|  |
| --- |
|  |
| **LABORATORIO N°15** |
| Big Data |

|  |
| --- |
| Integrantes: Cristian Álvarez – Jorge González |

Introducción

En este laboratorio se utilizará una MV especifica entregada para realizar actividades con SQLSpark y Spark Streaming, en estas actividades se realizarán una seguidilla de pasos para finalmente responder un cuestionario.

Parte 1. Actividades con SQLSpark

***Paso1.*** En este paso abrimos Jupyter Python Notebook para SparkSQL y asi poder trabajar en los siguientes pasos.

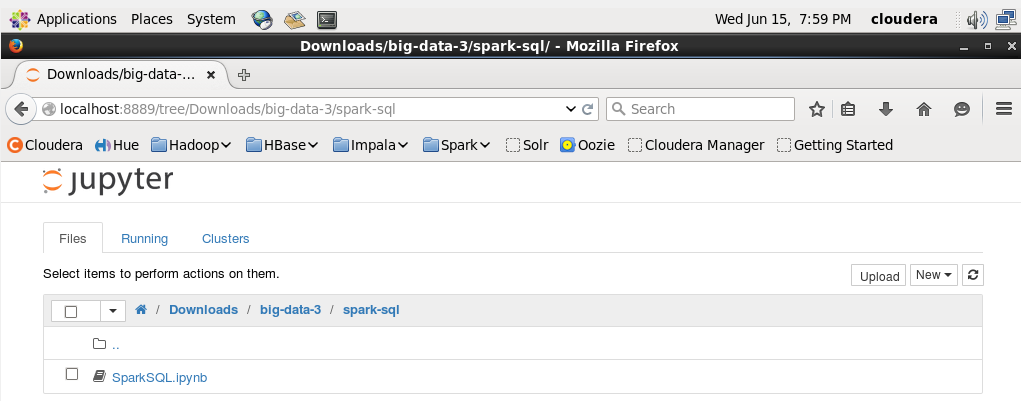
Primero vamos al terminal y ejecutamos:

pyspark

Vamos a la direccion siguiente en el navegador:

localhost:8889/tree/Downloads/big-data-3/spark-sql:

Abrimos SparkSQL.ipynb en Jyputer



***Paso2.*** En este paso ingresaremos a la tabla postgres ejecutando unos comandos específicos.

Hacemos la importación siguiente:

from pyspark.sql import SQLContext

Creamos un SQLContext desde Spark:

sqlsc = SQLContext(sc)

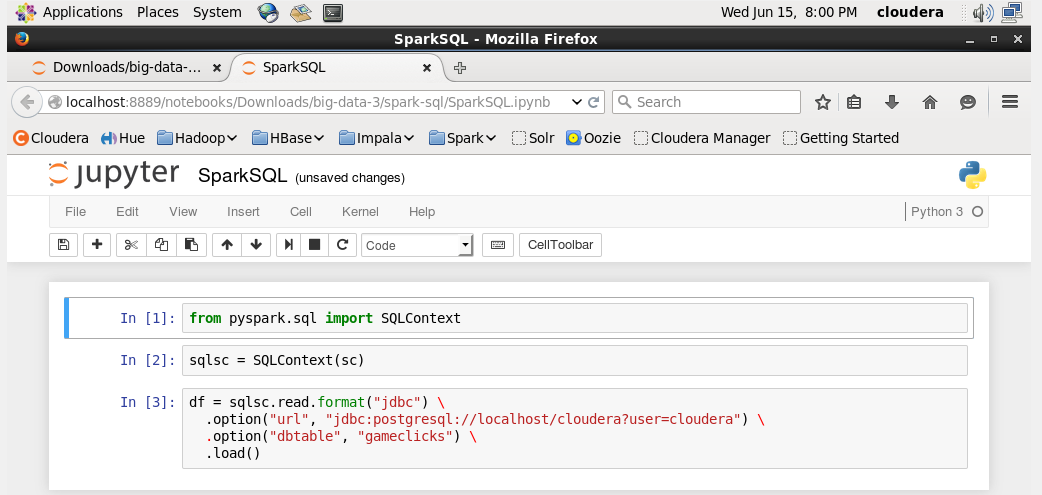
Creamos un nuevo Spark DataFrame para Postgres:

