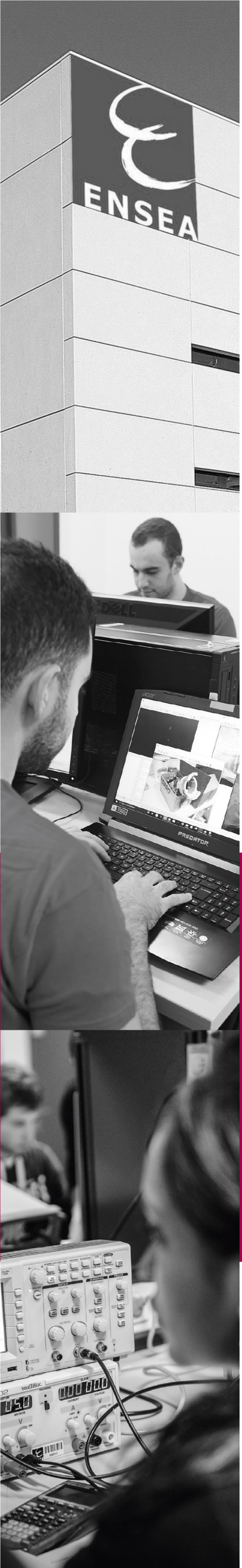


Beyond Engineering

AI & BIG DATA

Big Data

Lab report : Big Log Data Analytics

2nd year student

Erwan BROUDIN
16 février 2023

Useful informations

With this report, we return two ipynb files :

— The first one, named BIG_Data_TP2, contains the code we made for this exercise.

— The second one, named BIG_Data_TP2_Timer contains the code we have rewritten, with timers that display the times of each question.

We had to make two documents because if we kept only the BIG_Data_TP2_Timer file, we could not see the RDDs' displays with certain functions such as take.

We also calculated the execution time of the whole notebook by trying to set a timer at the very beginning, one at the very end and displaying the time taken by doing a global evaluation.

We can therefore state that the execution time of the entire notebook is 692.461 seconds (which is equivalent to 11 minutes and 32 seconds).

Exercise n°0 : Download and Prepare your Log File

In this part, we have to download the .zip file in our Google Drive. Then, we will use the following to code to unzip the file :

```python
import zipfile
from google.colab import drive
drive.mount('/content/drive')
!apt install unzip
!unzip -u "/content/drive/MyDrive/pagecounts-20160101-000000_parsed.out.zip" -d "/content/drive/MyDrive/"
```

Code 1 – Unzipping the file (the page views of Wikipedia projects)

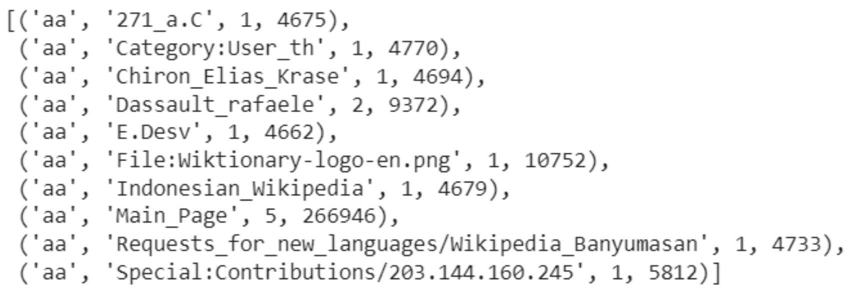
Obviously, we could have done the unzipping part directly in the Drive without any coding, but we found it more interesting to use the method shown in the tutorial document on Moodle.

Afterward, we can load the document as an RDD :

```python
from pyspark import SparkFiles

Wiki_url = "/content/drive/MyDrive/pagecounts-20160101-000000_parsed.out"
spark.sparkContext.addFile(Wiki_url)
Wiki_rdd_v1=sc.textFile(SparkFiles.get("pagecounts-20160101-000000_parsed.out"))
```

