Table 1:

For table 1, I first dropped the first 15 rows because the information these rows contained are redundant and meaningless. This table has some problems that might violate the tidy data principles. For example, after I renamed some columns by using *"DataFrame.column"*, they didn't only appear multiple variables stored in 1 column such as "`both_sexs2000`", but also their column labels were values not variable names, which violated the principles 1 & 2. To correct them, I used the *"DataFrame.melt"* function to unpivot table one from wide to long format. By doing so, I kept the columns that are "**Sort order**", "**Major area, region, country or area of destination**", "**Notes**", "**Country code**", and "**Type of data (a)**", and I used "*DataFrame.assign*" function to assign new columns to the table. To be specific, I named the new column "**Genderandyear**" by using "*var_name*", and I used "*value_name*" for inputting my values into the column. The purpose of using "*DataFrame.assign*" is that I wanted to separate the value of "Gender" and "Year" where both are in the "**Genderandyear**" column, and I needed to create two new columns to separately display "Gender" and "Year" and I used a lambda function "*Gender = lambda x: x.Genderandyear.str[0].astype(str), Year = lambda x: x.Genderandyear.str[-4:].astype(str)*" to fit my needs. Further, I used "*DataFrame.*replace" to replace the values from "f", "b", and "m" to "female", "both sexes", and "male" for styling purposes, and I dropped the invalid column "**Genderandyear**" by using "*DataFrame.drop*". In the end, I reset the index by using "*Columns.tolist*" and "*Dataframe.insert*". Finally, I got a cleaned table with 4770 rows and 7 columns.

Table 2:

Same steps as mentioned in Table 1.

Table 3:

Same steps as mentioned in Table 1.

Table 4:

Same steps as mentioned in Table 1.

Table 5:

I noticed that table 5 has a unique time span which is a five-year period that is different from previous tables.

Table 6:

The situation in table 6 is more complicated, so I decided to clean the table separately. First of all, I still kept the columns "**Sort order**", "**Major area, region, country or area of destination**", "**Notes**", "**Country code**", and "**Type of data (a)**" as before, and I divided the table into three different sub-tables by using "*Dataframe.drop*" function. For my first sub-table, I deleted the columns which belong to "**Refugees as a percentage of the international migrant stock**" and "**Annual rate of change of the refugee stock**", and I duplicated what I've done previously for tables 1-5. In sub-table 2, I cleaned the data for "**Refugees as a percentage of the international migrant stock**"; In sub-table 3, I cleaned the data for "**Refugees as a percentage of the international migrant stock**".

Merge:

Eventually, I found that table one and table three actually display similar content, so I decided to merge them together by using "*Dataframe.merge*" function.

Conclusion:

I think my dataset is cleaned. My column names are informative, and variable names. Meanwhile, I corrected some wrong columns that consisted of multiple variables such as "**Genderandyear**" which I mentioned above. Also, the variables in the dataset are not in rows and columns, they are stored in cells. Finally, each table in the dataset only has one singular data type and single observational units. I didn't merge every table into one because I thought some of them that have different time-span and observational units. Therefore, I chose to only merge table 1 and table 3 when they focus on the same topic.

Difficulty:

I've faced a lot of difficulties. For example, I initially aimed to combine all columns from table 1, table 2, table 3, and partial columns from table 5 and table 6 into a new table called final_table_1 and combine the rest columns into another table called final_table_2, because these columns use the same time spam. However, I failed because after I merged them, I found the length of the table was too long and that might be wrong. When I used "*Dataframe.merge*", it was difficult for me to decide whether to use "*left_on*", "*right_on*", or "*on*" based on my situation. Moreover, I failed to style the tables. I wanted to make each row be ordered by country name.