Df = sqlsc.read.format(“jdbc”) \

.option(“url”, “jdbc:postgresql://localhost/cloudera?user=cloudera”) \

.option(“dbtable”, “gameclicks”) \

.load()



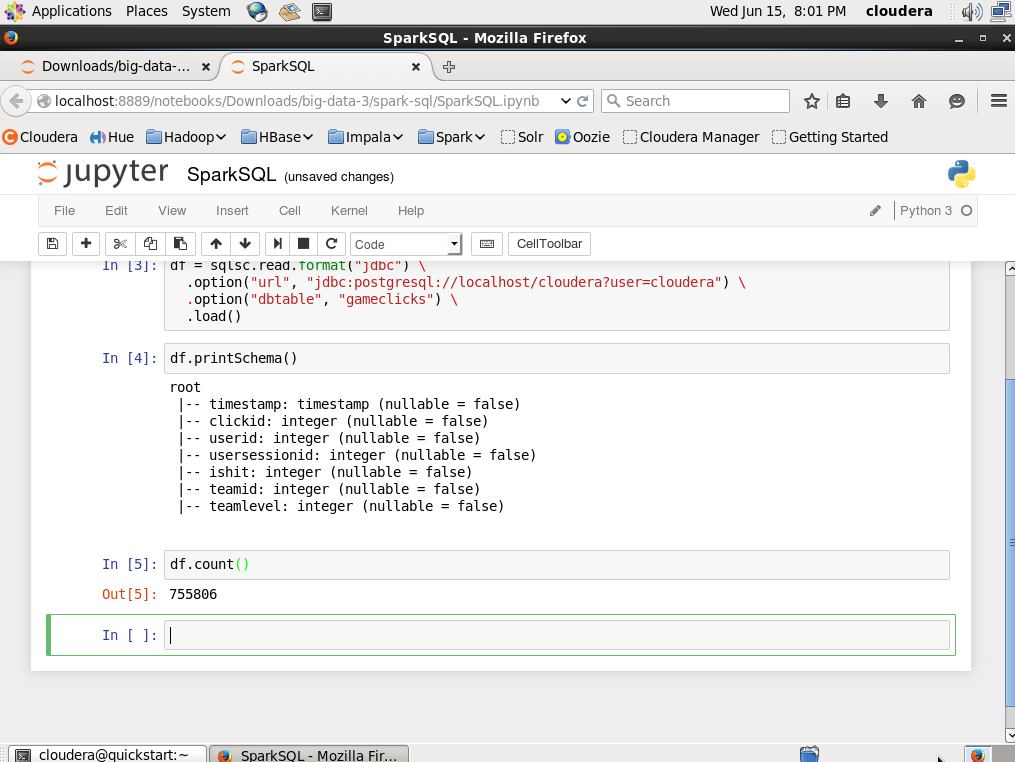
***Paso3.*** En este paso vemos el esquema de Spark DataFrame y continuamos contando las filas con determinados comandos.

Ejecutamos:

df.printSchema()

Contamos el número de filas existentes:

df.count()



