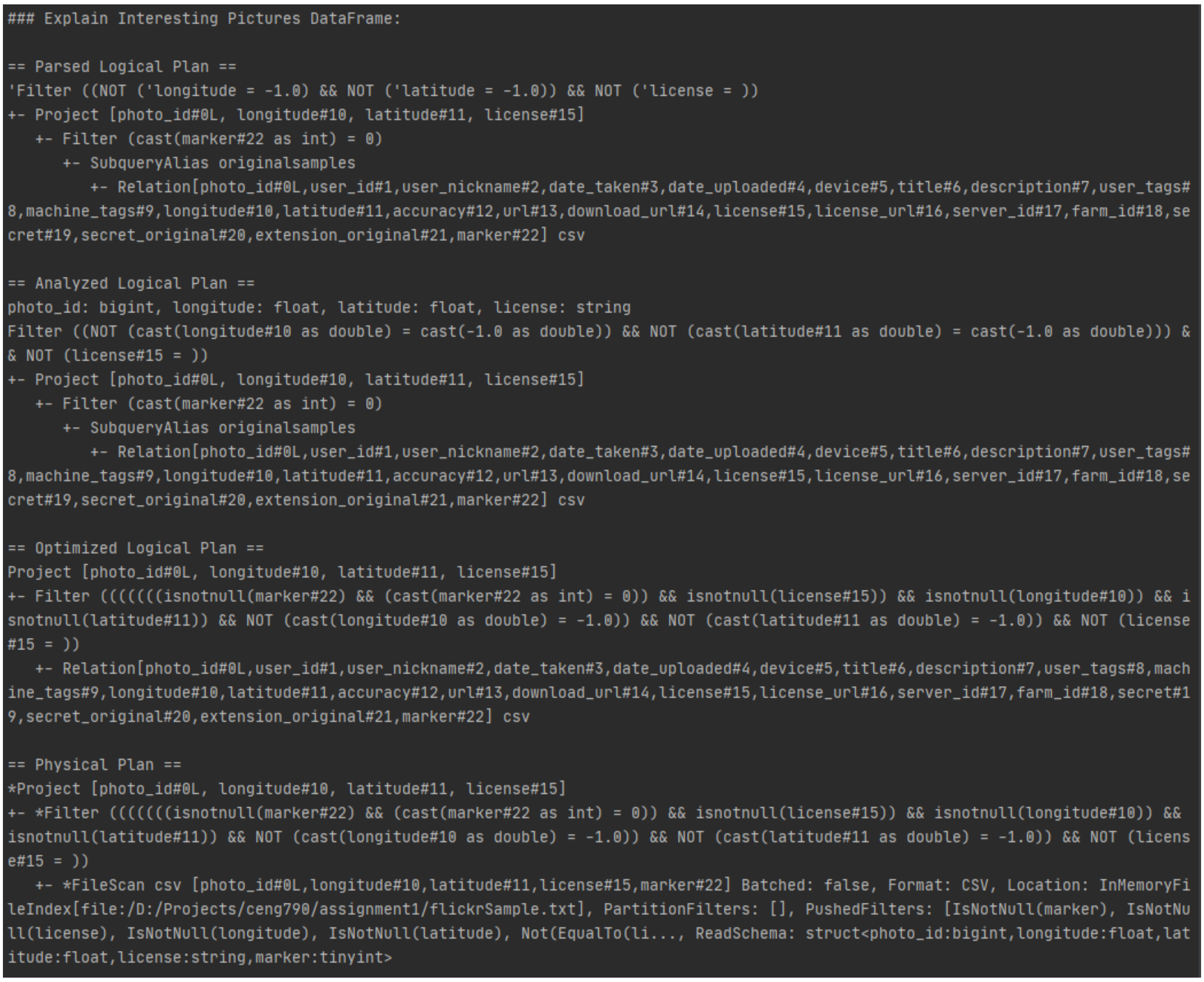


```
val interestingPictures = samplesWithReducedColumns
  .filter( conditionExpr = "longitude <> -1.0 AND latitude <> -1.0 AND license <> ''")
interestingPictures.createOrReplaceTempView( viewName = "interestingPictures")
```

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

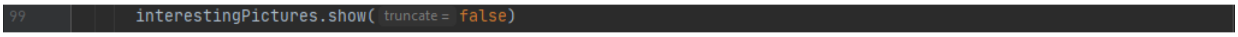


```
println("### Explain Interesting Pictures DataFrame:\n")
interestingPictures.explain( extended = true)
```

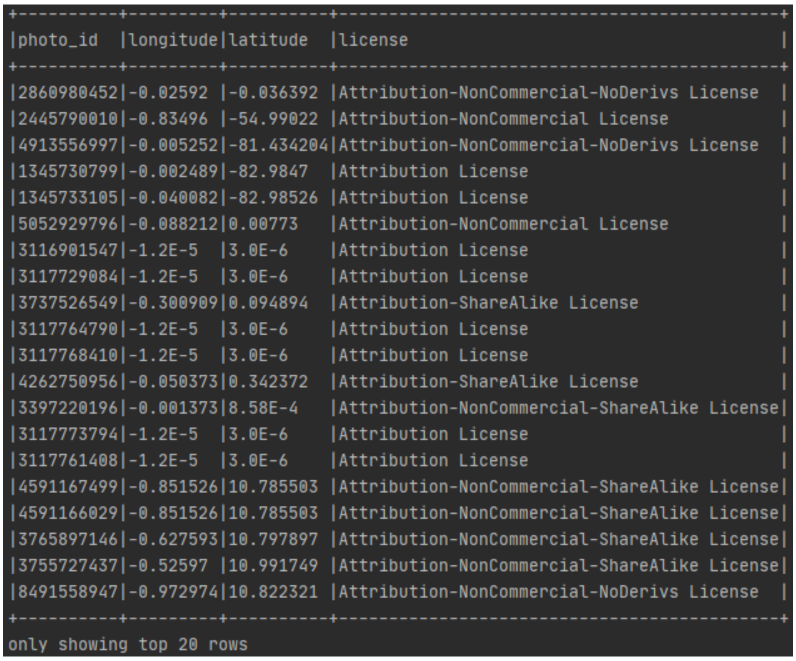


```
### Explain Interesting Pictures DataFrame:

== Parsed Logical Plan ==
'Filter ((NOT ('longitude = -1.0) && NOT ('latitude = -1.0)) && NOT ('license = ))
+- Project [photo_id#0L, longitude#10, latitude#11, license#15]
   +- Filter (cast(marker#22 as int) = 0)
      +- SubqueryAlias originalsamples
         +- Relation[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,se
cret#19,secret_original#20,extension_original#21,marker#22] csv

== Analyzed Logical Plan ==
photo_id: bigint, longitude: float, latitude: float, license: string
Filter ((NOT (cast(longitude#10 as double) = cast(-1.0 as double)) && NOT (cast(latitude#11 as double) = cast(-1.0 as double))) &
& NOT (license#15 = ))
+- Project [photo_id#0L, longitude#10, latitude#11, license#15]
   +- Filter (cast(marker#22 as int) = 0)
      +- SubqueryAlias originalsamples
         +- Relation[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,se
cret#19,secret_original#20,extension_original#21,marker#22] csv

== Optimized Logical Plan ==
Project [photo_id#0L, longitude#10, latitude#11, license#15]
+- Filter (((((((isnotnull(marker#22) && (cast(marker#22 as int) = 0)) && isnotnull(license#15)) && isnotnull(longitude#10)) && i
snotnull(latitude#11)) && NOT (cast(longitude#10 as double) = -1.0)) && NOT (cast(latitude#11 as double) = -1.0)) && NOT (license
#15 = ))
   +- Relation[photo_id#0L,user_id#1,user_nickname#2,date_taken#3,date_uploaded#4,device#5,title#6,description#7,user_tags#8,mach
ine_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#1
9,secret_original#20,extension_original#21,marker#22] csv

== Physical Plan ==
*Project [photo_id#0L, longitude#10, latitude#11, license#15]
+- *Filter (((((((isnotnull(marker#22) && (cast(marker#22 as int) = 0)) && isnotnull(license#15)) && isnotnull(longitude#10)) &&
isnotnull(latitude#11)) && NOT (cast(longitude#10 as double) = -1.0)) && NOT (cast(latitude#11 as double) = -1.0)) && NOT (licens
e#15 = ))
   +- *FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15,marker#22] Batched: false, Format: CSV, Location: InMemoryFi
leIndex[file:/D:/Projects/ceng790/assignment1/flickrSample.txt], PartitionFilters: [], PushedFilters: [IsNotNull(marker), IsNotNu
ll(license), IsNotNull(longitude), IsNotNull(latitude), Not(EqualTo(li..., ReadSchema: struct<photo_id:bigint,longitude:float,lat
itude:float,license:string,marker:tinyint>
```

