Practical Data Science (with Python)

# Data Preparation

After loading the data of the CSV file, it's evident that each column holds "object" data type, encompassing both percentage and year information. Determine the sequence for addressing errors based on the types encountered during the course. I think that task 1 should start with removing redundant whitespaces and handling missing data. It is convenient to handle other error types

## Redundant white space

Utilize the sum function in conjunction with the string function to count occurrences of consecutive spaces in each column. Recognize that columns 1, 2, and 5 necessitate addressing redundant whitespaces, which can be confirmed by printing. Using a lambda function in combination with the applymap() method to remove leading and trailing spaces from all string values in a DataFrame[1]. Subsequently, validate the space removal by utilizing the sum function in conjunction with the string function.

## Missing values

Use DataFrame.isna()[source] to identify NAs within the DataFrame. Columns with "true" values indicate the presence of NA[2]. Employ astype() to transform data types[3]. We can't delete but fill missing values with mean because there is too little data[4][16]. Use the apply() function with a lambda function to format the values in a DataFrame column as percentages[5].

## Mistakes during data entry

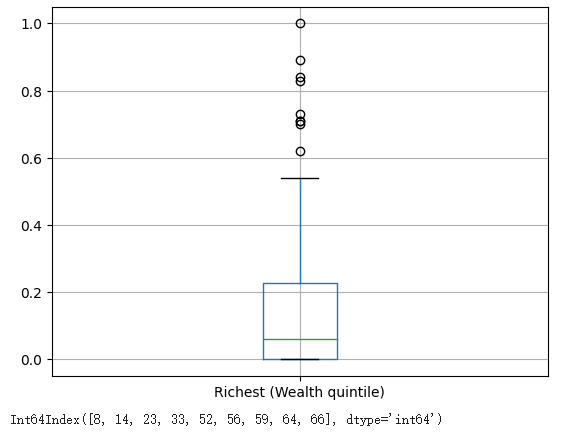
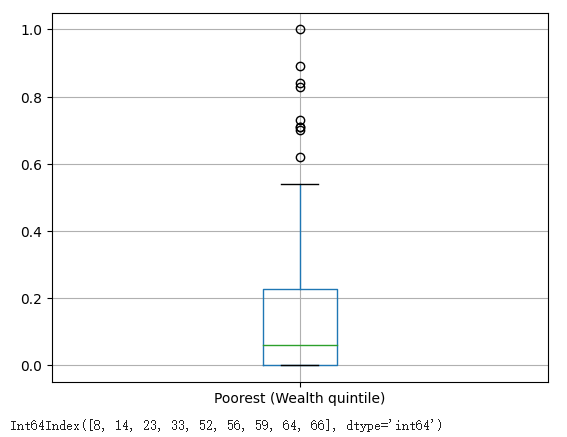
Percentage data is vulnerable to 'input errors.' Upon inspecting the output, it becomes evident that some data entries contain double percent signs. Utilize the count function to identify columns with such instances[6]. Subsequently, accurately modify them using the replace() function[7]. Moreover, apply the counts() function to identify obvious misspellings in a string, and rectify them using the replace() function[6].

## Impossible values

When addressing "Errors in data entry," it was observed that the "Time period" column contains two improbable years: "2076" and "2562." First, it's essential to standardize the year expression format by extracting the first four digits from each entry[8]. This means that if a year is represented as a range, it will be converted to the first year number of that range. Leverage the lambda function along with Python's datetime module to process the 'Time period' column in a DataFrame[5][9]. The objective is to replace years greater than the current year with the current year while keeping other years unchanged.

## Outliers

Utilize matplotlib.pyplot and pandas to create boxplots for outlier detection. Start by converting percentage data into floating-point numbers and transforming all percentage columns into boxplots[3][7][10]. By analyzing the graphs, identify outliers in two specific columns. Calculate the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for each column. Identify outliers based on the calculated threshold, print their indices, and replace them with the median value to mitigate their impact[11].



## Remove duplicate (records)

The duplicates DataFrame is created using the duplicated() method to locate duplicate rows, and the school\_cleaned DataFrame is obtained by removing the duplicate rows using the drop\_duplicates() method[12][13].

## Irrelevant observations

Read the contents of part2, identify columns for deletion, use the drop function to remove them, convert floating-point data back to percentages, and rename the cleaned files[14][5].

