Assignment 1: Data Cleaning and Summarising

Ridge Tagala (s3934367)

Data Preparation (Section 1)

This highlights the first step in the data science process, which is preparing the data. This consists of getting rid of any unwanted or incorrect data, such as data which may contain typos, inconsistencies or incorrect data. Section 1 of this report highlights how I approached this assignment, further leading to how I cleansed the data as part of the first step in the data science process.

While I had a couple errors along the way, not many in task 1, but mostly in task 2 when starting task 2.1. I had multiple errors – one of which was a key error.

Furthering more research, a key error was when two spreadsheets being assigned and compared to each other had a different number of columns, eg. Spreadsheet-1 = spreadsheet-new2, a key error would mean that spreadsheet 1 has less or more columns of data than spreadsheet-new2, thus giving a key error. I solved this by googling and asking my practical teacher, to which I found the solution: creating a dataframe. A dataframe is a data structure which can be used for arithmetic operations, and poses as a two-dimensional dictionary (python) or array (pandas).

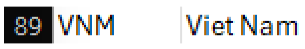
To solve most of the errors, id attend my practical session on Thursday and have a discussion with my practical teacher, also sending him emails of questions when id get stuck. (practical teacher = Chengyao Xie). For the other times, id use the official pandas documentation, stackOverFlow, Tutorialspoint and geeksforgeeks (see references)

Data was given to us in a excel spreadsheet, (.csv) file. Opening the file, I had a look at the columns specified in the document; ISO3, Countries, Region, Sub-Region, Income Group, Total, Residence (Rural), Residence (Urban), Wealth quintile (Poorest), Wealth quintile (Richest) and Time period.

From observing the data located in the spreadsheet, I have noticed the following errors:

Error Type 1: Typos and repeated values in 'Countries' column

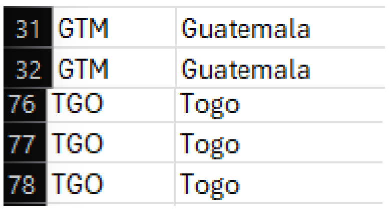
Looking through the 'Countries' column, we notice that on line 89 Veitnam is spelt incorrectly, with a whitespace inbetween.



89	VNM	Viet Nam

Figure 1: Vietnam spelt incorrectly with a whitespace in-between.

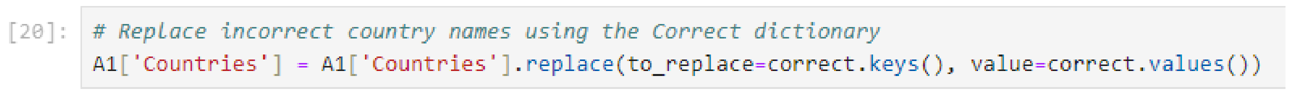
Upon a further inspection of the column, we can see that there are a couple countries that have been repeated multiple times along the column. Those countries being Tongo (line 76) and Guatemala (line 31)



31	GTM	Guatemala
32	GTM	Guatemala
76	TGO	Togo
77	TGO	Togo
78	TGO	Togo

Figure 2 and 3: Repeated countries Guatemala and Tongo located in the data

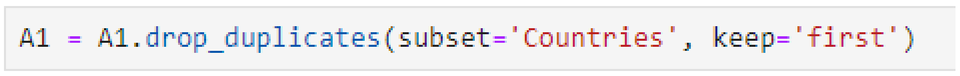
To fix these errors, I used Python, along with the pandas library. Firstly, a list of all the countries spelt correctly was made, then a list of the countries with typos was made. These were made to be compared to each other and would be corrected later in the code. They were then combined into a list named correct, where syntax was correctSpelling : incorrectSpelling, then python was used to check if the country was spelt incorrectly by comparing the string to the list, then if it was spelt incorrectly, it would be edited and overwritten with the country from the list correct.



```
[20]: # Replace incorrect country names using the Correct dictionary
      A1['Countries'] = A1['Countries'].replace(to_replace=correct.keys(), value=correct.values())
```

Figure 4: code which replaces the incorrect countries with the correct spelling in the list created, as mentioned above

