**Jupyter Part**

**Let’s code**

Remove duplicates and write here how many duplicates were.

**Answer**

* There were a total of 10001 records
* There were a total of 9617 unique records
* There were a total of 384 duplicates

Add code for first cluster here

**Code**

# Read the CSV file

myRdd = sc.textFile("gs://kamal\_at\_ie/lab2Dataset.csv")

# Check the total no. of records and pprint that

total = myRdd.count()

print("Total:" + str(total))

# Create a new RDD containing only unique keys

uniqueRdd = myRdd.distinct()

# Print the total no. of unique keys

unique = uniqueRdd.count()

print("Unique:" + str(unique))

# Get the total no. of duplicates and print the same

dups = total - unique

print("Dups:" + str(dups))

Number of duplicates: 384

Create new RDD with ids whose length is bigger than 3.

Save RDD as text in gs bucket you created before. Choose a different file

name.

**Code (Continued from before):**

# Take the original RDD that we had created earlier and create a new RDD that contains the keys of length bigger than 3

idsgt3 = myRdd.filter(lambda line: len(line) > 3)

# Do a sanity check

idsgt3.take(10)

# Print the no. of strings in the new RDD

print("Count of IDs bigger than length 3:" + str(idsgt3.count()))

# Save this file in storage

idsgt3.saveAsTextFile("gs://kamal\_at\_ie/idsgt3.csv")

**Delete cluster 1.**

**DataLab part**

Read text file generated by cluster 1 and add it to rdd\_main.

rdd\_main = sc.textFile("gs://kamal\_at\_ie/idsgt3.csv")

Get number of partitions and reduce the partitions to 2

# get no. of partitions in the rdd

rdd\_main.getNumPartitions()

2

# Reduce the partitions to 2; # although, we should do only if #the existing no. of partitions is > 2

rddCoalesed = rdd\_main.coalesce(2)

# again check if the no. of partitions are <=2

rdd\_main.getNumPartitions()

2

Obtain the length of all the ids added together.

For example if ids are:

[alex, pepe, domin] then total would be 3 from alex plus 4 from pepe plus 5

from domin so 12

# get the length of all IDs put together

recLenRdd = rddCoalesed.map(lambda rec: len(rec))

totalLength = recLenRdd.reduce(lambda a, b: a + b)

print("Total length of all strings:" + str(totalLength))

Total length of all strings:91545

Sort alphabetically and display the first 10 ids (order from a to z)

#Sort the IDs alphabetically and print the first 10

sortedRdd = rddCoalesed.map(lambda x: (x, 1)).sortByKey().keys()

sortedRdd.take(10)

[u'02strich',

u'0m4r',

u'0x860111',

u'0x90',

u'0xen',

u'0xhacker',

u'10K35H 5H4KY4',

u'14256424',

u'150GritSandpaper',

u'1ifbyLAN2ifbyC']

Save the file to your bucket with the name sorted\_rddids.txt

sortedRdd.saveAsTextFile("gs://kamal\_at\_ie/sorted\_rddids.txt")

From rdd\_main obtain the number of ids that share the first two characters.

For example if we have ids: aax , aat, aaron, bbt we would have aa,3 and

bb,1

rddWordCount = rdd\_main.map(lambda x: (x[:2], 1)).reduceByKey(lambda a, b: a + b)

# print the first 10

rddWordCount.take(10)

[(u'gw', 1),

(u'gu', 7),

(u'gs', 4),

(u'ge', 7),

(u'gc', 1),

(u'ga', 13),

(u'go', 11),

(u'gm', 4),

(u'gk', 1),

(u'm\_', 1)]

Add ALL your code here

# read the file that contains IDs with length bigger than 3, from the storage bucket

rdd\_main = sc.textFile("gs://kamal\_at\_ie/idsgt3.csv")

# Do some sanity check

print(rdd\_main.count())

# get no. of partitions in the rdd

rdd\_main.getNumPartitions()

# Reduce the partitions to 2;# although, we should do only if the existing no. of partitions is > 2

rddCoalesed = rdd\_main.coalesce(2)

# again check if the no. of partitions are <=2

rdd\_main.getNumPartitions()

# get the length of all IDs put together

recLenRdd = rddCoalesed.map(lambda rec: len(rec))

totalLength = recLenRdd.reduce(lambda a, b: a + b)

print("Total length of all strings:" + str(totalLength))

#Sort the IDs alphabetically and print the first 10

sortedRdd = rddCoalesed.map(lambda x: (x, 1)).sortByKey().keys()

sortedRdd.take(10)

# Store the sorted RDD in gs

sortedRdd.saveAsTextFile("gs://kamal\_at\_ie/sorted\_rddids.txt")

#From rdd\_main obtain the number of ids that share the first 2 chars

#For example if we have ids: aax , aat, aaron, bbt we would have aa,3 and bb,1

rddWordCount = rdd\_main.map(lambda x: (x[:2], 1)).reduceByKey(lambda a, b: a + b)

# print the first 10

rddWordCount.take(10)