KTH Royal Institute of Technology
Stockholm

School of Electrical Engineering and
Computer Science

Data-Intensive Computing - ID2221

Lab 1 - Report

Author
Emil Ståhl

Author
Selemawit Fsha

September 26, 2021

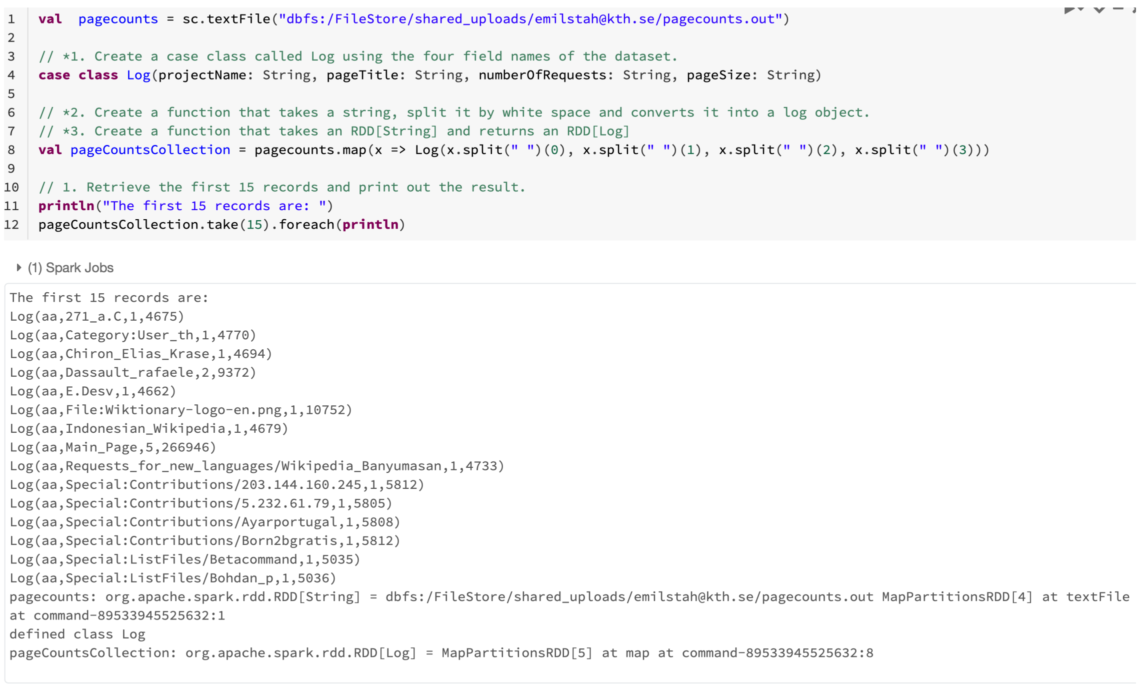
Lab 1 - Report - ID2221

Emil Ståhl and Selemawit Fsha

September 2021

1 Task 1 - Spark

1.1 Retrieve the first 15 records and print out the result



```
val  pagecounts = sc.textFile("dbfs:/FileStore/shared_uploads/emilstah@kth.se/pagecounts.out")

// *1. Create a case class called Log using the four field names of the dataset.
case class Log(projectName: String, pageTitle: String, numberOfRequests: String, pageSize: String)

// *2. Create a function that takes a string, split it by white space and converts it into a log object.
// *3. Create a function that takes an RDD[String] and returns an RDD[Log]
val pageCountsCollection = pagecounts.map(x => Log(x.split(" ")(0), x.split(" ")(1), x.split(" ")(2), x.split(" ")(3)))

// 1. Retrieve the first 15 records and print out the result.
println("The first 15 records are: ")
pageCountsCollection.take(15).foreach(println)
```

(1) Spark Jobs

```
The first 15 records are:
Log(aa,271_a.C,1,4675)
Log(aa,Category:User_th,1,4770)
Log(aa,Chiron_Elias_Krase,1,4694)
Log(aa,Dassault_rafaele,2,9372)
Log(aa,E.Desv,1,4662)
Log(aa,File:Wiktionary-logo-en.png,1,10752)
Log(aa,Indonesian_Wikipedia,1,4679)
Log(aa,Main_Page,5,266946)
Log(aa,Requests_for_new_languages/Wikipedia_Banyumasan,1,4733)
Log(aa,Special:Contributions/203.144.160.245,1,5812)
Log(aa,Special:Contributions/5.232.61.79,1,5805)
Log(aa,Special:Contributions/Ayarportugal,1,5808)
Log(aa,Special:Contributions/Born2bgratis,1,5812)
Log(aa,Special:ListFiles/Betacommand,1,5035)
Log(aa,Special:ListFiles/Bohdan_p,1,5036)
pagecounts: org.apache.spark.rdd.RDD[String] = dbfs:/FileStore/shared_uploads/emilstah@kth.se/pagecounts.out MapPartitionsRDD[4] at textFile
at command-89533945525632:1
defined class Log
pageCountsCollection: org.apache.spark.rdd.RDD[Log] = MapPartitionsRDD[5] at map at command-89533945525632:8
```

