COMP9321 Data Services Engineering

Assignment 1: Data Ingestion, Manipulation, and Visualisation (15 marks)

Changelog

Date	Version	Description
28/02/2024	v1	Initial draft
01/03/2024	v2	Revised Windows instructions to use `python` / `pip` instead of `python3` / `pip3`. Clarified that CSV files should be saved after pre-processing the DataFrame.

Introduction

As budding data scientists, we're no doubt interested in the job market that awaits us. In this assignment we'll be building a data pipeline to help inform our future decisions about entering the workforce.

Our primary dataset is a CSV file containing data science jobs from the past few years. It's been scraped from ai-jobs.net and, to some degree, pre-processed.

We'll also be using a number of secondary datasets, which we'll need to scrape ourselves. These will help inform our analysis:

Source	CSE Mirror
Cost of Living Index by Country 2023	[CSE Mirror]
Foreign currency exchange rates for the year ending 31 December 2023	[CSE Mirror]
Two-letter country codes	[CSE Mirror]

Important Note: We will **not** be scraping data from the source, but rather from the **CSE Mirror** pages. This is to prevent placing undue burden on third-party servers, and to ensure the pages are static throughout the duration of the assignment.

You're encouraged to review these links and explore these datasets prior to attempting the assignment.

Please note that this is publicly available data, and we are not responsible for political correctness, or other data inaccuracies, as this is purely a learning exercise for data services engineering.

Accompanying each question below you'll find:

- **Output**: This includes the expected shape and column names of the DataFrame you create.
- **Marking Considerations**: These, along with the expected output, are provided to help you stay on track. It is not a marking rubric, but merely some points for you to be mindful of while completing each question.

A code template is provided. In completing this assignment:

- You **must** read and follow these instructions carefully.
- You **must** use the code template.
- You **must** rename the code template with your zID.
- You **must not** modify the code template, except where indicated. For example, you **must not**:
  - Import additional third-party libraries.
  - Modify the function signature for each `question_X` function.
  - Modify the `main` function.
  - Modify the `log` function.
  - Disable or modify the calls to the `log` and `plt.savefig` functions within each `question_X` function.
  - Add any global variables.
- You **must** setup a virtual environment as per the instructions below.
- You **must** use either Python 3.11 (which is installed on CSE) or Python 3.12 (which is the current release).
- You **must** use pandas features to solve each question.
- You **must not** iterate over the rows of any DataFrame.
- You **must not** convert any DataFrame to a native data type (e.g. a `list` or `dict`) in order to process the data.
- You **must not** hard-code any file paths or URL's within the code you write.
  - These are specified in the `main` function, which you **must not** modify, and are passed as parameters to the `question_X` functions that you will complete. You **must** use these local variables instead of hard-coded values.
- You **must not** display any plots (e.g. using `plt.show`).
  - The code template is configured to save your plot to disk.
- You **must not** manually edit any datasets.
  - During marking a clean copy of the CSV file will be used, and web data will be freshly scraped.
- You **may** import and use any of the Python 3.11 or Python 3.12 standard libraries.
- You **may** write helper functions and include them where indicated in the code template.
- You **may** iterate over `DataFrame.columns`, if you feel it necessary.

For your reference, the third-party libraries you may use, and their versions, are listed below. Of these, the only library you are required to use is pandas. In setting up your virtual environment, these packages, along with their dependencies, will be installed automatically.

Package	Version
matplotlib	3.8.2
numpy	1.26.0

Package	Version
pandas	2.2.0
thefuzz	0.22.1

Part 0: Setting up your Virtual Environment

We'll be setting up a virtual environment in which to work on the assignment. This is not only good practice, but **critical** to ensure your code can be run during marking.

For this, we'll be using Python's built-in venv tool to create and activate the environment, and then pip to install the permitted libraries within the virtual environment.

Instructions below are based on a typical macOS/Linux or Windows environment. If your environment differs, please see the documentation for venv and pip.

1. Decide on a directory in which you'd like to work on the assignment, for example:
   - macOS: `~/9321/ass1`
   - Windows: `C:\Users\username\9321\ass1`
2. Create the virtual environment using:

   ```
   python3 -m venv /path/to/my/assignment                    # macOS
   python  -m venv C:\path\to\my\assignment                  # Windows
   ```

   For the example directory it would be:

   ```
   python3 -m venv ~/9321/ass1                               # macOS
   python  -m venv C:\Users\username\9321\ass1               # Windows
   ```

   Running this command creates the target directory, if it doesn't already exist, along with any necessary parent directories.
3. Activate the virtual environment using:

   ```
   source /path/to/my/assignment/bin/activate                # macOS
   C:\path\to\my\assignment\Scripts\activate.bat             # Windows
   ```

   For the example directory it would be:

   ```
   source ~/9321/ass1/bin/activate                           # macOS
   C:\Users\username\9321\ass1\Scripts\activate.bat           # Windows
   ```
4. Download the provided requirements.txt file to your assignment directory (you may need to right-click → save as...), and then install the listed packages within the virtual environment:

```
pip3 install -r /path/to/my/assignment/requirements.txt         # macOS
pip  install -r C:\path\to\my\assignment\requirements.txt       # Windows
```

For the example directory it would be:

```
pip3 install -r ~/9321/ass1/requirements.txt                    # macOS
pip  install -r C:\Users\username\9321\ass1\requirements.txt    # Windows
```

5. You're now ready to begin working on the assignment. For example, you could open the assignment directory as a workspace in Visual Studio Code:

   ```
   code /path/to/my/assignment                                  # macOS
   code C:\path\to\my\assignment                                # Windows
   ```

   For the example directory it would be:

   ```
   code ~/9321/ass1                                             # macOS
   code C:\Users\username\9321\ass1                             # Windows
   ```

   Tip: if you are using VS Code, make sure you have the Python extension installed. You may also like to review the documentation on Python environments in VS Code.

6. Once you've finished working on the assignment, you can deactivate the virtual environment by simply executing:

   ```
   deactivate
   ```

Important Note: You will need to repeat steps 3 and 6, to activate and deactivate the virtual environment, each time you work on the assignment.

Part 1: Data Ingestion and Cleaning (3.5 marks)

Question 1 (0.5 marks)

First things first, we need to load our primary dataset.

- Download the code template to your assignment directory.
  - Rename the file according to your zID.
- Download the data science jobs CSV file to the same directory as your Python script.
- Read it into a DataFrame.

Hint: You may need to right-click → save as... to download each of the files.

Output

Shape:

```
(3755, 11)
```

Columns:

```
['work_year', 'experience_level', 'employment_type', 'job_title',
 'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
 'remote_ratio', 'company_location', 'company_size']
```

Marking Considerations

- The Python script has been correctly renamed.
- The CSV file is not hard coded within the function, but rather uses the argument passed to the function.
- The CSV file is read correctly into a DataFrame and returned by the function.

Question 2 (1 mark)

In assessing the job market, it's great to know the salaries that are out there, but for it to be meaningful, we also need to consider the cost of living in each market.

- Scrape the table from the cost of living page into a DataFrame.
- Convert all the column names to lower case and replace spaces with underscores.
- Because we're ethically minded data scientists, once you've completed the above pre-processing, save the DataFrame as a CSV file (without the index), so that on future executions of your script, instead of scraping the data it will be read from disk.

Hint: use pandas.read_html().

Hint: don't write the code to save the CSV file until you're confident that you're scraping the website correctly, otherwise you'll need to delete the file before each execution.

Note: you may find the `rank` column is empty. This is fine. We can always calculate it ourselves, if we need to.

Output

Shape:

```
(140, 8)
```

Columns:

```
['rank', 'country', 'cost_of_living_index', 'rent_index',
 'cost_of_living_plus_rent_index', 'groceries_index',
 'restaurant_price_index', 'local_purchasing_power_index']
```

Marking Considerations

- The CSV file and URL are not hard coded within the function, but rather the arguments passed to the function are used.
- The website is only scraped if the CSV file doesn't exist, and:
  - The column names are correctly sanitised.
  - The CSV file is correctly saved.
- The CSV file is read if it does exist.
- The dataset is loaded correctly into a DataFrame and returned by the function.

Question 3 (1 mark)

You'll notice the jobs dataset includes the salary for each job in its local currency as well as its US dollar equivalent. For our purposes, it would be better to have the salaries in Australian dollars. We'll be focussing mainly on the most recent job listings, in 2023, and fortunately the Australian Taxation Office provides average currency exchange rates for each calendar year.

- Scrape the table from the currency exchange rates page into a DataFrame.
- You'll see this table actually has two header rows:
  - Remove all columns under `Nearest actual exchange rate` along with the headers.
  - Remove the top level header row.
- The remaining header row and the data contain some non-breaking spaces, which can be problematic. Replace all such spaces with regular spaces.
- Remove the `30 Jun 23` column.
- Rename the `31 Dec 23` column as `rate`.
- Convert all column names to lower case.
- Because we're ethically minded data scientists, once you've completed the above pre-processing, save the DataFrame as a CSV file (without the index), so that on future executions of your script, instead of scraping the data it will be read from disk.

Hint: a non-breaking space can be specified in Python as unicode `\u00A0` or hex code `\xa0`.