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.



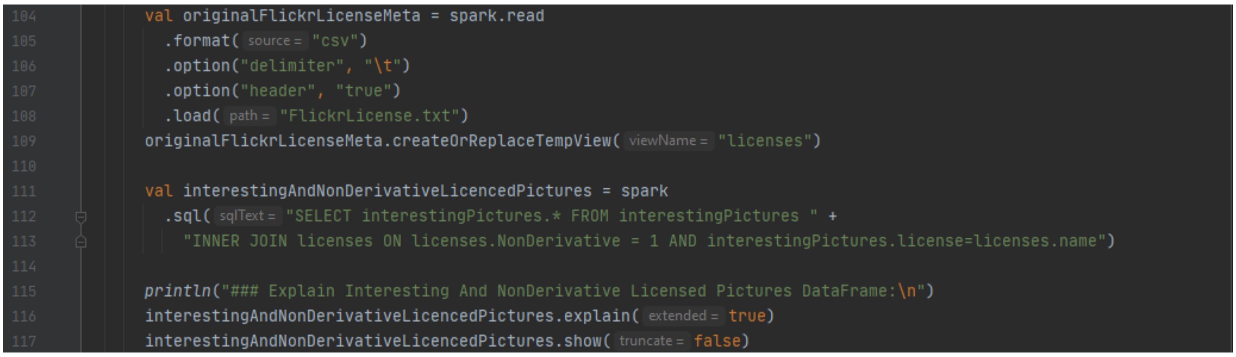
```
interestingPictures.show( truncate = false)
```



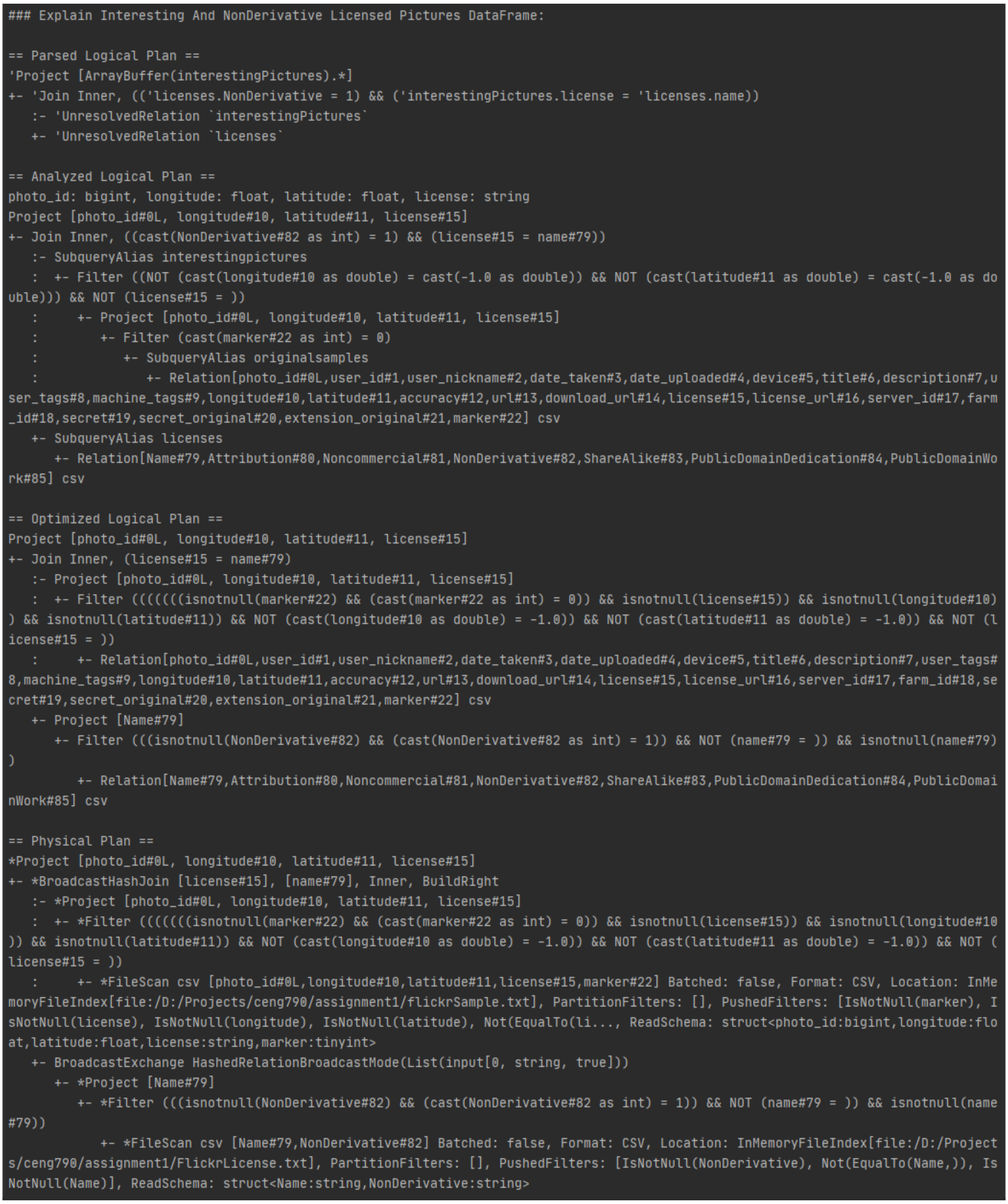
```
+----------+---------+----------+-----------------------------------------+
|photo_id  |longitude|latitude  |license                                  |
+----------+---------+----------+-----------------------------------------+
|2860980452|-0.02592 |-0.036392 |Attribution-NonCommercial-NoDerivs License |
|2445790010|-0.83496 |-54.99022 |Attribution-NonCommercial License        |
|4913556997|-0.005252|-81.434204|Attribution-NonCommercial-NoDerivs License |
|1345730799|-0.002489|-82.9847  |Attribution License                      |
|1345733105|-0.040082|-82.98526 |Attribution License                      |
|5052929796|-0.088212|0.00773   |Attribution-NonCommercial License        |
|3116901547|-1.2E-5  |3.0E-6    |Attribution License                      |
|3117729084|-1.2E-5  |3.0E-6    |Attribution License                      |
|3737526549|-0.300909|0.094894  |Attribution-ShareAlike License           |
|3117764790|-1.2E-5  |3.0E-6    |Attribution License                      |
|3117768410|-1.2E-5  |3.0E-6    |Attribution License                      |
|4262750956|-0.050373|0.342372  |Attribution-ShareAlike License           |
|3397220196|-0.001373|8.58E-4   |Attribution-NonCommercial-ShareAlike License|
|3117773794|-1.2E-5  |3.0E-6    |Attribution License                      |
|3117761408|-1.2E-5  |3.0E-6    |Attribution License                      |
|4591167499|-0.851526|10.785503 |Attribution-NonCommercial-ShareAlike License|
|4591166029|-0.851526|10.785503 |Attribution-NonCommercial-ShareAlike License|
|3765897146|-0.627593|10.797897 |Attribution-NonCommercial-ShareAlike License|
|3755727437|-0.52597 |10.991749 |Attribution-NonCommercial-ShareAlike License|
|8491558947|-0.972974|10.822321 |Attribution-NonCommercial-NoDerivs License |
+----------+---------+----------+-----------------------------------------+
only showing top 20 rows
```