Code 2 – Loading the file as an RDD

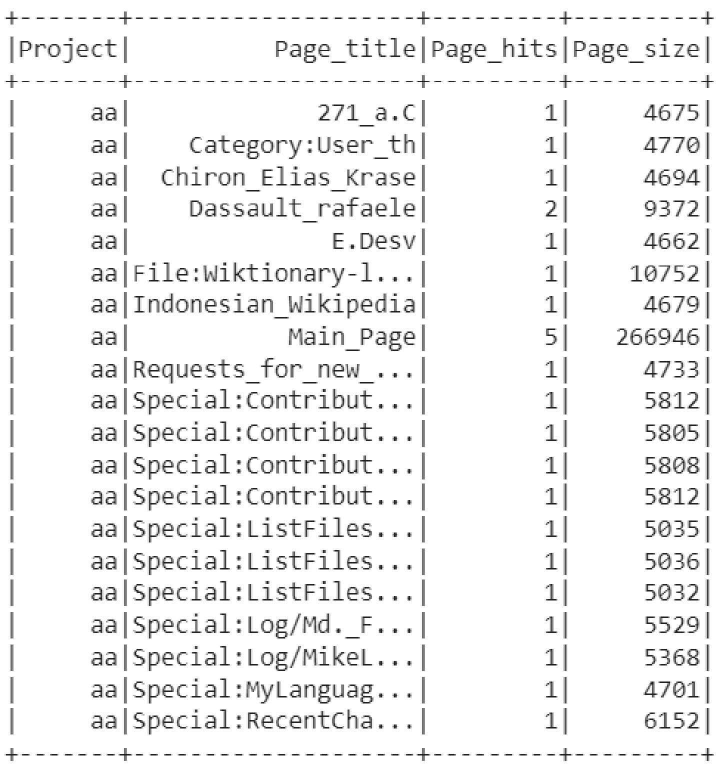
And this is the RDD we get :



```
[('aa', '271_a.C', 1, 4675),
 ('aa', 'Category:User_th', 1, 4770),
 ('aa', 'Chiron_Elias_Krase', 1, 4694),
 ('aa', 'Dassault_rafaele', 2, 9372),
 ('aa', 'E.Desv', 1, 4662),
 ('aa', 'File:Wiktionary-logo-en.png', 1, 10752),
 ('aa', 'Indonesian_Wikipedia', 1, 4679),
 ('aa', 'Main_Page', 5, 266946),
 ('aa', 'Requests_for_new_languages/Wikipedia_Banyumasan', 1, 4733),
 ('aa', 'Special:Contributions/203.144.160.245', 1, 5812)]
```

Image 1 – Original RDD

Using the **createDataFrame** function, we now create the dataframe for exercise two and obtain :



```
+-------+--------------------+---------+---------+
|Project|          Page_title|Page_hits|Page_size|
+-------+--------------------+---------+---------+
|     aa|             271_a.C|        1|     4675|
|     aa|    Category:User_th|        1|     4770|
|     aa|   Chiron_Elias_Krase|        1|     4694|
|     aa|    Dassault_rafaele|        2|     9372|
|     aa|              E.Desv|        1|     4662|
|     aa|File:Wiktionary-l...|        1|    10752|
|     aa|Indonesian_Wikipedia|        1|     4679|
|     aa|           Main_Page|        5|   266946|
|     aa|Requests_for_new_...|        1|     4733|
|     aa|Special:Contribut...|        1|     5812|
|     aa|Special:Contribut...|        1|     5805|
|     aa|Special:Contribut...|        1|     5808|
|     aa|Special:Contribut...|        1|     5812|
|     aa|Special:ListFiles...|        1|     5035|
|     aa|Special:ListFiles...|        1|     5036|
|     aa|Special:ListFiles...|        1|     5032|
|     aa|Special:Log/Md._F...|        1|     5529|
|     aa|Special:Log/MikeL...|        1|     5368|
|     aa|Special:MyLanguag...|        1|     4701|
|     aa|Special:RecentCha...|        1|     6152|
+-------+--------------------+---------+---------+
```

Image 2 – Original DataFrame

Exercise n°1 : Explore Web Logs with Spark RDDs

The objective of this lab will be to manipulate RDDs again, but with the ability to manipulate the attributes of RDD values more easily. We will have each line with four attributes, all of which can be called using ".attname".
To do this, we first create a structure using **StructType**, consisting of four fields **StructField** which also allows us to declare objects such as character strings or integers. This is how we do it :

```
Log = StructType([
    StructField("project", StringType(), True),
    StructField("title", StringType(), True),
    StructField("hits", IntegerType(), True),
    StructField("size", IntegerType(), True)
])
```

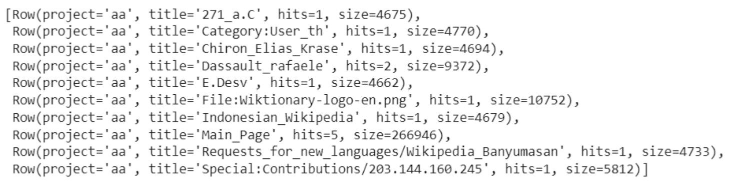
Code 3 – Structure

We then need to separate the elements of the RDD with a function we code and call "separe_elements", which will separate the RDD with space characters.

```
def separe_elements(ligne):
    elements = ligne.split(" ")
    LogEx = Row(project=elements[0], title=elements[1], hits=int(elements[2]), size=int(
    elements[3]))
    return LogEx
```

Code 4 – separe_element function

This function will be called as a criterion for mapping.
Here is the constructed RDD, consisting of lines of 4 elements :



```
[Row(project='aa', title='271_a.C', hits=1, size=4675),
 Row(project='aa', title='Category:User_th', hits=1, size=4770),
 Row(project='aa', title='Chiron_Elias_Krase', hits=1, size=4694),
 Row(project='aa', title='Dassault_rafaele', hits=2, size=9372),
 Row(project='aa', title='E.Desv', hits=1, size=4662),
 Row(project='aa', title='File:Wiktionary-logo-en.png', hits=1, size=10752),
 Row(project='aa', title='Indonesian_Wikipedia', hits=1, size=4679),
 Row(project='aa', title='Main_Page', hits=5, size=266946),
 Row(project='aa', title='Requests_for_new_languages/Wikipedia_Banyumasan', hits=1, size=4733),
 Row(project='aa', title='Special:Contributions/203.144.160.245', hits=1, size=5812)]
```

