Midterm Write Up

Emily Clarke

INF 1340

Professor Shion Guha

November 16, 2022

Introduction

Tidy data is a standard in data science that is focused on the mapping and cleaning of a dataset structure. It is an established standard in data science that determines if a dataset is messy or tidy depending on how the rows, columns and tables are aligned with the observations, variables, and types. Tidy data is structured by five fundamental principles, which are the following: principle 1 states that column headers should be variable names, not values; principle 2 states that multiple variables should not be stored in one column; principle 3 states that variables should not be stored in both rows and columns; principle 4 states that multiple types of observational units should not be stored in the same table; and lastly, principle 5 states that a single observational unit should not be stored in multiple tables.

Within the scope of this midterm assignment, I will be applying tidy data principles to the United Nations (UN) excel sheet titled Migrant Stock: 2015 Revision. Of the nine sheets located in the excel sheet, there are six tables that will be tidied and three tables that will be excluded, which are the contents, annex, and notes page. The purpose of this write-up is to analyze my overall tidy data process and touch on limitations that I encountered.

Table 1

Table 1 was the first sheet in the UN dataset that I decided to clean. As shown in Figure 1 below, I used the .columns function to identify the column framework. Evidently, this arrangement is very messy as it is showing all of the columns in the index as an unnamed variable. This is not what is expressed in the excel sheet, which was a clear indication that this violated the principles of tidy data.

```
In [58]: sheet1.columns

Out[58]: Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed:
         4',
                'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed:
         9',
                'Unnamed: 10', 'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13',
                'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17',
                'Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21',
                'Unnamed: 22'],
               dtype='object')
```

Figure 1: Opening the column types from Table 1 in the UN dataset.

As found in Figure 2 below, this is the data input that I used for tidying the majority of Table 1. Included in Figure 2 is the rename column function. I renamed the columns to reflect the order from Table 1 in the UN excel sheet, beginning with creating a dictionary for the column titles. Next, the pandas melt function was utilized with the consideration of principle 1 and principle 2. The melt function was slightly cut-off in Figure 2, so it should be noted that the var_name = “Demographic” and value_name = “International Migrant Stock at Mid-Year”. The outcome of the melt function and the insertion of the columns “Demographic” and “International Migrant Stock at Mid-Year” can be viewed in Figure 3, which is the output of Figure 2.

```
In [699]: Table1 = pd.DataFrame.from_records(
           sheet1.
           rename(columns={"Unnamed: 0": "Sort Order",
                           "Unnamed: 1": "Major area, region, country or area of destination",
                           "Unnamed: 2": "Notes",
                           "Unnamed: 3": "Country code",
                           "Unnamed: 4": "Type of data",
                           "Unnamed: 5": "b1990",
                           "Unnamed: 6": "b1995",
                           "Unnamed: 7": "b2000",
                           "Unnamed: 8": "b2005",
                           "Unnamed: 9": "b2010",
                           "Unnamed: 10": "b2015",
                           "Unnamed: 11": "m1990",
                           "Unnamed: 12": "m1995",
                           "Unnamed: 13": "m2000",
                           "Unnamed: 14": "m2005",
                           "Unnamed: 15": "m2010",
                           "Unnamed: 16": "m2015",
                           "Unnamed: 17": "f1990",
                           "Unnamed: 18": "f1995",
                           "Unnamed: 19": "f2000",
                           "Unnamed: 20": "f2005",
                           "Unnamed: 21": "f2010",
                           "Unnamed: 22": "f2015"}).
           melt(id_vars = ["Sort Order", "Major area, region, country or area of destination", "Notes", "Country code", "Type of
           drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).
           drop(columns = ['Country code', 'Sort Order']).
           reset_index(drop=True).
           head(5026)
          )
```

Figure 2: Data input for tidying Table 1. The functions include the rename columns, melt, drop, drop columns, reset index, and head.

