Hive, Pig and Spark Comparison

# Importing the data

The table must be created in Apache Hive first before importing the data into the table.

Create the table in Apache Hive:

create table clickstream(id string, datetime string, siteid string, offerid string, category string, merchant string, countrycode string, browserid string, devid string) row format delimited fields terminated by ',';

load data local inpath '/home/dante/Downloads/clickdata.csv' overwrite into table clickstream;

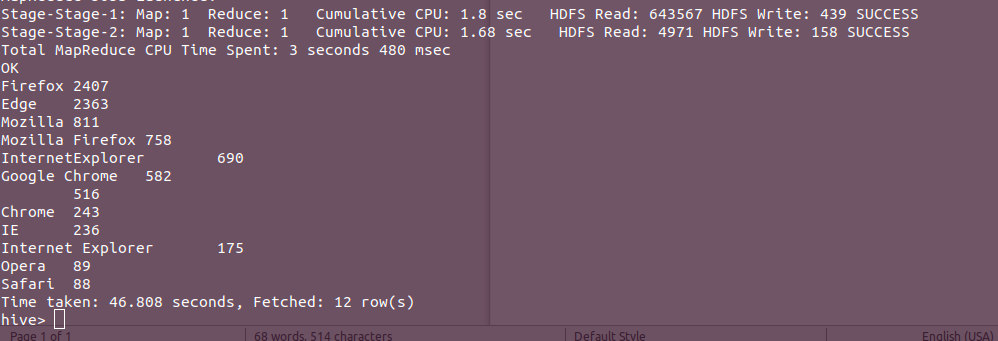
load data inpath ‘/user/dante/clickdata.csv’ into table clickstream

In Pig and Spark, there is no need to create a table, the file can be read directly from HDFS.

# Aggregation of browserID

Hive:

SELECT browserid, COUNT(\*) AS freq FROM clickstream GROUP BY browserid ORDER BY freq DESC;



Pig:

-- group the data by browser

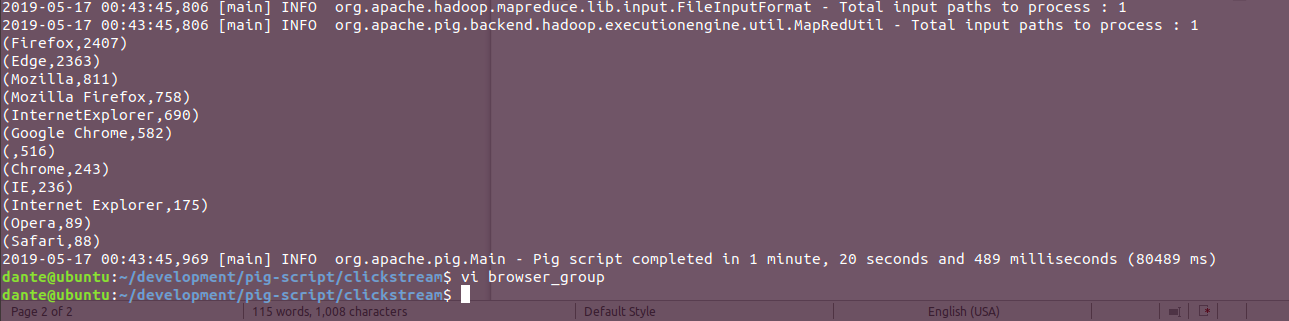
streamed\_data = LOAD '/user/dante/clickdata.csv' USING PigStorage(',') AS (id:chararray, datetime:chararray, siteid:chararray, offerid:chararray, category:chararray, merchant:chararray, countrycode:chararray, browserid:chararray, devid:chararray);

by\_browser = GROUP streamed\_data by browserid;

by\_browser\_count = FOREACH by\_browser GENERATE FLATTEN(group) as browserid, COUNT($1);

sorted\_browser\_count = ORDER by\_browser\_count BY $1 DESC;

DUMP sorted\_browser\_count;



Hive has a slight edge over pig in term of performance. Hive took 46.808 seconds to produce the aggregation and Pig took 80 seconds to produce the same result.

# Cleansing the data

## Redundancy in the data

From the aggregation result above, it is seen that there is redundancy in the dataset, same value is presented in several forms for instance, mozilla, firefox, mozilla firefox are all referring to the same browser which is Mozilla Firefox. These instances need to be standardized.

## Date format

The date format in the file is in the form of “22/1/2017 9:55” which has an issue when importing into Hive. This is because the format doesn’t conform with Hive’s timestamp format which is in the form of YYYY-MM-DD HH:MM:SS.fffffffff. My current table structure import the date time as string.