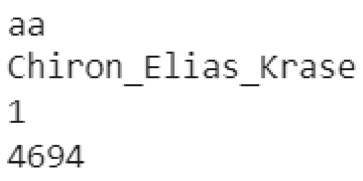
Image 3 – RDD with rows

For example, we can display the values of the third line of the previous capture using the method of the **.attname** as follows :

```
log_RDD_Thirdline = log_RDD.take(3)[-1]

print(log_RDD_Thirdline.project)
print(log_RDD_Thirdline.title)
print(log_RDD_Thirdline.hits)
print(log_RDD_Thirdline.size)
```

Code 5 – Get the informations of the thrid line

And we get the following display, consistent with the display of the first 10 lines of the RDD as we did before :



```
aa
Chiron_Elias_Krase
1
4694
```

Image 4 – Values on the third line of the RDD

The execution time for this question is 0.843 second.

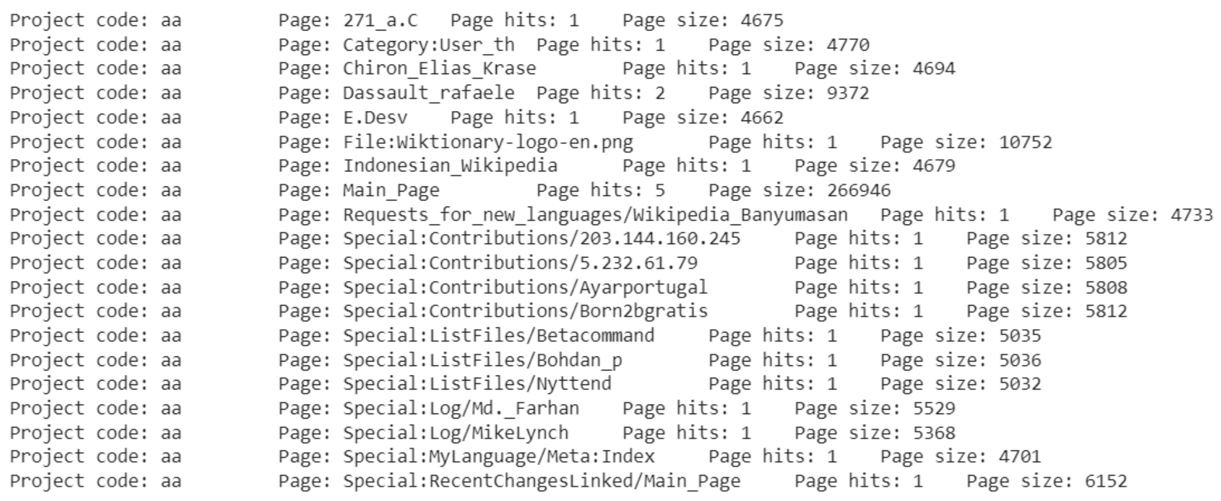
Question 1

The purpose of this question is to call the previous code and make the data in the first 20 lines appear more readable. We will create a simple function called **print_record**, the code for which is as follows :

```python
def print_record(rdd, number):
  myLine = rdd.take(number)[-1]
  print("Project code: " + myLine.project + "\t Page: " + myLine.title + "\t Page hits: "
     + str(myLine.hits) + "\t Page size: " + str(myLine.size))
```

Code 6 – print_record function

Once this function has been created, we call it in a loop **for**, and we obtain the following display :



```
Project code: aa         Page: 271_a.C    Page hits: 1    Page size: 4675
Project code: aa         Page: Category:User_th  Page hits: 1    Page size: 4770
Project code: aa         Page: Chiron_Elias_Krase        Page hits: 1    Page size: 4694
Project code: aa         Page: Dassault_rafaele  Page hits: 2    Page size: 9372
Project code: aa         Page: E.Desv    Page hits: 1    Page size: 4662
Project code: aa         Page: File:Wiktionary-logo-en.png       Page hits: 1    Page size: 10752
Project code: aa         Page: Indonesian_Wikipedia      Page hits: 1    Page size: 4679
Project code: aa         Page: Main_Page         Page hits: 5    Page size: 266946
Project code: aa         Page: Requests_for_new_languages/Wikipedia_Banyumasan   Page hits: 1    Page size: 4733
Project code: aa         Page: Special:Contributions/203.144.160.245     Page hits: 1    Page size: 5812
Project code: aa         Page: Special:Contributions/5.232.61.79         Page hits: 1    Page size: 5805
Project code: aa         Page: Special:Contributions/Ayarportugal        Page hits: 1    Page size: 5808
Project code: aa         Page: Special:Contributions/Born2bgratis        Page hits: 1    Page size: 5812
Project code: aa         Page: Special:ListFiles/Betacommand     Page hits: 1    Page size: 5035
Project code: aa         Page: Special:ListFiles/Bohdan_p        Page hits: 1    Page size: 5036
Project code: aa         Page: Special:ListFiles/Nyttend         Page hits: 1    Page size: 5032
Project code: aa         Page: Special:Log/Md._Farhan    Page hits: 1    Page size: 5529
Project code: aa         Page: Special:Log/MikeLynch     Page hits: 1    Page size: 5368
Project code: aa         Page: Special:MyLanguage/Meta:Index     Page hits: 1    Page size: 4701
Project code: aa         Page: Special:RecentChangesLinked/Main_Page     Page hits: 1    Page size: 6152
```

IMAGE 5 – Beautiful print of the first 20 records

The execution time for this question is 8.222 seconds.