***Paso 4.*** Aquí veremos el contenido que tiene DataFrame llamando algunos métodos.

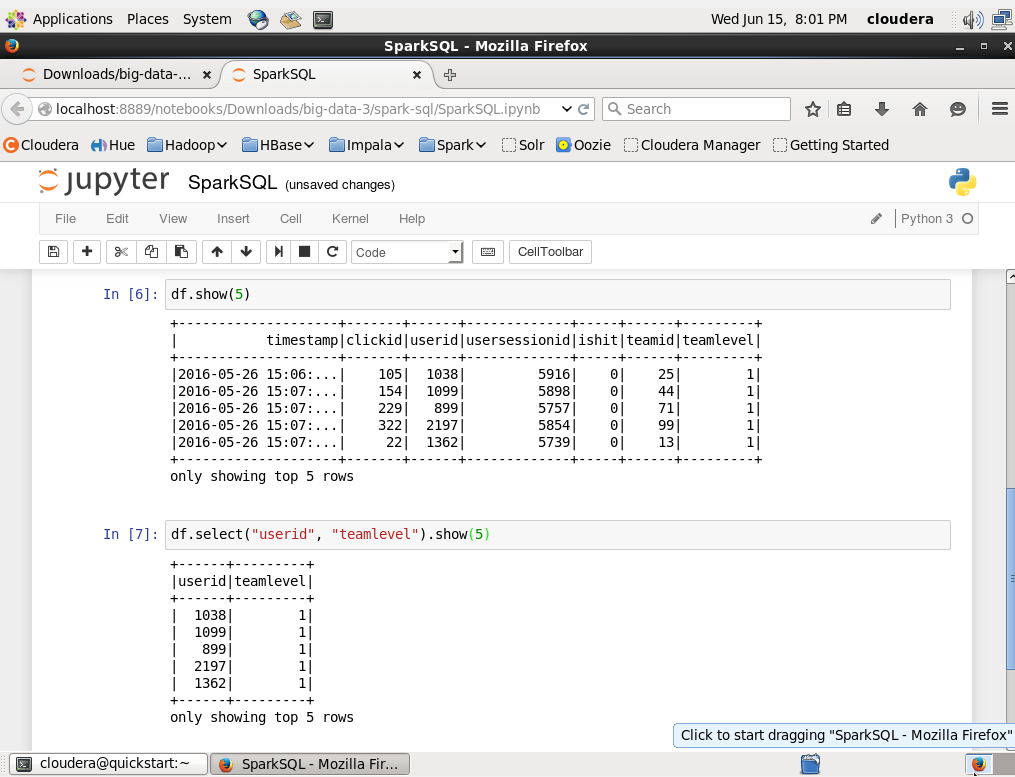
Ejecutamos:

df.show(5)

***Paso 5.*** A continuación filtraremos por una o más columnas en DataFrame

Ejecutamos:

df.select(“userid”, “teamlevel”).show(5)



*(Imagen de pasos 4 y 5)*

***Paso 6.*** Realizamos filtros según criterios generales o específicos

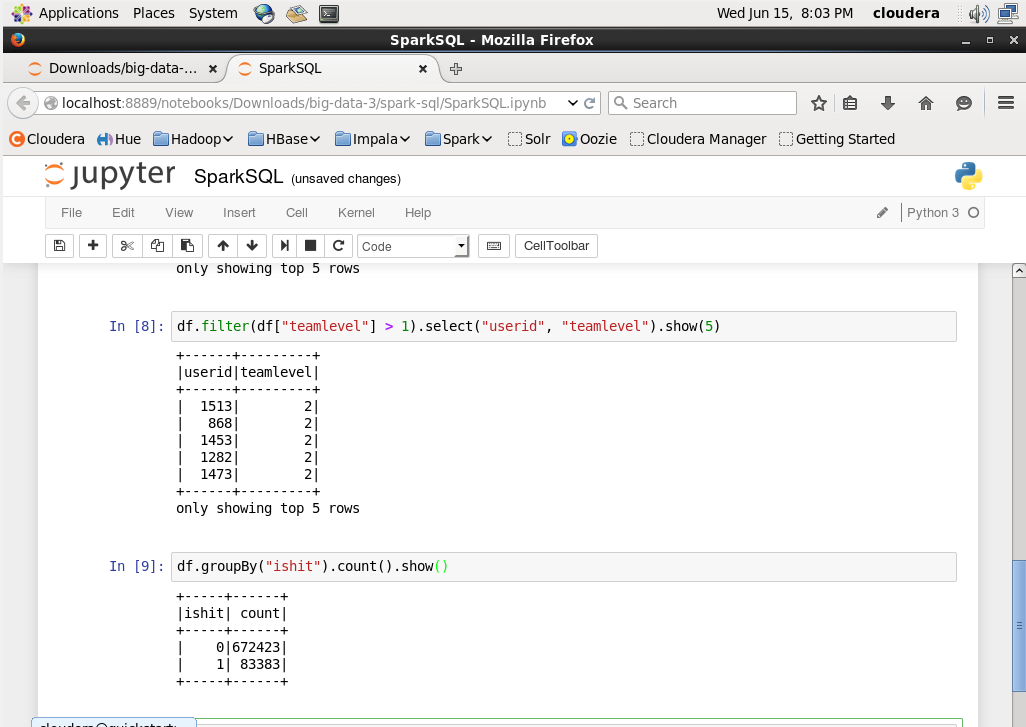
Ejecutamos:

df.filter(df[“teamlevel”] > 1).select(“userid”, “teamlevel”).show(5)

