The purpose of this assignment was to clean the document “Trend in International Migrant Stock: The 2015 Revision” published by the United Nations. This document was comprised of six tables and related contextual information. Tables 1 and 2 are separated according to different regions, continents, and countries and indicate the number of migrants which moved to these destinations according to year and sex. Tables 3 to 6 present the rate of change and percentage of the movement of migrant stock to various regions again according to year and sex. While the UN dataset is useful as is for the analysis of specific countries or regions, as data it is quite messy. In order to be able to create an effective and clear analysis of the dataset, cleaning was required.

The cleaning process was done following the principles of tidy data: (1) every column is a single variable; (2) each row is a single observation; (3) every cell is a singular value. The UN dataset violated the first two principles. Column headers often contained values, such as years, multiple variables, such as years and sexes were contained within the same column, and many observations were found in the same row (for Table 6 exclusively).

To solve the first two issues, I proceeded the using the following method. 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.

In [1]:
```python
import pandas as pd

#read Excel and get Table 4 sheet
xls = pd.ExcelFile("https://github.com/shionguha/inf1340-programmingfordatascience-fa22/raw/60b7f5d757553308a4b5db8c439c360ea244e
t4 = pd.read_excel(xls, "Table 4")

t4.head()
```

Out[1]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	United Nations	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	Population Division	NaN	NaN	NaN	NaN	NaN	NaN

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”.

```
In [2]: import pandas as pd

        #read Excel and get Table 4 sheet
        xls = pd.ExcelFile("https://github.com/shionguha/inf1340-programmingfordatascience-fa22/raw/60b7f5d757553308a4b5db8c439c360ea244
        t4 = pd.read_excel(xls, "Table 4")

        #drop all the context at the top of the sheet
        t4 = t4.drop(t4.index[0:13])

        #change column headers to data from first row
        new_header = t4.iloc[0]
        t4 = t4[1:]
        t4.columns = new_header

        t4.head()
```

Out[2]:

13	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Female migrants as a percentage of the international migrant stock	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	2015.000000
15	1	WORLD	NaN	900	NaN	49.03915	49.16879	49.112244	48.832993	48.305660	48.249769
16	2	Developed regions	(b)	901	NaN	51.123977	51.149024	51.113307	51.171501	51.658932	51.866687
17	3	Developing regions	(c)	902	NaN	46.592099	46.500135	46.128444	45.134297	43.319780	43.327078
18	4	Least developed countries	(d)	941	NaN	47.261155	47.571664	46.826689	45.157406	45.499573	45.942752

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.

```
#drops first column and first row which contain useless info
t4 = t4.drop(columns=["Sort\norder", "Notes", "Country code", "Type of data (a)"])
t4 = t4.drop([14])

#renames certain column headers to years
t4.columns = ["Migration destination", "f1990", "f1995", "f2000", "f2005", "f2010", "f2015"]

t4.head()
```

Out[3]:

	Migration destination	f1990	f1995	f2000	f2005	f2010	f2015
15	WORLD	49.03915	49.16879	49.112244	48.832993	48.305660	48.249769
16	Developed regions	51.123977	51.149024	51.113307	51.171501	51.658932	51.866687
17	Developing regions	46.592099	46.500135	46.128444	45.134297	43.319780	43.327078
18	Least developed countries	47.261155	47.571664	46.826689	45.157406	45.499573	45.942752
19	Less developed regions excluding least develop...	46.466684	46.279022	46.009598	45.130768	43.043672	42.984398

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.

```
#moves year values into the table by creating year column with migrant values
t4 = t4.melt(id_vars = ["Migration destination"],
             var_name = "Years", value_name = "Migrants")

#seperates gender and year
t4 = (t4.assign(Year = lambda x: x.Years.str[1:].astype(str), Sex = lambda x: x.Years.str[0].astype(str))
      .drop("Years", axis = 1))

#reorganizes columns
t4 = t4[["Migration destination", "Year", "Sex", "Migrants"]]


t4.head()
```

Out[5]:

	Migration destination	Year	Sex	Migrants
0	WORLD	1990	f	49.03915
1	Developed regions	1990	f	51.123977
2	Developing regions	1990	f	46.592099
3	Least developed countries	1990	f	47.261155
4	Less developed regions excluding least develop...	1990	f	46.466684

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.

```
#full word for sex
#remove .. values which would cause an error
t4 = t4.replace(to_replace = ["f", ".."], value = ["Female", ""])


t4.head()
```

Out[6]:

	Migration destination	Year	Sex	Migrants
0	WORLD	1990	Female	49.03915
1	Developed regions	1990	Female	51.123977
2	Developing regions	1990	Female	46.592099
3	Least developed countries	1990	Female	47.261155
4	Less developed regions excluding least develop...	1990	Female	46.466684

