CS2316 - HOMEWORK 04: NETFLIX ANALYST

Purpose

Data cleaning and reorganizing are important parts of working with data. Data scientists spend a significant portion of time preparing the data before attempting to analyze it. See this article.

Skills

This assignment will give you practice cleaning the data stored in two types of input files--csv files and json formatted files--both of which are common data exchange formats used by companies.

Knowledge

The types of problems you will encounter working with data sets and attempting to get them into a consistent format is similar to that of industry professionals working with real-world data sets.

Information

Please read each section and question carefully before you start coding. The goal of this homework is to showcase your knowledge of File I/O and data exchange formats. You should test out the functions first on your own computer, then when you have one or more of them working, upload the entire file (which must be named HW04.py) to GradeScope to see how well your code performs on the test cases we have created. You can submit the homework file as many times as you'd like before the deadline.

Allowed imports: `csv, json, datetime`

- Due Date: **02/26/2020 at 11:59 pm**
- This homework is graded out of **100** points however, the autograder will only show up to **90 points** when you submit your code. We are saving some test cases for the final autograder that we will use. This is to encourage you to start developing ways to test features of your code when you aren't given test cases in an autograder. We also reserve the right to change the test cases later if we need to. Therefore, the grade reflected in Gradescope does not reflect your final grade on HW04.py.
- This is an individual assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 2316, the TAs and the instructor. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

- For Help:
  - TA Helpdesk (Schedule posted on class website)
  - Email TA's or use Piazza Forums Notes
  - How to Think Like a Computer Scientist Textbook
  - CSV Module Documentation
  - JSON Module Documentation
  - Python File I/O Documentation
- Comment out or delete all your function calls. Only global variables, and comments are okay to be outside the scope of a function. When your code is run, all it should do is run without any errors.
- Do not wait until the last minute to do this assignment in case you run into problems.
- Read the entire specifications document before starting this assignment.
- Having function calls or extraneous code outside the scope of functions will result in an automatic 0.
- If your code cannot run because of an error, it is a 0%.

Background

The raw data of this assignment comes from Kaggle – Netflix Movies and TV Shows dataset. Although the dataset includes a variety of fields related to movies and TV shows found on Netflix as of 2019, you will only be required to use a subset of the following data. **DISCLAIMER:** The dataset is for educational purposes only. We do not endorse any inappropriate content that could be found within the dataset.

The data we provided you consist of 12 total fields:

- **show_id:** Unique ID for every Movie / TV Show
- **type:** Identifies if the video is a Movie or TV Show
- **title:** Title of the Movie / TV Show
- **director:** Director of the Movie
- **cast:** Actors involved in the Movie / TV Show
- **country:** Country where the Movie / TV Show was produced
- **date_added:** Date it was added to Netflix
- **release_year:** Actual release year of the Movie / TV Show
- **rating:** TV rating of the Movie / TV Show
- **duration:** Total duration – in minutes or number of seasons
- **listed_in:** Genre
- **description:** The summary description of the Movie / TV Show

You are required to work with the **show_id**, **type**, **title**, **director**, **cast**, **country**, **date_added**, **release_year**, **listed_in**, and **description** fields for this assignment. The dataset is split into two files: `netflix.txt` and `netflix.json`.
The files you should have downloaded from Canvas for this assignment are `netflix.txt`, `netflix.json`, and `bee_movie.txt`. Make sure you have placed them in the same directory as your HW04.py file.

IMPORTANT: if you open up `netflix.txt` and observe its structure, you can see that it is actually a CSV (Comma-Separated Value) file. However, it has the .txt extension rather than the .csv extension. We purposefully provided this CSV file in .txt format because files with the .csv extension often get converted to Excel spreadsheets automatically by computers, which would potentially mess up the original data. Remember that you can still use Python's csv module to read in a .txt file, as long as it has the structure of a CSV file.

