1340 Midterm Project – Write Up

Yuetong Jiang
1004339343

Introduction:

Data preparation and data cleaning are essential parts of data analysis, which provides a robust and clean environment for future data extraction, mining, analysis, and visualization. The data set 'Trends in International Migrant Stock' contains six tables that involved multiple untidy problems such as non-consistent data structures, clustered variables, and observations.

The following five problems can be found in the Migrant Stock dataset and 5 tidy principles will be applied to accommodate the problems:

#	Problems	Principles
1	Column headers are values, not variable names	column headers should be names, no values
2	Multiple variables are stored in one column	each column needs to consist of one and only one variable
3	Variables are stored in both rows and columns	variables need to be in cells, not rows and columns
4	Multiple types of observational units are stored in the same table	each table column needs to have a singular data type
5	A single observational unit is stored in multiple tables	a single observational unit must be in 1 table

Since the overall structures between tables are quite similar, the cleaning process will be organized by tables and some repetitive steps will be compressed as functions and applied when appropriate.

Methods and Results

Parsing Dataset

The excel file containing multiple tabs can be read by the Pandas read_excel function. Since Table names and notes are placed on the top of the data, and the header occupied two rows: row 15 and row 16, for simplicity, we avoid multiple indexes and headers by specifying the header row index as 15 (row 16), and read the rest rows as data. The excel sheets are stored and sorted as a dictionary, where keys are sheet names, and values are data frames.

```
In [2]: UN = 'UN_MigrantStockTotal_2015.xlsx'
        #  read all the tables into pandas dataframes,
        #  row 14 & 15 as row names, for simplicity we use row 15 as header
        #  and read the rest rows as data
        dfs = pd.read_excel(UN, sheet_name=None, header=[15])
```

```
In [3]: # sheets are sorted as a dictionary, where
        #   keys are sheet names
        #   values are data frames
        dfs.keys()
```

```
Out[3]: dict_keys(['CONTENTS', 'Table 1', 'Table 2', 'Table 3', 'Table 4',
        'Table 5', 'Table 6', 'ANNEX', 'NOTES'])
```

Table 1: International Migrant Stock at Mid-Year

Because of the multiple-header in the original file, there will be some columns with no variable names in the data frames, which is shown in Dataframe 1. For clarity, we first give proper names to each variable. Preparing for the later cleaning, we drop non-useful columns and replace the missing value '..' with NaN, which yields Dataframe 2.

```
In [4]: table1 = dfs['Table 1']
        table1.head()
```

Out[4]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	1990	1995	2000	2005	2010	...	2000.1	2005.1	2010.1	
0	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	1261
1	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	676
2	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	584
3	4	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	64
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	56778501	59244124	64272611	79130668	...	31986141	35265888	45069923	520

5 rows × 23 columns

Dataframe 1: first five rows of Table1. Five columns missed variable names.

	Order	Destination	Country Code	1990	1995	2000	2005	2010	2015	1990.1	...	2000.1	2005.1	2010.1
0	1	WORLD	900	152563212.0	160801752.0	172703309.0	191269100.0	221714243	243700236	77747510.0	...	87884839.0	97866674.0	114613714
1	2	Developed regions	901	82378628.0	92306854.0	103375363.0	117181109.0	132560325	140481955	40263397.0	...	50536796.0	57217777.0	64081077
2	3	Developing regions	902	70184584.0	68494898.0	69327946.0	74087991.0	89153918	103218281	37484113.0	...	37348043.0	40648897.0	50532637
3	4	Least developed countries	941	11075966.0	11711703.0	10077824.0	9809634.0	10018128	11951316	5843107.0	...	5361902.0	5383009.0	5462714
4	5	Less developed regions excluding least develop...	934	59105261.0	56778501.0	59244124.0	64272611.0	79130668	91262036	31641006.0	...	31986141.0	35265888.0	45069923

5 rows × 21 columns

Dataframe 2: The first 5 rows of Table1, where first three columns are now assigned with descriptive names.

We can find that the column names are year values but not variable names, where '1990' to '2015' columns represents the international migrant stock for both sexes, '1990.1' to '2015.1' represents the international migrant stock for male, and the rest represents international migrant stock for female. **Since the year variable is spread across columns, by tidy data principle 1, column headers should be names, not values, we need to turn year columns into rows.**

For consistency, we first rename columns for both sexes by adding '.0' at the end of the years, and then we use Pandas melt function to turn columns into rows, which yields Dataframe 3.

