c Python for Data Science Project

**Problem Description**

In 2012, URL shortening service Bitly partnered with the US government website USA.gov to provide a feed of anonymous data gathered from users who shorten links ending with .gov or .mil.

The text file comes in JSON format and here are some keys and their description. They are only the most important ones for this task.

|  |  |
| --- | --- |
| **@@@@key** | **description** |
| a | Denotes information about the web browser and operating system |
| tz | time zone |
| r | URL the user comes from |
| u | URL where the user headed to |
| t | Timestamp when the user starts using the website in UNIX format |
| hc | Timestamp when the user exits the website in UNIX format |
| cy | The city from which the request was initiated |
| ll | Longitude and Latitude |

**Required**

Write a script that can transform the JSON files to a DataFrame and commit each file to a separate CSV file in the target directory and consider the following:

All CSV files must have the following columns

* web\_browser: The web browser that has requested the service
* operating\_sys: the operating system that initiated this request
* from\_url: The main URL the user came from

**note**: If the retrieved URL was in a long format http://www.facebook.com/l/7AQEFzjSi/1.usa.gov/wfLQtf

make it appear in the file in a short format like this www.facebook.com

* to\_url: The same applied like `to\_url`
* city: The city from which the request was sent
* longitude: The longitude where the request was sent
* latitude: The latitude where the request was sent
* time\_zone: The time zone that the city follows
* time\_in: The time when the request started
* time\_out: The time when the request is ended

**NOTE** :

Because some instances of the file are incomplete, you may encounter some NaN values in your transformation. Make sure that the final data frames have no NaNs at all.

**In Short, your Script should do:**

* Take a three arguments from the user:
  + **-i:** The input files path
  + **-o:** The output files path
  + **-u:** optional parameter; if passed the timestamps in data will be kept in UNIX format, if not, convert it to timestamp
* Reads JSON file from a directory that the user entered.
* Extracts the data and cleans and transforms it.
* Print the number of records of the entered path.
* Check if the files have any duplicates and remove them.
* Uses the optional argument "-u" to maintain the UNIX format for the timestamp, otherwise convert it to human readable timestamp format.
* Prints a message after converting the files with the number of rows transformed and the file's path.
* Create a CSV file containing the final output in a CSV format, and save the output file in the output path provided by the user.
* Prints the total execution time.