ASSIGNMENT 1

Mapper T1
We load the dataset into memory, one record at a time and we eliminate records if any of the required data values contains a null, and further eliminate them if the record does not satisfy the given set of conditions. If the record satisfies all conditions, we extract its start time hour from its timestamp and then output the same from the mapper. “Start_time_hour 1”.

Reducer T1
We read the results of the mapper from stdin and sum the count of accidents that occur at each hour and output the results to stdout.” Start_time_hour count”. We use a list to implement the same.

Mapper T2
We load the dataset into memory, one record at a time and we eliminate records if any of the required data values contains a null. We extract the coordinates from each line and from the command line arguments to calculate the Euclidean Distance and compare it with the given constraint value of distance. If the record does satisfy the condition, we post a request containing the Coordinates in a JSON string to the IP Address and get the city and state name in the response and output the same. “StateName CityName 1”

Reducer T2
We read the results of the mapper “StateName CityName 1” from stdin and sum the count of accidents that occur in each city and each state and output the results to stdout in the required format. Since the input to the reducer is already sorted, we process the records incrementally, eliminating the need to store data at any point.

ASSIGNMENT 2

Mapper T1
The input to the mapper is a network of pages that is loaded into the program one line at a time. Each line is split to get the source and destination page and is printed again sorted in a lexicographic way.
“SourcePage DestinationPage”

Reducer T1
We read the results of the mapper from stdin where each line is split to get the source and the destination. If there’s no initial source we make the ‘prev_source=0’ and append the destination

in the adjacency list. Now if the source is the same as the ‘prev_source’, we append it in the same list and then finally print the source along with the adjacency list. If the source is not the same as the ‘prev_source’ we append the destination in a different list. At last, we print the last source in the adjacency list.

MAPPER T2

The mapper receives two command-line arguments, the absolute path to the v file and the absolute path to the page embedding file. The input to the mapper is the adjacency list generated from the previous task. We split each line into the source and adjacency list . Similarity and contribution are computed and we print the contribution for each destination.

REDUCER T2

We read the results of the mapper from stdin where each line is split to get the page and the contribution. It computes the new page ranks. For each page in the network, we display the page’s ID along with its updated page rank on a single line. The values are comma-separated and the newline delimited. The output is in the lexicographical order of the node. “Node PageRank”

ASSIGNMENT 3

TASK 1

The input to the file Task1.py consists of the path to the city.csv file taken from the command line argument. The city.csv file is stored as a data frame. The data frame is filtered by the country name, also provided in the command line. It is then grouped by the various cities to calculate the average (“Avg_Temp”) of the “AverageTemperature” for each city and stored in another data frame. These two data frames are joined by the “City” name and rows with “ AverageTemperature " greater than the calculated “Avg_Temp " are filtered out. The count of each such row is calculated and printed as “CityName Count ” in a sorted manner.

TASK 2

The input to the file Task2.py consists of the path to the files city.csv and global.csv which are taken from the command line argument. Both are stored as data frames, “df_city” and “df_global”. Df_city is grouped in terms of the country and the maximum average temperature of each country is found on every date. df_global is joined with the result, which is then filtered by the number of occurrences where the country’s maximum average temperature on a particular date turned out to be higher than the worldwide land average temperature on the same date. It is then grouped by date and the count of each such instance is calculated and printed as “Country Name Count” in a sorted manner.