I decided to drop the first 15 rows as those rows because they reflected a brief and unnecessary introduction to Table 1. The columns that I decided to drop were “Country code” and “Sort Order”. I dropped the “Country code” considering that “Major area, region, country or area of destination” were synonymous with one another. In my perspective, having the “Country code” was redundant and had no order which could be potentially confusing when it came to analyzing this data. I dropped “Sort Order” as it violated principle 4. There were two indexes and “Sort Order” was not something that should be kept.

Out[699]:

	Major area, region, country or area of destination	Notes	Type of data	Demographic	International Migrant Stock at Mid-Year
0	WORLD	NaN	NaN	b1990	152563212
1	Developed regions	(b)	NaN	b1990	82378628
2	Developing regions	(c)	NaN	b1990	70184584
3	Least developed countries	(d)	NaN	b1990	11075966
4	Less developed regions excluding least develop...	NaN	NaN	b1990	59105261
...	...	...	...	...	...
5020	Samoa	NaN	B	f2015	2460.0
5021	Tokelau	NaN	B	f2015	254.0
5022	Tonga	NaN	B	f2015	2604.0
5023	Tuvalu	NaN	C	f2015	63.0
5024	Wallis and Futuna Islands	NaN	B	f2015	1411.0

5025 rows × 5 columns

Figure 3: Output of data before using the lambda function to separate the Demographic column.

Figure 3 illustrates the “Demographic” variable that was created by me using the rename function and the melt function. The sex function was renamed to be in the same cells as the year, which violates principle 2. The melt function then corrected the violation to principle 1 and principle 2, as the years were previously located in the column headers. This puts more order to things, but it is not completely correct just yet. To correct this issue, as pictured in Figure 4, the lambda function was used to separate the “Demographic” column into two separate columns “Sex” and “Year”.

```
In [746]: Table1 = (Table1.assign(Sex = lambda x: x.Demographic.str[0].astype(str),
                                  Year = lambda x: x.Demographic.str[1:].astype(str)).drop("Demographic", axis=1))
          Table1.head(10)
```

Out[746]:

	Major area, region, country or area of destination	Notes	Type of data	International Migrant Stock at Mid-Year	Sex	Year
0	WORLD	NaN	NaN	152563212	b	1990
1	Developed regions	(b)	NaN	82378628	b	1990
2	Developing regions	(c)	NaN	70184584	b	1990
3	Least developed countries	(d)	NaN	11075966	b	1990
4	Less developed regions excluding least develop...	NaN	NaN	59105261	b	1990
5	Sub-Saharan Africa	(e)	NaN	14690319	b	1990
6	Africa	NaN	NaN	15690623	b	1990
7	Eastern Africa	NaN	NaN	5964031	b	1990
8	Burundi	NaN	B R	333110	b	1990
9	Comoros	NaN	B	14079	b	1990

Figure 4: Lambda Function and Final Table Output for Table 1. Under the Sex variable name, it should be noted that variable b is for both genders, variable m for male, variable f for female.

Table 2

Table 2 is the second sheet within the UN dataset that required cleaning. As shown in Figure 5 below, it illustrates my process throughout tidying Table 2. To begin, as principle 1 suggests, column headers should be variable names. I used the rename column function to create a dictionary of the correct column headers as reflected in the raw UN dataset. This violates principle 1 in the fullest extent, as values should not be in the headers.

```
In [695]: Table2 = pd.DataFrame.from_records(
           sheet2.
           rename(columns={"Unnamed: 0": "Sort Order",
                           "Unnamed: 1": "Major area, region, country or area of destination",
                           "Unnamed: 2": "Notes",
                           "Unnamed: 3": "Country code",
                           "Unnamed: 4": "b1990",
                           "Unnamed: 5": "b1995",
                           "Unnamed: 6": "b2000",
                           "Unnamed: 7": "b2005",
                           "Unnamed: 8": "b2010",
                           "Unnamed: 9": "b2015",
                           "Unnamed: 10": "m1990",
                           "Unnamed: 11": "m1995",
                           "Unnamed: 12": "m2000",
                           "Unnamed: 13": "m2005",
                           "Unnamed: 14": "m2010",
                           "Unnamed: 15": "m2015",
                           "Unnamed: 16": "f1990",
                           "Unnamed: 17": "f1995",
                           "Unnamed: 18": "f2000",
                           "Unnamed: 19": "f2005",
                           "Unnamed: 20": "f2010",
                           "Unnamed: 21": "f2015"}).
           melt(id_vars = ["Sort Order", "Major area, region, country or area of destination", "Notes", "Country code"], var_name
           drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).
           drop(columns = ['Country code', 'Sort Order']).
           reset_index(drop=True).
           head(5025)
          )
          Table2.head(5025)
```

