Design Laboratory (CS69202)

Spring Semester 2024

Assignment 5.2: NoSQL - Mapper, Combiner, Reducer

Assignment date: February 02, 2024 (In Class Assignment)

Important Instructions:

1. You need to write Mapper, Reducer and Combiner codes for the Queries. You have to fill the folders for Queries. You also need to write Make files so that on executing 'make' command in the terminal, the file does the needful and gives the necessary outputs.
2. Write Python code for the above Queries.
3. You must NOT use any libraries apart from sys, datetime, time module.
4. You can use 1D lists and Dictionaries as and when required based on Queries. You CANNOT use any other data structures for the assignment.
5. Not adhering to these instructions can incur a penalty (worst case being 0 marks).
6. You can write a readme file to provide any particular instructions related to program execution steps, input format, or anything that you might think is useful for the evaluator while evaluating the assignment.
7. Not writing Makefiles for the Queries will incur 50% of the marks against corresponding Queries
8. Plagiarism in any form is not allowed. Students found copying/sharing code will be awarded 0 marks. Any discussions among students are NOT allowed for the assignment.
9. All errors should be handled properly.
10. Also submit a README file.
11. Save this in a folder named in the format: <Roll No.>_DesLab_A5_2. Compress this folder to zip format, creating a compressed file <Roll No.>_DesLab_A5_2.zip. Upload this compressed file to moodle. Example: If your roll no. is 22CS60R05, the folder should be 22CS60R05_DesLab_A5_2, and the compressed file should be 22CS60R05_DesLab_A5_2.zip.

Problem Statement :

Tour de Rentals are renting cycles to its customers. They stored the data for January 2021 in 6 files event_1.txt, event_2.txt,..., event_6.txt. The attributes are as mentioned below (from left to right) :

ride_id : Unique ID of Customer
rideable_type : Type of bike used
started_at : Start Time of Ride
ended_at : End Time of Ride
start_station_name : Originating Ride Station
start_station_id : ID of the Originating station
end_station_name : Ending Ride Station
end_station_id : ID of the Ending Ride station
member_casual : Whether customer is casual rider or member of the Rentals

A screenshot of 50 such data tuples of event_i.txt is as given along with this assignment in file Dataset.png . All the tuples mentioned are separated by tabs.

Due to a glitch in the database, some fields are missing in the dataset. Keeping this detail in mind, the rental company wants you to give them answers to some of their queries. Also, they want you to retrieve results using the appropriate attribute as key (key of the <key,value> pair) and other details as either a single value or a list of values depending upon the use case. Help them resolve their queries by performing the following tasks.

The zip file Assignment 2nd NoSQL.zip contains the six folders for six queries to be written. The file also contains the dataset event_<event_no>.txt. The dataset.png image shows a sample snapshot of the contents of the dataset.

For all the queries write Mapper, Combiner and Reducer codes. Mapper may contain 1D lists for storing list of required attributes as value (use minimum number of attributes in the list to get the accurate result. Use of more than the required attributes will incur penalties). Combiner too may contain 1D lists. No other data structures are to be used in Mapper, Combiner codes. Also, datetime module may be used in Reducer.py. Note that while creating a value list, use appropriate delimiters. You can use dictionaries in reducer codes.

Example :

ABC123 [abc:def:ghi:llp]

ABC123 is the key and [abc:def:ghi:llp] is the value list with four attribute values abc, def, ghi and llp respectively separated by colon (**:**) .

Query 1 [20 marks]:

Print the IDs of the customers who have missing fields in either of start_station_name, start_station_id, end_station_name or end_station_ids and store them in result_q1.txt.

Hint : ID is the key in <key,value> pair. The attributes {start_station_name, start_station_id, end_station_name, end_station_ids} form the value list. Similarly, you have to figure out the key for solving the next queries.

Output file :

<ID 1>
<ID 2>
…

Query 2 [20 marks]:

Find the count of members of the rental company who are using different types of bikes by ordering it in lexicographical order of bike names (i.e. electric_bike, docked_bike, classic_bike..etc). Store it in result_q2.txt

Output file :

classic_bike <no of classic bike for members>
docked_bike <no of docked bike for members>
…

Query 3 [20 marks]:

Calculate the total revenue generated by renting bikes. For members, rent costs $5/hour and for casuals, rent costs $10/hour. Store it in result_q3.txt

Note if one rents a bike for 1 second after a complete hour, charge for another hour.

Example - One rents a bike for 2 hours 1 second. Charge for 3 hours.
One rents a bike for 1 minute, charge for 1 hour
One rents a bike for 1 hour 59 minutes, 30 seconds, charge for 2 hours. So on and so forth.

Output file:

Total Revenue : <Total Revenue obtained>

Query 4 [20 marks]:

Find the top 5 popular start points(station name as well as IDs) from where the members of the rental company rent their cycles.(Higher the frequency of the start point, higher the popularity). Store it in result_q4.txt

Output:

<start_station_name_1,start_station_id_1>
<start_station_name_2,start_station_id_2>
<start_station_name_3,start_station_id_3>
<start_station_name_4,start_station_id_4>
<start_station_name_5,start_station_id_5>

Query 5 [20 marks]:

Out of the casual customers, the rental company wants to convert those customers to their member customers who rented more than the average rental duration of casual customers. Find the IDs of such customers and store it in result_q5.txt.

Output :

<Casual Customer ID 1>
<Casual Customer ID 2>
<Casual Customer ID 3>
<Casual Customer ID 4>
<Casual Customer ID 5>
…

Deliverables :

5 Mapper.py, 5 Reducer.py, 5 Combiner.py, 5 Makefiles, a Readme file explaining the logistics of the assignment (execution and approach of assignment). Folder will be provided along with the dataset. Write the codes against respective queries, zip it and save it with the filename convention given in the instructions.Each Query folder will contain one Mapper.py, one Reducer.py, one Combiner.py and 1 Makefile.

Makefile/Execution Command :

(python mapper.py event_1.txt | sort | python combiner.py & python mapper.py event_2.txt | sort | python combiner.py & ... | sort | python reducer.py)

Note on Concurrent Execution and Printing Issues and Bonus :

While running the code as given above, your code will be prone to concurrency issues i.e., you code will be concatenating outputs of multiple files into one. This will result in erroneous outputs generated from the mappers and combiners (even mappers too if large data is fed) and will cause a problem in the reducer if your conditions in reducer are restricted and if error messages are not given, the desired result won't be retrieved. Try to mitigate the issue.

It is seen that running the command a few times does the trick of concurrency issues. But try to mitigate this issue by using appropriate measures. You can use time module to mitigate or other OS related methods for smooth execution of the file. But you cannot store the data in some variable and print it as a whole to avoid concurrency issues. If you are found to do so, marks will be deducted.

You can use linux commands/ other python modules related to concurrency/threading to handle your queries. Update your readme file regarding the same. [Bonus]

An example of such case is :

A.txt

1 4
102 7
3 10

B.txt

11 73
201 3
90 1

Python mapper.py A.txt & python mapper.py B.txt

1 4201 3102 7 #*

3 10
90 1
11 73

#* line 1 of A.txt, line 2 of B.txt and line 2 of B.txt are concatenated due to concurrency