1.2 Determine the number of records the dataset has in total

```scala
// 2. Determine the number of records the dataset has in total.
val totalNumbeOfRecords = pageCountsCollection.count()
println("Total number of records in the dataset is " + totalNumbeOfRecords)
```

▸ (1) Spark Jobs

```
Total number of records in the dataset is 3324129
```

1.3 Compute the min, max, and average page size

```scala
// 3. Compute the min, max, and average page size.
val maxPages = pageCountsCollection.reduce((acc,value) => {
  if(acc.pageSize < value.pageSize) value else acc})
println("The maximum page size is " + maxPages.pageSize)

val minPages = pageCountsCollection.reduce((acc,value) => {
  if(acc.pageSize > value.pageSize) value else acc})
println("The minimum page size is " + minPages.pageSize)

val average = pageCountsCollection.map(_.pageSize.toLong).sum/totalNumbeOfRecords.toLong
println("The average page size is " + average)
```

▸ (3) Spark Jobs

```
The maximum page size is 99999
The minimum page size is 0
The average page size is 132239.56957446598
```

1.4 Determine the record(s) with the largest page size. If multiple records have the same size, list all of them

```scala
// 4. Determine the record(s) with the largest page size. If multiple records have the same size, list all of
val largePageSize = pageCountsCollection.filter(page => page.pageSize == maxPages.pageSize)
println("List of records with largest page size: ")
largePageSize.collect.foreach(println)
```

▸ (1) Spark Jobs

```
List of records with largest page size:
Log(en,DC_Super_Hero_Girls,7,99999)
Log(en,Old_Masters,1,99999)
Log(en,Patricia_Roc,5,99999)
Log(en,User_talk:5_albert_square,7,99999)
Log(en,Whitney:_The_Greatest_Hits,2,99999)
largePageSize: org.apache.spark.rdd.RDD[Log] = MapPartitionsRDD[9] at filter at command-89533945525637:2
```

1.5 Determine the record with the largest page size again. But now, pick the most popular

```scala
// 5. Determine the record with the largest page size again. But now, pick the most popular.
val largePageSizeAndPopular = largePageSize.reduce((acc,value) => {
  if(acc.numberOfRequests < value.numberOfRequests) value else acc})
println("Records with largest page size and that are the most popular " + largePageSizeAndPopular)
```

▸ (1) Spark Jobs

```
Records with largest page size and that are the most popular Log(en,DC_Super_Hero_Girls,7,99999)
largePageSizeAndPopular: Log = Log(en,DC_Super_Hero_Girls,7,99999)
```

