Spark

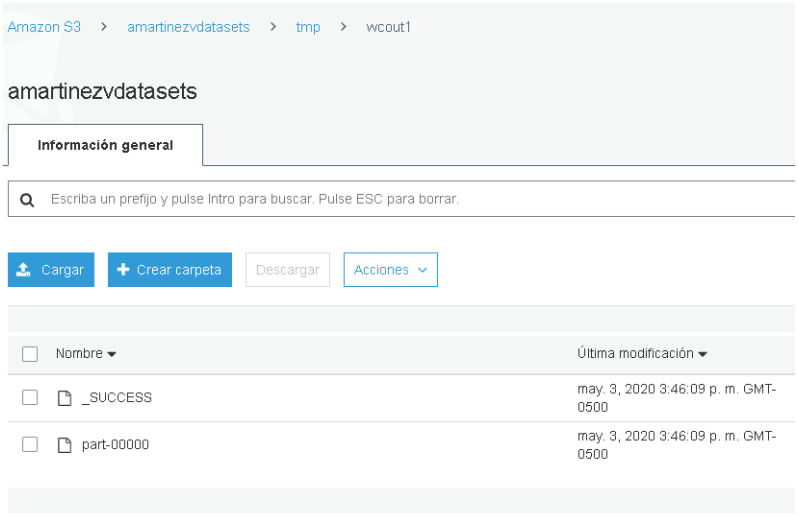
domingo, 3 de mayo de 2020 13:29

- Tutorial Zeppelin

- Wordcount en python:

```
%spark.pyspark
files = sc.textFile("s3://amartinezvdatasets/datasets/onu/datasets/gutenberg-small/*.txt")
wc = files.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
wc.coalesce(1).saveAsTextFile("s3://amartinezvdatasets/tmp/wcout1")
```

Took 20 sec. Last updated by anonymous at May 03 2020, 3:46:08 PM.



- wordcount en spark.sql

```
%spark.sql
CREATE EXTERNAL TABLE default.docs2 (line STRING) stored as textfile location 's3://amartinezvdatasets/datasets/onu/datasets/gutenberg-small/*.txt'
```

```
%spark.sql
SELECT word, count(1) AS count FROM (SELECT explode(split(line,' ')) AS word FROM docs2) w GROUP BY word ORDER BY word
```

settings

word	count
	27298
"	8
"'ARTICLE	1
"'But	1
"'FROM	1
"'KANSAS	1
"'Nothing	1
"'One	1

- Worcount en jupiter

```
In [1]: sc
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1588536233611_0002	pyspark	idle	Link	Link	✓

```
SparkSession available as 'spark'.
<SparkContext master=yarn appName=livy-session-0>
```

```
In [2]: # WORDCOUNT COMPACTO
files_rdd = sc.textFile("s3://amartinezvdatasets/datasets/onu/datasets/gutenberg-small/*.txt")
wc_unsort = files_rdd.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
wc = wc_unsort.sortBy(lambda a: -a[1])
```

```
for tupla in wc.take(10):
      print(tupla)
```

▸ Spark Job Progress

```
('the', 44647)
('of', 28020)
('', 27298)
('to', 23208)
('and', 20444)
('in', 13174)
('that', 12265)
('I', 10880)
('a', 10431)
('is', 7776)
```

```
# WORDCOUNT PASO A PASO
```

```
files = sc.textFile("s3://amartinezvdatasets/datasets/onu/datasets/gutenberg-small/*.txt")
for f in files.take(10):
    print(f)
```

▸ Spark Job Progress

```
LINCOLN LETTERS

By Abraham Lincoln


Published by The Bibilophile Society
```

```
tokens = files.flatMap(lambda line: line.split(' '))
for t in tokens.take(10):
    print(t)
```

▸ Spark Job Progress

```
LINCOLN
LETTERS

By
Abraham
Lincoln


Published
```

```
wc1 = tokens.map(lambda word: (word, 1))
for c in wc1.take(10):
    print(c)
```

▸ Spark Job Progress

```
('', 1)
('LINCOLN', 1)
('LETTERS', 1)
('', 1)
('By', 1)
('Abraham', 1)
('Lincoln', 1)
```

```
wc = wc1.reduceByKey(lambda a, b: a + b)
for c in wc.take(10):
    print(c)
```

▸ Spark Job Progress

```
('', 27298)
('thoroughly', 15)
('themselves', 192)
('them.', 371)
('letter', 312)
('A.', 1456)
('ORIGINALS', 1)
('THEY', 1)
('sum', 59)
('singular', 18)
```