Output

Shape:

```
(18, 3)
```

Columns:

```
['country', 'rate', 'currency']
```

Marking Considerations

- The CSV file and URL are not hard coded within the function, but rather the arguments passed to the function are used.
- The website is only scraped if the CSV file doesn't exist, and:
  - Irrelevant columns are discarded.
  - The column names are correctly sanitised and renamed.

    - The dataset is correctly sanitised.
    - The CSV file is correctly saved.
- The CSV file is read if it does exist.
- The dataset is loaded correctly into a DataFrame and returned by the function.

Question 4 (1 mark)

You'll also notice that the jobs dataset includes the country of employment as a two-letter country code. We'd like to translate these to actual country names. This will be useful when it comes time to communicate our analysis (e.g. through visualisations), but we'll also use it later on to help connect each job with its cost of living.

- Scrape the "Officially assigned code elements" table from the two-letter country codes page into a DataFrame.
- Remove the columns `Year`, `ccTLD`, and `Notes`.
- Rename the columms `Country name (using title case)` to `country` and `Code` to `code`.
- Because we're ethically minded data scientists, once you've completed the above pre-processing, save the DataFrame as a CSV file (without the index), so that on future executions of your script, instead of scraping the data it will be read from disk.

Output

Shape:

```
(249, 2)
```

Columns:

```
['code', 'country']
```

Marking Considerations

- The CSV file and URL are not hard coded within the function, but rather the arguments passed to the function are used.
- The website is only scraped if the CSV file doesn't exist, and:
    - Irrelevant columns are discarded.
    - Columns are correctly renamed.
- The CSV file is read if it does exist.
- The dataset is loaded correctly into a DataFrame and returned by the function.

Part 2: Data Exploration (1 mark)

Question 5 (1 mark)

Before we move any further with our data manipulation and analysis, we'd like to profile some aspects of our jobs dataset. Using the jobs DataFrame from question 1, produce a summary DataFrame such that:

- Each column from the jobs DataFrame is a row index of the new DataFrame, e.g. `work_year`, `experience_level`, etc., should each be a row index.
- The new DataFrame contains 3 columns, `observations`, `distinct`, and `missing`, such that:
  - `observations` is the number of cells in the corresponding column from the jobs DataFrame that contain values (i.e. that are not missing values).
  - `distinct` is the number of distinct values (not including any missing values) from the corresponding column in the jobs DataFrame.
  - `missing` is the number of missing values from the corresponding column in the jobs DataFrame.

Hint: for clarity, `observations` + `missing` should always equal the number of rows in the jobs DataFrame.

Output

Shape:

```
(11, 3)
```

Columns:

```
['observations', 'distinct', 'missing']
```

Marking Considerations

- Summary DataFrame is correctly produced.

Part 3: Data Manipulation (6.5 marks)

Question 6 (0.5 marks)

The jobs data set includes a categorical field to describe the experience level of each position. Unfortunately, because it's text encoded, it doesn't capture the implicit ordering of each experience level, so we'd like to correct that:

- Add an `experience_rating` column to the jobs DataFrame from question 1, based on the `experience_level` column, such that:
  - Entry-level `EN` → `1`
  - Mid-level `MI` → `2`
  - Senior-level `SE` → `3`
  - Executive-level `EX` → `4`

Output

Shape:

```
(3755, 12)
```

Columns:

```
['work_year', 'experience_level', 'employment_type', 'job_title',
 'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
 'remote_ratio', 'company_location', 'company_size',
 'experience_rating']
```

Marking Considerations

- New column is correctly added.

Question 7 (1 mark)

We'd now like to translate the two-letter country code in our jobs dataset into an actual country name.

- Add a `country` column to the jobs DataFrame from question 6 by linking the `employee_residence` with the `code` from the country codes DataFrame from question 4.

Output

Shape:

```
(3755, 13)
```

Columns:

```
['work_year', 'experience_level', 'employment_type', 'job_title',
 'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
 'remote_ratio', 'company_location', 'company_size', 'experience_rating',
 'country']
```

Marking Considerations

- New column is correctly added.

Question 8 (1 mark)

Focussing on recent job listings, we'd also like to translate each position's salary into our local currency, the Australian dollar. This will help ensure our analysis has the appropriate context.

- Filter the jobs DataFrame from question 7 to only consider the `work_year` 2023, and add an integer-type `salary_in_aud` column by converting `salary_in_usd` using the currency conversion rates DataFrame from question 3.
    - The appropriate rate must be found programmatically, rather than hard-coded, but you can reference "United States" or "United States dollar" in order to find the rate.