***Paso 7.*** Agrupamos las columnas y contamos las veces que se agrupa la columna “ishit”.

Ejecutamos:

df.groupBy(“ishit”).count().show()



*(Imagen de pasos 6 y 7)*

***Paso 8.*** En este paso importamos las bibliotecas de Python para poder calcular promedio y suma de las columnas de DataFrames.

Ejecutamos:

From pyspark.sql.funtions import \*

df.select(mean(‘ishit’), sum(‘ishit’)).show()

***Paso 9.*** Unimos dos DataFrames en una columna.

Ejecutamos:

df2 = sqlsc.read.format(“jdbc”) \

.option(“url”, “jdbc:postgresql://localhost/cloudera?user=cloudera”) \

.option(“dbtable”, “adclicks”) \

.load()

Vemos las columnas en df2:

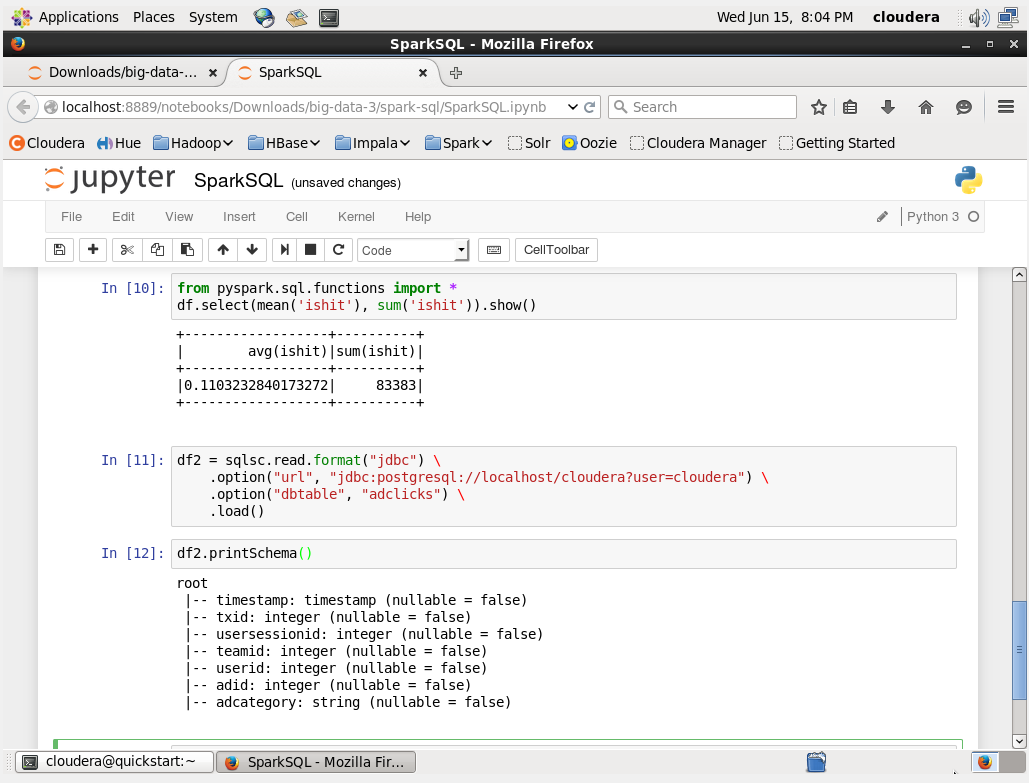
Df2.printSchema()