	Order	Destination	Country Code	Year	International Migrant Stock
0	1	WORLD	900	1990.0	152563212.0
1	2	Developed regions	901	1990.0	82378628.0
2	3	Developing regions	902	1990.0	70184584.0
3	4	Least developed countries	941	1990.0	11075966.0
4	5	Less developed regions excluding least develop...	934	1990.0	59105261.0
5	6	Sub-Saharan Africa	947	1990.0	14690319.0
6	7	Africa	903	1990.0	15690623.0
7	8	Eastern Africa	910	1990.0	5964031.0
8	9	Burundi	108	1990.0	333110.0
9	10	Comoros	174	1990.0	14079.0

Dataframe 3: The first ten rows of Table 1. The column has been renamed to Year, and value to International Migrant Stock.

However, now the Year column contains not only years but also gender that there are multiple variables stored in 1 column. **By tidy data principle 2: each column needs to consist of one and only one variable, thus we split 'Year' column into 'Year' and 'Gender', which yields Dataframe 4.**

	Order	Destination	Country Code	Year	International Migrant Stock	Gender
0	1	WORLD	900	1990	152563212.0	Both Sexes
1	2	Developed regions	901	1990	82378628.0	Both Sexes
2	3	Developing regions	902	1990	70184584.0	Both Sexes
3	4	Least developed countries	941	1990	11075966.0	Both Sexes
4	5	Less developed regions excluding least develop...	934	1990	59105261.0	Both Sexes

Dataframe 4: The first 5 rows of Table 1. The column Year has been split into Year and renamed to Year and Gender.

Tidy_table Function

We can find the structures of Tables are very similar. For example, Table 2 consists of the same 'Order' 'Destination' and 'Country Code' columns, and the population values are

stored by sex and years. It means that we can define a function tidy_table to organize the steps of data cleaning and enable repetitive use.

Tidy Table Steps:

1. clean Table by replacing missing values with NaN
2. by principle 1, year-related columns to a single variable Year
3. by principle 2, split 'Year' column into 'Year' and 'Gender'

```
# function takes two variables: table t, and a name for the main values stored in this table
#    returns the tidy table, splitted major_area table, splitted countries table
def tidy_table(t, name):
    t = (t.melt(id_vars=["Order","Destination","Country Code"],  # principle 1
                var_name = "Year", value_name = name).
         assign(Gender = lambda x: x.Year.str[-1]))            # principle 2

    # replace gender values to descriptive values
    t['Gender'] = t['Gender'].apply(lambda x: "Male" if (x == "1")
                                    else ("Both Sexes" if (x == "0") else "Female"))
    t = (t.assign(Year = lambda x: x.Year.str[:-2].astype(str)))
    return t
```

Pic 1: Tidy_table function

Table 2: Total population at mid-year (thousands)

As shown in Dataframe 5, We first cleaned Table by giving proper variable names for Table and dropping not useful columns to prepare for using the tidy function. Then we apply tidy_table function onto the dataframe5 and yield the cleaned Table2.

	Order	Destination	Country Code	1990.0	1995.0	2000.0	2005.0	2010.0	2015.0	1990.1	...	2000.1	2005.1	
0	1	WORLD	900	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	2670423.701	...	3084537.662	3285082.249	3493
1	2	Developed regions	901	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	555255.626	...	578010.218	587962.213	599
2	3	Developing regions	902	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	2115168.075	...	2506527.444	2697120.036	2894
3	4	Least developed countries	941	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	254042.556	...	331482.475	375757.715	422
4	5	Less developed regions excluding least develop...	934	3655147.008	3980172.519	4273424.303	4557911.390	4849094.485	5143963.209	1861125.519	...	2175044.969	2321362.321	2471

5 rows × 21 columns

Dataframe 5: First 5 rows in the preliminary cleaned Table 2.

	Order	Destination	Country Code	Year	Total Population (thousands)	Gender
0	1	WORLD	900	1990	5309667.699	Both Sexes
1	2	Developed regions	901	1990	1144463.062	Both Sexes
2	3	Developing regions	902	1990	4165204.637	Both Sexes
3	4	Least developed countries	941	1990	510057.629	Both Sexes
4	5	Less developed regions excluding least develop...	934	1990	3655147.008	Both Sexes

Dataframe 5: First 5 rows in the cleaned Table 2 by tidy function.