A similar approach was taken to correct the repeated values in the countries column, using the same list, but adding a validation check to remove any repeated values that isn't the first value.



```
A1 = A1.drop_duplicates(subset='Countries', keep='first')
```

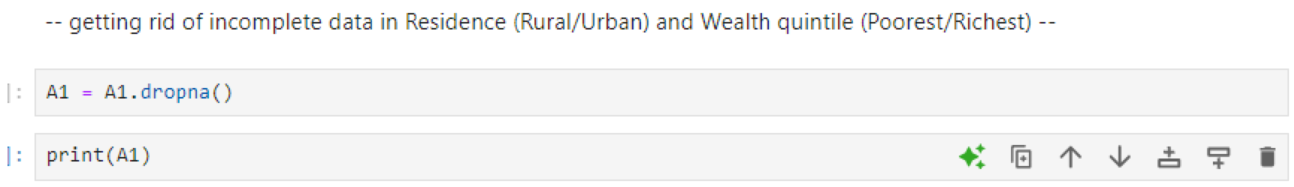
Figure 5: code to drop duplicates but keeps the first copy

Implementation of the following code ensures that there are no more typos or repeated values present in the 'Countries' column, making it ready for stage 2 of the data science process: Data exploration.

Error Type 2: Incomplete/empty data in Residence (Rural/Urban) and Total columns

In the given data set, we have multiple rows with empty data (NaN – Not a number) present, with those countries being Argentina (line 4), Egypt (line 27), Mexico (line 49), Niger (line 56), South Africa (line 68), Ukraine (line 85) and United Kingdom (line 86).

To fix this, I used python with the pandas library, using the function dropna() to drop the columns that are empty.



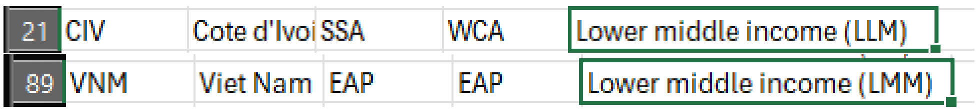
-- getting rid of incomplete data in Residence (Rural/Urban) and Wealth quintile (Poorest/Richest) --

```
[ ]: A1 = A1.dropna()

[ ]: print(A1)
```

The function dropna() deletes all the columns which contain empty or NaN valules. Since the columns with countries specified above contain empty columns, meaning that they are incomplete data, the function automatically deletes them.
Deleting the columns with empty values better cleans and prepares the data, ensuring that the data is ready for exploration.

Error Type 3: Typo in 'Income Group' column

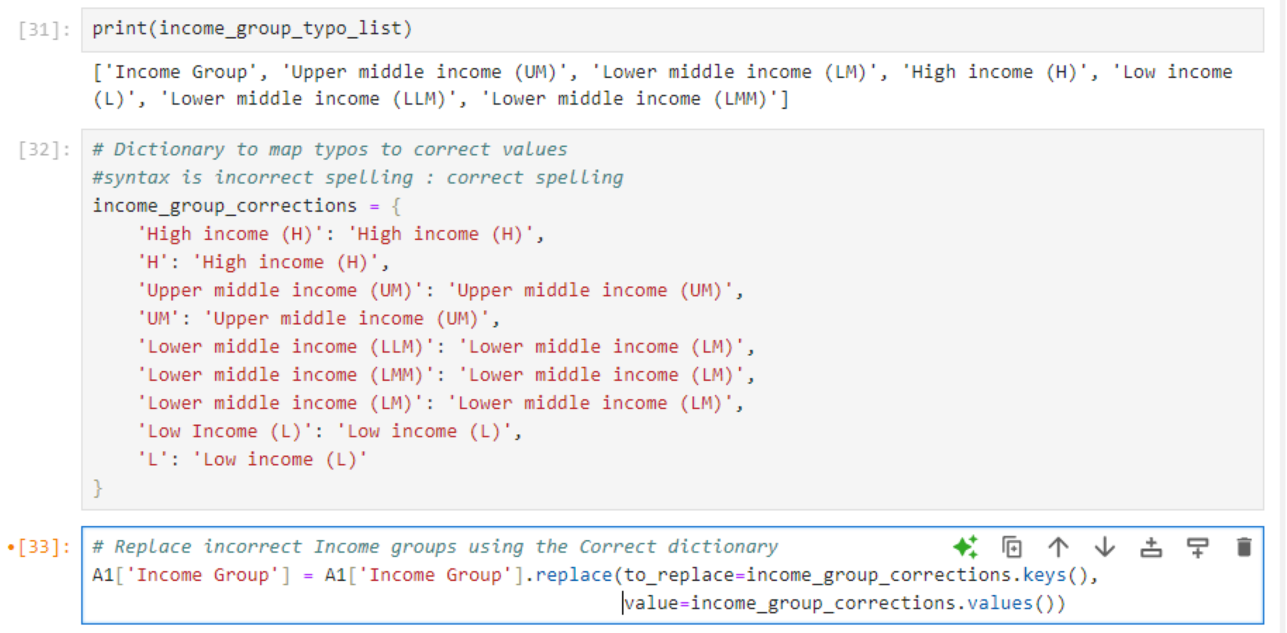
The given data set has incorrect abbreviations for the Lower Middle Income group, abbreviated as (LM), but looking through the data we see (LLM) and (LMM) (as seen in the screenshots below).