Figure 5: Data input for tidying Table 2. The functions include the rename columns, melt, drop, drop columns, reset index, and head.

To address the violation of principle 1 and principle 2, the melt function was used to place the appropriate values into their columns. In Figure 5, the code is partially cut-off for the melt function, it should be noted that the var_name = “Demographic” and value_name = “Total Population at Mid-Year”. These are the new column headers to resolve the violation of principle 1 and principle 2, these results can be viewed in Figure 6.

The first 15 rows of Table 2 as they reflected an unnecessary introduction and offered no data to the Table. The columns that I decided to drop were “Country code” and “Sort Order”. “Country code” was dropped because “Major area, region, country or area of destination” were already included in the dataset. I thought to remove the redundancy, it would be useful to remove “Country code”. “Sort Order” was dropped as it violated principle 4 and overlapped with the additional index.

Out[718]:

	Major area, region, country or area of destination	Notes	Demographic	Total Population at Mid-Year
0	WORLD	NaN	b1990	5309667.699
1	Developed regions	(b)	b1990	1144463.062
2	Developing regions	(c)	b1990	4165204.637
3	Least developed countries	(d)	b1990	510057.629
4	Less developed regions excluding least develop...	NaN	b1990	3655147.008
...	...	...	...	...
5020	Samoa	NaN	f2015	93.584
5021	Tokelau	NaN	f2015	..
5022	Tonga	NaN	f2015	52.931
5023	Tuvalu	NaN	f2015	..
5024	Wallis and Futuna Islands	NaN	f2015	..

5025 rows × 4 columns

Figure 6: Output from the input for Table 2. Prior to the use of the lambda function to separate the Demographic column.

```
In [729]: Table2 = (Table2.assign(Sex = lambda x: x.Demographic.str[0].astype(str),
                                  Year = lambda x: x.Demographic.str[1:].astype(str)).drop("Demographic", axis=1))
          Table2.head(10)
```

Out[729]:

	Major area, region, country or area of destination	Notes	Total Population at Mid-Year	Sex	Year
0	WORLD	NaN	5309667.699	b	1990
1	Developed regions	(b)	1144463.062	b	1990
2	Developing regions	(c)	4165204.637	b	1990
3	Least developed countries	(d)	510057.629	b	1990
4	Less developed regions excluding least develop...	NaN	3655147.008	b	1990
5	Sub-Saharan Africa	(e)	491497.691	b	1990
6	Africa	NaN	631614.304	b	1990
7	Eastern Africa	NaN	198231.687	b	1990
8	Burundi	NaN	5613.141	b	1990
9	Comoros	NaN	415.144	b	1990

Figure 7: Lambda Function and Final Table Output for Table 2.

The "Demographic" variable, as shown in Figure 6, violates principle 2. I previously recoded the years in Figure 5 to indicate the "Sex" before the "Year" (ex. b1990 for both sexes and year). Under the "Sex" variable name, it should be noted that variable b is for both genders, variable m for male, variable f for female. Having multiple variables in one column is an incorrect format according to principle 2, which is why I used the lambda function to separate the "Demographic" column. "Demographic" was then split into two columns "Sex" and "Year" to reflect the columns separately.

Please see the conclusion where I address the limitations I encountered in Table 2.