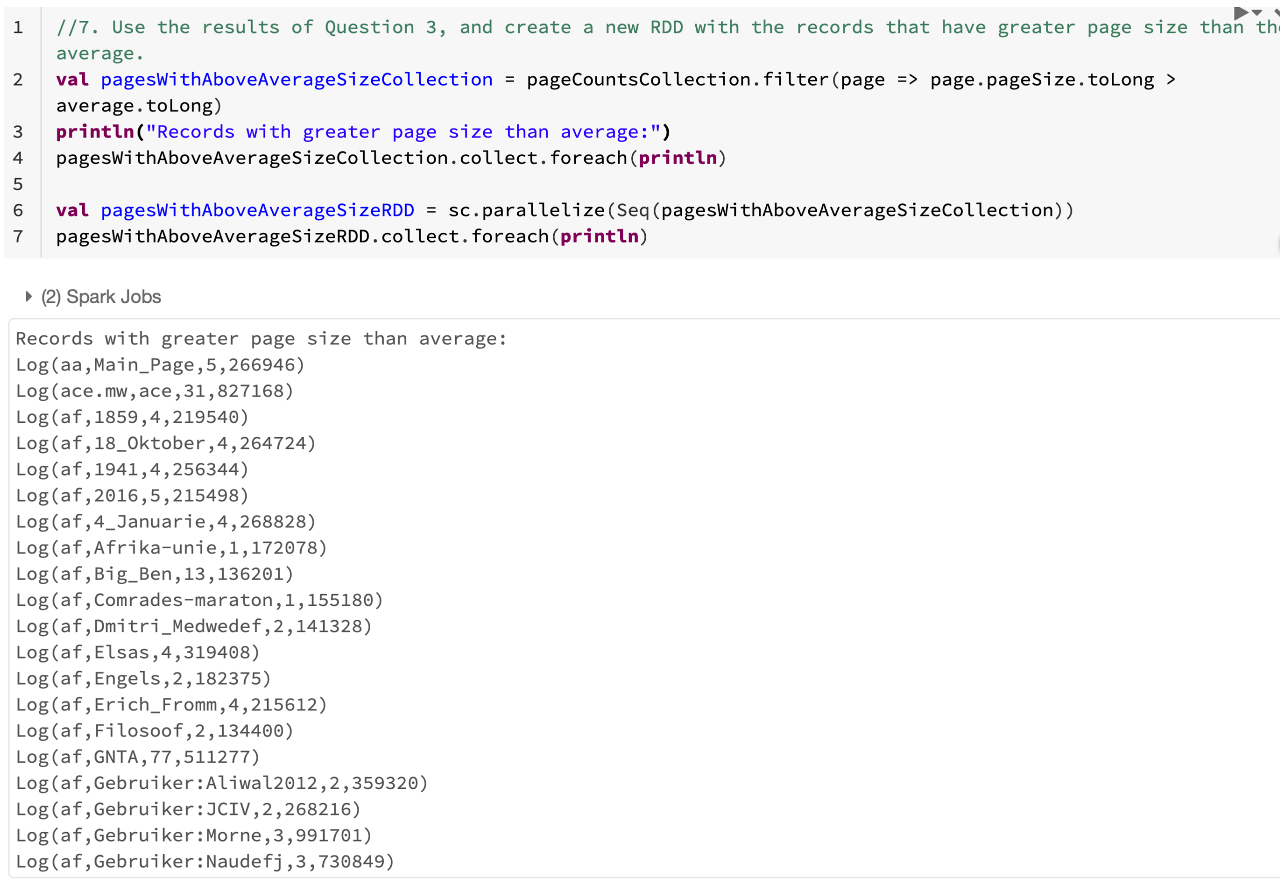
1.6 Determine the record(s) with the largest page title. If multiple titles have the same length, list all of them

```scala
// 6. Determine the record(s) with the largest page title. If multiple titles have the same length, list all of
them.
val maxTitleLength = pageCountsCollection.reduce((acc,value) => {
  if(acc.pageTitle.length() < value.pageTitle.length()) value else acc})

val largePageTitle = pageCountsCollection.filter(page => page.pageTitle.length() ==
maxTitleLength.pageTitle.length())

println("List of records with longest page title: ")
largePageTitle.collect.foreach(println)
```

▸ (2) Spark Jobs

```
List of records with longest page title:
Log(zh,Special:e8b18ee6baafefbda5efbdbfe89cb7e6829fefbdbfe88b93e29980e89e9fefbda9e89eb3efbda425636f256d6725736f2573732
56f38257373256f38257373256f38256b6d73efbdaa256e6b256678256f6b2c687474703a2f2f7777772e653662313966653861356266656f2d6f3
53930386365356266393765383831386165356134613965356165613426636f2e6d672e732e736f2e382e73736f386b2e6d2e372e73736f3873736
f386b6d37332e752e622e61616e6b66786f6b2e70772f2ce8b18ee6baafefbda5efbdbfe89cb7e6829fefbdbfe88b93e29980e89e9fefbda9e89eb
3efbda425636f256d6725736f257373256f38257373256f38257373256f38256b6d73efbdaa256e6b256678256f6b/,1,6043)
```

1.7 Use the results of Question 3, and create a new RDD with the records that have greater page size than the average



```
//7. Use the results of Question 3, and create a new RDD with the records that have greater page size than th
average.
val pagesWithAboveAverageSizeCollection = pageCountsCollection.filter(page => page.pageSize.toLong >
average.toLong)
println("Records with greater page size than average:")
pagesWithAboveAverageSizeCollection.collect.foreach(println)

val pagesWithAboveAverageSizeRDD = sc.parallelize(Seq(pagesWithAboveAverageSizeCollection))
pagesWithAboveAverageSizeRDD.collect.foreach(println)
```