Question 2

We only need to count the number of entries in the dataset, so we call the **count** function and store the information in a variable we call **log_RDD_Count**. We run the code and we see that there are 3,324,129 entries in this dataset.
The execution time for this question is 12.679 seconds.

Question 3

In this question, we are interested in the maximum, minimum and average size of the pages in this dataset. To do this, we map the previous RDD (which is possible because this first RDD is made up of Rows), keeping only the third column, and applying the functions **min**, **max** and **sum** (which we then divide by the variable **log_Count** recorded in the previous question to obtain the average). Once these variables have been calculated, we display them on the screen using **print**, and we obtain the following information :

— The largest page is 141 180 155 987
— the smallest page size is 0
— the average size of the pages is 132 239.569 574 465 98

The execution time for this question is 60.393 seconds.

Question 4

For this question, we want to display the entries in the dataset such that the pages are the maximum size. So we simply create the function **keepMax** which will allow us to check this, and we apply it as a filter on the RDD named **log_RDD**.

```python
def keepMax(x):
  if x[3]==log_RDD_Max:
    return x
```

Code 7 – keepMax function

Once done, we display the entries of the dataset and we observe :

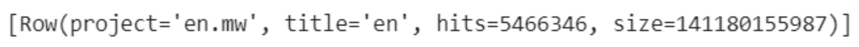

```
[Row(project='en.mw', title='en', hits=5466346, size=141180155987)]
```

Image 6 – Records with the biggest page size

We note that there is only one entry whose page is the maximum size.
The execution time for this question is 14.303 seconds.

Bonus

We wanted to test in the same way if there were several entries in the dataset such that their sizes were of the minimum size. We therefore created **keepMin** in the same way :

```
def keepMin(x):
  if x[3]==log_RDD_Min:
    return x
```

Code 8 – keepMin function

And when we apply it as a filter criterion to **log_RDD**, we obtain the following display :

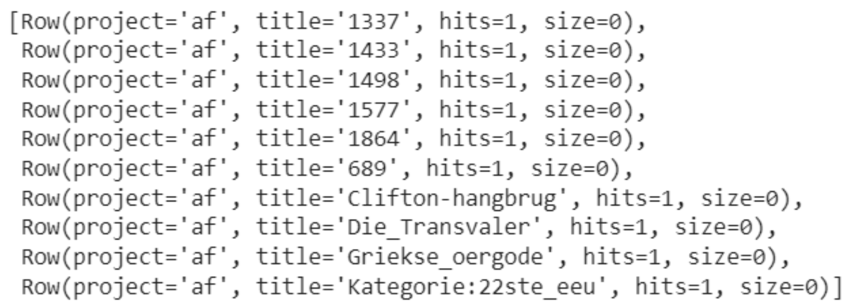

```
[Row(project='af', title='1337', hits=1, size=0),
 Row(project='af', title='1433', hits=1, size=0),
 Row(project='af', title='1498', hits=1, size=0),
 Row(project='af', title='1577', hits=1, size=0),
 Row(project='af', title='1864', hits=1, size=0),
 Row(project='af', title='689', hits=1, size=0),
 Row(project='af', title='Clifton-hangbrug', hits=1, size=0),
 Row(project='af', title='Die_Transvaler', hits=1, size=0),
 Row(project='af', title='Griekse_oergode', hits=1, size=0),
 Row(project='af', title='Kategorie:22ste_eeu', hits=1, size=0)]
```

Image 7 – 10 first records with the smallest page size

We can conclude from this question that there is only one entry that is the maximum size, but that there are several of the minimum size.

The execution time for this question is 0.659 seconds.

Question 5

First, we look only at the number of hits of the biggest page we found in the previous question, and we notice that this page contains 5 466 346 hits.
A bit like question 4, we are now going to create a function **keepMaxHits** which is going to be the criterion of the **filter** of our RDD. Here is the function **keepMaxHits** :

```
def keepMaxHits(x):
  if x[2]==log_RDD_MaxHits:
    return x
```

Code 9 – keepMaxHits function

We then apply it as a filter to our RDD. We then have the following output when trying to observe the first 10 samples :

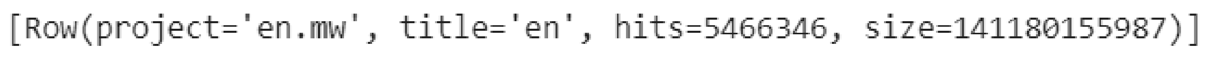

```
[Row(project='en.mw', title='en', hits=5466346, size=141180155987)]
```

Image 8 – Records with the most Hits

We notice that there is only one page that reaches the maximum level of hits. This page is also the one found in the previous question, with the largest page size.