```
wcsort = wc.sortBy(lambda a: -a[1])
for c in wcsort.take(10):
    print(c)
```

▸ Spark Job Progress

```
('the', 44647)
('of', 28020)
('', 27298)
('to', 23208)
('and', 20444)
```

```
('in', 13174)
('that', 12265)
('I', 10880)
```

- Ejecución de Data_processing_using_PySpark.ipynb

```
%%configure -f
{ "conf":{
"spark.pyspark.python": "python3",
"spark.pyspark.virtualenv.enabled": "true",
"spark.pyspark.virtualenv.type":"native",
"spark.pyspark.virtualenv.bin.path":"/usr/bin/virtualenv"
}}
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
2	application_1588536233611_0004	pyspark	idle	Link	Link	✓

```
SparkSession available as 'spark'.
```

```
Current session configs: {'conf': {'spark.pyspark.python': 'python3', 'spark.pyspark.
'spark.pyspark.virtualenv.type': 'native', 'spark.pyspark.virtualenv.bin.path':
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1588536233611_0002	pyspark	idle	Link	Link	
2	application_1588536233611_0004	pyspark	idle	Link	Link	✓

a. Carga de datos csv en spark desde un bucket S3.

```
[2]: #create spar session object
spark=SparkSession.builder.appName('data_processing').getOrCreate()
```

```
[3]: # Load csv Dataset
df=spark.read.csv('s3://amartinezvdatasets/datasets/onu/datasets/spark/sample_data.csv',inferSchema=True,header=True)
```

▸ Spark Job Progress

```
[4]: #columns of dataframe
df.columns
```

```
['ratings', 'age', 'experience', 'family', 'mobile']
```

```
[11]: #info about dataframe
df.describe().show()
```

▸ Spark Job Progress

```
+-------+------------------+------------------+------------------+------------------+------+
|summary|           ratings|               age|        experience|            family|mobile|
+-------+------------------+------------------+------------------+------------------+------+
|  count|                33|                33|                33|                33|    33|
|   mean|3.5757575757575757|30.484848484848484|10.303030303030303|1.8181818181818181|  null|
| stddev|1.1188806636071336|  6.18527087180309| 6.770731351213326|1.8448330794164254|  null|
|    min|                 1|                22|               2.5|                 0| Apple|
|    max|                 5|                42|              23.0|                 5|  Vivo|
+-------+------------------+------------------+------------------+------------------+------+
```

b. Se añade columna 'age_after_10_years' : Edad de la población después de diez años

```
#with column
df.withColumn("age_after_10_yrs",(df["age"]+10)).show(10,False)
```

▸ Spark Job Progress

```
+-------+---+----------+------+-------+----------------+
|ratings|age|experience|family|mobile |age_after_10_yrs|
+-------+---+----------+------+-------+----------------+
|3      |32 |9.0       |3     |Vivo   |42              |
|3      |27 |13.0      |3     |Apple  |37              |
|4      |22 |2.5       |0     |Samsung|32              |
|4      |37 |16.5      |4     |Apple  |47              |
|5      |27 |9.0       |1     |MI     |37              |
|4      |27 |9.0       |0     |Oppo   |37              |
|5      |37 |23.0      |5     |Vivo   |47              |
|5      |37 |23.0      |5     |Samsung|47              |
|3      |22 |2.5       |0     |Apple  |32              |
|3      |27 |6.0       |0     |MI     |37              |
+-------+---+----------+------+-------+----------------+
only showing top 10 rows
```

c. Se filtran algunos datos

Quienes tienen el mobile 'Vivo'

```
#filter the records
df.filter(df['mobile']=='Vivo').show()
```

▸ Spark Job Progress

```
+-------+---+----------+------+------+
|ratings|age|experience|family|mobile|
+-------+---+----------+------+------+
|      3| 32|       9.0|     3|  Vivo|
|      5| 37|      23.0|     5|  Vivo|
|      4| 37|       6.0|     0|  Vivo|
|      5| 37|      13.0|     1|  Vivo|
|      4| 37|       6.0|     0|  Vivo|
+-------+---+----------+------+------+
```

d. Mínimo de ratings, edad, experiencia y familia de los tipos de móviles

```
# Value counts
df.groupBy('mobile').min().show(5,False)
```

▸ Spark Job Progress