(2) Spark Jobs

```
Records with greater page size than average:
Log(aa,Main_Page,5,266946)
Log(ace.mw,ace,31,827168)
Log(af,1859,4,219540)
Log(af,18_Oktober,4,264724)
Log(af,1941,4,256344)
Log(af,2016,5,215498)
Log(af,4_Januarie,4,268828)
Log(af,Afrika-unie,1,172078)
Log(af,Big_Ben,13,136201)
Log(af,Comrades-maraton,1,155180)
Log(af,Dmitri_Medwedef,2,141328)
Log(af,Elsas,4,319408)
Log(af,Engels,2,182375)
Log(af,Erich_Fromm,4,215612)
Log(af,Filosoof,2,134400)
Log(af,GNTA,77,511277)
Log(af,Gebruiker:Aliwal2012,2,359320)
Log(af,Gebruiker:JCIV,2,268216)
Log(af,Gebruiker:Morne,3,991701)
Log(af,Gebruiker:Naudefj,3,730849)
```

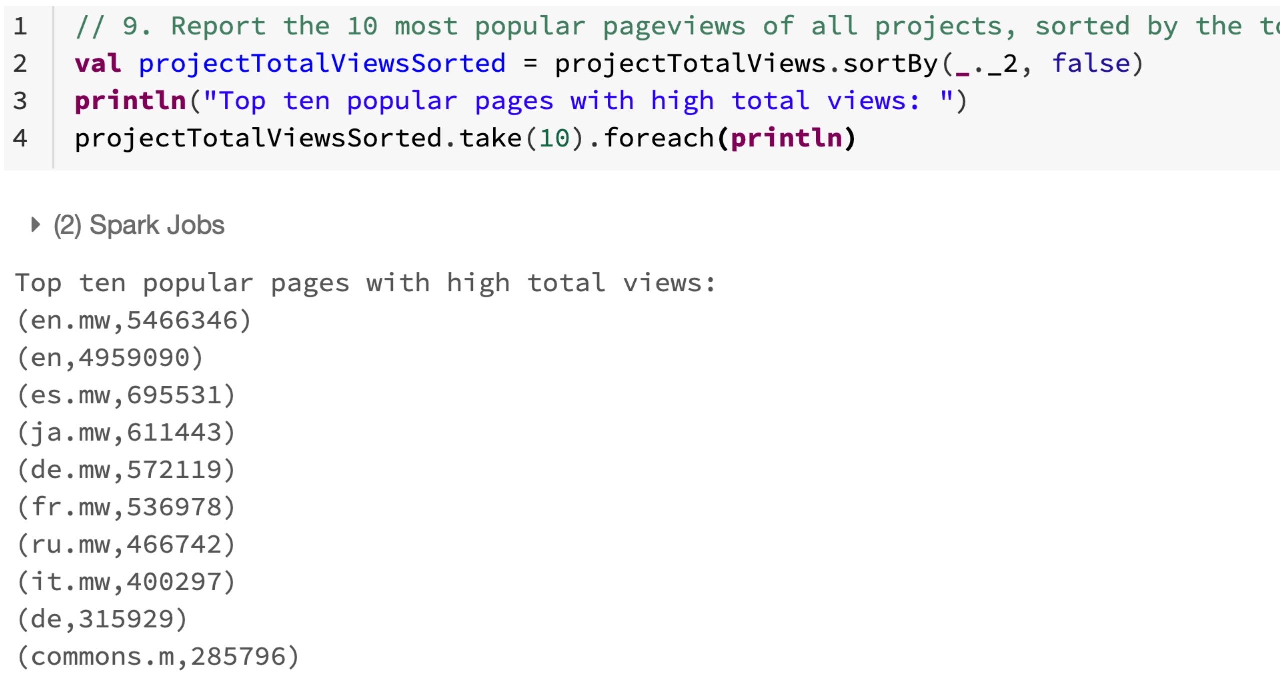
1.8 Compute the total number of pageviews for each project (as the schema shows, the first field of each record contains the project code)

```scala
// 8. Compute the total number of pageviews for each project (as the schema shows, the first field of each rec
val projectViewCollection = pageCountsCollection.map(page => (page.projectName, page.numberOfRequests.toInt))
//projectViewCollection.take(15).foreach(println)
val projectTotalViews = projectViewCollection.reduceByKey((x,y)=>x+y)
projectTotalViews.collect().foreach(println)
```