### Pig

-- Cleansing the clickstream data with Pig

streamed\_data = LOAD '/user/dante/clickdata.csv' USING PigStorage(',') AS (id:chararray, datetime:chararray, siteid:chararray, offerid:chararray, category:chararray, merchant:chararray, countrycode:chararray, browserid:chararray, devid:chararray);

-- Convert the datetime from string to datetime format

datetime\_col = FOREACH streamed\_data GENERATE id, ToDate($1, 'dd/MM/yyyy HH:mm');

-- Cleansing the redundancy in browserid column

cleansed\_firefox = FOREACH streamed\_data GENERATE id, REPLACE(browserid, '^Firefox$', 'Mozilla Firefox') AS browserid;

cleansed\_mozilla = FOREACH cleansed\_firefox GENERATE id, REPLACE(browserid, '^Mozilla$', 'Mozilla Firefox') AS browserid;

cleansed\_inte = FOREACH cleansed\_mozilla GENERATE id, REPLACE(browserid, '^InternetExplorer$', 'Internet Explorer') AS browserid;

cleansed\_ie = FOREACH cleansed\_inte GENERATE id, REPLACE(browserid, '^IE$', 'Internet Explorer') AS browserid;

cleansed\_chrome = FOREACH cleansed\_ie GENERATE id, REPLACE(browserid, '^Chrome$', 'Google Chrome') AS browserid;

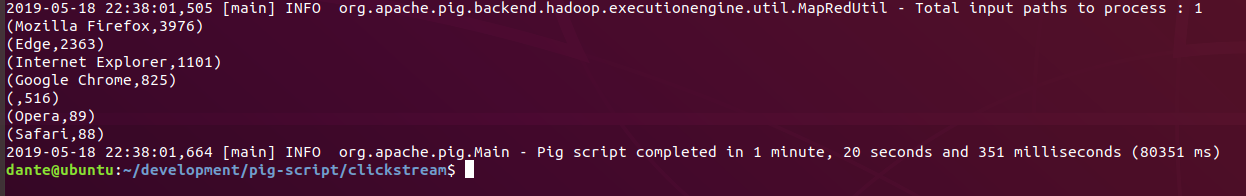
-- Join the columns back together

join\_data = JOIN streamed\_data by id, datetime\_col by id, cleansed\_chrome by id;

cleansed\_data = FOREACH join\_data GENERATE $0 AS id, $10 AS datetime, $2 AS siteid, $3 AS offerid, $4 AS category, $5 AS merchant, $6 AS countrycode, $12 AS browserid, $8 AS devid;

STORE cleansed\_data INTO '/user/dante/cleansed\_clickstream' USING PigStorage(',');

Aggregation result after cleansing



### Hive

Hive database is a data warehouse so it won’t support updating or editing on the value of the columns. However there is a work around to cleanse the data in the column.

The command below cleansed the browser column and aggregate the browser count.

SELECT cleansed.n\_browserid, COUNT(\*) AS freq FROM (SELECT \*, CASE WHEN click.browserid = 'Mozilla' THEN "Mozilla Firefox"

WHEN click.browserid = 'Firefox' THEN 'Mozilla Firefox'

WHEN click.browserid = 'Mozilla Firefox' THEN 'Mozilla Firefox'

WHEN click.browserid = 'InternetExplorer' THEN 'Internet Explorer'

WHEN click.browserid = 'IE' THEN 'Internet Explorer'

WHEN click.browserid = 'Internet Explorer' THEN 'Internet Explorer'

WHEN click.browserid = 'Chrome' THEN 'Google Chrome'

WHEN click.browserid = 'Google Chrome' THEN 'Google Chrome'

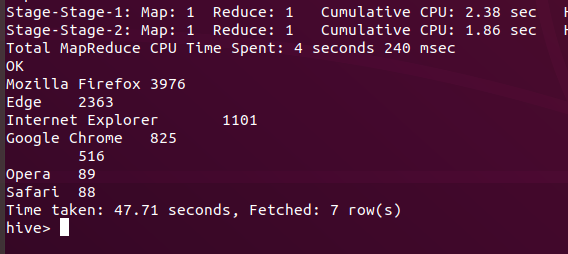
WHEN click.browserid = 'Edge' THEN 'Edge'

WHEN click.browserid = 'Opera' THEN 'Opera'

WHEN click.browserid = 'Safari' THEN 'Safari'

