# Big Data Analytics

## Assignment 6 (hands-on with Hive and Spark SQL)

This assignment has two short programming problems. For both problems, you need to use hiveql in spark and turn in your spark program.

## Problem I ( 11 points).

For this problem, you are given two input files:

1. **ratings dataset** containing the following information. Each row in this dataset is in the following format:

   User_id movie_id1#movie_id2#movie_id3,....

   The first column is the id of a user, and the second column is the ids of all the movies that this user watched. The movie ids are separated by '#'.
2. **Movies dataset.** This file contains movies information and has the following format (genres are delimited by "|") :

   Movie_id#movie_title#genre1|genre2|genre3|...

Write a spark code that **finds all pairs of users who watched more than 50 common movies and sort them by the number of common movies.** Your output should be a table with the following schema:

**User_id1 user_id2 count_of_common_movies a comma-separated list of the titles of the common movies watched by user_id1 and user_id2**

**SOME GUIDELINES:**

1- Use spark to pre-process the files and extract the information you need. Then use Hive Queries in spark to find common movies for each pair of users and filter the ones who have more than 50 common movies. Before using hiveQL, you probably need to put each user_id,movie_id pair in the ratings file into a separate row.

2- To produce a comma-separated list of common movies watched by a pair of users, you can use the aggregate function "collect_set(col)" in hive, which returns a comma-separated list of the items in a columnor group. Please refer to hive documentation for more information: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

**Running your Script**

The size of the data is not very big (only 2.3MB for ratings dataset and 160KB for movies dataset). However, you need to use join on this dataset and that could produce several of GB of intermediate and final results.

I strongly recommend that you test your program first on a much smaller sample of dataset. For example, copy the first 50 lines of ratings.dat in another file and test your code on this smaller input.

Once you are sure that your code works correctly on a smaller sample, run it on the entire ratings.dat dataset. Before running your spark, please make sure that you clear up all the old files from your HDFS.

**What you need to turn in:**

**Your spark code ( .scala file for spark scala) or (.py for spark python)**

## Problem II (4 points).

For this problem, you are to write a hive query in spark that finds the top most 10 frequent 3-grams (trigrams) that appear across a collection of Shakespeare's books .

**The Input Dataset:**

The input dataset is a directory of 10 files where each file contains the plain text of a single Shakespeare's book. The name of the file is the title of the book. You can download the zip file from blackboard.

**Output File:**

Your program should produce an output file containing the top 10 frequent trigrams. Here is a sample record from the output file your program should produce:

```
{"ngram":["my","lord","i"],"estfrequency":35.0}
```

**Hint:** You need to create a table consisting of a single column "line" and partition your table by the book_name. Load all the books into the table (Note: you need to have a separate load statement for each book) The rest is a piece of cake :D just use the sample statement provided in the lecture notes to retrieve the ngrams).

**What you need to turn in:**

1. **Your spark code ( .scala file for spark scala) or (.py for spark python)**
2. **Your output file** with the top 10 most frequent ngrams. If you have multiple output parts, please merge it into one file. In unix you can merge multiple files , e.g., file1, file2, file3 as follows:
   **Cat file1 file2 file3 >> merged_file**

## Optional Bonus Problem (+5 bonus points)

For this bonus question, in addition to the top 10 ngrams in all books, your spark program should also output the title of the books that contain the ngram and the number of times that the ngram occurred in each book. For example, for the ngram "my lord I" you should get the following records

in the output:

```
{"ngram":["my","lord","i"],"estfrequency":33.0}    Hamlet      13
{"ngram":["my","lord","i"],"estfrequency":33.0}    King Richard III 6
{"ngram":["my","lord","i"],"estfrequency":33.0}    The Tragedy of
King Lear  6
{"ngram":["my","lord","i"],"estfrequency":33.0}    Othello    5
{"ngram":["my","lord","i"],"estfrequency":33.0}    The Tragedy of
Julius Casear                                      2
{"ngram":["my","lord","i"],"estfrequency":33.0}    A Midsummer
Night Dream 1
```

This means that the trigram “my lord I” had an estimated frequency of 33 in all books and it appeared 13 times in Hamlet, 6 times in King Richard III6, etc.

You might be interested to know that Google maintains a similar n-gram dataset on all the google books.you can read more about this datasethere: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html )

Also this Amazon article explains how you can use Hive to identify trending topics on google’s ngrams dataset.