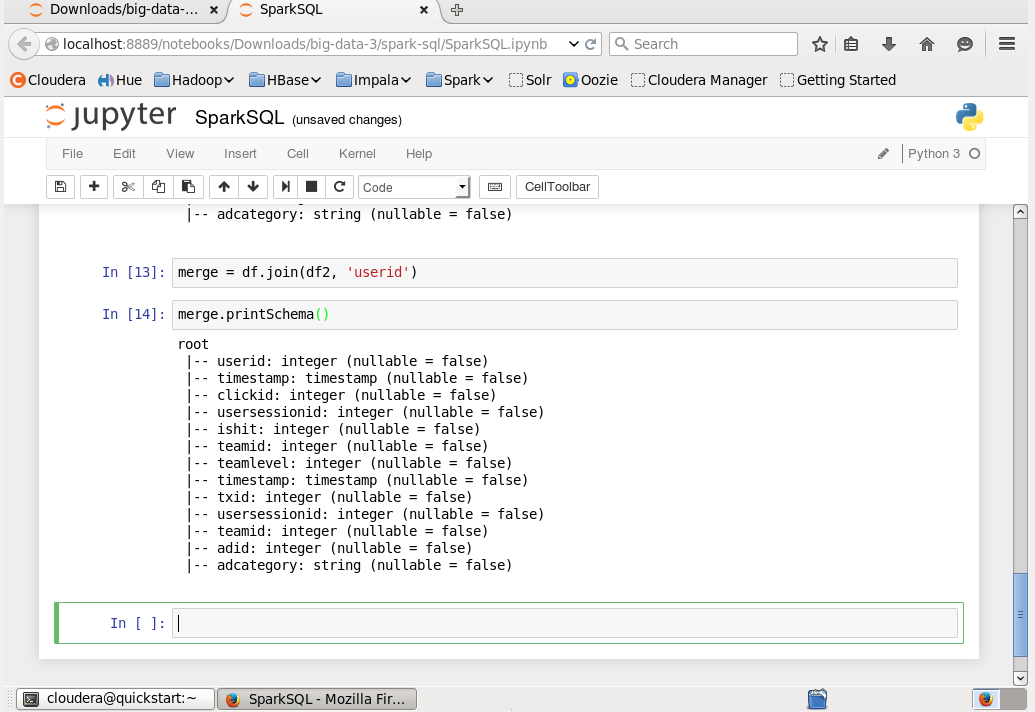
Combinamos gameclicks y adclicksDataFranes, almacenándolos en DataFrames con el siguiente comando:

merge = df.join(df2, ‘userid’)

Vemos el schema resultante:

merge.printSchema()