21	CIV	Cote d'Ivoi	SSA	WCA	Lower middle income (LLM)
89	VNM	Viet Nam	EAP	EAP	Lower middle income (LMM)

Using python and the Pandas library, I fixed this error by creating a list similar to how we fixed the typos in the country column. A list containing the (incorrectSpelling) : (correctSpelling), then I used the replace() method in python to replace the said typo. (see image below).

This successfully replaced the incorrect abbreviated income groups in the column, minimising the potential errors in the data.

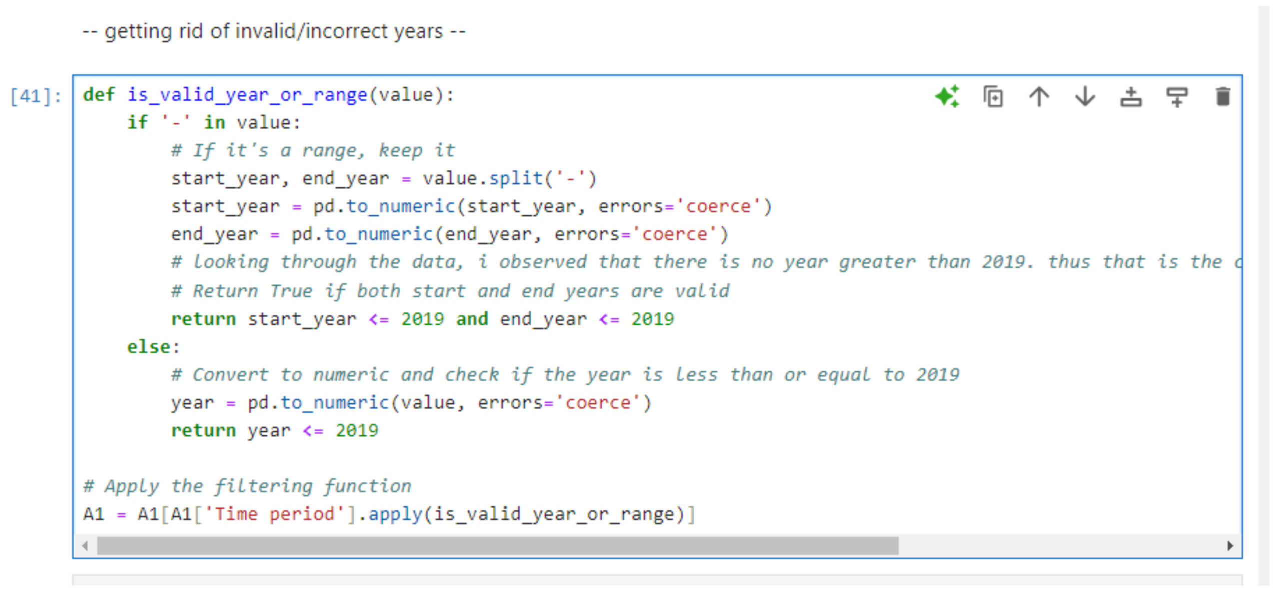


```
[31]: print(income_group_typo_list)

      ['Income Group', 'Upper middle income (UM)', 'Lower middle income (LM)', 'High income (H)', 'Low income
      (L)', 'Lower middle income (LLM)', 'Lower middle income (LMM)']

[32]: # Dictionary to map typos to correct values
      #syntax is incorrect spelling : correct spelling
      income_group_corrections = {
          'High income (H)': 'High income (H)',
          'H': 'High income (H)',
          'Upper middle income (UM)': 'Upper middle income (UM)',
          'UM': 'Upper middle income (UM)',
          'Lower middle income (LLM)': 'Lower middle income (LM)',
          'Lower middle income (LMM)': 'Lower middle income (LM)',
          'Lower middle income (LM)': 'Lower middle income (LM)',
          'Low Income (L)': 'Low income (L)',
          'L': 'Low income (L)'
      }

•[33]: # Replace incorrect Income groups using the Correct dictionary
      A1['Income Group'] = A1['Income Group'].replace(to_replace=income_group_corrections.keys(),
                                                      value=income_group_corrections.values())
```