Displayed rows are truncated to 20 rows even though I used *df*.show(truncate = false) to display every row.

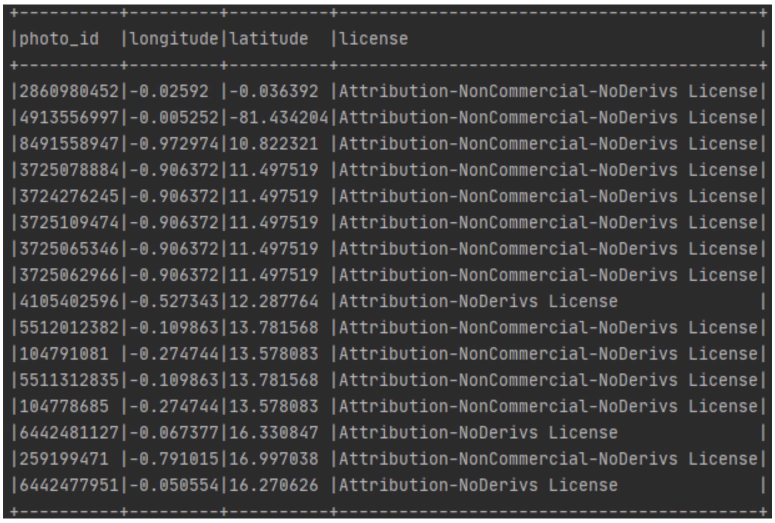
5. Our goal is now to select the pictures whose license is NonDerivative. To this end we will use a second file containing the properties of each license. Load this file in a DataFrame and do a join operation to identify pictures that are both interesting and NonDerivative. Examine the execution plan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



```
val originalFlickrLicenseMeta = spark.read
  .format( source = "csv")
  .option("delimiter", "\t")
  .option("header", "true")
  .load( path = "FlickrLicense.txt")
originalFlickrLicenseMeta.createOrReplaceTempView( viewName = "licenses")

val interestingAndNonDerivativeLicencedPictures = spark
  .sql( sqlText = "SELECT interestingPictures.* FROM interestingPictures " +
    "INNER JOIN licenses ON licenses.NonDerivative = 1 AND interestingPictures.license=licenses.name")

println("### Explain Interesting And NonDerivative Licensed Pictures DataFrame:\n")
interestingAndNonDerivativeLicencedPictures.explain( extended = true)
interestingAndNonDerivativeLicencedPictures.show( truncate = false)
```