▸ (1) Spark Jobs

```
(tr.mw,125999)
(nso,108)
(it.s,1444)
(lb.mw,158)
(ckb,25)
(sk.mw,9548)
(hak,54)
(frp.mw,11)
(ik.d,1)
(ik,57)
(yi.mw,70)
(az.q,9)
(mk.b,2)
(ak.v,2)
(ky.b,2)
(pt.voy,30)
(fr.v,264)
(ca.v,1)
(eml.mw,30)
(xh.d,8)
(bxr,12)
```

1.9 Report the 10 most popular pageviews of all projects, sorted by the total number of hits

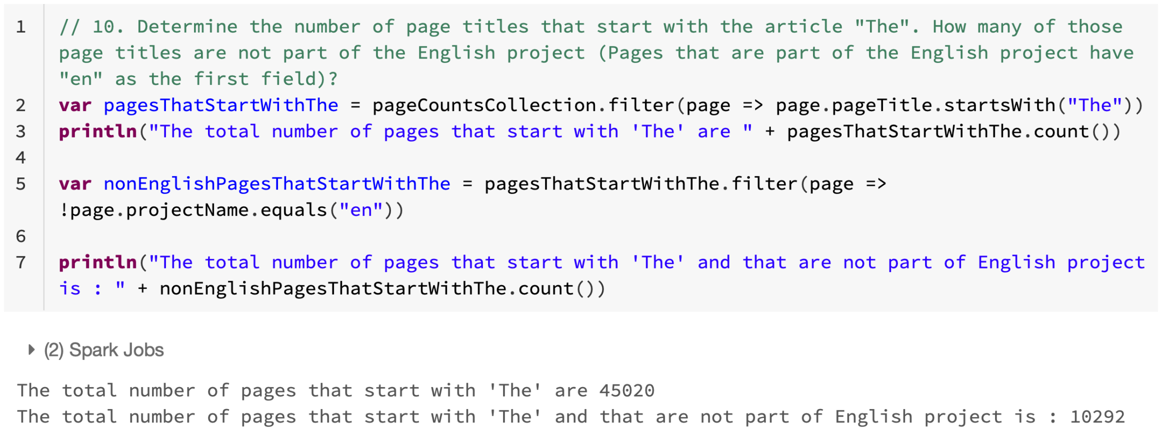


```scala
// 9. Report the 10 most popular pageviews of all projects, sorted by the to
val projectTotalViewsSorted = projectTotalViews.sortBy(_._2, false)
println("Top ten popular pages with high total views: ")
projectTotalViewsSorted.take(10).foreach(println)
```

(2) Spark Jobs

```
Top ten popular pages with high total views:
(en.mw,5466346)
(en,4959090)
(es.mw,695531)
(ja.mw,611443)
(de.mw,572119)
(fr.mw,536978)
(ru.mw,466742)
(it.mw,400297)
(de,315929)
(commons.m,285796)
```

1.10 Determine the number of page titles that start with the article "The". How many of those page titles are not part of the English project (Pages that are part of the English project have "en" as the first field)?

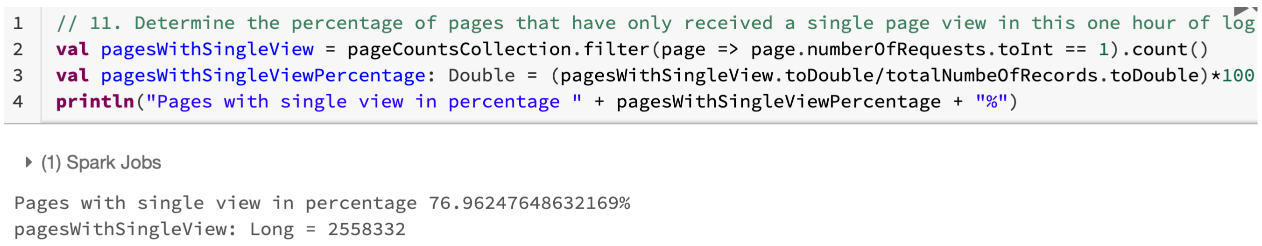


```scala
// 10. Determine the number of page titles that start with the article "The". How many of those
page titles are not part of the English project (Pages that are part of the English project have
"en" as the first field)?
var pagesThatStartWithThe = pageCountsCollection.filter(page => page.pageTitle.startsWith("The"))
println("The total number of pages that start with 'The' are " + pagesThatStartWithThe.count())

var nonEnglishPagesThatStartWithThe = pagesThatStartWithThe.filter(page =>
!page.projectName.equals("en"))

println("The total number of pages that start with 'The' and that are not part of English project
is : " + nonEnglishPagesThatStartWithThe.count())
```