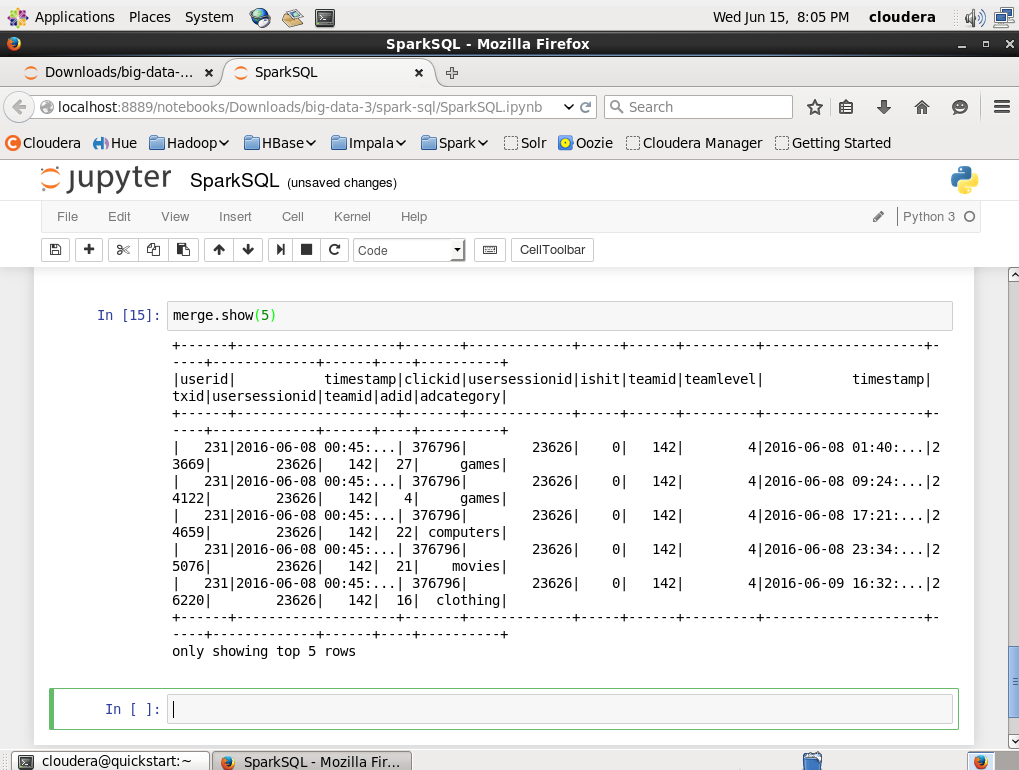




***Final.*** Visualizamos el contenido completo de merge con todas las columnas de gameclicks y adclicks

Ejecutamos:

merge.show(5)



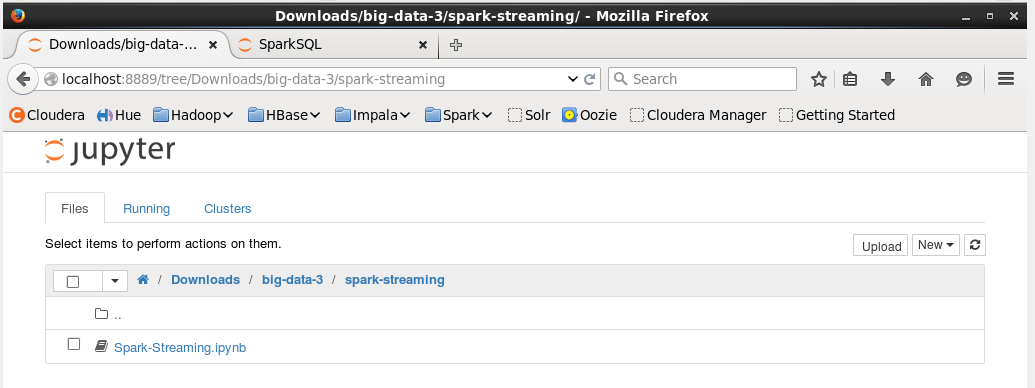
Parte 2. Actividades con SparkStreaming

***Paso 1.*** Para este paso abrimos Jupyter Notebook para Spark Streaming,

Para ello abrimos un navegador y entramos en el siguiente enlace:

localhost:8889/tree/Downloads/big-data-3/spark-streaming

Abrimos Spark-Streaming.lpynb



***Paso 2.*** Podemos mirar el formato del sensor y los tipos de medición con ejemplos que se nos entregan, teniendo cada línea marca de tiempo y conjunto de medidas, el cual estaremos interesados en la dirección promedio del viento (Dm)

Ejecutamos el siguiente comando para analizar cada línea y recibir el valor de Dm:

import re

def parse(line):

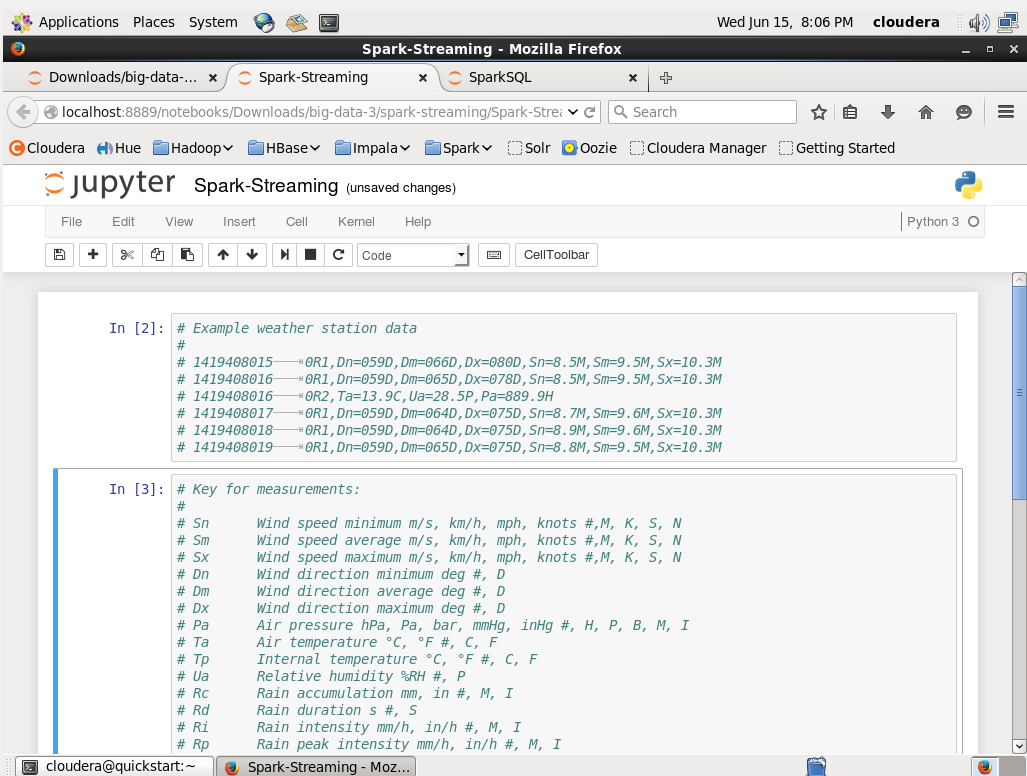
match = re.search(“Dm=(\d+)”, line)

