My first step to clean up the dataset was to start with the 'easiest' table: Table 4. This table was the easiest as it only had one sex (female). To start, I imported the Excel file and table 4 and used the head function to see what I had to work with. Immediately, I realized that the header of the Excel sheet was showing up and I could get rid of the first 14 lines, so I used a drop function to drop indexes 0-13. After that, I needed to change the names of the headers, so I used iloc on the first row and assigned it to a variable called "new_header". I then modified Table 4 (t4) to include all the rows expect row 1. Finally, I then assigned the columns headers to the values of the variable "new_header".

Now that I had a cleaner version of the table to work with, I started to identify columns and rows which would not be of use for a data analysis. I thus deleted the columns called "Sort\norder", "Notes", "Country", "Type of data (a)" and the deleted the first row. Then I renamed the columns to better represent their values. I renamed the first column "Migration destination" and the other the year they represented along with the sex of the migrants (in this case female). This step created columns which looked like "f1990" and thus resembled one of the datasets we cleaned in class.

Following the model we did in class and to respect the first and second principles of tidy data, I then used the melt function to bring the year/sex column names into the dataset assigning their value the name "Migrants". I then separated the letter for sex and the number for the year using two lambda functions within an assign function and dropped the "Years" column (I copied this almost directly from the course github page). Now that I had four columns, I reorganized them to have the destination first, followed by year, sex, and finally the number of migrants. Continuing to follow the model we did in class, I then replace the placeholder letter I had used for the sex with the full word using the replace function.

At the same time, I also replaced all the ". ." values with a blank space. The reason for this is that when I originally tried to use the pivot function later, I kept getting an error called "int64" for the "Migrants" column. I understood through research that this was because I needed to convert all of the values in the "Migrants" column to int64 values using the pd.to_numeric function. I thus had to remove all the ". ." values which caused errors with the pd.to_numeric function.

With the ". ." error solved, I was able to successfully convert all the values of the "Migrants" column and pivot the table, keeping "Migration Destination" and "Year" as they were, but assigning the values of the "Migrants" column to the values of the "Sex" column. I was not too sure whether this step violates the first principle of tidy data as I was worried that by bringing the sex back into the column headers would mean that these contained useful information. However, upon thinking further I realized that the values under female were the number of female migrants which moved to different regions. Therefore, changing the name of the column to include the sex did not violate the first principle. My decision was further supported when I tackled the other tables.

The main differences in my code from table 4 and the code for all the other tables was the addition of other sexes (and in the case of table 6, the percentage of migrant stock, and rate of change). Like with table 4, I merged the years and sexes at first to rename the columns and then separated them again after I had used the melt function. Except for a few column names and the inclusion of year range for table 6, the code is mostly the same for all the tables.