Table 3: International Migrant Stock as Percentage of Total Population

Table 3 stores the International migrant stock as a percentage of the total population. Recall that table 1 stores International migrant stock at mid-year and table 2 stores the Total population at mid-year (thousands). We can find that values in table 3 can be easily derived by: table1 dividing Table2. Moreover, values in the three tables are all organized by the same index sets: 'Order', 'Destination', and 'Country Code', which means there are many redundant values stored in those three tables. **By principle 5, a single observational unit must be in 1 table, thus we can combine these three tables.**

In Dataframe 6, we combined Table1 with Table2 by adding a new column ‘Total Population (thousands)’ to Table1. Then we can either add a new column ‘Percentage’ from Table3 to the joined table, or calculate the value manually by International Migrant Stock/(Total Population * 1000) * 100%, columns manipulated for unit integrity. The combined table is shown in Dataframe 7.

	Order	Destination	Country Code	Year	International Migrant Stock	Gender	Total Population (thousands)
0	1	WORLD	900	1990	152563212.0	Both Sexes	5309667.699
1	2	Developed regions	901	1990	82378628.0	Both Sexes	1144463.062
2	3	Developing regions	902	1990	70184584.0	Both Sexes	4165204.637
3	4	Least developed countries	941	1990	11075966.0	Both Sexes	510057.629
4	5	Less developed regions excluding least develop...	934	1990	59105261.0	Both Sexes	3655147.008

Dataframe 6: First 5 rows of the combination of Table1 and Table2

	Order	Destination	Country Code	Year	International Migrant Stock	Gender	Total Population (thousands)	Migrant Percentage of Population
0	1	WORLD	900	1990	152563212.0	Both Sexes	5309667.699	2.873310
1	2	Developed regions	901	1990	82378628.0	Both Sexes	1144463.062	7.198015
2	3	Developing regions	902	1990	70184584.0	Both Sexes	4165204.637	1.685021
3	4	Least developed countries	941	1990	11075966.0	Both Sexes	510057.629	2.171513
4	5	Less developed regions excluding least develop...	934	1990	59105261.0	Both Sexes	3655147.008	1.617042

Dataframe 7: First 5 rows of the joined table (combination to Table1, Table2, Table3)

Furthermore, we can observe that the column 'Destination' contains two types of data: major areas/regions and countries, where major areas have country codes greater or equal to 900, and each individual country has the code less than 900. **According to principle 4: to make sure we have singular data type in one column/table, we need to split them into two tables**.

Before splitting, to be more informative, we can add one more column to store the region of each destination and reorder the columns for clarity.

```
# add a new column region for each country to the table
def region_col(table):
    region = []
    major_areas = []     # to store all unique major_areas
    for i in range(len(table)):
        if table.iloc[i,]["Country Code"] >= 900:
            major_areas.append(table.iloc[i,]["Destination"])  # store the major area
        region.append(major_areas[-1])  # current country's region is the last value in the region list
    table["Region"] = region
    return table
```

Pic 2: Function to add a region column into a table.

	Order	Destination	Region	Country Code	Year	Gender	International Migrant Stock	Total Population (thousands)	Migrant Percentage of Population
166	167	Slovenia	Southern Europe	705	1990	Both Sexes	178077.0	2006.520	8.874918
167	168	Spain	Southern Europe	724	1990	Both Sexes	821605.0	39192.055	2.096356
168	169	The former Yugoslav Republic of Macedonia	Southern Europe	807	1990	Both Sexes	95142.0	1996.227	4.766091
169	170	Western Europe	Western Europe	926	1990	Both Sexes	16237829.0	175615.377	9.246246
170	171	Austria	Western Europe	40	1990	Both Sexes	793239.0	7706.571	10.293021

Dataframe 8: randomly picked 5 consecutive rows in the joined table, after adding a new column 'Region' and reordering columns.

After splitting Table into two independent tables (based on the country code) and reset index, we got Table: tidy_major_area and Table: tidy_countries.

	Order	Destination	Country Code	Year	Gender	International Migrant Stock	Total Population (thousands)	Migrant Percentage of Population
0	1	WORLD	900	1990	Both Sexes	152563212.0	5309667.699	2.873310
1	2	Developed regions	901	1990	Both Sexes	82378628.0	1144463.062	7.198015
2	3	Developing regions	902	1990	Both Sexes	70184584.0	4165204.637	1.685021
3	4	Least developed countries	941	1990	Both Sexes	11075966.0	510057.629	2.171513
4	5	Less developed regions excluding least develop...	934	1990	Both Sexes	59105261.0	3655147.008	1.617042

Dataframe 9: first 5 rows in the tidy_major_area table, after dropping the column region which is exactly the same as the destination column.