Error Type 4: Invalid Years in 'Time period' column

The data has incorrect years which jeopardise the validity of the data. From reading the data, I observed that the time frame is between 2010-2019, thus there is no year or time frame smaller than or bigger than the two values (none smaller than 2010, none bigger than 2019). However, looking through the data, on line 74, we have a year "3562", which is a massive outlier due to the above observation. Also, on line 87 we have the date range 2012-2099, which is also invalid as data from the future cannot be collected.

To fix this and make the time frames valid, I wrote a python script which consists of a function (def is_valid_year_or_range), where it checks if the year is greater than 2019, or less than 2010. If the year fulfils one of the two, or both statements the row is removed as it is deemed invalid and compromises the data validity.



-- getting rid of invalid/incorrect years --

```
[41]: def is_valid_year_or_range(value):
          if '-' in value:
              # If it's a range, keep it
              start_year, end_year = value.split('-')
              start_year = pd.to_numeric(start_year, errors='coerce')
              end_year = pd.to_numeric(end_year, errors='coerce')
              # looking through the data, i observed that there is no year greater than 2019. thus that is the
              # Return True if both start and end years are valid
              return start_year <= 2019 and end_year <= 2019
          else:
              # Convert to numeric and check if the year is less than or equal to 2019
              year = pd.to_numeric(value, errors='coerce')
              return year <= 2019

      # Apply the filtering function
      A1 = A1[A1['Time period'].apply(is_valid_year_or_range)]
```

Data Exploration

Task 2.1

This task focuses on the total percentage of children in a school attendance age which have internet connection at home, filtered by Region.

The task asked to plot a box plot separated by region to fulfil this requirement.

To do this, I first had to organise the data with what I needed and didn't need. I removed all columns excluding 'Countries', 'Total' and 'Region'. This simplified the whole table and data gathering process, minimising the possibility of mismatch/conversion errors from other columns.

Then, I converted 'Total' to a float data type, as it was currently a string. (We did not need to convert Region as it was already a string.) To convert Total to a float, we first created a dataframe, followed by a list of all the values, using a for loop to replace the '%' character with an empty string character. (' '). After this, we then used the astype() function to convert it to a float, then using the boxplot() function to plot the given boxplot.

Task 2.2

Task 2.2 asked us to compute the mean of the 'Total' column for the wealth quintile columns (Richest/Poorest), then asked to display the top 10 countries in both.

I did the same in Task 2.1 and removed all other columns except the ones needed – those being 'Countries', 'Wealth quintile (Richest)' and 'Wealth quintile (Poorest)'

This was similar to Task 2.1, where we had to create a dataframe for both data sets – creating a separate list for Wealth quintile (Richest) and Wealth quintile (Poorest), replacing the '%' with an empty character, followed by a dataframe.

When converting, i also used the astype() method, then when calculating the mean of the wealth quintile, I used the mean() function, later using the print() statement to print the mean value, assigned to the variable mean_poorest and mean_richest respectively.

To display the top 10 countries, created a new list of countries and used the same list for wealth quintiles. (to save time and code).