Table 3

Table 3 is the third sheet in the UN dataset that I cleaned. The process of cleaning was quite similar to Table 1 and Table 2. As shown in Figure 8, principle 1 was violated considering the headers did not have a variable name. This was corrected by using the rename column function, where I created a dictionary of column headers to reflect the UN dataset. Once this is complete, it then violates principle 1 further, as column headers should be variable names, not values. To rectify this, the melt function was used to place these values into their appropriate columns.

```
In [692]: Table3 = pd.DataFrame.from_records(
 sheet3.
 rename(columns={"Unnamed: 0": "Sort Order",
                 "Unnamed: 1": "Major area, region, country or area of destination",
                 "Unnamed: 2": "Notes",
                 "Unnamed: 3": "Country code",
                 "Unnamed: 4": "Type of data",
                 "Unnamed: 5": "b1990",
                 "Unnamed: 6": "b1995",
                 "Unnamed: 7": "b2000",
                 "Unnamed: 8": "b2005",
                 "Unnamed: 9": "b2010",
                 "Unnamed: 10": "b2015",
                 "Unnamed: 11": "m1990",
                 "Unnamed: 12": "m1995",
                 "Unnamed: 13": "m2000",
                 "Unnamed: 14": "m2005",
                 "Unnamed: 15": "m2010",
                 "Unnamed: 16": "m2015",
                 "Unnamed: 17": "f1990",
                 "Unnamed: 18": "f1995",
                 "Unnamed: 19": "f2000",
                 "Unnamed: 20": "f2005",
                 "Unnamed: 21": "f2010",
                 "Unnamed: 22": "f2015"}).
 melt(id_vars = ["Sort Order", "Major area, region, country or area of destination", "Notes", "Country code", "Type of
 drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).
 drop(columns = ['Country code', 'Sort Order']).
 reset_index(drop=True).
 head(5025)
)
Table3.head(5025)
```

Figure 8: Data input for tidying Table 3. The functions include the rename columns, melt, drop, drop columns, reset index, and head.

The drop function was used to drop the first 15 rows as it does not offer valuable information to Table 3. The drop column function was used to drop “Country code” and “Sort Order”. The reasoning is the same as stated in Table 1 and Table 2.

Out[692]:

	Major area, region, country or area of destination	Notes	Type of data	Demographic	International Migrant Stock as a Percentage
0	WORLD	NaN	NaN	b1990	2.87331
1	Developed regions	(b)	NaN	b1990	7.198015
2	Developing regions	(c)	NaN	b1990	1.685021
3	Least developed countries	(d)	NaN	b1990	2.171513
4	Less developed regions excluding least develop...	NaN	NaN	b1990	1.617042
...	...	...	...	...	...
5020	Samoa	NaN	B	f2015	2.628654
5021	Tokelau	NaN	B	f2015	..
5022	Tonga	NaN	B	f2015	4.919612
5023	Tuvalu	NaN	C	f2015	..
5024	Wallis and Futuna Islands	NaN	B	f2015	..

5025 rows × 5 columns

Figure 9: Output from the input for Table 3. Prior to the use of the lambda function to separate the Demographic column.

Figure 9 illustrates the “Demographic” variable, which is the variable I created with the melt function. Under the Sex variable name, it should be noted that variable b is for both genders, variable m for male, variable f for female. This column violates principle 2 considering

it has two variables in the column, which is “Sex” and “Year”. To split these into two column headers, the lambda function was used to rectify the violation. The lambda function and final output table is shown in Figure 10 below.

```
In [736]: Table3 = (Table3.assign(Sex = lambda x: x.Demographic.str[0].astype(str),
                                  Year = lambda x: x.Demographic.str[1:].astype(str)).drop("Demographic", axis=1))
          Table3.head(10)
```

Out[736]:

	Major area, region, country or area of destination	Notes	Type of data	International Migrant Stock as a Percentage	Sex	Year
0	WORLD	NaN	NaN	2.87331	b	1990
1	Developed regions	(b)	NaN	7.198015	b	1990
2	Developing regions	(c)	NaN	1.685021	b	1990
3	Least developed countries	(d)	NaN	2.171513	b	1990
4	Less developed regions excluding least develop...	NaN	NaN	1.617042	b	1990
5	Sub-Saharan Africa	(e)	NaN	2.988889	b	1990
6	Africa	NaN	NaN	2.48421	b	1990
7	Eastern Africa	NaN	NaN	3.008616	b	1990
8	Burundi	NaN	B R	5.934467	b	1990
9	Comoros	NaN	B	3.391353	b	1990

Figure 10: Lambda Function and Final Table Output for Table 3.