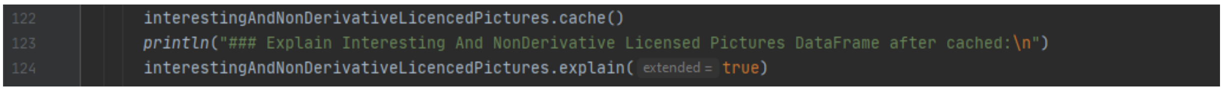


```
### Explain Interesting And NonDerivative Licensed Pictures DataFrame:

== Parsed Logical Plan ==
'Project [ArrayBuffer(interestingPictures).*]
+- 'Join Inner, (('licenses.NonDerivative = 1) && ('interestingPictures.license = 'licenses.name))
   :- 'UnresolvedRelation `interestingPictures`
   +- 'UnresolvedRelation `licenses`

== Analyzed Logical Plan ==
photo_id: bigint, longitude: float, latitude: float, license: string
Project [photo_id#0L, longitude#10, latitude#11, license#15]
+- Join Inner, ((cast(NonDerivative#82 as int) = 1) && (license#15 = name#79))
   :- SubqueryAlias interestingpictures
   :  +- Filter ((NOT (cast(longitude#10 as double) = cast(-1.0 as double)) && NOT (cast(latitude#11 as double) = cast(-1.0 as do
uble))) && NOT (license#15 = ))
   :     +- Project [photo_id#0L, longitude#10, latitude#11, license#15]
   :        +- Filter (cast(marker#22 as int) = 0)
   :           +- SubqueryAlias originalsamples
   :              +- Relation[photo_id#0L,user_id#1,user_nickname#2,date_taken#3,date_uploaded#4,device#5,title#6,description#7,u
ser_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] csv
   +- SubqueryAlias licenses
      +- Relation[Name#79,Attribution#80,Noncommercial#81,NonDerivative#82,ShareAlike#83,PublicDomainDedication#84,PublicDomainWo
rk#85] csv

== Optimized Logical Plan ==
Project [photo_id#0L, longitude#10, latitude#11, license#15]
+- Join Inner, (license#15 = name#79)
   :- Project [photo_id#0L, longitude#10, latitude#11, license#15]
   :  +- Filter ((((((isnotnull(marker#22) && (cast(marker#22 as int) = 0)) && isnotnull(license#15)) && isnotnull(longitude#10)
) && isnotnull(latitude#11)) && NOT (cast(longitude#10 as double) = -1.0)) && NOT (cast(latitude#11 as double) = -1.0)) && NOT (l
icense#15 = ))
   :     +- Relation[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,se
cret#19,secret_original#20,extension_original#21,marker#22] csv
   +- Project [Name#79]
      +- Filter (((isnotnull(NonDerivative#82) && (cast(NonDerivative#82 as int) = 1)) && NOT (name#79 = )) && isnotnull(name#79)
)
         +- Relation[Name#79,Attribution#80,Noncommercial#81,NonDerivative#82,ShareAlike#83,PublicDomainDedication#84,PublicDomai
nWork#85] csv

== Physical Plan ==
*Project [photo_id#0L, longitude#10, latitude#11, license#15]
+- *BroadcastHashJoin [license#15], [name#79], Inner, BuildRight
   :- *Project [photo_id#0L, longitude#10, latitude#11, license#15]
   :  +- *Filter ((((((isnotnull(marker#22) && (cast(marker#22 as int) = 0)) && isnotnull(license#15)) && isnotnull(longitude#10
)) && isnotnull(latitude#11)) && NOT (cast(longitude#10 as double) = -1.0)) && NOT (cast(latitude#11 as double) = -1.0)) && NOT (
license#15 = ))
   :     +- *FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15,marker#22] Batched: false, Format: CSV, Location: InMe
moryFileIndex[file:/D:/Projects/ceng790/assignment1/flickrSample.txt], PartitionFilters: [], PushedFilters: [IsNotNull(marker), I
sNotNull(license), IsNotNull(longitude), IsNotNull(latitude), Not(EqualTo(li..., ReadSchema: struct<photo_id:bigint,longitude:flo
at,latitude:float,license:string,marker:tinyint>
   +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
      +- *Project [Name#79]
         +- *Filter (((isnotnull(NonDerivative#82) && (cast(NonDerivative#82 as int) = 1)) && NOT (name#79 = )) && isnotnull(name
#79))
            +- *FileScan csv [Name#79,NonDerivative#82] Batched: false, Format: CSV, Location: InMemoryFileIndex[file:/D:/Project
s/ceng790/assignment1/FlickrLicense.txt], PartitionFilters: [], PushedFilters: [IsNotNull(NonDerivative), Not(EqualTo(Name,)), Is
NotNull(Name)], ReadSchema: struct<Name:string,NonDerivative:string>
```

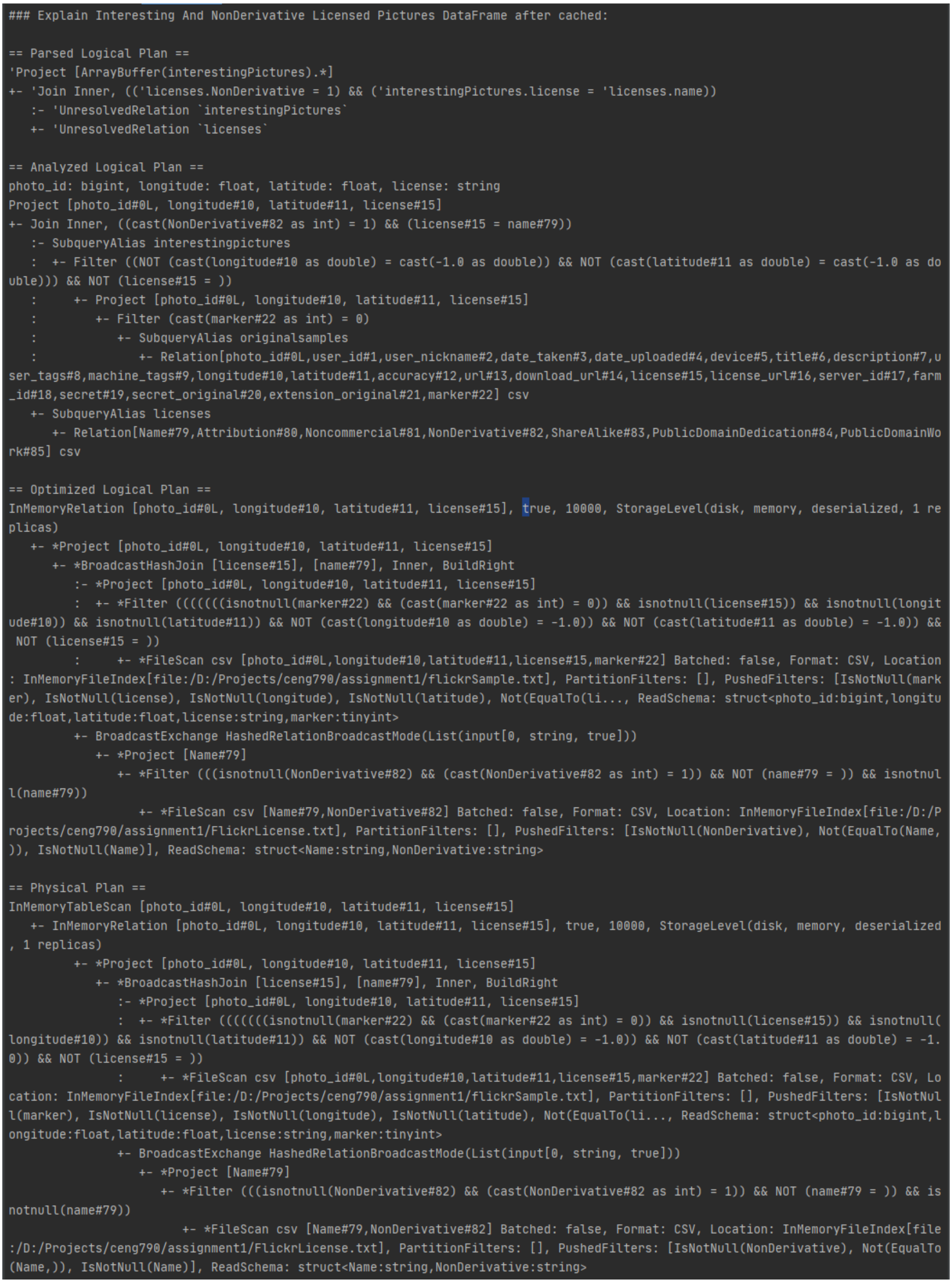


```
+----------+---------+----------+------------------------------------------+
|photo_id  |longitude|latitude  |license                                   |
+----------+---------+----------+------------------------------------------+
|2860980452|-0.02592 |-0.036392 |Attribution-NonCommercial-NoDerivs License|
|4913556997|-0.005252|-81.434204|Attribution-NonCommercial-NoDerivs License|
|8491558947|-0.972974|10.822321 |Attribution-NonCommercial-NoDerivs License|
|3725078884|-0.906372|11.497519 |Attribution-NonCommercial-NoDerivs License|
|3724276245|-0.906372|11.497519 |Attribution-NonCommercial-NoDerivs License|
|3725109474|-0.906372|11.497519 |Attribution-NonCommercial-NoDerivs License|
|3725065346|-0.906372|11.497519 |Attribution-NonCommercial-NoDerivs License|
|3725062966|-0.906372|11.497519 |Attribution-NonCommercial-NoDerivs License|
|4105402596|-0.527343|12.287764 |Attribution-NoDerivs License              |
|5512012382|-0.109863|13.781568 |Attribution-NonCommercial-NoDerivs License|
|104791081 |-0.274744|13.578083 |Attribution-NonCommercial-NoDerivs License|
|5511312835|-0.109863|13.781568 |Attribution-NonCommercial-NoDerivs License|
|104778685 |-0.274744|13.578083 |Attribution-NonCommercial-NoDerivs License|
|6442481127|-0.067377|16.330847 |Attribution-NoDerivs License              |
|259199471 |-0.791015|16.997038 |Attribution-NonCommercial-NoDerivs License|
|6442477951|-0.050554|16.270626 |Attribution-NoDerivs License              |
+----------+---------+----------+------------------------------------------+
```