Output

Shape:

```
(1785, 14)
```

Columns:

```
['work_year', 'experience_level', 'employment_type', 'job_title',
 'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
 'remote_ratio', 'company_location', 'company_size', 'experience_rating',
 'country', 'salary_in_aud']
```

Marking Considerations

- Rows correctly filtered.
- New column is correctly added.
- New column is correctly calculated.
- New column has the correct type.
- Currency conversion rate isn't hard-coded.

Question 9 (1 mark)

Unfortunately our cost of living indices also aren't in the appropriate context for our analysis. They're currently relative to the cost of living in New York City, so we'd like to re-scale these to be relative to Australia.

- Remove all columns other than `country` and `cost_of_living_plus_rent_index`.
- Re-calculate the `cost_of_living_plus_rent_index` for each country by dividing by the index for Australia, and multiplying by 100.
  - The index for "Australia" must be found programmatically rather than hard-coded.
- Round the calculated values to 1 decimal place.
- Sort the DataFrame by increasing `cost_of_living_plus_rent_index`.

Hint: if the index for Australia is `80`, and the index for New Zealand is `120`, then the re-scaled index for New Zealand would be `120 / 80 * 100`.

Output

Shape:

```
(140, 2)
```

Columns:

```
['country', 'cost_of_living_plus_rent_index']
```

Marking Considerations

- Irrelevant columns removed.

- Cost of living in Australia found programmatically.
- Cost of living correctly re-calculated.
- Cost of living correctly rounded.
- DataFrame correctly sorted.

Question 10 (2 marks)

Now we'd like to incorporate our cost of living data with our jobs data.

- Add a `cost_of_living` column to the jobs DataFrame from question 8 by linking it with the cost of living DataFrame from question 9.
  - This will require a fuzzy match between the `country` column of each DataFrame, for which we'll use a library called thefuzz.
  - The library will use the Levenshtein distance to determine a similarity, out of 100, between two strings. Set a match threshold of 90.
- Remove any rows that don't yield a match.

Hint: See this (slightly out of date) code example on stackoverflow.

Output

Shape:

```
(1783, 15)
```

Columns:

```
['work_year', 'experience_level', 'employment_type', 'job_title',
 'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
 'remote_ratio', 'company_location', 'company_size', 'experience_rating',
 'country', 'salary_in_aud', 'cost_of_living']
```

Marking Considerations

- Rows correctly matched.
- Umatched rows correctly removed.
- Column correctly renamed.

Question 11 (1 mark)

As a first step in our analysis, we'd like to look at the average salary in each country according to the experience level of the jobs.

- Using the jobs DataFrame from question 10, create a pivot table of the average `salary_in_aud` by `country` and `experience_rating`.
  - `country` should be the index.
  - Each `experience_rating` should be a columns, in ascending order.
  - Any missing values should be set to 0, and all values should be converted to integers.

- Sort the table by decreasing average `salary_in_aud`, first by `experience_rating` 1, then 2, then 3, then 4.

Output

Shape:

```
(39, 4)
```

Columns:

```
[('salary_in_aud', 1), ('salary_in_aud', 2), ('salary_in_aud', 3),
 ('salary_in_aud', 4)]
```

Marking Considerations

- Table correctly indexed.
- Correct columns, in the correct order.
- Values correctly calculated.
- Missing values correctly filled.
- Data types correctly converted.
- Table correctly sorted.

Part 4: Data Visualisation (4 marks)

Question 12 (4 marks)

As a final step, we'd like to develop a visual representation to better understand, interpret, and gain important insights from the data.

Using the jobs DataFrame from question 10, create a visualisation to help inform the best markets in which to apply for jobs. Minimally, it must consider the `country`, `salary_in_aud`, and `cost_of_living` data. However, you are encouraged to explore the dataset, and you are free to incorporate additional metrics, if you believe them to be informative.

You are also free to transform the data as you see fit, for example through aggregation, filtering, outlier removal, or normalisation.

Should you choose to restrict the dataset to a subset of countries, then please note that a minimum of 5 countries, including Australia, must be represented.

You may also use subplots, if you deem them appropriate.

Note: The code you submit must not display your visualisation (e.g. using `plt.show()`). The code template is setup to save your visualisation. Do NOT modify this code.

Output

Shape:

```
N/A
```

Columns:

```
N/A
```

Marking Considerations

- Suitable choice of visualisation.
- Appropriate use of scale.
- Appropriate inclusion of title, labels, legend, with suitable sizing.
- Appropriate use of colour.
- Visualisation is self-explanatory and informative.
- Image is correctly saved to disk, rather than shown.

```
$ echo "Congratulations, you've reached the end of the assignment!"
```