(2) Spark Jobs

```
The total number of pages that start with 'The' are 45020
The total number of pages that start with 'The' and that are not part of English project is : 10292
```

1.11 Determine the percentage of pages that have only received a single page view in this one hour of log data

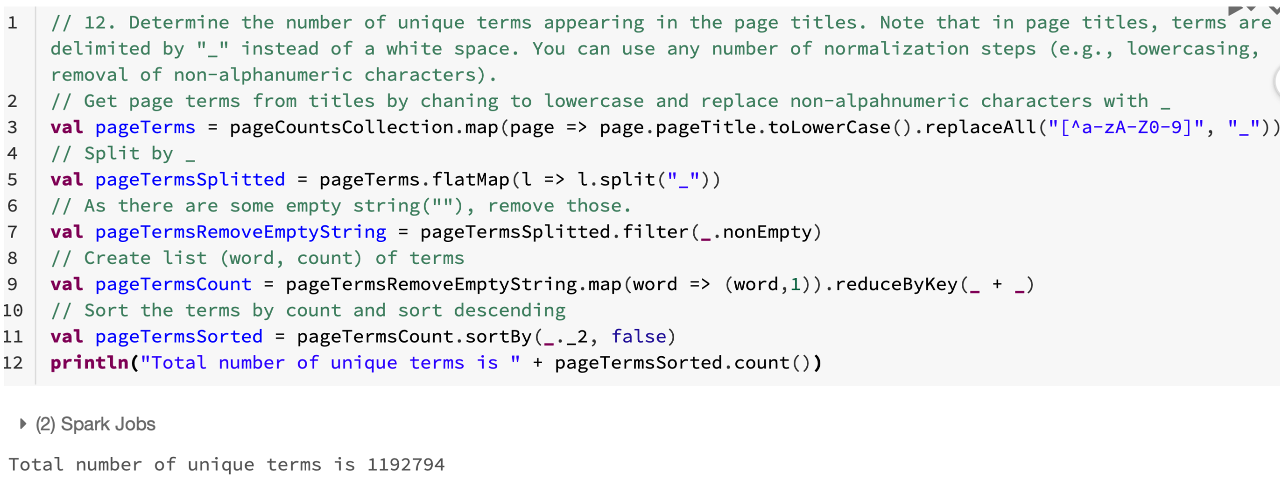


```
// 11. Determine the percentage of pages that have only received a single page view in this one hour of log
val pagesWithSingleView = pageCountsCollection.filter(page => page.numberOfRequests.toInt == 1).count()
val pagesWithSingleViewPercentage: Double = (pagesWithSingleView.toDouble/totalNumbeOfRecords.toDouble)*100
println("Pages with single view in percentage " + pagesWithSingleViewPercentage + "%")
```

▸ (1) Spark Jobs

```
Pages with single view in percentage 76.96247648632169%
pagesWithSingleView: Long = 2558332
```

1.12 Determine the number of unique terms appearing in the page titles. Note that in page titles, terms are delimited by “ ” instead of a white space. You can use any number of normalization steps (e.g., lowercasing, removal of non-alphanumeric characters)

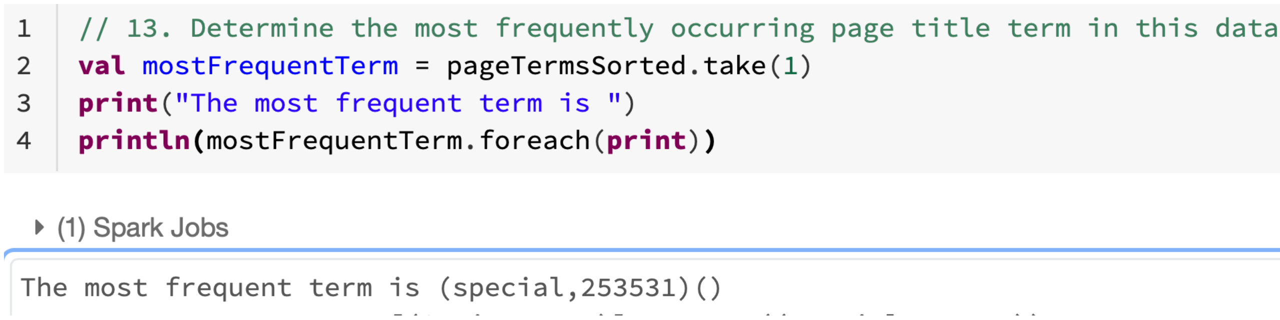


```
// 12. Determine the number of unique terms appearing in the page titles. Note that in page titles, terms are
delimited by "_" instead of a white space. You can use any number of normalization steps (e.g., lowercasing,
removal of non-alphanumeric characters).
// Get page terms from titles by chaning to lowercase and replace non-alpahnumeric characters with _
val pageTerms = pageCountsCollection.map(page => page.pageTitle.toLowerCase().replaceAll("[^a-zA-Z0-9]", "_"))
// Split by _
val pageTermsSplitted = pageTerms.flatMap(l => l.split("_"))
// As there are some empty string(""), remove those.
val pageTermsRemoveEmptyString = pageTermsSplitted.filter(_.nonEmpty)
// Create list (word, count) of terms
val pageTermsCount = pageTermsRemoveEmptyString.map(word => (word,1)).reduceByKey(_ + _)
// Sort the terms by count and sort descending
val pageTermsSorted = pageTermsCount.sortBy(_._2, false)
println("Total number of unique terms is " + pageTermsSorted.count())
```