6. 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?



```
interestingAndNonDerivativeLicencedPictures.cache()
println("### Explain Interesting And NonDerivative Licensed Pictures DataFrame after cached:\n")
interestingAndNonDerivativeLicencedPictures.explain( extended = true)
```

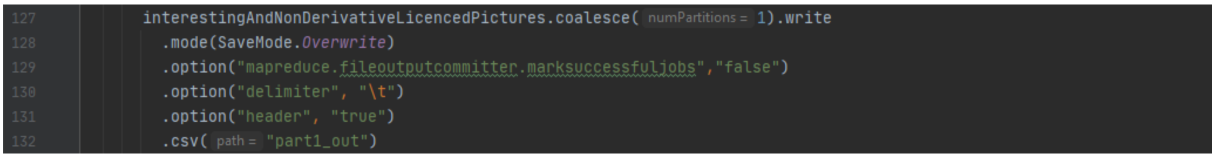


```
### Explain Interesting And NonDerivative Licensed Pictures DataFrame after cached:

== Parsed Logical Plan ==
'Project [ArrayBuffer(interestingPictures).*]
+- 'Join Inner, (('licenses.NonDerivative = 1) && ('interestingPictures.license = 'licenses.name))
   :- 'UnresolvedRelation `interestingPictures`
   +- 'UnresolvedRelation `licenses`

== Analyzed Logical Plan ==
photo_id: bigint, longitude: float, latitude: float, license: string
Project [photo_id#0L, longitude#10, latitude#11, license#15]
+- Join Inner, ((cast(NonDerivative#82 as int) = 1) && (license#15 = name#79))
   :- SubqueryAlias interestingpictures
   :  +- Filter ((NOT (cast(longitude#10 as double) = cast(-1.0 as double)) && NOT (cast(latitude#11 as double) = cast(-1.0 as do
uble))) && NOT (license#15 = ))
   :     +- Project [photo_id#0L, longitude#10, latitude#11, license#15]
   :        +- Filter (cast(marker#22 as int) = 0)
   :           +- SubqueryAlias originalsamples
   :              +- Relation[photo_id#0L,user_id#1,user_nickname#2,date_taken#3,date_uploaded#4,device#5,title#6,description#7,u
ser_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] csv
   +- SubqueryAlias licenses
      +- Relation[Name#79,Attribution#80,Noncommercial#81,NonDerivative#82,ShareAlike#83,PublicDomainDedication#84,PublicDomainWo
rk#85] csv

== Optimized Logical Plan ==
InMemoryRelation [photo_id#0L, longitude#10, latitude#11, license#15], true, 10000, StorageLevel(disk, memory, deserialized, 1 re
plicas)
   +- *Project [photo_id#0L, longitude#10, latitude#11, license#15]
      +- *BroadcastHashJoin [license#15], [name#79], Inner, BuildRight
         :- *Project [photo_id#0L, longitude#10, latitude#11, license#15]
         :  +- *Filter ((((((isnotnull(marker#22) && (cast(marker#22 as int) = 0)) && isnotnull(license#15)) && isnotnull(longit
ude#10)) && isnotnull(latitude#11)) && NOT (cast(longitude#10 as double) = -1.0)) && NOT (cast(latitude#11 as double) = -1.0)) &&
 NOT (license#15 = ))
         :     +- *FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15,marker#22] Batched: false, Format: CSV, Location
: InMemoryFileIndex[file:/D:/Projects/ceng790/assignment1/flickrSample.txt], PartitionFilters: [], PushedFilters: [IsNotNull(mark
er), IsNotNull(license), IsNotNull(longitude), IsNotNull(latitude), Not(EqualTo(li..., ReadSchema: struct<photo_id:bigint,longitu
de:float,latitude:float,license:string,marker:tinyint>
         +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
            +- *Project [Name#79]
               +- *Filter (((isnotnull(NonDerivative#82) && (cast(NonDerivative#82 as int) = 1)) && NOT (name#79 = )) && isnotnul
l(name#79))
                  +- *FileScan csv [Name#79,NonDerivative#82] Batched: false, Format: CSV, Location: InMemoryFileIndex[file:/D:/P
rojects/ceng790/assignment1/FlickrLicense.txt], PartitionFilters: [], PushedFilters: [IsNotNull(NonDerivative), Not(EqualTo(Name,
)), IsNotNull(Name)], ReadSchema: struct<Name:string,NonDerivative:string>

== Physical Plan ==
InMemoryTableScan [photo_id#0L, longitude#10, latitude#11, license#15]
   +- InMemoryRelation [photo_id#0L, longitude#10, latitude#11, license#15], true, 10000, StorageLevel(disk, memory, deserialized
, 1 replicas)
         +- *Project [photo_id#0L, longitude#10, latitude#11, license#15]
            +- *BroadcastHashJoin [license#15], [name#79], Inner, BuildRight
               :- *Project [photo_id#0L, longitude#10, latitude#11, license#15]
               :  +- *Filter ((((((isnotnull(marker#22) && (cast(marker#22 as int) = 0)) && isnotnull(license#15)) && isnotnull(
longitude#10)) && isnotnull(latitude#11)) && NOT (cast(longitude#10 as double) = -1.0)) && NOT (cast(latitude#11 as double) = -1.
0)) && NOT (license#15 = ))
               :     +- *FileScan csv [photo_id#0L,longitude#10,latitude#11,license#15,marker#22] Batched: false, Format: CSV, Lo
cation: InMemoryFileIndex[file:/D:/Projects/ceng790/assignment1/flickrSample.txt], PartitionFilters: [], PushedFilters: [IsNotNul
l(marker), IsNotNull(license), IsNotNull(longitude), IsNotNull(latitude), Not(EqualTo(li..., ReadSchema: struct<photo_id:bigint,l
ongitude:float,latitude:float,license:string,marker:tinyint>
               +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
                  +- *Project [Name#79]
                     +- *Filter (((isnotnull(NonDerivative#82) && (cast(NonDerivative#82 as int) = 1)) && NOT (name#79 = )) && is
notnull(name#79))
                        +- *FileScan csv [Name#79,NonDerivative#82] Batched: false, Format: CSV, Location: InMemoryFileIndex[file
:/D:/Projects/ceng790/assignment1/FlickrLicense.txt], PartitionFilters: [], PushedFilters: [IsNotNull(NonDerivative), Not(EqualTo
(Name,)), IsNotNull(Name)], ReadSchema: struct<Name:string,NonDerivative:string>
```