	Order	Destination	Region	Country Code	Year	Gender	International Migrant Stock	Total Population (thousands)	Migrant Percentage of Population
0	9	Burundi	Eastern Africa	108	1990	Both Sexes	333110.0	5613.141	5.934467
1	10	Comoros	Eastern Africa	174	1990	Both Sexes	14079.0	415.144	3.391353
2	11	Djibouti	Eastern Africa	262	1990	Both Sexes	122221.0	588.356	20.773307
3	12	Eritrea	Eastern Africa	232	1990	Both Sexes	11848.0	3139.083	0.377435
4	13	Ethiopia	Eastern Africa	231	1990	Both Sexes	1155390.0	48057.094	2.404203

Dataframe 10: first 5 rows in Table tidy_countries, after resetting index

Table 4: Female Migrants as Percentage of International Migrant stock

Similarly, we cleaned Table by dropping non-useful columns, adding appropriate column names, and replacing "..'' with NaN. Since Table only contains one gender, we can drop the gender column obtained by the tidy_table() function.

	Order	Destination	Country Code	Year	Female Migrants Percentage
0	1	WORLD	900	1990	49.03915
1	2	Developed regions	901	1990	51.123977
2	3	Developing regions	902	1990	46.592099
3	4	Least developed countries	941	1990	47.261155
4	5	Less developed regions excluding least develop...	934	1990	46.466684

Dataframe 11: first 5 rows in cleaned Table 4.

Table 5: Annual rate of change of the migrant stock (by sex)

Table 5 contains the annual rate of change of the migrant stock, thus the year column contains not a single value but a range.

	Order	Destination	Country Code	Year	Annual Rate of Change of The Migrant Stock	Gender
0	1	WORLD	900	1990-1995	1.051865	Both Sexes
1	2	Developed regions	901	1990-1995	2.275847	Both Sexes
2	3	Developing regions	902	1990-1995	-0.487389	Both Sexes
3	4	Least developed countries	941	1990-1995	1.118175	Both Sexes
4	5	Less developed regions excluding least develop...	934	1990-1995	-0.803244	Both Sexes

Dataframe 12: first 5 rows in cleaned Table 5.

Since each country/region's annual rate of change has three categories 'Both sexes', 'Male' and 'Female'. For clarity, we can also convert the gender column into three individual columns by principle 3.

	Order	Destination	Country Code	Year	Both Sexes	Female	Male
0	1	WORLD	900	1990-1995	1.051865	1.104667	1.000922
1	1	WORLD	900	1995-2000	1.428058	1.405044	1.450294
2	1	WORLD	900	2000-2005	2.042124	1.928080	2.151575
3	1	WORLD	900	2005-2010	2.954160	2.737012	3.159228
4	1	WORLD	900	2010-2015	1.890991	1.867837	1.912603

Dataframe 13: first 5 rows in cleaned Table 5 which split the gender column into three columns, each contains the annual rate of change of migrant stock for each gender.

Table 6: Estimated refugee stock at mid-year

Table 6 is slightly different from the previous tables, which contains multiple types of data:

- Column '1990.0' to '2015.0' represents Estimated refugee stock at mid-year (both sexes).
- Column '1990.1' to '2015.1' represents Refugees as a percentage of the international migrant stock.
- Column '1990-1995' to '2010-2015' represents Annual rate of change of the refugee stock.

	Order	Destination	Country Code	1990.0	1995.0	2000.0	2005.0	2010.0	2015.0	1990.1	1995.1	2000.1	2005.1	2010.1
0	1	WORLD	900	18836571.0	17853840.0	15827803.0	13276733.0	15370755	19577474	12.346732	11.103013	9.164736	6.941389	6.932687
1	2	Developed regions	901	2014564.0	3609670.0	2997256.0	2361229.0	2046917	1954224	2.445494	3.910511	2.899391	2.015025	1.544140
2	3	Developing regions	902	16822007.0	14244170.0	12830547.0	10915504.0	13323838	17623250	23.968236	20.795958	18.507035	14.733162	14.944759
3	4	Least developed countries	941	5048391.0	5160131.0	3047488.0	2363782.0	1957884	3443582	45.565880	44.041961	30.221557	24.082430	19.533425
4	5	Less developed regions excluding least develop...	934	11773616.0	9084039.0	9783059.0	8551722.0	11365954	14179668	19.919743	15.999082	16.513130	13.305391	14.363526

Dataframe 14: first five rows of Table 6.

Thus, by principle 4, we need to split it into several tables and tidy them using the principle 1 and principle 2.