ELSE ''

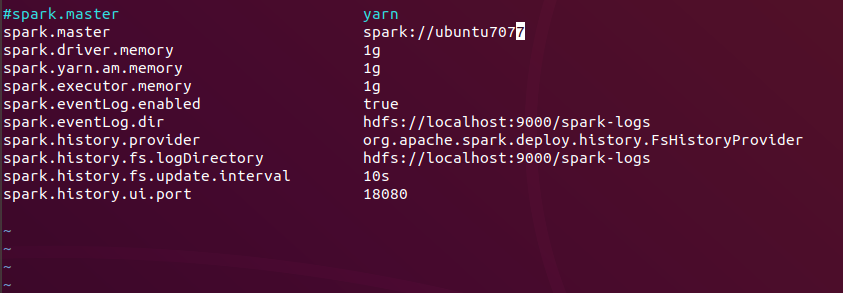
END AS n\_browserid FROM clickstream click) cleansed GROUP BY cleansed.n\_browserid ORDER BY freq DESC;



# Apache Spark

Installing Apache Spark

1. Download the tar.gz file from <https://spark.apache.org/downloads.html>
2. Extract the file and move it to a desired location
3. Install Scala
4. Include spark/bin and scala/bin directory into the PATH
5. Configure conf/spark-defaults.conf



1. start spark master, slaves and history server
2. start spark shell with scala with the command: spark-shell –master spark:IP:PORT
3. or start spark shell with python with the command: pyspark –master spark:IP:PORT

## Pyspark

### Aggregation by browserid

df = spark.read.format("csv").option("header", "true").load("hdfs:////user/dante/clickdata\_header.csv")

browserGroup = df.groupBy("browserid").count()

browserGroup.orderBy('count', ascending=False).show()

### Cleansing

from pyspark.sql.functions import when

df2 = df.withColumn("browserid", when (df['browserid'] == 'Firefox', 'Mozilla Firefox').otherwise(df['browserid']))

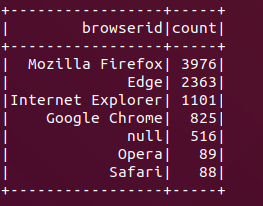
df2 = df2.withColumn('browserid', when (df2['browserid'] == 'Mozilla', 'Mozilla Firefox').otherwise(df2['browserid']))

df2 = df2.withColumn('browserid', when (df2['browserid'] == 'InternetExplorer', 'Internet Explorer').otherwise(df2['browserid']))

df2 = df2.withColumn('browserid', when (df2['browserid'] == 'IE', 'Internet Explorer').otherwise(df2['browserid']))

df2 = df2.withColumn('browserid', when (df2['browserid'] == 'Chrome', 'Google Chrome').otherwise(df2['browserid']))

df2.write.csv('hdfs:////user/dante/spark-cleansed')



## Spark Shell (Scala)

### Aggregation by browserid

val df = spark.read.format("csv").option("header","true").load("hdfs:///user/dante/clickdata\_header.csv")

val browserGroup = df.groupBy("browserid").count()

browserGroup.sort(desc("count")).show()

### Cleansing

import org.apache.spark.sql.functions.\_

val df2 = df.withColumn("browserid", when(col("browserid") === "Firefox", "Mozilla Firefox").otherwise(col("browserid")))

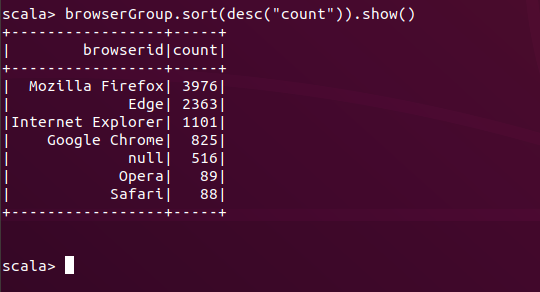
val df3 = df2.withColumn("browserid", when(col("browserid") === "Mozilla", "Mozilla Firefox").otherwise(col("browserid")))

val df4 = df3.withColumn("browserid", when(col("browserid") === "InternetExplorer", "Internet Explorer").otherwise(col("browserid")))

val df5 = df4.withColumn("browserid", when(col("browserid") === "IE", "Internet Explorer").otherwise(col("browserid")))

val df6 = df5.withColumn("browserid", when(col("browserid") === "Chrome", "Google Chrome").otherwise(col("browserid")))

df6.write.csv("hdfs:///user/dante/scala-cleansed")



With python program