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 now contained values. However, upon thinking further I realized female was a variable and that the values in the table were the number of migrants assigned to the female variable. Therefore, changing the name of the column to include the sex did not violate the first principle. I thus continued to use this approach when cleaning tables 1-3 and table 5 with the addition of the male variable and the both sexes variable.

```python
#Replace migrants with each sex category
#use pd.to_numeric to fix error mentioned above
t4["Migrants"] = t4["Migrants"].apply(pd.to_numeric)

t4.pivot_table(index = ["Migration destination","Year"],
               columns = "Sex",
               values = "Migrants")
```

Out[9]:

Migration destination	Year	Female
Afghanistan	1990	43.559963
	1995	45.324516
	2000	43.559414
	2005	43.557847
	2010	43.558672
...	...	...
Zimbabwe	1995	42.950564
	2000	42.970825
	2005	42.965625
	2010	42.957493
	2015	42.993637

1575 rows × 1 columns

Table 6 was different and needed a new approach. Not only did table 6 also bring in the male and both gender variables, but this table also violated the second principle of tidy data. Each row contained three different observations: (1) estimated refugee stock as mid-year (both sexes), (2) refugees as a percentage of the international migrants stock, (3) annual rate of change of the refugee stock.

My first step was to once again bring the table into a format I could work with by removing the header, creating new headers, dropping unnecessary columns and renaming the new headers.

```python
t6.columns = new_header

#drops first column and first row which contain useless info
t6 = t6.drop(columns=["Sort\norder", "Notes", "Country code", "Type of data (a)"])
t6 = t6.drop([14])

#renames certain column headers to years
t6.columns = ["Refugee destination", "b1990", "b1995", "b2000", "b2005", "b2010", "b2015",
              "m1990", "m1995", "m2000", "m2005", "m2010", "m2015",
              "f1990-1995", "f1995-2000", "f2000-2005", "f2005-2010", "f2010-2015"]

t6.head()
```

Out[30]:

	Refugee destination	b1990	b1995	b2000	b2005	b2010	b2015	m1990	m1995	m2000	m2005	m2010	m2015	f1990-1995
15	WORLD	18836571	17853840	15827803	13276733	15370755.0	19577474.0	12.346732	11.103013	9.164736	6.941389	6.932687	8.033424	-2.123497
16	Developed regions	2014564	3609670	2997256	2361229	2046917.0	1954224.0	2.445494	3.910511	2.899391	2.015025	1.544140	1.391085	9.388424
17	Developing regions	16822007	14244170	12830547	10915504	13323838.0	17623250.0	23.968236	20.795958	18.507035	14.733162	14.944759	17.073768	-2.839417
18	Least developed countries	5048391	5160131	3047488	2363782	1957884.0	3443582.0	45.56588	44.041961	30.221557	24.08243	19.533425	28.801534	-0.680327
19	Less developed regions excluding least develop...	11773616	9084039	9783059	8551722	11365954.0	14179668.0	19.919743	15.999082	16.51313	13.305391	14.363526	15.537313	-4.3836

I then proceeded to separate this one table into three tables to account for each observation. Pictured below is table 6_1 which shows estimated refugee stock for both sexes.

```
#split table so that each observation is in a single table
t6_1 = t6[["Refugee destination", "b1990", "b1995", "b2000", "b2005", "b2010", "b2015"]].copy()
t6_2 = t6[["Refugee destination", "m1990", "m1995", "m2000", "m2005", "m2010", "m2015"]].copy()
t6_3 = t6[["Refugee destination", "f1990-1995", "f1995-2000", "f2000-2005", "f2005-2010", "f2010-2015"]].copy()

t6_1.head()
```

Out[23]:

	Refugee destination	b1990	b1995	b2000	b2005	b2010	b2015
15	WORLD	18836571	17853840	15827803	13276733	15370755.0	19577474.0
16	Developed regions	2014564	3609670	2997256	2361229	2046917.0	1954224.0
17	Developing regions	16822007	14244170	12830547	10915504	13323838.0	17623250.0
18	Least developed countries	5048391	5160131	3047488	2363782	1957884.0	3443582.0
19	Less developed regions excluding least develop...	11773616	9084039	9783059	8551722	11365954.0	14179668.0

Then I continued as I had for the tables 1-5 using the melt and pivot functions until the each of the three new sub-tables were cleaned.

The results of this cleaning process changed six messy datasets into eight clean long datasets which respected all three principles of tidy data. Throughout this process I learned a few crucial skills for cleaning datasets. (1) It is important to identify what are variables, what are observations, and what are values before you begin your cleaning process. If I had done this step, it would have saved me some time trying to understand whether sex was a variable or a value after having already cleaned table 4. (2) Identifying what is crucial data and what is not. When I was first faced with the datasets, I had a hard time trying to identify which columns I could remove. However, after reflecting on which values were essential to analyze the data in a visual way, I was able to identify useless columns. (3) When to stop. Originally, I was going to separate the major areas, regions, and countries into three different columns, creating a wide dataset. Here, I was thinking as a historian, but I realized that as a data analyst this was not necessary as these were all values of the same variable – migrant/refugee destination. So, learning when to stop cleaning was also a challenge. In brief, this assignment taught me a great deal about cleaning messy datasets like those in the UN Migrant file using the principles of tidy data.