```
+-------+------------+--------+---------------+-----------+
|mobile |min(ratings)|min(age)|min(experience)|min(family)|
+-------+------------+--------+---------------+-----------+
|MI     |1           |27      |2.5            |0          |
|Oppo   |2           |22      |6.0            |0          |
|Samsung|2           |22      |2.5            |0          |
|Vivo   |3           |32      |6.0            |0          |
|Apple  |3           |22      |2.5            |0          |
+-------+------------+--------+---------------+-----------+
```

e. Se añade la columna price_range para saber si el tipo de móvil tuvo un costo alto o bajo

```
#normal function
def price_range(brand):
    if brand in ['Samsung','Apple']:
        return 'High Price'
    elif brand =='MI':
        return 'Mid Price'
    else:
        return 'Low Price'
```

```
#create udf using python function
brand_udf=udf(price_range,StringType())
#apply udf on dataframe
df.withColumn('price_range',brand_udf(df['mobile'])).show(10,False)
```

▸ Spark Job Progress

```
+-------+---+----------+------+-------+-----------+
|ratings|age|experience|family|mobile |price_range|
+-------+---+----------+------+-------+-----------+
|3      |32 |9.0       |3     |Vivo   |Low Price  |
|3      |27 |13.0      |3     |Apple  |High Price |
|4      |22 |2.5       |0     |Samsung|High Price |
|4      |37 |16.5      |4     |Apple  |High Price |
|5      |27 |9.0       |1     |MI     |Mid Price  |
|4      |27 |9.0       |0     |Oppo   |Low Price  |
|5      |37 |23.0      |5     |Vivo   |Low Price  |
|5      |37 |23.0      |5     |Samsung|High Price |
|3      |22 |2.5       |0     |Apple  |High Price |
|3      |27 |6.0       |0     |MI     |Mid Price  |
+-------+---+----------+------+-------+-----------+
only showing top 10 rows
```

f. Se añade la columna age_group y se utliza la funcion lambda para definir un rango de edades y clasificarlo en 'junior' o 'senior'

```
#using lambda function
age_udf = udf(lambda age: "young" if age <= 30 else "senior", StringType())
#apply udf on dataframe
df.withColumn("age_group", age_udf(df.age)).show(10,False)
```

▸ Spark Job Progress

```
+-------+---+----------+------+-------+---------+
|ratings|age|experience|family|mobile |age_group|
+-------+---+----------+------+-------+---------+
|3      |32 |9.0       |3     |Vivo   |senior   |
|3      |27 |13.0      |3     |Apple  |young    |
|4      |22 |2.5       |0     |Samsung|young    |
|4      |37 |16.5      |4     |Apple  |senior   |
|5      |27 |9.0       |1     |MI     |young    |
|4      |27 |9.0       |0     |Oppo   |young    |
|5      |37 |23.0      |5     |Vivo   |senior   |
|5      |37 |23.0      |5     |Samsung|senior   |
|3      |22 |2.5       |0     |Apple  |young    |
|3      |27 |6.0       |0     |MI     |young    |
+-------+---+----------+------+-------+---------+
only showing top 10 rows
```

g. Se crea la columna yrs_left donde se calcula la resta de 100-edad por medio de una funcion creada previamente.

```
#create python function
def remaining_yrs(age):
    yrs_left=100-age

    return yrs_left
```

▸ Spark Job Progress

```
#create udf using python function
length_udf = pandas_udf(remaining_yrs, IntegerType())
#apply pandas udf on dataframe
df.withColumn("yrs_left", length_udf(df['age'])).show(10,False)
```

```
+-------+---+----------+------+-------+---------+
|ratings|age|experience|family|mobile |yrs_left|
+-------+---+----------+------+-------+---------+
|3      |32 |9.0       |3     |Vivo   |68       |
|3      |27 |13.0      |3     |Apple  |73       |
|4      |22 |2.5       |0     |Samsung|78       |
|4      |37 |16.5      |4     |Apple  |63       |
|5      |27 |9.0       |1     |MI     |73       |
|4      |27 |9.0       |0     |Oppo   |73       |
|5      |37 |23.0      |5     |Vivo   |63       |
|5      |37 |23.0      |5     |Samsung|63       |
|3      |22 |2.5       |0     |Apple  |78       |
|3      |27 |6.0       |0     |MI     |73       |
+-------+---+----------+------+-------+---------+
only showing top 10 rows
```

h. Se crea la función prod donde se multiplica el ratings por la experiencia y se guarda en una nueva columna

In [91]:
```
#udf using two columns
def prod(rating,exp):
    x=rating*exp
    return x
```

▸ Spark Job Progress

In [92]:
```
#create udf using python function
prod_udf = pandas_udf(prod, DoubleType())
#apply pandas udf on multiple columns of dataframe
df.withColumn("product", prod_udf(df['ratings'],df['experience'])).show(10,False)
```

▸ Spark Job Progress