The execution time for this question is 26.452 seconds.

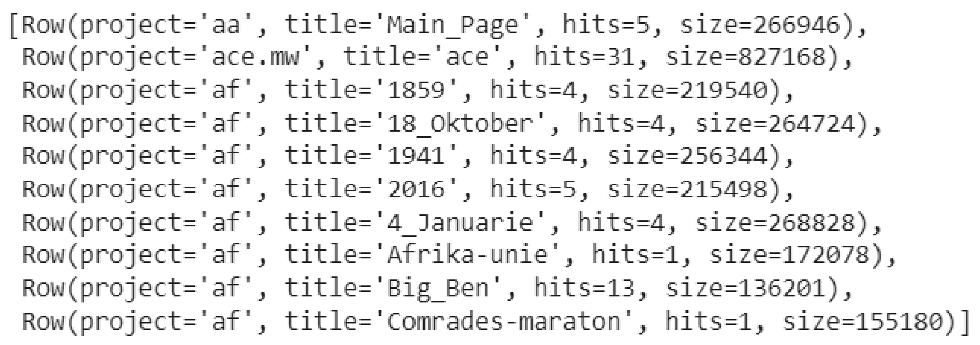
Question 6

Again, we create a function that will keep only those entries whose page size is greater than the average value we calculated earlier. We then code the following function :

```python
def keepSupAverage(x):
  if x[3]>log_RDD_Mean:
    return x
```

Code 10 – keepSupAverage function

And we then apply it as a filter on the RDD. Here is the result when we display the first 10 entries with the **take** command :



```
[Row(project='aa', title='Main_Page', hits=5, size=266946),
 Row(project='ace.mw', title='ace', hits=31, size=827168),
 Row(project='af', title='1859', hits=4, size=219540),
 Row(project='af', title='18_Oktober', hits=4, size=264724),
 Row(project='af', title='1941', hits=4, size=256344),
 Row(project='af', title='2016', hits=5, size=215498),
 Row(project='af', title='4_Januarie', hits=4, size=268828),
 Row(project='af', title='Afrika-unie', hits=1, size=172078),
 Row(project='af', title='Big_Ben', hits=13, size=136201),
 Row(project='af', title='Comrades-maraton', hits=1, size=155180)]
```

IMAGE 9 – Records with the page size above average

We can see that the size of each of these pages is indeed larger than the average size of 132,240.

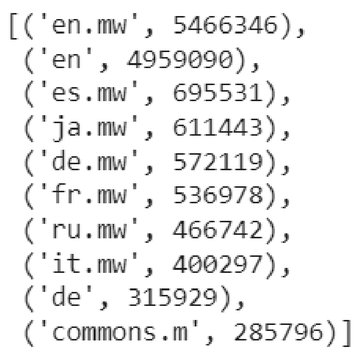
The execution time for this question is 0.507 seconds.

Question 7

For this part of the question, we are only interested in the project names and the number of hits. We'll then map these two columns, then use **reduceByKey** to sum the number of hits, and finally we'll sort them in descending order of number of hits. Here is the coded line :

```python
log_RDD_HitsPerProject = log_RDD.map(lambda x : (x[0], x[2])).reduceByKey(lambda x, y : x
    +y).sortBy(lambda row : row[1], ascending=False)
```

This gives us the ranking, in descending order, of the projects with the highest number of hits :



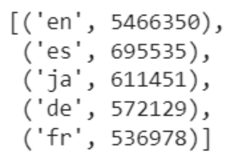
```
[('en.mw', 5466346),
 ('en', 4959090),
 ('es.mw', 695531),
 ('ja.mw', 611443),
 ('de.mw', 572119),
 ('fr.mw', 536978),
 ('ru.mw', 466742),
 ('it.mw', 400297),
 ('de', 315929),
 ('commons.m', 285796)]
```

IMAGE 10 – Projects with the most hits

For the rest of this question, we need to display the 5 most popular pages. So we map to keep only the page names and the number of hits, and we use a **reduceByKey** again to get the number of occurrences. We then write the following line of code :

```python
log_RDD_HitsPerPage = log_RDD.map(lambda x : (x[1], x[2])).reduceByKey(lambda x, y : x+y)
    .sortBy(lambda row : row[1], ascending=False)
```

We then obtain the ranking of the 5 most popular results :



```
[('en', 5466350),
 ('es', 695535),
 ('ja', 611451),
 ('de', 572129),
 ('fr', 536978)]
```

IMAGE 11 – Pages with the most hits

We notice that these are the pages for selecting the language ! There is en for English, es for Spanish, ja for Japanese, de for German and finally fr for French !
The execution time for this question is 59.741 seconds.

