```
/*
Lab 1
Data-intensive computing
Emil Ståhl
Selemawit Fsha
Date: 2021-09-26
Task 1
 */

val pagecounts = sc.textFile("/FileStore/tables/
pagecounts_20160101_000000_parsed-1.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)
// 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)

// 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)

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

// 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)

// 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)

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

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

// 8. Compute the total number of pageviews for each project (as the
schema shows, the first field of each record contains the project
code).
val projectViewCollection = pageCountsCollection.map(page =>
(page.projectName, page.numberOfRequests.toInt))
val projectTotalViews =
projectViewCollection.reduceByKey((x,y)=>x+y)

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

// 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"))
// var pagesThatStartWithTheCount = pagesThatStartWithThe.count()
// pagesThatStartWithThe.take(3).foreach(println)
// println(pagesThatStartWithThe)
println("The total number of pages that start with 'The' are " +
pagesThatStartWithThe.count())
```

```
var nonEnglishPagesThatStartWithThe =
pagesThatStartWithThe.filter(page => !page.projectName.equals("en"))
// var nonEnglishPagesThatStartWithTheCount =
nonEnglishPagesThatStartWithThe.count()
// pagesThatStartWithThe.take(3).foreach(println)
println("The total number of pages that start with 'The' and that
are not part of English project is : " +
nonEnglishPagesThatStartWithThe.count())

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


// 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())


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

/*
Task 2
*/
val pageCountsDataFrame =
spark.createDataFrame(pageCountsCollection)
```

```
// pageCountsDataFrame.show()
import org.apache.spark.sql.functions._

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

// 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)

// 7. Use the results of Question 3, and create a new RDD with the
records that have greater page size than the average.
val avgPageSize =
pageCountsDataFrame.select(avg($"pageSize")).take(1)(0)(0)

val pageSizeGreaterThanAvgRDD =
pageCountsDataFrame.filter($"pagesize" > avgPageSize).rdd

pageSizeGreaterThanAvgRDD.take(15).foreach(println)


/*
  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)


/*
  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)
```