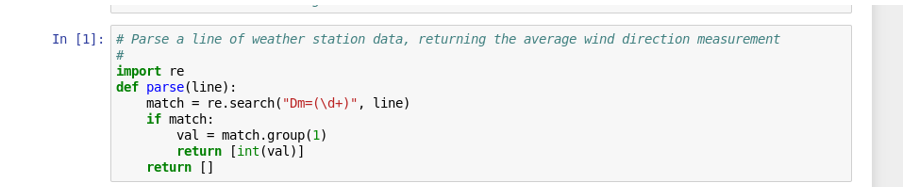
if match:

val = match.group(1)

return [int(val)]

return []





***Paso 3.*** Importamos y creamos una variable apra StreamingContext .

Ejecutamos:

from pyspark.streaming import StreamingContect

ssc = StreamingContext(sc,1)

***Paso 4.*** En este paso abrimos una conexión con el clima de transmisión

Ejecutamos:

lines = ssc.socketTextStream(“rtd.hpwren.ucsd.edu”, 12028)



*(Imagen de los pasos 3 y 4)*

***Paso 5.*** Ahora nos disponemos a leer la velocidad media del viento de cada linea y la guardamos en un Dstream

Ejecutamos:

vals = lines.flatMap(parse)

***Paso 6.*** Creamos una ventana sobre las mediciones

Ejecutamos:

window = vals.window(10, 5)

***Paso 7.*** Definimos una función de análisis ahora encontrar el máximo y el mínimo y luego poder llamarla para dada RDD en nuestra ventana creada anteriormente.

Ejecutamos:

Def stats(rdd):

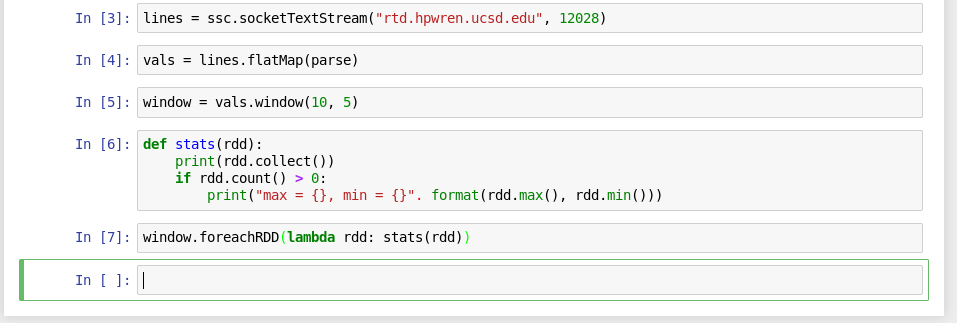
Print(rdd.collect())

If rdd.count() > 0:

Print(“max = {}, min = {}.format(rdd.max(), rdd.min()))

Llamamos a la function:

Window,foreachRDD(lambda rdd: stats(rdd))