Parsed Logical Plan and Analyzed Logical Plan are the same in both DataFrames’ execution plans. However, Optimized Logical Plans and Physical Plans of cached and uncached DataFrames have differences such as:

- Cached DataFrame have “*” characters (i.e. *Project, *Filter) in Optimized Logical Plan. I am not sure but “*” character might mean that the transformations are already executed.
- Cached DataFrame’s Optimized Logical Plan and Physical Plan has lines starting with “*InMemoryRelation*” and “*InMemoryTableScan*” which can be interpreted as the table and the relation is retrieved from the memory.
- Cached DataFrame’s Optimized Logical Plan have “*BroadcastHashJoin*” and “*BroadcastExchange*” instead of “*Join*”, “*FileScan*” instead of “*Relation*”.
- Cached DataFrame’s Physical Plan is almost identical to the uncached DataFrame’s plan. Only difference is the use of “InMemoryTableScan” and “InMemoryRelation”.

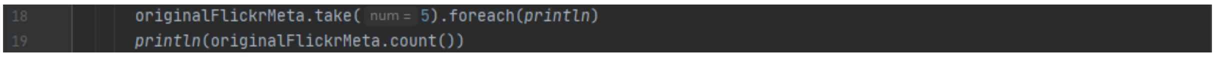
7. Save the final result in a csv file (write). Don’t forget to add a header to reuse it more easily.



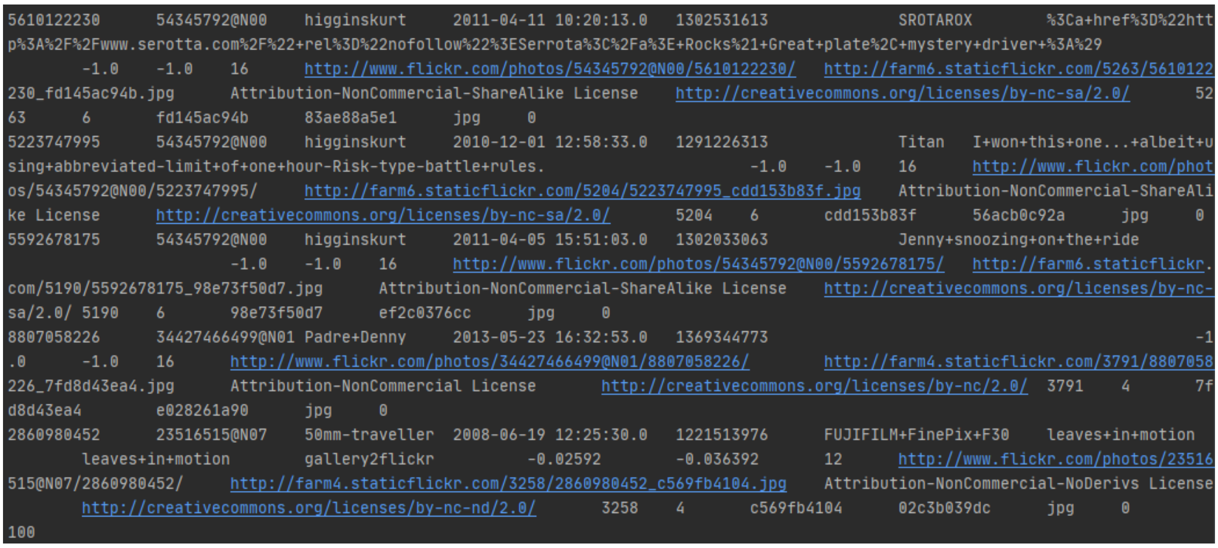
```
    interestingAndNonDerivativeLicencedPictures.coalesce( numPartitions = 1).write
      .mode(SaveMode.Overwrite)
      .option("mapreduce.fileoutputcommitter.marksuccessfuljobs","false")
      .option("delimiter", "\t")
      .option("header", "true")
      .csv( path = "part1_out")
```

Part 2: Processing data using RDDs

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

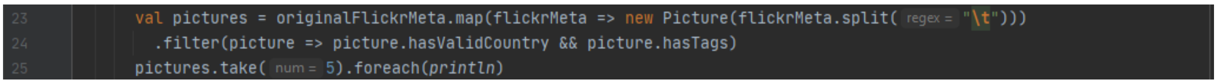


```
    originalFlickrMeta.take( num = 5).foreach(println)
    println(originalFlickrMeta.count())
```