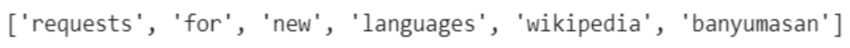
Question 8

In the same way as the previous lab, we will code the function **cleaningFunc**. The purpose of this function will be to transform all letters into lower case characters, and to transform each punctuation with the space character. Here is how we have coded this function :

```python
def cleaningFunc(text):
  text = text.lower()
  translator = text.maketrans(string.punctuation,"_" * len(string.punctuation))
  text = text.translate(translator)
  text = text.split("_")
  return text
```

Code 11 – cleaningFunc function

We can test it on a basic string, such as **"Requests_for.new :languages/Wikipedia-Banyumasan"** in order to verify that it works. Here is the output we get :



```
['requests', 'for', 'new', 'languages', 'wikipedia', 'banyumasan']
```

IMAGE 12 – Example of the cleaningFunc function

Now we want to "unroll" the RDD : each entry can contain several words in the title, and we want an entry to correspond to only one word. We use a flatMap to which we apply the previous function **cleaningFunc**. Then we filter to obtain only the results that contain letters (no numbers), then we do a mapping that allows us to keep only the title. Finally, we use a reduceByKey to count the occurrences of each word.
We will need to record this RDD (named log_RDD_PageNameWithOccurences) for the next question.
For this question, we want words that only occur once, so we'll want to filter by number of occurrences, keeping only those that have an occurrence of 1. We map to keep only those words.
Here is how we code these few steps :

```python
log_RDD_PageName2 = log_RDD.flatMap(lambda x : cleaningFunc(x[1])) #unfold the table

log_RDD_PageNameWordOnly = log_RDD_PageName2.filter(lambda x: x.isalpha()).map(lambda x : (x, 1))

log_RDD_PageNameWithOccurences = log_RDD_PageNameWordOnly.reduceByKey(lambda x, y : x+y)

log_RDD_Page_UniqueWord = log_RDD_PageNameWithOccurences.filter(lambda x : x[1]==1).map(lambda x : x[0])
```

And here is the result of the first lines :

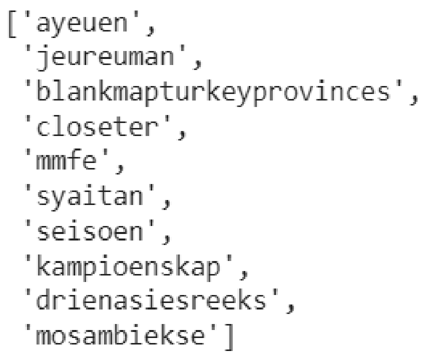

```
['ayeuen',
 'jeureuman',
 'blankmapturkeyprovinces',
 'closeter',
 'mmfe',
 'syaitan',
 'seisoen',
 'kampioenskap',
 'drienasiesreeks',
 'mosambiekse']
```

IMAGE 13 – RDD containing one word per record, each word is unique in the original RDD

We note that the filter criterion using Python's **isalpha** function is much more efficient than using a criterion with "[a-z]+" because it removes all words containing numbers, as well as all words containing only one letter, which are not real words.

The execution time for this question is 40.317 seconds.

Question 9

For this question, we will arrange the RDD **log_RDD_PageNameWithOccurences** created in question 8 in descending order, again using the function **sortBy**. Once this is done, we call the function **first** to display the word that is most present in the list of all titles. The result is then :

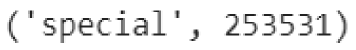

```
('special', 253531)
```

IMAGE 14 – Most used word in title, with its occurence

The most frequently used word is **special**, it is used 253 531 times.

The execution time for this question is 5.237 seconds.

Exercise n°2 : Query Web Logs with Spark SQL

Introduction

The aim of this second exercise is to repeat questions 3, 5, 6 and 8 from the previous section, this time using dataframes.

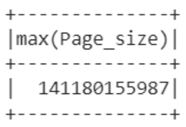
Question 3 bis

In this question we want to re-determine values for page size : minimum, maximum and average.
To calculate the maximum, we simply write :

```
wiki_DF.select(max("Page_size")).show()
```

Code 12 – Determining the maximum page size

And we get the following display :



```
+--------------+
|max(Page_size)|
+--------------+
|  141180155987|
+--------------+
```

IMAGE 15 – Determining the maximum page size (with dataframes)

Similarly, replacing **max** with **min** and **mean** in the previous line of code, we find the following values :
- The largest page is 141 180 155 987
- the smallest page size is 0
- the average size of the pages is 132 239.569 574 465 98

These are exactly the same values we found with the RDDs !
The execution time for this question is 44.020 seconds.

Question 5 bis

In this question, we will try to determine which page has the most hits. To do this, we first recorded in variables the values that correspond to the max, min and average found previously. This is how we did it with the collect function :

```
maxDF = wiki_DF.select(max("Page_size")).collect()[0][0]
minDF = wiki_DF.select(min("Page_size")).collect()[0][0]
meanDF = wiki_DF.select(mean("Page_size")).collect()[0][0]
```

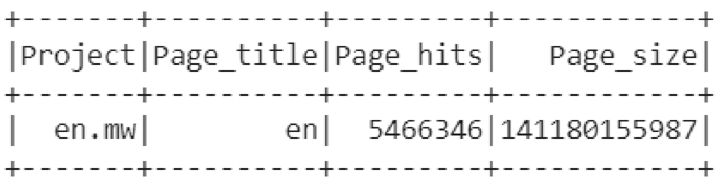
Code 13 – Recording of extremums and average