Table 4

Table 4 is the fourth sheet within the UN dataset that required cleaning. As illustrated in Figure 11, it includes the input process of cleaning Table 4. Firstly, principle 1 was violated, therefore the rename column headers were used to reflect the headers in the UN dataset. The outcome of this ended up being that the column headers had both variable names and values, which further violates principle 1. To address this violation, the melt function was used to move the values from the column headers to their appropriate columns. The code is partially cut-off in Figure 11, so it should be noted that var_name = “Demographic” and value_name = “Female Migrants as a Percentage of the International Migrant Stock”. These are the two new column headers, which are shown in Figure 12.

In [691]:
```
Table4 = pd.DataFrame.from_records(
 sheet4.
 rename(columns={"Unnamed: 0": "Sort Order",
                 "Unnamed: 1": "Major area, region, country or area of destination",
                 "Unnamed: 2": "Notes",
                 "Unnamed: 3": "Country code",
                 "Unnamed: 4": "Type of data",
                 "Unnamed: 5": "f1990",
                 "Unnamed: 6": "f1995",
                 "Unnamed: 7": "f2000",
                 "Unnamed: 8": "f2005",
                 "Unnamed: 9": "f2010",
                 "Unnamed: 10": "f2015"}).
 melt(id_vars = ["Sort Order", "Major area, region, country or area of destination", "Notes", "Country code", "Type of
 drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).
 drop(columns = ['Country code', 'Sort Order']).
 reset_index(drop=True).
 head(1665)
)
Table4.head(1665)
```

Figure 11: Data input for tidying Table 4. The functions include the rename columns, melt, drop, drop columns, reset index, and head.

The first 15 rows were dropped for the same reasons as in Table 1, Table 2, and Table 3. The columns dropped, which are "Country code" and "Sort Order", were dropped for the same reasons in Table 1 and Table 2.

Out[691]:

	Major area, region, country or area of destination	Notes	Type of data	Demographic	Female Migrants as a Percentage of the International Migrant Stock
0	WORLD	NaN	NaN	f1990	49.03915
1	Developed regions	(b)	NaN	f1990	51.123977
2	Developing regions	(c)	NaN	f1990	46.592099
3	Least developed countries	(d)	NaN	f1990	47.261155
4	Less developed regions excluding least develop...	NaN	NaN	f1990	46.466684
...	...	...	...	...	...
1660	Samoa	NaN	B	f2015	49.908704
1661	Tokelau	NaN	B	f2015	52.156057
1662	Tonga	NaN	B	f2015	45.437096
1663	Tuvalu	NaN	C	f2015	44.680851
1664	Wallis and Futuna Islands	NaN	B	f2015	49.52615

Figure 12: Output from the input for Table 4. Prior to the use of the lambda function to separate the Demographic column.

In Figure 12, the "Demographic" variable that I recoded violates principle 2. Previously, I recoded "Sex" and "Year" to be in the same cell, with "Sex" being represented as f. Due to this, it is a violation to principle 2, and they require their own separate columns. Therefore, in Figure 3, the lambda function was used to separate into "Sex" and "Year" columns, respectively.

```
In [739]: Table4 = (Table4.assign(Sex = lambda x: x.Demographic.str[0].astype(str),
                                  Year = lambda x: x.Demographic.str[1:].astype(str)).drop("Demographic", axis=1))
          Table4.head(10)
```

Out[739]:

	Major area, region, country or area of destination	Notes	Type of data	Female Migrants as a Percentage of the International Migrant Stock	Sex	Year
0	WORLD	NaN	NaN	49.03915	f	1990
1	Developed regions	(b)	NaN	51.123977	f	1990
2	Developing regions	(c)	NaN	46.592099	f	1990
3	Least developed countries	(d)	NaN	47.261155	f	1990
4	Less developed regions excluding least develop...	NaN	NaN	46.466684	f	1990
5	Sub-Saharan Africa	(e)	NaN	47.276121	f	1990
6	Africa	NaN	NaN	47.232408	f	1990
7	Eastern Africa	NaN	NaN	48.504812	f	1990
8	Burundi	NaN	B R	50.987061	f	1990
9	Comoros	NaN	B	52.290646	f	1990

Figure 13: Lambda Function and Final Table Output for Table 4.

Table 5

Table 5 is the fifth sheet within the UN dataset. Figure 14 depicts the data input process for tidying Table 5. Similar to the previous tables, principle 1 was the first principle identified that was violated. Again, the rename function was used to rename the original column headers in the UN dataset and the melt function was used to create new columns for the values that were in the column headers. It should be noted that var_name = "Demographic" and value_name = "Annual Rate of Change of the Migrant Stock", as it is cut-off in Figure 14.

```
In [707]: Table5 = pd.DataFrame.from_records(
           sheet5.
           rename(columns={"Unnamed: 0": "Sort Order",
                           "Unnamed: 1": "Major area, region, country or area of destination",
                           "Unnamed: 2": "Notes",
                           "Unnamed: 3": "Country code",
                           "Unnamed: 4": "Type of data",
                           "Unnamed: 5": "b1990-1995",
                           "Unnamed: 6": "b1995-2000",
                           "Unnamed: 7": "b2000-2005",
                           "Unnamed: 8": "b2005-2010",
                           "Unnamed: 9": "b2010-2015",
                           "Unnamed: 10": "m1990-1995",
                          "Unnamed: 11": "m1995-2000",
                          "Unnamed: 12": "m2000-2005",
                          "Unnamed: 13": "m2005-2010",
                          "Unnamed: 14": "m2010-2015",
                          "Unnamed: 15": "f1990-1995",
                          "Unnamed: 16": "f1995-2000",
                          "Unnamed: 17": "f2000-2005",
                          "Unnamed: 18": "f2005-2010",
                          "Unnamed: 19": "f2010-2015",}).
           melt(id_vars = ["Sort Order", "Major area, region, country or area of destination", "Notes", "Country code", "Type of
           drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).
           drop(columns = ['Country code', 'Sort Order']).
           reset_index(drop=True).
           head(4185)
          )
          Table5.head(4185)
```

Figure 14: Data input for tidying Table 5. The functions include the rename columns, melt, drop, drop columns, reset index, and head.

The first 15 rows were dropped for the same reasons as in Table 1, Table 2, and Table 3. The columns dropped, which are “Country code” and “Sort Order”, were dropped for the same reasons in Table 1 and Table 2.

Out[707]:

	Major area, region, country or area of destination	Notes	Type of data	Demographic	Annual Rate of Change of the Migrant Stock
0	WORLD	NaN	NaN	b1990-1995	1.051865
1	Developed regions	(b)	NaN	b1990-1995	2.275847
2	Developing regions	(c)	NaN	b1990-1995	-0.487389
3	Least developed countries	(d)	NaN	b1990-1995	1.118175
4	Less developed regions excluding least develop...	NaN	NaN	b1990-1995	-0.803244
...	...	...	...	...	...
4180	Samoa	NaN	B	f2010-2015	-0.545343
4181	Tokelau	NaN	B	f2010-2015	2.60325
4182	Tonga	NaN	B	f2010-2015	2.526318
4183	Tuvalu	NaN	C	f2010-2015	-1.819436
4184	Wallis and Futuna Islands	NaN	B	f2010-2015	0.516899

4185 rows × 5 columns

Figure 15: Output from the input for Table 5. Prior to the use of the lambda function to separate the Demographic column.

Figure 15 shows the “Demographic” variable, which violates principle 2. This is the column that I intentionally renamed to have the gender and the year in the same cell. “Sex” is represented as b for both sexes, m for male, and f for female. The lambda function, as exhibited in Figure 16, was used to split the “Demographic” variable to “Sex” and “Year”.