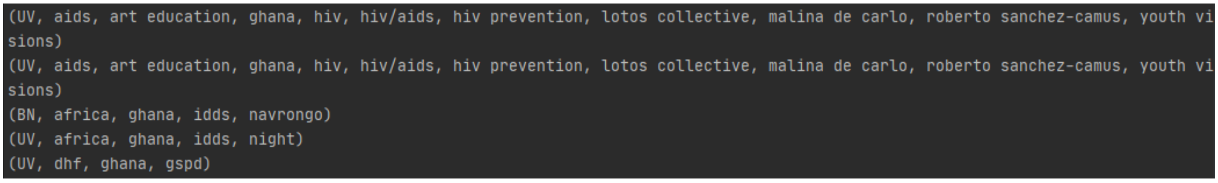


```
5610122230      54345792@N00    higginskurt     2011-04-11 10:20:13.0   1302531613              SROTAROX        %3Ca+href%3D%22htt
p%3A%2F%2Fwww.serotta.com%2F%22+rel%3D%22nofollow%22%3ESerrota%3C%2Fa%3E+Rocks%21+Great+plate%2C+mystery+driver+%3A%29
        -1.0    -1.0    16      http://www.flickr.com/photos/54345792@N00/5610122230/   http://farm6.staticflickr.com/5263/5610122
230_fd145ac94b.jpg      Attribution-NonCommercial-ShareAlike License    http://creativecommons.org/licenses/by-nc-sa/2.0/       52
63      6       fd145ac94b      83ae88a5e1      jpg     0
5223747995      54345792@N00    higginskurt     2010-12-01 12:58:33.0   1291226313              Titan   I+won+this+one...+albeit+u
sing+abbreviated-limit+of+one+hour-Risk-type-battle+rules.                              -1.0    -1.0    16      http://www.flickr.com/phot
os/54345792@N00/5223747995/     http://farm6.staticflickr.com/5204/5223747995_cdd153b83f.jpg    Attribution-NonCommercial-ShareAli
ke License      http://creativecommons.org/licenses/by-nc-sa/2.0/       5204    6       cdd153b83f      56acb0c92a      jpg     0
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.staticflickr.
com/5190/5592678175_98e73f50d7.jpg      Attribution-NonCommercial-ShareAlike License    http://creativecommons.org/licenses/by-nc-
sa/2.0/ 5190    6       98e73f50d7      ef2c0376cc      jpg     0
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/3791/8807058
226_7fd8d43ea4.jpg      Attribution-NonCommercial License       http://creativecommons.org/licenses/by-nc/2.0/  3791    4       7f
d8d43ea4        e028261a90      jpg     0
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/photos/23516
515@N07/2860980452/     http://farm4.staticflickr.com/3258/2860980452_c569fb4104.jpg    Attribution-NonCommercial-NoDerivs License
        http://creativecommons.org/licenses/by-nc-nd/2.0/       3258    4       c569fb4104      02c3b039dc      jpg     0
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 5 elements.

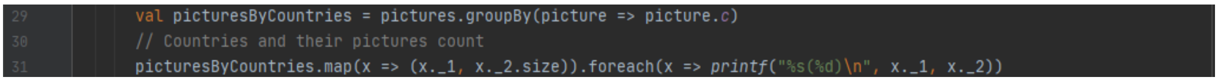


```
    val pictures = originalFlickrMeta.map(flickrMeta => new Picture(flickrMeta.split( regex = "\t")))
      .filter(picture => picture.hasValidCountry && picture.hasTags)
    pictures.take( num = 5).foreach(println)
```

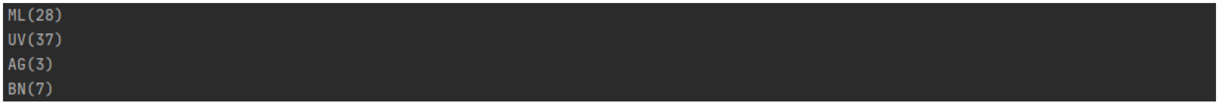


```
(UV, aids, art education, ghana, hiv, hiv/aids, hiv prevention, lotos collective, malina de carlo, roberto sanchez-camus, youth vi
sions)
(UV, aids, art education, ghana, hiv, hiv/aids, hiv prevention, lotos collective, malina de carlo, roberto sanchez-camus, youth vi
sions)
(BN, africa, ghana, idds, 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?



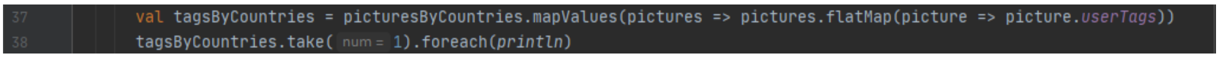
```
val picturesByCountries = pictures.groupBy(picture => picture.c)
// Countries and their pictures count
picturesByCountries.map(x => (x._1, x._2.size)).foreach(x => printf("%s(%d)\n", x._1, x._2))
```



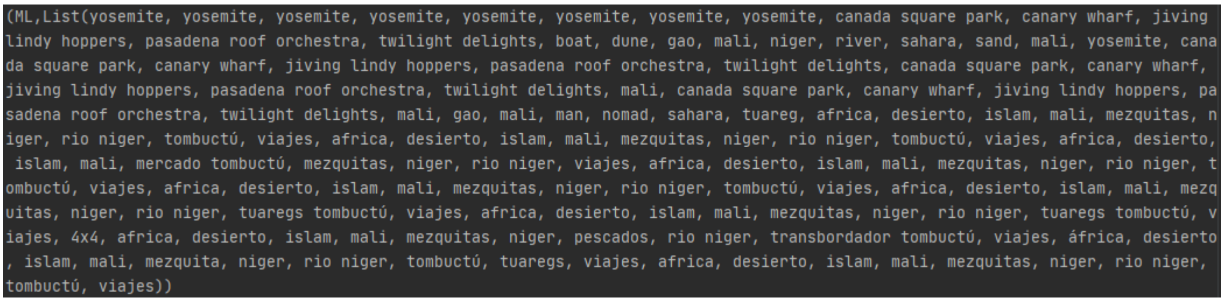
```
ML(28)
UV(37)
AG(3)
BN(7)
```

The type of picturesByCountries is RDD[(Country, Iterable[Picture])]. Additionally, I printed picture counts of countries.

4. 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.

