7CCSMBDT – Big Data Technologies Coursework 2

Coursework assigned: 8 March 2019.

Coursework deadline: 4:00pm, 22 March 2019.

Feedback: 22 April 2019.

Late submission deadline (capped at 50%): 4:00pm, 23 March 2019.

Overview: The coursework aims to make you familiar with the following technologies: (i) MongoDB, (ii) Apache Spark, and (iii) Hive.

This coursework is formally assessed and is worth 10% of your final mark.

You will receive feedback as part of the marking of the coursework.

Submission: Include:

(i) A file, **Coursework2.PDF**, containing your answers. For questions that require writing code, write your code as part of the answer **in this PDF** (see each task below for details). For questions that require output of a program provide the output (or a small part- first few lines – if the output is too large) **in this PDF**.

(ii) A file, **Coursework2_code.ZIP**, containing queries, query outputs, and code (see each task below for details).

Please make sure that the code in the PDF is exactly the same as the code in the .zip, unless otherwise requested. Inconsistencies will be penalized.

Evaluation: The maximum number of marks (out of 100) for each task is given in square brackets [] next to each question.

Plagiarism: “Plagiarism is passing off someone else’s work as your own, or submitting a piece of your own work that you have already submitted as part of a different programme, module or at a different institution. The penalties for plagiarising by the College can be severe. Uploading work to KEATS is regarded by the Department as a statement by the student concerned, confirming that the work has not been plagiarised.”

Late submission: "If you are submitting your coursework after the deadline, you must submit an Extension Request Form to your Programme Administrator, with evidence to justify why you have not submitted on time. If you do not do this or your reasons are not acceptable, your coursework may be given a mark of zero." Please check your handbook and contact your personal tutor for more information.

Task 1. mongoDB queries [total marks 35]

1) Load the file championsleauge_1.json (available in KEATS) into a database cw2 and a collection cl. **[5]**

Write **a single** query, for each subtask 2 to 6 below.
It is ok if your queries output "_id" too.

2) Write a query that returns the name, number of followers, and number of friends, of each user with fewer than 25 friends, whose name starts with "A" (case insensitive) and ends with "es" (case sensitive). The results must be sorted in decreasing order of displayName. **[6]**

3) Write a query that returns the average number of followers of users with more than 100000 friends. **[6]**

4) Write a query that returns the average ratio between numbers of followers and number of friends, over all documents. Note that the number of ~~followers~~ must be >0 for the ratio to be defined. **[6]** friends

5) Write a query that returns the number of users who have at least 1000 friends, posted a tweet (the field verb has the value "post") and whose tweet (field body) contains the string “Madrid” (even as part of a word). **[6]**

6) Write a query that displays the number of followers of users who have more than 200 and fewer than 203 statuses (the number of statuses is in statusesCount). **[6]**

Submission instructions: Your PDF file should contain each query and a small part of its output. Your zip file should contain a directory “task1” with files query1.txt , ... , query6.txt. Each txt file should contain the query and then the output. Inconsistencies between the pdf and .txt files will be penalized.

Task 2. Apache Spark [total marks 35]

Download the file u.user from KEATS. This file has lines that correspond to users with the following attributes: user id, age, gender, occupation, zipcode. For example, the following part of u.user has 5 attribute values (namely 1, 24, M, technician, 85711):
1|24|M|technician|85711

Complete the following program, so that it outputs all occupations that are performed by users in the age group [40,50) and by users in the age group [50,60) and are among the 10 most frequent occupations for the users in each age group.

```
from pyspark import SparkContext
from operator import add
sc = SparkContext( 'local', 'pyspark')
def age_group(age):
    if age < 10 :
        return '0-10'
    elif age < 20:
        return '10-20'
    elif age < 30:
        return '20-30'
    elif age < 40:
        return '30-40'
    elif age < 50:
        return '40-50'
    elif age < 60:
        return '50-60'
    elif age < 70:
        return '60-70'
    elif age < 80:
        return '70-80'
    else :
        return '80+'


def parse_with_age_group(data):
    userid,age,gender,occupation,zip = data.split("|")
    return userid, age_group(int(age)),gender,occupation,zip,int(age)
# WRITE YOUR CODE HERE
```

Important note: Your program must make use of Apache Spark commands and it must include comments that explain its operation. Note that data structures in Apache Spark (RDDs) are distributed, while python data structures (e.g., dictionaries) are not. Your program must make use of distributed data structures, whenever possible, so that it is scalable.

Submission instructions: Your PDF file should contain the entire code (including the part of code given above). Your zip file should contain a directory task2, with two files, one with the entire code (e.g., task2.py) and one with the output (e.g.,task2.out). Recall that, to execute such a script, you need to write

spark-submit task2.py --master local[2]

in Cloudera. Inconsistencies between the PDF and the zip file will be penalized.

Task 3. HIVE [total marks 30]

Download the file query_logs.txt from KEATS. This file contains searches made by different users. Specifically, there are three tab-separated attributes user, time, and query. The user attribute may appear in many lines, meaning that the user has issued different queries (attribute query), at different times (attribute time).

For example, in the following part of query_logs.txt, there are two users, 2A9EABFB35F5B954 and BED75271605EBD0C. The first user issued one query +md foods +proteins at time 970916105432 and the third user issued three queries (each of them was yahoo chat) at times 970916001949, 970916001954, and 970916003523.

2A9EABFB35F5B954 970916105432 +md foods +proteins

BED75271605EBD0C970916001949 yahoo chat

BED75271605EBD0C 970916001954 yahoo chat

BED75271605EBD0C 970916003523 yahoo chat

There are also lines where query is empty. These correspond to visits where a query was not issued. For example, the user below visited the website at time 970916074828 but issued no query.

893C3ADD0EFBBECB 970916074828

1) Create a database log_db and a table logs that has the attributes user, time, query, all of which are of type STRING

The table must contain all records of the file query_logs.txt Note that the fields are terminated by '\t'. **[6]**

2) Write a query that returns the maximum number of visits of all users. Note that in each page visit a user may or may not issue a query. **[6]**

3) Write a query that returns, for each user, the number of queries that the user performed and end with the string "business". **[6]**

4) Write a query that returns all attributes of each user who issues a query that contains the string "job". Note that "job" may appear within a word (e.g.,"job" is part of "jobs"). **[6]**

5) Write a query that returns the number of distinct users who issue a (non-empty) query between 21:00:00 (inclusive) and 22:59:59 (inclusive). Note that the second attribute in query_logs.txt is in the form yyMMddHHmmss . For example, in 970916105432 yy=97, MM=09, dd=16, HH=10, mm=54, ss=32, which means that the query was issued in 1997, September, 16th, and at time 10:54:32. **[6]**

Submission instructions: Your pdf file should contain the query and a small part of the output. Your zip file should contain a directory "task3" with files query1.txt , ... , query5.txt. Each txt file should contain the query and then the output. Inconsistencies between the PDF and zip file will be penalized.

[End of Coursework]