# Data Exploration

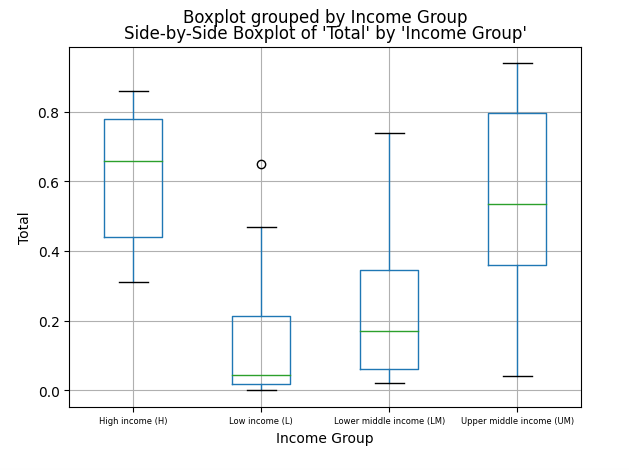
## Task 2.1 Explore the overall percentage of children in a school attendance age that have

## internet connection at home (column F). Create the side-by-side boxplot (as one

## graph/chart) having the data separated/groupded by Income Group. Note that

## there are four categories of Income Group. Compare/compute the Mean (of the total

## percentage) for each category of Income Group.

  
Upon reading the new dataset, convert it into floating-point values[5]. Subsequently, employ boxplots to visualize the dataset's distribution, categorized into four groups based on income groups[10]. Utilize the groupby() function to compute the mean value within each group[15][16]. Comparing the mean across the four groups yields the result: High income (H) > Upper middle income (UM)> Lower middle income (LM)> Low income (L).

## Task 2.2 Compare/compute the Median (of the percentage of school-age children who have

## internet connection at home) with respect to Residence (Rural) and Residence (Urban), respectively. Display the top 10 countries with the highest percentage with

## respect to Residence (Rural) and Residence (Urban), respectively.

Utilize the median() function to compute medians for both rural and urban areas. Upon comparison, it becomes evident that the median value is higher in urban areas than in rural areas. Employ the sort\_values() function to arrange cities based on their residence in rural or urban areas. Subsequently, use head(10) to retain the top ten results.[16]

The top 10 countries with the highest percentage with respect to Residence (Rural) :  
The top 10 countries with the highest percentage with respect to Residence (Rural):



## Task 2.3 Explore the data about children that are from the Upper middle income (UM) group.

## Compare the percentage of different categories of Wealth quintile (Poorest versus

## Richest), using at least three statistics measures.

Defining datasets for upper middle income groups. Then, utilize the mean(), median(), and std() functions to calculate the percentages of the poorest and richest segments respectively. Upon careful calculation, the mean, standard deviation and median of Wealth quintile (Poorest versus Richest) are indeed the same, 18%, 10%, 18%.

REFERENCES  
  
[1]“Apply strip() to all cells in dataframe with multiple data types,” Stack Overflow. https://stackoverflow.com/questions/73514510/apply-strip-to-all-cells-in-dataframe-with-multiple-data-types

[2]Vijetha, “Pandas DataFrame isna() Function,” Spark By {Examples}, Dec. 27, 2022. https://sparkbyexamples.com/pandas/pandas-dataframe-isna-function/ (accessed Aug. 27, 2023).  
[3]“pandas.Series.astype — pandas 2.0.0 documentation,” pandas.pydata.org. <https://pandas.pydata.org/docs/reference/api/pandas.Series.astype.html>  
[2]“pandas.DataFrame.fillna — pandas 1.4.1 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html  
[5]NNK, “Pandas apply() with Lambda Examples,” Spark By {Examples}, Jan. 12, 2022. https://sparkbyexamples.com/pandas/pandas-apply-with-lambda-examples/  
[6]“pandas.Series.value\_counts — pandas 1.3.4 documentation,” pandas.pydata.org. <https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html>  
[7]“pandas.Series.str.replace — pandas 2.0.3 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.Series.str.replace.html (accessed Aug. 28, 2023).

[8]“pandas.Series.str.extract — pandas 1.5.3 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.Series.str.extract.html

[9]“Datetime current year and month in Python,” Stack Overflow. https://stackoverflow.com/questions/28189442/datetime-current-year-and-month-in-python

[10]“pandas.DataFrame.boxplot — pandas 1.3.4 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.boxplot.html

[11]“Cleaning up Data from Outliers | Pluralsight,” www.pluralsight.com. https://www.pluralsight.com/guides/cleaning-up-data-from-outliers

[12]“pandas.DataFrame.duplicated — pandas 1.4.1 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.duplicated.html

[13]“pandas.DataFrame.drop\_duplicates — pandas 1.2.4 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop\_duplicates.html

[14]“pandas.DataFrame.drop — pandas 1.2.4 documentation,” pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html

[15]“Group by: split-apply-combine — pandas 1.2.3 documentation,” pandas.pydata.org. <https://pandas.pydata.org/docs/user_guide/groupby.html>  
[16]“Data Manipulation with pandas - Yulei’s Sandbox,” yuleii.github.io. https://yuleii.github.io/2020/06/27/data-manipulation-with-pandas.html