Helper Function

Function name: **`date_cruncher`**
Parameters: `dirty_date` (str)
Return Type: str
Description: This is a given helper function provided for you in the HW04.py skeleton file. You may use this function to help you convert the date strings within the **date_added** column to the desired format. This function takes in a string representing the raw date string from the **date_added** column and returns a cleaned date string in the `'mm-dd-yyyy'` format.
Example:

```
>>> date_cruncher('September 8, 2018')
'09-08-2018'
```

Required Functions

Note: See the grading rubric at the end of the document for point distribution.

Function name: **`read_and_clean_csv`**
Parameters: `csv_name` (str)
Return Type: list
Description: Write a function that takes in a CSV filename as a string and cleans the corresponding CSV file and returns a list of sublists containing the data. The order of the sublists should match the order of the rows in the CSV file. Each sublist should represent a row of information within the CSV file. For the sake of this homework, only include the following columns in this exact order: **show_id**, **type**, **title**, **director**, **cast**, **country**, **date_added**, **release_year**, **listed_in**, and **description**.

In addition to the order specified above, follow the proper formatting provided for each column specified below:

Column	Expected Data Type	Other Requirements
show_id	int	
type	str	
title	str	
director	str	Set to `None` if empty
cast	str	Set to `None` if empty
country	str	Set to `'Unknown'` if empty
date_added	str	Should be in "`mm-dd-yyyy`"; set to `None` if empty
release_year	int	
listed_in	str	
description	str	

Failure to follow the above formatting will result in your code to fail in GradeScope. The returned list should not contain the header row of the CSV file. The string parameter `csv_name` will **not** include the file extension. If you encounter encoding errors when attempting to read in your CSV file, try opening the file with an encoding parameter: `open(f'{csv_name}.txt', 'r', encoding = 'utf8')`

Important: The file provided is `netflix.txt`, which is not a .csv extension but rather a .txt extension file. However, the structure remains the same and you are encouraged to use the csv module to read the .txt file. More information can be found in the **Background** section.

Test Case:

```
>>> read_and_clean_csv('netflix')
[
[80058654, 'TV Show', 'Transformers: Robots in Disguise', None,
'Will Friedle, Darren Criss, Constance Zimmer, Khary Payton,
Mitchell Whitfield, Stuart Allan, Ted McGinley, Peter Cullen',
'United States', '09-08-2018', 2016, "Kids' TV", 'When a prison
ship crash unleashes hundreds of Decepticons on Earth, Bumblebee
leads a new Autobot force to protect humankind.'],
```

```
[70299204, 'Movie', 'Kidnapping Mr. Heineken', 'Daniel
Alfredson', 'Jim Sturgess, Sam Worthington, Ryan Kwanten,
Anthony Hopkins, Mark van Eeuwen, Thomas Cocquerel, Jemima West,
David Dencik', 'Netherlands, Belgium, United Kingdom, United
States', '09-08-2017', 2015, 'Action & Adventure, Dramas,
International Movies', 'When beer magnate Alfred "Freddy"
Heineken is kidnapped in 1983, his abductors make the largest
ransom demand in history.'],
...]
```

Function name: **read_and_clean_json**
Parameters: `json_name` (str)
Return Type: list
Description: The returned list for this function will follow the exact format as `read_and_clean_csv`. However, your returned list will not have the same elements as `clean_and_read_csv`. Write a function that takes in a JSON filename as a string and cleans the corresponding JSON file and returns a list of sublists containing the data. Follow the same formatting as listed above in `read_and_clean_csv`. If you encounter encoding errors when attempting to read in your JSON file, try opening the file with an encoding parameter:

```
open(f'{json_name}.json', 'r', encoding = 'utf8')
```

Test Case:

```
>>> read_and_clean_json('netflix')
[
[372195, 'Movie', "Child's Play", 'Tom Holland', 'Catherine
Hicks, Alex Vincent, Brad Dourif, Chris Sarandon, Dinah Manoff,
Tommy Swerdlow, Jack Colvin, Neil Giuntoli, Juan Ramírez, Alan
Wilder', 'United States', '12-31-2019', 1988, 'Cult Movies,
Horror Movies', 'When a rash of murders unfolds, 6-year-old Andy
knows that his toy doll Chucky is the killer, but neither his
mom nor the cops believe him.'],
[880640, 'Movie', 'Pulp Fiction', 'Quentin Tarantino', 'John
Travolta, Samuel L. Jackson, Uma Thurman, Harvey Keitel, Tim
Roth, Amanda Plummer, Maria de Medeiros, Ving Rhames, Eric
Stoltz, Rosanna Arquette, Christopher Walken, Bruce Willis',
'United States', '01-01-2019', 1994, 'Classic Movies, Cult
Movies, Dramas', 'This stylized crime caper weaves together
stories featuring a burger-loving hit man, his philosophical
partner and a washed-up boxer.'],
...
]
```