Next, we create the **wiki_DF_MaxPages** dataframe which is a filter of the dataframe, keeping only those whose size is the maximum size maxDF. Here is how it's done :

```
wiki_DF_MaxPages = wiki_DF.filter(wiki_DF.Page_size==maxDF)
```

Code 14 – Determining the records with the largest page size

We display it and get :



```
+-------+----------+---------+------------+
|Project|Page_title|Page_hits|   Page_size|
+-------+----------+---------+------------+
|  en.mw|        en|  5466346|141180155987|
+-------+----------+---------+------------+
```

IMAGE 16 – Records with the largest page size

We observe that we obtain only one record, which is the page with the most hits. This is what we verify :

```
wiki_DF_MaxHits = wiki_DF.filter(wiki_DF.Page_hits==maxHitsDF)
```

Code 15 – Determining the records with the most hits

Displaying this dataframe gives us exactly the same result as the one displayed previously.
The execution time for this question is 41.051 seconds.

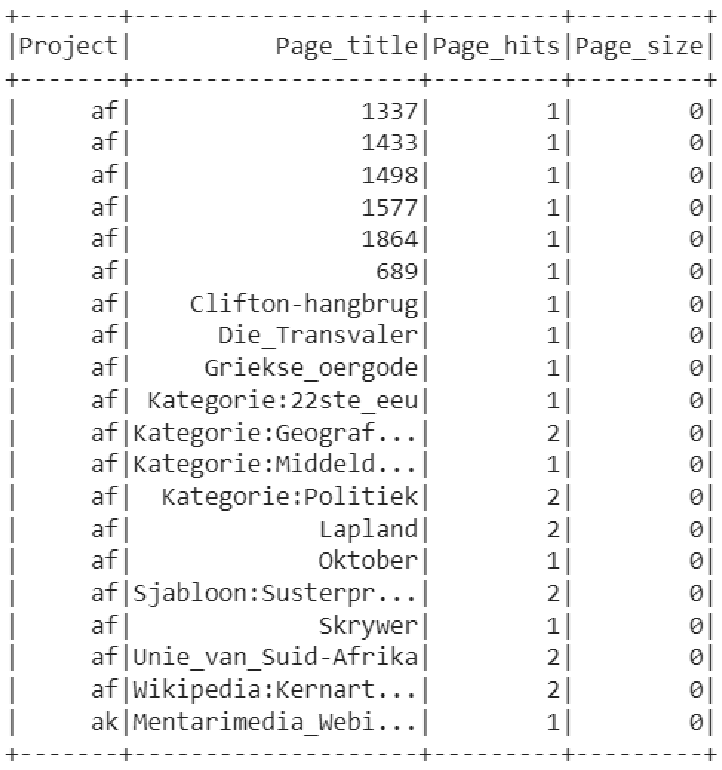
Bonus

As we did with the RDDs, we created a dataframe containing only the pages with the smallest page sizes. Here is how we did it :

```
wiki_DF_MaxHits = wiki_DF.filter(wiki_DF.Page_hits==maxHitsDF)
```

Code 16 – Determining the records with the smallest page size

And this is the display we get when we use the **show** command :



```
+-------+--------------------+---------+---------+
|Project|          Page_title|Page_hits|Page_size|
+-------+--------------------+---------+---------+
|     af|                1337|        1|        0|
|     af|                1433|        1|        0|
|     af|                1498|        1|        0|
|     af|                1577|        1|        0|
|     af|                1864|        1|        0|
|     af|                 689|        1|        0|
|     af|     Clifton-hangbrug|        1|        0|
|     af|       Die_Transvaler|        1|        0|
|     af|      Griekse_oergode|        1|        0|
|     af|  Kategorie:22ste_eeu|        1|        0|
|     af|Kategorie:Geograf...|        2|        0|
|     af|Kategorie:Middeld...|        1|        0|
|     af|   Kategorie:Politiek|        2|        0|
|     af|              Lapland|        2|        0|
|     af|              Oktober|        1|        0|
|     af|Sjabloon:Susterpr...|        2|        0|
|     af|              Skrywer|        1|        0|
|     af|Unie_van_Suid-Afrika|        2|        0|
|     af|Wikipedia:Kernart...|        2|        0|
|     ak|Mentarimedia_Webi...|        1|        0|
+-------+--------------------+---------+---------+
```

IMAGE 17 – Records with the smallest page size

Comparing with what we had previously obtained, we get exactly the same records.
The execution time for this question is 4.211 seconds.

