Capítulo 2

a. Descargar el Quijote

https://gist.github.com/jsdario/6d6c69398cb0c73111e49f1218960f79

Aplicar no solo count (para obtener el número de líneas) y show sino probar distintas sobrecargas del método show (con/sin truncate, indicando/sin indicar num de filas, etc) así como también los métodos, head, take, first (diferencias entre estos 3?)

Take puede mostrarte un numero n de las primeras líneas, first solo la primera y head si no se le pasan valores muestra solo la primera pero si le indicas un numero te muestra esos n resultados

b. Del ejercicio de M&M aplicar:

i. Otras operaciones de agregación como el Max con otro tipo de ordenamiento (descendiente).

ii. hacer un ejercicio como el “where” de CA que aparece en el libro pero indicando más opciones de estados (p.e. NV, TX, CA, CO).

iii. Hacer un ejercicio donde se calculen en una misma operación el Max, Min, Avg, Count. Revisar el API (documentación) donde encontrarán este ejemplo:

ds.agg(max($"age"), avg($"salary"))

ds.groupBy().agg(max($"age"), avg($"salary"))

NOTA: $ es un alias de col()

iv. Hacer también ejercicios en SQL creando tmpView

Capítulo 3

a. Realizar todos los ejercicios propuestos de libro

b. Leer el CSV del ejemplo del cap2 y obtener la estructura del schema dado por defecto.

StructType(Array(StructField(“State”, StringType, false), StructField(“Color”, StringType, false), StructField(“Count”, IntegerType, false)))

c. Cuando se define un schema al definir un campo por ejemplo

StructField('Delay', FloatType(), True) ¿qué significa el último

parámetro Boolean?

Indica si puede ser nulo o no

d. Dataset vs DataFrame (Scala). ¿En qué se diferencian a nivel de código?

Que un dataframe es una fila de un dataset

e. Utilizando el mismo ejemplo utilizado en el capítulo para guardar en parquet y guardar los datos en los formatos:

i. JSON

ii. CSV (dándole otro nombre para evitar sobrescribir el fichero origen)

iii. AVRO

ds.write.format("json").save("C:/Users/GAME/Desktop/ds_json")

ds.write.format("csv").save("C:/Users/GAME/Desktop/ds_csv")

ds.write.format("avro").save("C:/Users/GAME/Desktop/ds_avro") *da error

f. Revisar al guardar los ficheros (p.e. json, csv, etc) el número de ficheros creados, revisar su contenido para comprender (constatar) como se guardan.

i. ¿A qué se debe que hayan más de un fichero?

Hay 2 por cada particion, el porpio archivo con los datos y un crc

ii. ¿Cómo obtener el número de particiones de un DataFrame?

A través de webui

iii. ¿Qué formas existen para modificar el número de particiones de un DataFrame?

val dspart = ds.repartition(partitionExprs = col("battery_level"), numPartitions = 1) * preguntar por mas

iv. Llevar a cabo el ejemplo modificando el número de particiones a 1 y revisar de nuevo el/los ficheros guardados.

Vemos que ya solo hay un crc y un archivo con los datos

Capítulo 4

a. Realizar todos los ejercicios propuestos de libro

b. GlobalTempView vs TempView

El global puede usarse en todas las sparksession creadas dentro de la original, asi se pueden compartir información entre distintos puntos y trabajar todos con el mismo sin tener que pasar el archivo original

c. Leer los AVRO, Parquet, JSON y CSV escritos en el cap3

Capítulo 5

a. Realizar todos los ejercicios propuestos de libro (excepto los de hive en caso

de utilizar spark instalado en local y en el caso de RDBMS hacer únicamente

ejemplo especificado más adelante)

b. Pros y Cons utilizar UDFs

Pros: nos permite definir nuestras propias funciones que no se encuentran en spark y permitir que otros las utilicen

Contras: No se mantiene después de cerrar la sesión, no comprueba los valores nulos

c. Instalar MySQL, descargar driver y cargar datos de BBDD de empleados

https://dev.mysql.com/doc/employee/en/

i. Cargar con spark datos de empleados y departamentos

```
val jdbcDF = spark.read.format("jdbc").option("url",
"jdbc:mysql://127.0.0.1:3306/employees").option("driver",
"com.mysql.jdbc.Driver").option("dbtable", "employees").option("user",
"root").option("password", "stallman").load()
```

ii. Mediante Joins mostrar toda la información de los empleados además

de su título y salario.

```
JdbcDF.createOrReplaceTempView(“empleados”)

spark.sql("select empleados.emp_no, first_name, last_name, gender,
birth_date, salary from empleados inner join salarios on empleados.emp_no =
salarios.emp_no").show()
```

iii. Diferencia entre Rank y dense_rank (operaciones de ventana)

rank te devuelve el rango (si es el primero, segundo…) dentro de la particion dada, si hay un empate ambos pertenecen a la misma posición pero sigue contando y dense_rank hace lo mismo pero los rangos son consecutivos aunque haya varios empatados