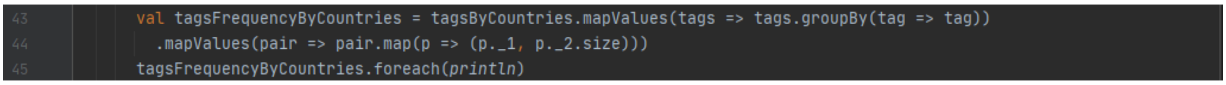


```
val tagsByCountries = picturesByCountries.mapValues(pictures => pictures.flatMap(picture => picture.userTags))
tagsByCountries.take( num = 1).foreach(println)
```

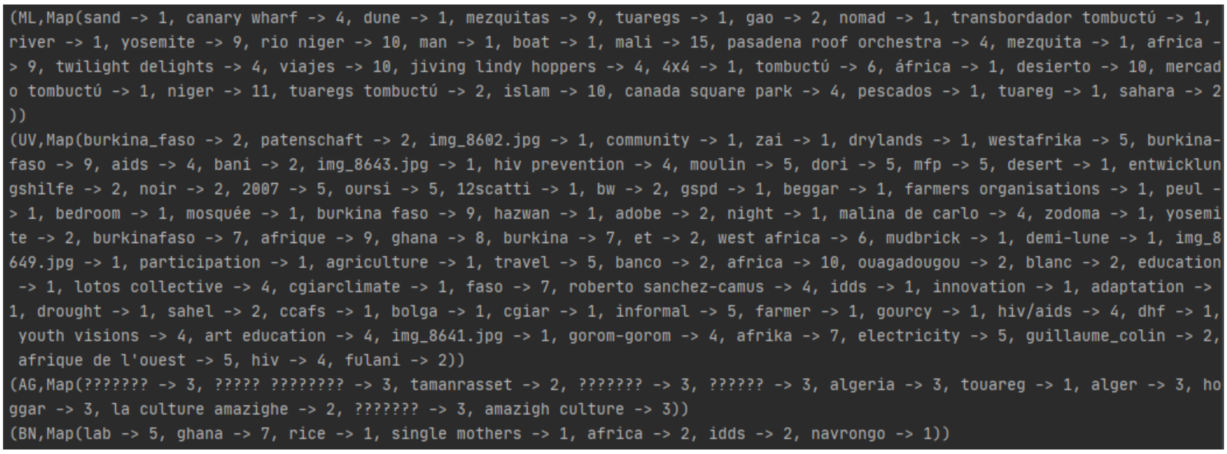


```
(ML,List(yosemite, yosemite, yosemite, yosemite, yosemite, yosemite, yosemite, yosemite, canada square park, canary wharf, jiving
lindy hoppers, pasadena roof orchestra, twilight delights, boat, dune, gao, mali, niger, river, sahara, sand, mali, yosemite, cana
da square park, canary wharf, jiving lindy hoppers, pasadena roof orchestra, twilight delights, canada square park, canary wharf,
jiving lindy hoppers, pasadena roof orchestra, twilight delights, mali, canada square park, canary wharf, jiving lindy hoppers, pa
sadena roof orchestra, twilight delights, mali, gao, mali, man, nomad, sahara, tuareg, africa, desierto, islam, mali, mezquitas, n
iger, rio niger, tombuctú, viajes, africa, desierto, islam, mali, mezquitas, niger, rio niger, tombuctú, viajes, africa, desierto,
 islam, mali, mercado tombuctú, mezquitas, niger, rio niger, viajes, africa, desierto, islam, mali, mezquitas, niger, rio niger, t
ombuctú, viajes, africa, desierto, islam, mali, mezquitas, niger, rio niger, tombuctú, viajes, africa, desierto, islam, mali, mezq
uitas, niger, rio niger, tuaregs tombuctú, viajes, africa, desierto, islam, mali, mezquitas, niger, rio niger, tuaregs tombuctú, v
iajes, 4x4, africa, desierto, islam, mali, mezquitas, niger, pescados, rio niger, transbordador tombuctú, viajes, áfrica, desierto
, islam, mali, mezquita, niger, rio niger, tombuctú, tuaregs, viajes, africa, desierto, islam, mali, mezquitas, niger, rio niger,
tombuctú, viajes))
```

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.



```
val tagsFrequencyByCountries = tagsByCountries.mapValues(tags => tags.groupBy(tag => tag))
  .mapValues(pair => pair.map(p => (p._1, p._2.size)))
tagsFrequencyByCountries.foreach(println)
```



```
(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 roof orchestra -> 4, mezquita -> 1, africa -
> 9, twilight delights -> 4, viajes -> 10, jiving lindy hoppers -> 4, 4x4 -> 1, tombuctú -> 6, áfrica -> 1, desierto -> 10, mercad
o tombuctú -> 1, niger -> 11, tuaregs tombuctú -> 2, islam -> 10, canada square park -> 4, pescados -> 1, tuareg -> 1, sahara -> 2
))
(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, dori -> 5, mfp -> 5, desert -> 1, entwicklun
gshilfe -> 2, noir -> 2, 2007 -> 5, oursi -> 5, 12scatti -> 1, bw -> 2, gspd -> 1, beggar -> 1, farmers organisations -> 1, peul -
> 1, bedroom -> 1, mosquée -> 1, burkina faso -> 9, hazwan -> 1, adobe -> 2, night -> 1, malina de carlo -> 4, zodoma -> 1, yosemi
te -> 2, burkinafaso -> 7, afrique -> 9, ghana -> 8, burkina -> 7, et -> 2, west africa -> 6, mudbrick -> 1, demi-lune -> 1, img_8
649.jpg -> 1, participation -> 1, agriculture -> 1, travel -> 5, banco -> 2, africa -> 10, ouagadougou -> 2, blanc -> 2, education
 -> 1, lotos collective -> 4, cgiarclimate -> 1, faso -> 7, roberto sanchez-camus -> 4, idds -> 1, innovation -> 1, adaptation ->
1, drought -> 1, sahel -> 2, ccafs -> 1, bolga -> 1, cgiar -> 1, informal -> 5, farmer -> 1, gourcy -> 1, hiv/aids -> 4, dhf -> 1,
 youth visions -> 4, art education -> 4, img_8641.jpg -> 1, gorom-gorom -> 4, afrika -> 7, electricity -> 5, guillaume_colin -> 2,
 afrique de l'ouest -> 5, hiv -> 4, fulani -> 2))
(AG,Map(??????? -> 3, ????? ???????? -> 3, tamanrasset -> 2, ??????? -> 3, ?????? -> 3, algeria -> 3, touareg -> 1, alger -> 3, ho
ggar -> 3, la culture amazighe -> 2, ??????? -> 3, amazigh culture -> 3))
(BN,Map(lab -> 5, ghana -> 7, rice -> 1, single mothers -> 1, africa -> 2, idds -> 2, navrongo -> 1))
```

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?

Instead of Creating RDD[(Country, Map[String, Int])], we could create RDD[(Country, String)] where String represents the tag then we could group by each country, tag pair or simply RDD[(Country, String, Int)] where String still represents the tag and Int is the frequency of that tag in the country.