Big Data Lab Answer Sheet.

Please complete this answer sheet and turn it in at the beginning of class on the due date posted in LEARN.

Part I

|  |  |
| --- | --- |
| **Part 1:** | Answer |
| **1**  (2 pts) | Temboo worked for me.  The data is delivered in a simple unformatted version that becomes easier to read when used as input for a JSON formatter. The data was organized by the information I was looking for in a list-like format. |
| **2**  (2 pts) | I chose JSON because I’m less familiar with it, so I wanted to give it a go. Also, the link was the first choice available, and the website seemed like it would do the job, which it did. |
| **3**  (2 pts) | 24.64%  24.77%  24.83% |
| **4a**  (1 pts) | There are 38 records. Adams is the last name. |
| **4b**  (1 pts) | There are still 38 records. |
| **4c**  (3 pts) | A full-text search is a search method that compares every word of a search request with every word in a selected document or database. |
| **4d**  (4 pts) | Databases are indexed to make relevant information easier to find. |
| **4e**  (1 pts) | Two records were found. If I were searching for a female with the last name “Adams,” using the filter would be very useful. The records are now far more useful if I were looking for the aforementioned information. |
| **4f**  (1 pts) | 12451 records were found. |
| **4g**  (1 pts) | 332 records were found. |
| **Part 2:** | Answer |
| **5**  (2 pts) | Each word in a sentence is assigned a value in a “key.” |
| **6**  (2 pts) | Each mapper uses a sentence per “step” as its piece of data. |
| **7**  (2 pts) | It’s easier to debug a program when it is segmented into smaller parts. The program could run faster as it becomes larger because more parts are taken out. |
| **8**  (2 pts) | The reducer grabs individual letters instead of entire words. |
| **Part 2: (cont’d)** | Answer |
| **9**  (2 pts) | The word list is chopped up into individual words by splitting at every space in between words. Each time the word is found in the word list, the word is added to an individual count per word. The word counts for every word are emitted for the reducer to log. |
| **10**  2 pts) | Each reducer uses jsmr context as input. Each reducer emits the word (“key”) and the value (amount) of each word. |
| **11**  (5 pts) | Log-in doesn’t work.  Different layout for a different language. The code was converted correctly. |
| **12**  (15 pts) | Nothing I can do because the log-in doesn’t work |
| **13**  (20 pts) | def mapper(key, value):  grade\_map = eval(key) # automatically parses the data in JSON format  # grade\_map['Alice'] contains 95 87 or 63 (and so on) depending on which row the mapper is reading  for student in grade\_map: # student is 'Alice', 'Bob', and so on  # get each of the 4 grades in your mapper data line  grade = grade\_map[student]  Wmr.emit(student, grade)  def reducer(key, values):  sum = 0  count = 0  for value in values:  sum = sum + float(value)  count = count + 1  if count > 0:  average = sum / count  Wmr.emit(key, average)  Used what worked already because the website doesn’t work. |