In [742]:
```
Table5 = (Table5.assign(Sex = lambda x: x.Demographic.str[0].astype(str),
                        Year = lambda x: x.Demographic.str[1:].astype(str)).drop("Demographic", axis=1))
Table5.head(100)
```

Out[742]:

	Major area, region, country or area of destination	Notes	Type of data	Annual Rate of Change of the Migrant Stock	Sex	Year
0	WORLD	NaN	NaN	1.051865	b	1990-1995
1	Developed regions	(b)	NaN	2.275847	b	1990-1995
2	Developing regions	(c)	NaN	-0.487389	b	1990-1995
3	Least developed countries	(d)	NaN	1.118175	b	1990-1995
4	Less developed regions excluding least develop...	NaN	NaN	-0.803244	b	1990-1995
...	...	...	...	...	...	...
95	Timor-Leste	NaN	B	1.688974	b	1990-1995
96	Viet Nam	NaN	C R	12.010796	b	1990-1995
97	Southern Asia	NaN	NaN	-4.729682	b	1990-1995
98	Afghanistan	NaN	B	4.299812	b	1990-1995
99	Bangladesh	NaN	B R	1.170107	b	1990-1995

Figure 16: Lambda Function and Final Table Output for Table 5.

Table 6

Table 6 is the final sheet in the UN dataset. In Figure 17, the code input of the tidy process is outlined. Principle 1 and principle 2 were violated, which was rectified with the rename function and the melt function. It should be noted that for the melt-function in Figure 17, var_name = “Demographic” and value_name = “Refugee Stock at Mid-Year”.

```
In [711]: Table6 = pd.DataFrame.from_records(
           sheet6.
           rename(columns={"Unnamed: 0": "Sort Order",
                           "Unnamed: 1": "Major area, region, country or area of destination",
                           "Unnamed: 2": "Notes",
                           "Unnamed: 3": "Country code",
                           "Unnamed: 4": "Type of data",
                           "Unnamed: 5": "E1990",
                           "Unnamed: 6": "E1995",
                           "Unnamed: 7": "E2000",
                           "Unnamed: 8": "E2005",
                           "Unnamed: 9": "E2010",
                           "Unnamed: 10": "E2015",
                          "Unnamed: 11": "P1990",
                          "Unnamed: 12": "P1995",
                          "Unnamed: 13": "P2000",
                          "Unnamed: 14": "P2005",
                          "Unnamed: 15": "P2010",
                          "Unnamed: 16": "P2015",
                          "Unnamed: 17": "R1990-1995",
                          "Unnamed: 18": "R1995-2000",
                          "Unnamed: 19": "R2000-2005",
                          "Unnamed: 20": "R2005-2010",
                          "Unnamed: 21": "R2010-2015"}).
           melt(id_vars = ["Sort Order", "Major area, region, country or area of destination", "Notes", "Country code", "Type of
           drop([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).
           drop(columns = ['Country code', 'Sort Order']).
           reset_index(drop=True).
           head(5000)
          )
```

Figure 17: Data input for tidying Table 6. The functions include the rename columns, melt, drop, drop columns, reset index, and head.

Out[711]:

	Major area, region, country or area of destination	Notes	Type of data	Demographic	Refugee Stock at Mid-Year
0	WORLD	NaN	NaN	E1990	18836571
1	Developed regions	(b)	NaN	E1990	2014564
2	Developing regions	(c)	NaN	E1990	16822007
3	Least developed countries	(d)	NaN	E1990	5048391
4	Less developed regions excluding least develop...	NaN	NaN	E1990	11773616
...	...	...	...	...	...
4740	Samoa	NaN	B	R2010-2015	..
4741	Tokelau	NaN	B	R2010-2015	..
4742	Tonga	NaN	B	R2010-2015	..
4743	Tuvalu	NaN	C	R2010-2015	..
4744	Wallis and Futuna Islands	NaN	B	R2010-2015	..