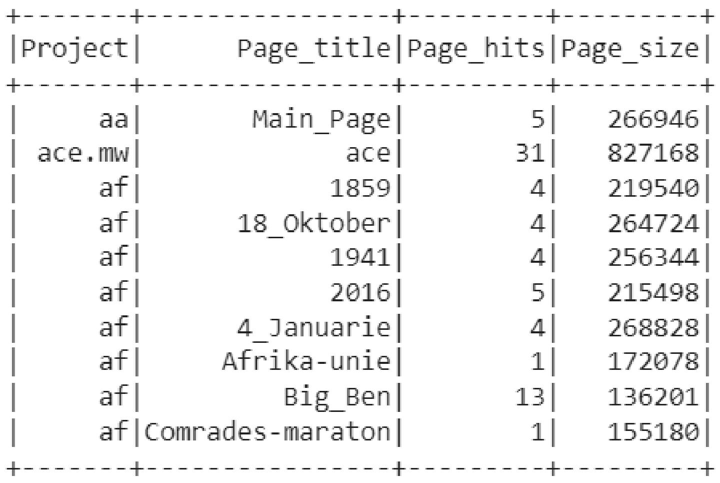
Question 6 bis

In the same way as in question 5 bis, in one line we will create a new dataframe which will only keep records whose page size is greater than the average page size, a value calculated previously and named **meanDF**. This is how we wrote it :

```
wiki_DF_AboveAverage = wiki_DF.filter(wiki_DF.Page_size>meanDF)
```

Code 17 – Determination of entries with an above-average page size

And this is the dataframe we get :



```
+-------+---------------+---------+---------+
|Project|     Page_title|Page_hits|Page_size|
+-------+---------------+---------+---------+
|     aa|      Main_Page|        5|   266946|
| ace.mw|            ace|       31|   827168|
|     af|           1859|        4|   219540|
|     af|     18_Oktober|        4|   264724|
|     af|           1941|        4|   256344|
|     af|           2016|        5|   215498|
|     af|     4_Januarie|        4|   268828|
|     af|    Afrika-unie|        1|   172078|
|     af|        Big_Ben|       13|   136201|
|     af|Comrades-maraton|        1|   155180|
+-------+---------------+---------+---------+
```

IMAGE 18 – Records with larger than average page sizes

Again, this dataframe corresponds exactly to the RDD we determined in question 6 : the records are the same !
The execution time for this question is 4.191 seconds.

Question 8 bis

We want to have a dataframe that contains all the words in the titles, and only those that occur once. We will then, by analogy with RDDs, create this dataframe using the following functions :

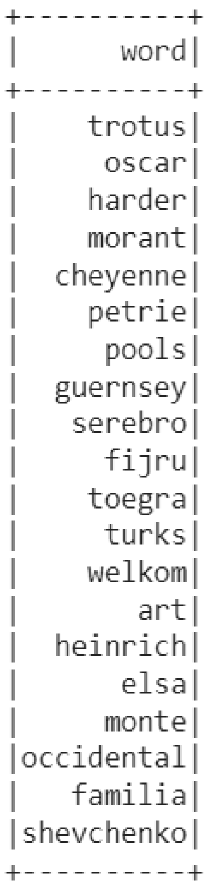
- **udf** will allow us to apply the cleaningFunc function created earlier to a dataframe
- **withColumn** adds a column that will contain all the words in lower case, with punctuation replaced by spaces
- **explode** is the equivalent of flatmap but for dataframe
- **distinct** allows us to keep only the words that appear only once in the dataframe

Here is more detail on how we did this :

```python
# define with udf a function applicable on a dataframe
clean_udf = udf(cleaningFunc, ArrayType(StringType()))

# we apply our new cleaning function to the title_page column
cleaned_DF = wiki_DF.withColumn("words", clean_udf(wiki_DF["Page_title"]))

# explode: equivalent to flatMap but for dataframe
exploded_DF = cleaned_DF.select(explode(cleaned_DF["words"]).alias("word"))

# we filter to keep only lowercase words
filtered_DF = exploded_DF.filter(regexp_extract(exploded_DF["word"], r"[a-z]+", 0) != "")

# only words that appear once are kept
unique_words_DF = filtered_DF.select("word").distinct()
```

Code 18 – Unique words of the title

Here is the result we get when we display this last dataframe with the show command :



```
+----------+
|      word|
+----------+
|    trotus|
|     oscar|
|    harder|
|    morant|
|  cheyenne|
|    petrie|
|     pools|
|  guernsey|
|   serebro|
|     fijru|
|    toegra|
|     turks|
|    welkom|
|       art|
|  heinrich|
|      elsa|
|     monte|
|occidental|
|   familia|
|shevchenko|
+----------+
```

IMAGE 19 – Unique words in titles

The execution time for this question is 68.493 seconds.

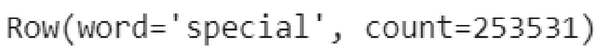
Bonus : Question 9 bis

Let's determine the most used words in page titles, using dataframes this time :

```python
# count() creates a new "count" column with the number of repetitions of the words in the
    "word" column
count_DF = filtered_DF.groupBy("word").count()

# we order in descending order of the count and display the first line
most_frequent_word = count_DF.orderBy(desc("count")).first()

print(most_frequent_word)
```

Code 19 – Most frequently occurring page title word in the dataset

Here is the result :



```
Row(word='special', count=253531)
```

IMAGE 20 – Most used title word

As with the RDDs, we can see that the word special is the most used word with an occurrence of 253 531.
The execution time for this question is 66.953 seconds.