▸ (2) Spark Jobs

```
Total number of unique terms is 1192794
```

1.13 Determine the most frequently occurring page title term in this dataset



```scala
// 13. Determine the most frequently occurring page title term in this data
val mostFrequentTerm = pageTermsSorted.take(1)
print("The most frequent term is ")
println(mostFrequentTerm.foreach(print))
```

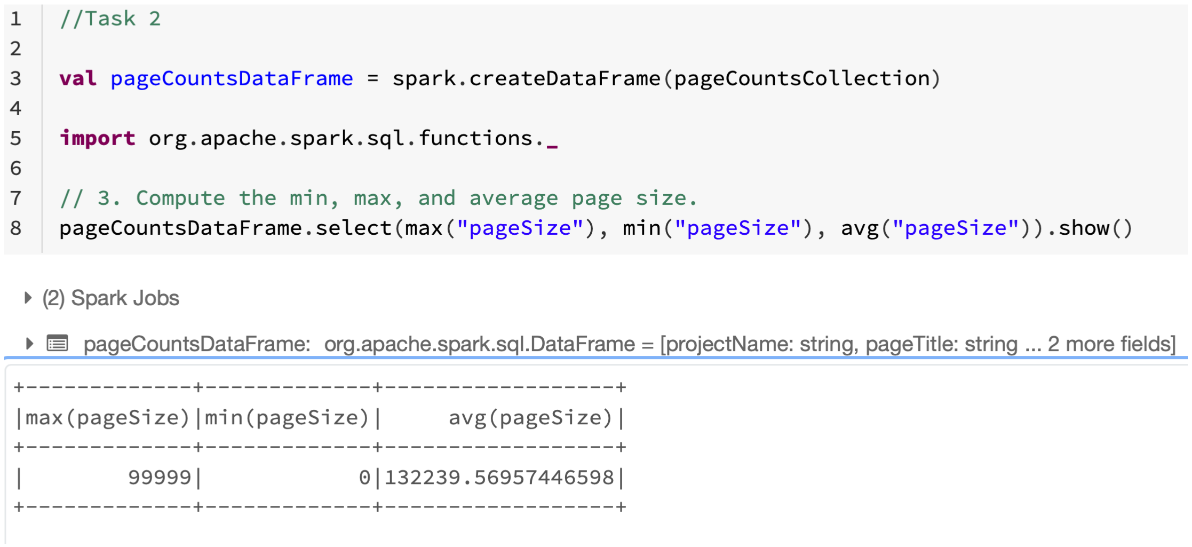
▸ (1) Spark Jobs

```
The most frequent term is (special,253531)()
```

2 Task 2 - Spark SQL

First, convert the pagecounts from RDD[String] into DataFrame (hint: you may need to transform RDD[String] into RDD[Log] and then DataFrame). Next, you must use your DataFrame to answer again to questions 3, 5, 7, 12, 13 of Task 1, but this time by running SQL queries programmatically.