```
+-------+---+----------+------+-------+-------+
|ratings|age|experience|family|mobile |product|
+-------+---+----------+------+-------+-------+
|3      |32 |9.0       |3     |Vivo   |27.0   |
|3      |27 |13.0      |3     |Apple  |39.0   |
|4      |22 |2.5       |0     |Samsung|10.0   |
```

```
|4      |37 |16.5      |4     |Apple  |66.0   |
|5      |27 |9.0       |1     |MI     |45.0   |
|4      |27 |9.0       |0     |Oppo   |36.0   |
|5      |37 |23.0      |5     |Vivo   |115.0  |
|5      |37 |23.0      |5     |Samsung|115.0  |
|3      |22 |2.5       |0     |Apple  |7.5    |
|3      |27 |6.0       |0     |MI     |18.0   |
+-------+---+----------+------+-------+-------+
only showing top 10 rows
```

i. Se guardan los resultados en un bucket público en S3

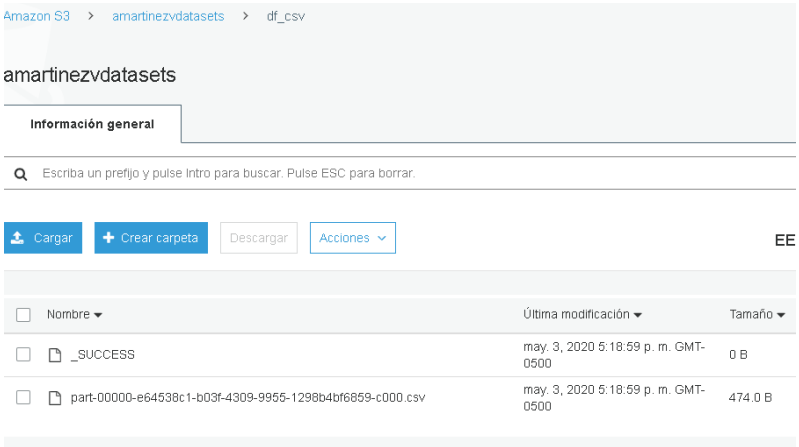
En csv

```
#target directory
write_uri='s3://amartinezvdatasets/df_csv'
```

▸ Spark Job Progress

```
#save the dataframe as single csv
df.coalesce(1).write.format("csv").option("header","true").save(write_uri)
```

▸ Spark Job Progress



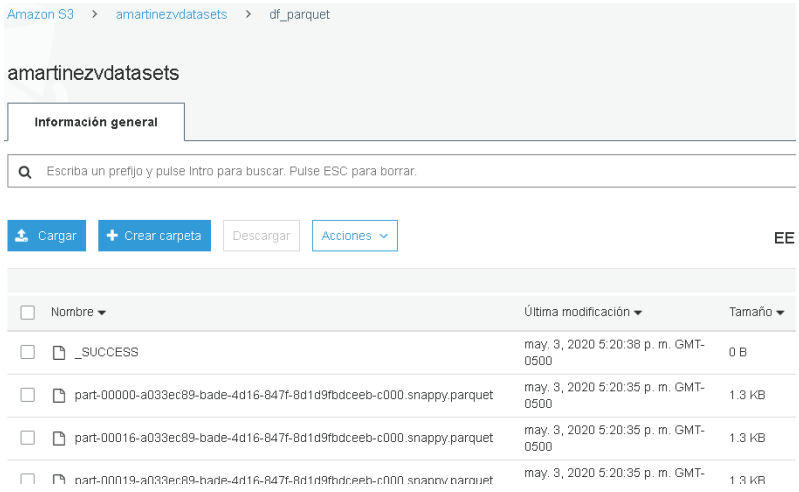
En Parquet

```
#target location
parquet_uri='s3://amartinezvdatasets/df_parquet'
```

▸ Spark Job Progress

```
#save the data into parquet format
df.write.format('parquet').save(parquet_uri)
```

▸ Spark Job Progress



Laboratorio Spark – Ejercicio COVID 19

La documentación se realizó en el mismo notebook

- finalmente grave los resultados en un bucket público en S3