Data Cleansing and Integration

Introduction

Nowadays there are many job hunting websites including seek.com, Azuna.com, etc. These job hunting sites all manage a job search system, where job hunters could search for relevant jobs based on keywords, salary, and categories, etc.

Job advertisement data analysis is becoming increasingly important and beneficial for job hunting sites, as they can be used to make improvements in the experience of users searching for jobs.

For this assessment, you are required to write Python (version 3.7) code to clean and integrate job advertisement datasets from different sources. There are two major tasks in this assessment, which has to be completed in order.

- In Task 1, you will need to find and fix problems in a given job advertisement dataset.
- Then in Task 2, you will integrate the cleaned dataset (the output from Task 1) and the 2nd dataset with different formatting.

Note that: Please strictly follow the instructions described below. Failure to do that will result in mark deduction.

Task 1. Auditing and Cleansing the Job dataset (70%)

In this task, you are given a job advertisement dataset **dataset1_with_error.csv**. You are required to inspect and audit this dataset to identify data problems and to fix those problems. The description of each column and its required format in the output cleaned dataset are shown in Table 1. Different generic and major data problems could be found in the data might include:

- Typos and spelling mistakes
- Irregularities, e.g., abnormal data values and data formats
- Violations of the Integrity constraint.
- Outliers
- Duplications
- Missing values
- Inconsistency, e.g., inhomogeneity in values and types in representing the same data

Hint: You might need to use non-graphical (e.g., statistics) and graphical (e.g., different plots) methods to explore the data in order to identify those problems.

After cleaning the dataset, the format of each column should be as follows:

Table 1. Columns, Descriptions and the Required Format of the Cleaned Job Dataset

COLUMN	[FORMAT] AND DESCRIPTION
Id	[Integer] 8 digit Id of the job advertisement
Title	[String] Title of the advertised job position.
Location	[String] Location of the advertised job position
Company	[String] Company (employer) of the advertised job position
ContractType	[String] The contract type of the advertised job position, it could be full_time, part_time or non_specified.
ContractTime	[String] The contract time of the advertised job position, it could be permanent, contract or non-specified.
Category	[String] The Category of the advertised job position, e.g., IT jobs, Engineering Jobs, etc.
Salary	[Float] Annual Salary of the advertised job position, e.g., 80000
OpenDate	[String] The opening time for applying for the advertised job position, e.g., 20120104T150000, means 3pm, 4th January 2012.
CloseDate	[String] The closing time for applying for the advertised job position, e.g., 20120104T150000, means 3pm, 4th January 2012.
SourceName	[String] The website where the job position is advertised.

After cleansing the dataset, you should output the clean dataset as **dataset1_solution.csv**

All Python code related to this task should be written in the jupyter notebook **task1.ipynb**.

Task 2. Integrating the Job datasets (30%)

In this task, you will be giving a 2nd job advertisement dataset **dataset2.csv**. You will then integrate this dataset with the output from Task 1 (**dataset1_solution.csv**).

To complete this task successfully, you are required to do the following:

1. **Resolving schema conflicts and merging data:** Inspect and compare the schema of **dataset1_solution.csv** and **dataset2.csv** to identify and resolve any schema conflicts. You will need to write Python code to
    a. Resolve any schema conflicts. You will need to **adopt** the schema in **dataset1_solution.csv** (refer to **Table 1**) as your global schema as much as you could (please **DO NOT** change the attribute names).
    b. Implement the semantic mapping and integrate the two data sets **dataset1_solution.csv** and **dataset2.csv** to produce one unified table.
2. **Resolving data conflicts:** Inspect tuples and instances for data conflicts in the unified table. In this step, you are required to do the following:
    a. Use Pandas libraries to detect and resolve duplications in the unified table.
    b. Identify a proper global key for the integrated job data and explain your chosen key in the notebook.
3. Finally, you should output the integrated dataset as **dataset_integrated.csv**

Note that all Python code related to Task 2 should be written in **task2.ipynb**.

Note also that you could assume data in **dataset2.csv** are clean, i.e., you don't need to clean data in this dataset.

Input and Output from the Tasks

The following is the summary of the input, output for the different tasks :

Task	**Input**	**Output**	**Jupyter notebook**
Task 1	dataset1_with_error.csv	dataset1_solution.csv	task1.ipynb
Task 2	dataset2.csv dataset1_solution.csv	dataset_integrated.csv	task2.ipynb

Documentation

Precise and concise comments/documents of your code are essential as part of assessment criteria. Do Not include things that are irrelevant in your code as that will reduce your code readability.

In addition, you need to submit a **final report** which covers the following points (but not limited to):

1. summarizing all generic and major data problems
2. showing the ways to fix those problems
3. listing any assumptions made when detecting or fixing the problems

Marking Criteria

- *Mechanical pass:* Your outputs will be compared against the expected output. Therefore, marking will be based on the similarity between what we expect (as discussed in the instructions) and what we receive from you. It is extremely important to carefully follow the instructions to produce the expected output. Otherwise, you may easily lose many points for simple mistakes (eg typos in the format of the files, not loading essential libraries, different file names/path, etc)
- *Expert pass:* Your code will be checked by an expert to validate the logic and flow, proper use of libraries and functions, and clarity of codes, comments, structure and presentation.
- The expert will **NOT** fix your code’s problem even if it is a simple typo in a file name or an imported library.

How to Submit

Once you have completed your work, take the following steps to submit it:

1. Rename “rename_me” directory to your student number. This directory should contain your Notebook(s), inputs(s), output(s), final report.
2. Before submission, you should restart your kernel and rerun your code from beginning to the end to make sure everything works as expected.
3. Save your notebook as a .ipynb file once again while you are keeping all the outputs (eg print outputs) in the notebook.
4. Make sure the output files are properly named according to instruction and your code does not produce any unnecessary file.
5. Zip the containing folder with the same name (ie <student_number>.zip) and upload for submission.

Good luck!