*(Imagen de los pasos 5, 6 y 7)*

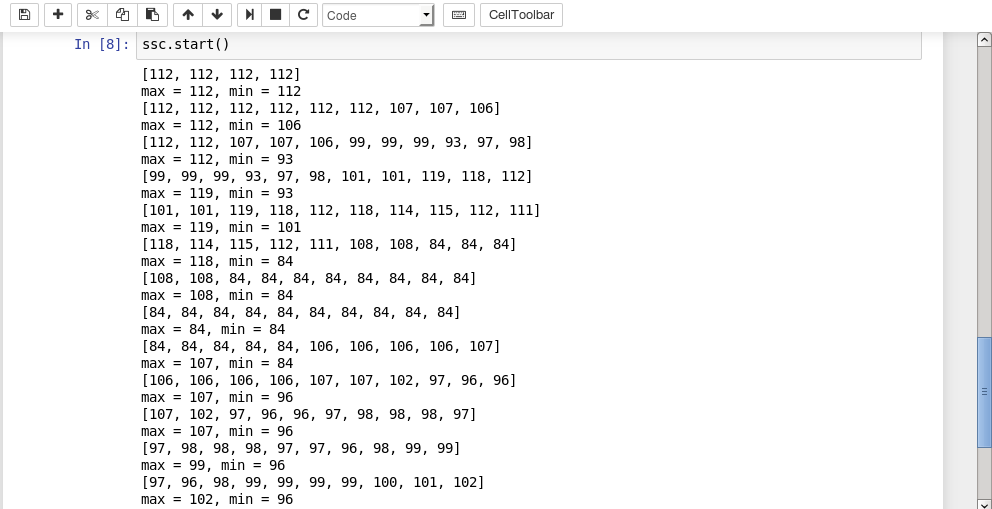
***Paso 8.*** Iniciamos el procesamiento del flujo, donde esperamos que los números de valores vaya en aumento al acumularse los datos y luego vaya reduciéndose las diferencias de números. Finalmente detenemos el proceso.

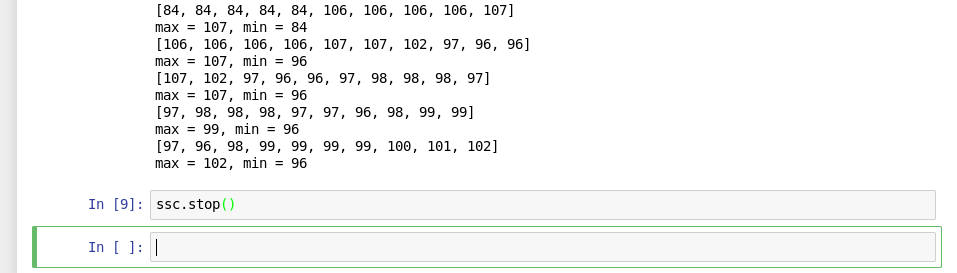
Ejecutamos:

ssc.start()

Detenemos el proceso:

ssc.stop()





Parte 3. Cuestionario

**Respuestas:**

**1)** What does the following filter line of code do? df.filter(df[“teamlevel”] > 1)

a) Select the first two columns of the data and filter each column to show only team levels

larger than 1.

b) Select the first two columns of the data and displays only team levels greater than 1.

**c) Filter each row to show only team levels larger than 1.**

d) Filter each column to show only team levels larger than 1.

**2)** What does the following do? df.select(“userid”, “teamlevel”).show(5)

a) Display all columns except “userid” and “teamlevel”.

b) Select the rows named “userid” and “teamlevel” and display first 5 rows.

c) Display all rows except “userid” and “teamlevel”.

**d) Select the columns named “userid” and “teamlevel” and display first 5 rows.**

**3)** What does the 1 represent in the following line of code? ssc = StreamingContext(sc,1)

a) To create one single context.

**b) A batch interval of 1 second.**

c) To specific debug output.

d) To create only one partition to manage the stream.

**4)** What does the following code do? window = vals.window(10, 5)

a) Creates 10 windows with 5 batch intervals inbetween.

**b) Creates a window that combines 10 seconds worth of data and moves by 5 seconds.**

c) Creates a batch interval between 10 seconds and 5 seconds.

d) Creates 10 windows with 5 seconds worth of data in them.