Function name: **country_appearances**
Parameters: `cleaned_list` (list)
Return Type: dict

CS2316 - HOMEWORK 04: NETFLIX ANALYST

Description: Write a function that takes in the cleaned and formatted list from either `read_and_clean_csv` or `read_and_clean_json` and returns a dictionary with each unique country name from the **country** column mapped to the number of times it appears in the cleaned list. Note that the **country** column may contain more than one country. If the **country** column contains more than one country, you should add each country as a key to your dictionary and increment the country's count accordingly (e.g. if the **country** column contains '`United States, China`', you should increment the count of '`United States`' and '`China`'). Include `'Unknown'` as a key and the number of times it appears in the `cleaned_list` as its value.

Test Case:

```
>>> cleaned_list = read_and_clean_csv('netflix')
>>> country_appearances(cleaned_list)
{
'United States': 296, 'Netherlands': 6, 'Belgium': 7, 'United
Kingdom': 56, 'Unknown': 64, 'France': 28,
...,
'Botswana': 1, 'Uruguay': 1, 'Malaysia': 1, 'Romania': 1
}
```

Function name: **`description_appearances`**
Parameters: `letter` (str), `cleaned_list` (list)
Return Type: list
Description: Write a function that takes in a `letter` and scans through all of the **descriptions** of movies/TV shows from `cleaned_list`. You should create a list of tuples with the 0th index of the tuple being the title of the movie/TV show (str) and the 1st index of the tuple being the number of times the `letter` appears in the **description** (int). Return the **top 50 results** of the list sorted **descending** by the number of times the `letter` appears within the **description**. If 2 or more movies/TV shows have the same number of `letter` appearances, you should further sort them by **title** alphabetically **descending**. Capitalization of the `letter` parameter and the **description** does not matter.
Hint: How to sort with a secondary element can be found here.

Test Case:

```
>>> cleaned_list = read_and_clean_json('netflix')
>>> description_appearances('W', cleaned_list)
[
('The Ultimatum', 7), ('Women of Mafia 2', 5), ('Rosario
Tijeras', 5), ('Off Camera', 5), ('Lila & Eve', 5), ('Kiss the
Girls', 5),
...,
('Tales by Light', 3), ('Star Trek', 3), ('Spyder', 3),
("Slobby's World", 3), ('Shrek the Musical', 3)
]
```

Function name: **`cast_appearances`**
Parameters: `actor` (str), `cleaned_list` (list)
Return Type: list
Description: Create a function that takes in an `actor` as a string and returns a list of tuples of all movies/TV shows that actor has appeared in from `cleaned_list`. The format of each tuple should be as follows:

- **director** - the 0th element (`str`)
- **type** of video - the 1st element (`str`)
- **title** of the video - the 2nd element (`str`)

If a movie/TV show does not have a cast, exclude it from the list. Return a sorted list based on the **title** in **descending** alphabetical order. Case sensitivity should not matter**.**

Test Case:

```
>>> cleaned_list = read_and_clean_csv('netflix')
>>> cast_appearances('NiCOlas cAge', cleaned _list)
[
('Peter Ramsey, Rodney Rothman, Bob Persichetti', 'Movie',
'Spider-Man: Into the Spider-Verse'),
('Hoyt Yeatman', 'Movie', 'G-Force')
]
```

