**CS6350 --Spring 2021**

**Homework #2**

**Submission Deadline: 11:59 pm, April 4, 2021**

**In this homework, you will use spark (pyspark or scala) to solve the following problems.**

**Q1:**

**Write a spark script to find total number of common friends for any possible friend pairs. The key idea is that if two people are friend then they have a lot of mutual/common friends.**

**For example**,

Alice’s friends are Bob, Sam, Sara, Nancy Bob’s friends are Alice, Sam, Clara, Nancy Sara’s friends are Alice, Sam, Clara, Nancy

As Alice and Bob are friend and so, their mutual friend list is [Sam, Nancy]

As Sara and Bob are not friend and so, their mutual friend list is empty. (**In this case you may exclude them from your output**).

Input files:

1. soc-LiveJournal1Adj.txt

**The input contains the adjacency list and has multiple lines in the following format:**

<User><TAB><Friends>

Here, <User> is a unique integer ID corresponding to a unique user and <Friends> is a comma-separated list of unique IDs (<User> ID) corresponding to the friends of the user. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A. The data provided is consistent with that rule as there is an explicit entry for each side of each edge. So when you make the pair, always consider (A, B) or (B, A) for user A and B but not both.

**Output: The output should contain one line per user in the following format:**

<User\_A>, <User\_B><TAB><Mutual/Common Friend Number>

where <User\_A> & <User\_B> are unique IDs corresponding to a user A and B (A and B are friend). < Mutual/Common Friend Number > is total number of common friends between user A and user B.

**Q2.**

**Please answer this question by using dataset from Q1.**

**Find friend pair(s) whose number of common friends is the maximum in all the pairs.**

Output Format:

<User\_A>, <User\_B><TAB><Mutual/Common Friend Number>

Please use the following dataset.

1. soc-LiveJournal1Adj.txt

Please use Apache Spark to derive some statistics from **Yelp Dataset.**

**Data set info:**

The dataset files are as follows and columns are separate using ‘**::**’

**business.csv. review.csv. user.csv.**

**Data set Description.**

The data set comprises of **three** csvfiles, namely user.csv, business.csv and review.csv.

**Business.csv** file contain basic information about local businesses.

**Business.csv** file contains the following columns: "business\_id"::"full\_address"::"categories"

'business\_id': (a unique identifier for the business)

'full\_address': (localized address),

'categories': [(localized category names)]

**review.csv** file contains the star rating given by a user to a business. Use user\_id to associate this review with others by the same user. Use business\_id to associate this review with others of the same business.

**review.csv** file contains the following columns: "review\_id"::"user\_id"::"business\_id"::"stars"

'review\_id': (a unique identifier for the review)

'user\_id': (the identifier of the reviewed business),

'business\_id': (the identifier of the authoring user),

'stars': (star rating, integer 1-5), the rating given by the user to a business

**user.csv file** contains aggregate information about a single user across all of Yelp

**user.csv file** contains the following columns:

"user\_id"::"name"::"url"

‘user\_id': (unique user identifier),

'name': (first name, last initial, like 'Matt J.'), this column has been made anonymous to preserve privacy

'url': url of the user on yelp

**Note**: :: is Column separator in the files.

**Q3:**

**Please list the 'user id' and 'rating' of users that reviewed businesses which are located in “Stanford”**

This will require you to use **review.csv** and **business.csv files.**

**Sample output:**

|  |  |
| --- | --- |
| **User id** | **Rating** |
| 0WaCdhr3aXb0G0niwTMGTg | 4.0 |

**Q4:**

**List the business\_ID, full address, and categories of the Top 10 businesses using the average ratings**

This will require you to use **review.csv** and **business.csv files.**

**Sample output:**

**business id full address categories avg rating**

xdf12344444444 CA 91711, List ['Local Services', 'Carpet Cleaning'] 5.0

**Q5:**

**For each category, find total number of business counts.**

**Q6:**

**Find top 10 categories which have the most business count**

**For example:**

+-----------------+--------+

|category |count|

+-----------------+--------+

| Restaurants |38917|

**Inverted Index:**

**Q7.**

Write a **Spark** program that will construct inverted index in the following ways.

The **map and additional** transformations parse each line in an input file, userdata.txt, and emit a sequence of <word, line number> pairs. The **reduce** and additional transformations accept all pairs for a given word, sort the corresponding line numbers, and emits a <word, list(line numbers)> pair. The set of all the output pairs forms a simple inverted index.

What to submit

(i) Submit the source code via the eLearning website. (ii) Submit the output file for each question.