I converted these values into a float, then used the sort_values() function with ascending=False to sort the values, then using head(10) to collect the top 10 values. This was the same for both wealth quintiles (richest and poorest)

Task 2.3

This task allowed us to choose the statistics measures (at least 3) of children which belong in the Lower Middle Income (LM) group. The 3 statistics measures I chose were **Mean, Median and Standard Deviation.**

These were chosen as pandas had already had a function for each of the statistics measures, as well as them being the most commonly used and recognised statistics measures in the field of statistics/data analysis.

To accomplish these tasks, I used the mean(), median() and std() functions which automatically calculate and do the math on the provided data (in this case, the spreadsheet), rounded to 2 decimal places using .2f

Looking at the data, children in the Lower Middle Income (LM) group located in more Rural areas had a lower mean, median and standard deviation, specifying that not many children in rural areas have an internet connection, those being 26.75%, 12.00% and 29.63% respectively.

Whereas the Lower Middle Income (LM) located in the Urban areas has a higher Mean at 43.17%, median at 41.00% and Standard Deviation at 30.86%.

This states that children living in more Urban areas have more frequent access to internet connection compared to children living in more Rural areas.

Use of AI Tools

ChatGPT:

ChatGPT was the main AI tool I used, and was used during this assignment in a way for extra help/guidance. ChatGPT model used was Chat-GPT model 3/3.5

AI Tools were used to help break down and explain certain errors (as mentioned above) in collaboration with the canvas sources for code formatting and my completion of the labs.

ChatGPT was also used to automate the creation of the list, for example, I would copy-paste the data from the excel sheet into chatgpt, and ask it to format and separate said data with a delimiter. (,), then use the formatted data in my assignment code.

Furthermore, it was also used to break down and explain terminology. It was also used to correctly reference (using APA 7th edition referencing), eliminating the need for a website or the need to do it by hand.

References: (7th apa referencing)

Aside from my practical teacher, Chengyao Xie, and the Data Science Lecture slides provided on the canvas shell, as well as the Labs provided every week, I also used these references:

Pandas Official Documentation:

- pandas. (n.d.). pandas.Series.str.replace. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/docs/reference/api/pandas.Series.str.replace.html
- pandas. (n.d.). pandas.read_csv. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
- pandas. (n.d.). pandas.to_numeric. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html
- pandas. (n.d.). Working with missing data. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/docs/user_guide/missing_data.html
- pandas. (n.d.). pandas.DataFrame.mean. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html
- pandas. (n.d.). pandas.DataFrame.map. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.map.html
- pandas. (n.d.). pandas.DataFrame.sort_values. pandas documentation. Retrieved September 1, 2024, from https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sort_values.html

Matplotlib Official Documentation:

- matplotlib. (n.d.). matplotlib.pyplot.boxplot. matplotlib documentation. Retrieved September 1, 2024, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html

W3Schools:

- W3Schools. (n.d.). Python arrays. Retrieved September 1, 2024, from https://www.w3schools.com/python/python_arrays.asp

GeeksforGeeks:

- GeeksforGeeks. (n.d.). pandas.DataFrame.mean() function. Retrieved September 1, 2024, from https://www.geeksforgeeks.org/python-pandas-dataframe-mean/

r/numpy Subreddit:

- Reddit. (2022, December 2). TypeError: ufunc 'true_divide' appeared unexpectedly. r/Numpy. Retrieved September 1, 2024, from https://www.reddit.com/r/Numpy/comments/zb59lp/typeerror_ufunc_true_divide_appeared_unexpectedly/

Stack Overflow Forums:

- Stack Overflow. (2017, March 10). Pandas convert string to int. Retrieved September 1, 2024, from https://stackoverflow.com/questions/42719749/pandas-convert-string-to-int
- Stack Overflow. (2013, April 8). Change column type in pandas. Retrieved September 1, 2024, from https://stackoverflow.com/questions/15891038/change-column-type-in-pandas
- Stack Overflow. (2018, January 4). Pandas convert data type from object to float. Retrieved September 1, 2024, from https://stackoverflow.com/questions/48094854/pandas-convert-data-type-from-object-to-float

- Stack Overflow. (2014, September 4). Convert percent string to float in pandas read_csv. Retrieved September 1, 2024, from https://stackoverflow.com/questions/25669588/convert-percent-string-to-float-in-pandas-read-csv