4745 rows × 5 columns

Figure 18: Output from the input for Table 6. Prior to the use of the lambda function to separate the Demographic column.

The first 15 rows were dropped for the same reasons as in Table 1, Table 2, and Table 3. The columns dropped, which are "Country code" and "Sort Order", were dropped for the same reasons in Table 1 and Table 2.

```
In [712]: Table6 = (Table6.assign(Calculation = lambda x: x.Demographic.str[0].astype(str), Year = lambda x: x.Demographic.str[
          Table6.head(100)
```

Out[712]:

	Major area, region, country or area of destination	Notes	Type of data	Refugee Stock at Mid-Year	Calculation	Year
0	WORLD	NaN	NaN	18836571	E	1990
1	Developed regions	(b)	NaN	2014564	E	1990
2	Developing regions	(c)	NaN	16822007	E	1990
3	Least developed countries	(d)	NaN	5048391	E	1990
4	Less developed regions excluding least develop...	NaN	NaN	11773616	E	1990
...	...	...	...	...	...	...
95	Timor-Leste	NaN	B	0	E	1990
96	Viet Nam	NaN	C R	21150	E	1990
97	Southern Asia	NaN	NaN	6906630	E	1990
98	Afghanistan	NaN	B	25	E	1990
99	Bangladesh	NaN	B R	73	E	1990

Figure 19: Lambda Function and Final Table Output for Table 6.

The "Demographic" variable violates principle 2. I purposely renamed the variables under "Demographic" to indicate the form of calculation before the year. Therefore, the lambda function was used to split the variables under "Demographic" to reflect two columns, which are "Calculation" and "Year". It should be noted that the variables under the "Calculation" column, E = estimated, P = percentage, and R = rate of change.

Please refer to the conclusion where I address the limitations I encountered in Table 6.

Conclusion

After cleaning the six tables in the UN dataset, I identified some principle violations that I was unable to address.

I attempted to rename the index function to "Sort Order", but it would give me error codes when attempting to rename it after resetting it. I decided to drop "Sort Order" across all tables because I dropped the first 15 rows, because Sort Order began its counting at 15, which prompted me to drop "Sort Order" all together, and reset the index for all tables. I was attempting to keep the variable name "Sort Order" at the top of the column, but had many difficulties with

this as I have explained. Therefore, I left the index with no name at the top, but as principle 1 suggests, column headers should be variable names.

One limitation that I identified was the violation of principle 2 under the "Type of Data" column. This was tricky considering that there are multiple observational units stored in the cells of "Type of Data", which violates Principle 2. "Type of Data" violates this as B refers to foreign-born population, C refers to foreign citizens, R refers to international migrants, and I refers to international migrants. As the combined variables still fall under "Type of Data", I would be inclined to rename/relabel the combined variables as their own entity. It would not be correct to have multiple tables of "Type of Data" as this would violate principle 4. Ultimately, I left it as-is as I could not find a solution to this, but recognize that it is currently violating principle 2.

In Table 2 of the UN dataset, the "Type of Data" column was not included in this Table. Upon further investigation, I realized that "Type of Data" is the same across all of the tables and recognized that this violates principle 5 and the column could be transferred to Table 2. This was a limitation for me as I could not find the solution to this. I attempted to run a For Loop to extract Type of Data from Table 1 (it could have been any Table, I just chose Table 1 at random), yet I continued to receive error codes. Overall, this violates principle 5, and I wanted to transport the "Type of Data" column from one of the other Tables to Table 2. Other than this situation, I could not locate any other violations to principle 5.

In Table 6, there are three columns "Estimated Refugee Stock at Mid-Year (both sexes)", "Refugees as a Percentage of the International Migrant Stock", and "Annual Rate of Change of the Refugee Stock". My current solution was running the lambda function to separate the calculations and the year. Considering that each of the titles represent different values and

populations, I do not think this is the correct solution. I believe the proper solution would be to split the titles into their three separate columns and have the year in another separate column.

I could not locate any violations to principle 3.