2.3 Compute the min, max, and average page size



```scala
//Task 2

val pageCountsDataFrame = spark.createDataFrame(pageCountsCollection)

import org.apache.spark.sql.functions._

// 3. Compute the min, max, and average page size.
pageCountsDataFrame.select(max("pageSize"), min("pageSize"), avg("pageSize")).show()
```

▸ (2) Spark Jobs

▸ pageCountsDataFrame: org.apache.spark.sql.DataFrame = [projectName: string, pageTitle: string ... 2 more fields]

```
+-------------+-------------+------------------+
|max(pageSize)|min(pageSize)|     avg(pageSize)|
+-------------+-------------+------------------+
|        99999|            0|132239.56957446598|
+-------------+-------------+------------------+
```

2.5 Determine the record with the largest page size again. But now, pick the most popular

```
// 5. Determine the record with the largest page size again. But now, pick the most popular.
pageCountsDataFrame.sort(col("pageSize").desc,col("numberOfRequests").desc).show(1)
```

▸ (1) Spark Jobs

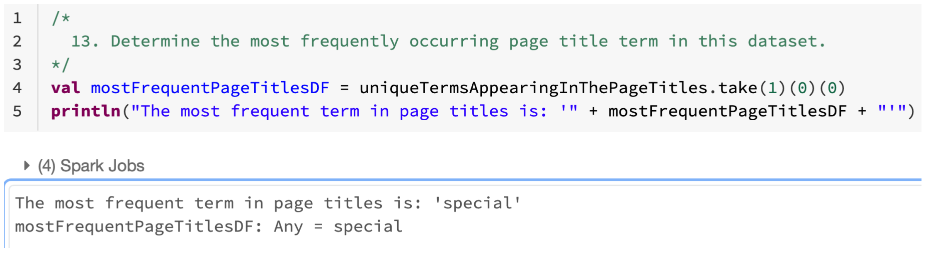
```
+-----------+-------------------+----------------+--------+
|projectName|          pageTitle|numberOfRequests|pageSize|
+-----------+-------------------+----------------+--------+
|         en|DC_Super_Hero_Girls|               7|   99999|
+-----------+-------------------+----------------+--------+
only showing top 1 row
```

2.12 Determine the number of unique terms appearing in the page titles

```scala
/*
  12.  Determine the number of unique terms appearing
        in the page titles. Note that in page titles, terms
        are delimited by \ " instead of a white space.
        You can use any number of normalization steps (e.g.,
        lowercasing, removal of non-alphanumeric characters).
*/

val pageCountsDataFrameToLower = pageCountsDataFrame.withColumn("pageTitle", lower(col("pageTitle")))

def replaceNonAlphaNumeric: String => String = _.replaceAll("[^a-zA-Z0-9 ]", "_")

val replaceNonAlphaNumericUDF = udf(replaceNonAlphaNumeric)

val pageCountsDataFrameNormalised = pageCountsDataFrameToLower.withColumn("pageTitle", replaceNonAlphaNumericUDF($"pageTitle"))

val wordsToRow = pageCountsDataFrameNormalised.withColumn("pageTitle", explode(split($"pageTitle", "\\_")))

val pageTitleWordCount = (wordsToRow.groupBy("pageTitle").count())
val pageTitleWordCountSorted = pageTitleWordCount.sort(col("count").desc)
val emptyStringRow = pageTitleWordCountSorted.first()

val uniqueTermsAppearingInThePageTitles = pageTitleWordCountSorted.filter(row => row != emptyStringRow)

uniqueTermsAppearingInThePageTitles.show()
val totalNumberOfUniqueTerms = uniqueTermsAppearingInThePageTitles.count
println("The total number of unique terms in page titles is " + totalNumberOfUniqueTerms)
```

```
+-------------------+------+
|          pageTitle| count|
+-------------------+------+
|            special|253531|
|               file|235019|
|                 of|195358|
|                jpg|192957|
|                the|129314|
|           category|125642|
|               talk|106441|
|                 in| 83470|
|               user| 74471|
|         entitydata| 72370|
|                 de| 67041|
|      whatlinkshere| 62358|
|recentchangeslinked| 48433|
|                and| 42169|
|                svg| 38486|
|               list| 38094|
|           template| 30138|
|                png| 30056|
|               film| 27253|
|                  a| 27057|
+-------------------+------+
only showing top 20 rows
The total number of unique terms in page titles is 1192794
```

2.13 Determine the most frequently occurring page title term in this dataset



```scala
/*
  13. Determine the most frequently occurring page title term in this dataset.
*/
val mostFrequentPageTitlesDF = uniqueTermsAppearingInThePageTitles.take(1)(0)(0)
println("The most frequent term in page titles is: '" + mostFrequentPageTitlesDF + "'")
```

```
The most frequent term in page titles is: 'special'
mostFrequentPageTitlesDF: Any = special
```