Function name: **`added_date`**
Parameters: `cleaned_list` (list), `beginning_year` (str), `ending_year` (str), `month` (str)
Return Type: list
Description: Create a function that returns a sorted list of tuples composed of the movies/TV shows within `cleaned_list`. The parameter `beginning_year` denotes the first year of **date_added** that you should begin including (inclusive), while the parameter `ending_year` denotes the final year you should include (inclusive). You should only include movies/TV shows that were added in the same month as the `month` parameter. If a movie/TV show's added date is `None`, you should exclude it from the list. The format of each tuple should be as follows:

- **date_added** - the 0th element (`str`)
- **show_id** - the 1st element (`int`)
- **title** - the 2nd element (`str`)

Your returned list should be sorted based on the **date_added** in **chronological (ascending)** order. If 2 or more movies/TV shows have the same month-date combination, you should then sort based on the **title** column in **ascending** order. You may assume `beginning_year` < `ending_year`.

Hint: How to sort with a secondary element can be found here.
Hint: How to sort dates chronologically can be found here (you may import datetime).

Test Case:

```
>>> cleaned_list = read_and_clean_json('netflix')
>>> added_date(cleaned_list, '2014', '2018', 'jaNuARy')
[
('01-24-2014', 70296733, 'Mitt'),
('01-01-2016', 70213235, 'Mighty Morphin Alien Rangers'),
('01-01-2016', 70221673, 'Power Rangers Samurai'),
('01-01-2017', 80158376, 'An American in Madras'),
('01-01-2017', 80117742, 'Hurricane Bianca'),
('01-01-2017', 80147318, 'Radiopetti'),
 ...,
('01-09-2018', 80233024, 'The House Next Door'),
('01-15-2018', 80168300, 'Unrest'),
('01-16-2018', 70114021, 'A Serious Man')
]

>>> added_date(cleaned_list, '2014', '2018', 'ocTober')
[
('10-01-2017', 70136140, 'Star Trek'),
('10-12-2018', 80178943, 'The Boss Baby: Back in Business'),
('10-16-2018', 70301553, 'Rake')
]
```

Function name: **`bee_script`**
Parameters: `file_name` (str), `output_file` (str)
Return Type: list
Description: You discover that a popular movie on Netflix is the Bee Movie. You decide to create a function that takes in `file_name` (a name representing the Bee Movie script text file, as a string) and writes to a new CSV file created by the `output_file` parameter. Parameters `file_name` and `output_file` will **NOT** include the file extensions.
The CSV file will contain the following information:

- Column 1 – **line** (`int`)
  - The corresponding line number of the script (starting at 1)
- Column 2 – **num_words** (`int`)
  - The number of words the current script line contains
- Column 3 – **contains_bee** (`str`)
  - `'YES'` if the word `'bee'` is present in the current script line, else `'NO'`. Case sensitivity does **NOT** matter
- Column 4 – **script_line** (`str`)
  - The current script line **WITHOUT** the trailing newline character

Return a list of dictionaries containing the information you wrote into the CSV file. This is what your returned list should look like...

Test Case:

```
>>> bee_script('bee_movie','bee_stats')
[
{'line': 1, 'num_words': 18, 'contains_bee': 'YES',
'script_line': 'According to all known laws of aviation, there
is no way a bee should be able to fly.'},
{'line': 2, 'num_words': 14, 'contains_bee': 'NO',
'script_line': 'Its wings are too small to get its fat little
body off the ground.'},
...,
]
```

`f'{file_name}.csv'` looks like…

```
line,num_words,contains_bee,script_line
1,18,YES,"According to all known laws of aviation, there is no way a bee
should be able to fly."
2,14,NO,Its wings are too small to get its fat little body off the ground.
3,15,YES,"The bee, of course, flies anyway because bees don't care what
humans think is impossible."
4,8,NO,"Yellow, black. Yellow, black. Yellow, black. Yellow, black."
```

Function name: **`write_to_json`**
Parameters: `file_name` (str), `combined_list` (list)
Return Type: dict
Description: Write a function that takes in `combined_list` (see test case) and writes a dictionary to a new JSON file created from the `file_name` parameter (again, **WITHOUT** file extension) with the following keys and values:

- Key i – the i^{th} unique **release_year** where i = 1, 2, … (str)
  - Value i – list of dictionaries corresponding to each movie/show:
    - Key 1 – **date_added**. If **date_added** is `None`, skip and move onto the next movie/show
      - Value 1 – date the movie/show was added (`str`)
    - Key 2 – **title**
      - Value 2 – the title of the movie/show (`str`)
    - Key 3 – **show_id**
      - Value 3 – The corresponding show id (`int`)
    - Key 4 – **director**
      - Value 4 – The corresponding director(s) (`str`)

- Key 5 – **num_countries**
  - Value 5 – Number of countries the movie/show was produced in. If **country** is 'Unknown', set the value to 0 (int)

You should sort combined_list based on the **release_year** in **descending order** first. You should also sort each list of dictionaries by **show_id** in **ascending order** before writing the dictionary to your JSON file. Do not include any movie/shows with the **dated_added** of None. Return the Python dictionary after you are finished.
Example of what your returned dictionary might look like…
Test Case:

```
>>> cleaned_csv_list = read_and_clean_csv('netflix')
>>> cleaned_json_list = read_and_clean_json('netflix')
>>> combined_list = cleaned_csv_list + cleaned_json_list
>>> write_to_json('combined_netflix', combined_list)
```

```
{
    "2020": [
        {
            "date_added": "01-03-2020",
            "title": "All the Freckles in the World",
            "show_id": 81006825,
            "director": "Yibr\u00e1n Asuad",
            "num_countries": 1
        },
        {
            "date_added": "11-13-2019",
            "title": "Maradona in Mexico",
            "show_id": 81034946,
            "director": None,
            "num_countries": 3
        },
        {
            "date_added": "01-01-2020",
            "title": "Ghost Stories",
            "show_id": 81088083,
            "director": "Anurag Kashyap, Dibakar Banerjee,
           Karan Johar, Zoya Akhtar",
            "num_countries": 1
        }
    ],
    "2019": [
        {
            "date_added": "09-19-2019",
            "title": "The Blacklist",
            "show_id": 70281312,
```

```
                "director": None,
                "num_countries": 1
            },
              ...
              ],
        ... ,
        "1942": [
            {
                "date_added": "03-31-2017",
                "title": "Prelude to War",
                "show_id": 60027945,
                "director": "Frank Capra",
                "num_countries": 1
            }
        ]
    }
```

`f'{file_name}.json'` looks like...

```
{
    "2020": [
        {
            "date_added": "01-03-2020",
            "title": "All the Freckles in the World",
            "show_id": 81006825,
            "director": "Yibr\u00e1n Asuad",
            "num_countries": 1
        },
        {
            "date_added": "11-13-2019",
            "title": "Maradona in Mexico",
            "show_id": 81034946,
            "director": null,
            "num_countries": 3
        },
        {
            "date_added": "01-01-2020",
            "title": "Ghost Stories",
            "show_id": 81088083,
            "director": "Anurag Kashyap, Dibakar Banerjee, Karan Johar, Zoya Akhtar",
            "num_countries": 1
        }
    ],
    "2019": [
        {
            "date_added": "09-19-2019",
            "title": "The Blacklist",
            "show_id": 70281312,
            "director": null,
            "num_countries": 1
        },
```

Gradescope Requirements

- NO PRINT STATEMENTS this will break the autograder
- NO FUNCTION CALLS outside of function definitions this will also break the autograder
- Make sure the file submitted is titled **HW04.py**
- Do not import any modules, packages, or libraries other than **csv**, **json,** and **datetime**
- Only submit the **HW04.py** file to Gradescope

Point Distribution

```
read_and_clean_csv:                      10 pts
read_and_clean_json:                     10 pts
country_appearances:                     10 pts
description_appearances:                 10 pts
cast_appearances:                        10 pts
added_date:                              15 pts
bee_script:                              15 pts
write_to_json:                           20 pts
_____________________________________________
Total                                100/100 pts
```

Note that the autograder on GradeScope does not include test cases for `cast_appearances`. Therefore, you are able to get up to 90 points on GradeScope before the assignment is due. We are saving those test cases for later use. You are encouraged to write your own test cases for that function.