	Order	Destination	Country Code	Year	Refugee Stock
0	1	WORLD	900	1990	18836571.0
1	2	Developed regions	901	1990	2014564.0
2	3	Developing regions	902	1990	16822007.0
3	4	Least developed countries	941	1990	5048391.0
4	5	Less developed regions excluding least develop...	934	1990	11773616.0

Dataframe 15: first five rows of Table Refugee_Stock.

	Order	Destination	Country Code	Year	Refugee Percentage
0	1	WORLD	900	1990	12.346732
1	2	Developed regions	901	1990	2.445494
2	3	Developing regions	902	1990	23.968236
3	4	Least developed countries	941	1990	45.565880
4	5	Less developed regions excluding least develop...	934	1990	19.919743

Dataframe 16: first five rows of Table Refugee_Percentage.

	Order	Destination	Country Code	Year	Rate of Change
0	1	WORLD	900	1990-1995	-2.123497
1	2	Developed regions	901	1990-1995	9.388424
2	3	Developing regions	902	1990-1995	-2.839417
3	4	Least developed countries	941	1990-1995	-0.680327
4	5	Less developed regions excluding least develop...	934	1990-1995	-4.383600

Dataframe 17: first five rows of Table Rate_of_Change.

We can find that the Refugees_Percentage tables shares some values with Table 4 (Female migrants as a percentage of the international migrant stock), therefore we can also join this table with table4. Similarly, the Rate_of_Change table also shares some commons with table 5 (Annual Rate of Change of the Migrant Stock by Sex). Thus we can join these two tables into one single table.

	Order	Destination	Country Code	Year	Female Migrants Percentage	Refugee Percentage
0	1	WORLD	900	1990	49.03915	12.346732
1	2	Developed regions	901	1990	51.123977	2.445494
2	3	Developing regions	902	1990	46.592099	23.968236
3	4	Least developed countries	941	1990	47.261155	45.565880
4	5	Less developed regions excluding least develop...	934	1990	46.466684	19.919743

Dataframe 18: first five rows of Table 4, which was added a new column for refugee migrants percentage of total population.

	Order	Destination	Country Code	Year	Both Sexes	Female	Male	Refugees
0	1	WORLD	900	1990-1995	1.051865	1.104667	1.000922	-2.123497
1	1	WORLD	900	1995-2000	1.428058	1.405044	1.450294	-3.837069
2	1	WORLD	900	2000-2005	2.042124	1.928080	2.151575	-5.557223
3	1	WORLD	900	2005-2010	2.954160	2.737012	3.159228	-0.025089
4	1	WORLD	900	2010-2015	1.890991	1.867837	1.912603	2.947267

Dataframe 19: first five rows of Table5, which was added a new column Refugee for the annual rate of change in refugee migrant stock.

	Order	Destination	Country Code	Year	Refugee Stock
0	1	WORLD	900	1990	18836571.0
1	2	Developed regions	901	1990	2014564.0
2	3	Developing regions	902	1990	16822007.0
3	4	Least developed countries	941	1990	5048391.0
4	5	Less developed regions excluding least develop...	934	1990	11773616.0

Dataframe 20: first five rows of Table6, left a column for refugee stock

Discussion:

Through the process of cleaning the Migrant Stock dataset, it's found that the logic and steps of data cleaning matter when dealing with a group of tables. In order to join the tables with the same structures and contents into a single table and reduce redundant information, it's important to keep the results consistent and robust, with high resilience to change. For example, we can find that data stored in table 3 can be easily derived from table 1 and table 2. Then it's intuitive to combine those two tables to link and illustrate that relationship. However, the structures and format of table 1 and table 2 were different at one point in the cleaning process, which brought much more workload as we need to reorganize all steps and conduct the analysis again, with the purpose of merging and concatenating. It may be more efficient when we first merge tables with similar contents and then clean the whole dataset. Similarly, Table 4 contains the data that can be derived directly from Table1, by using the female migrant stock / total migrant stock. However, adding such a percentage column to Table 1 will greatly influence the table structure and make it hard to understand. One solution is to simply remove this table as we can obtain the data quickly when needed, and the other solution is to combine Table 4 with Percentage-related columns in Table 6, as we showed in the previous section.

Conclusion:

By the end of data cleaning, we have cleaned the original 6 tables into the following tables:

	Order	Destination	Region	Country Code	Year	Gender	International Migrant Stock	Total Population (thousands)	Migrant Percentage of Population
0	1	WORLD	WORLD	900	1990	Both Sexes	152563212.0	5309667.699	2.873310
1	2	Developed regions	Developed regions	901	1990	Both Sexes	82378628.0	1144463.062	7.198015
2	3	Developing regions	Developing regions	902	1990	Both Sexes	70184584.0	4165204.637	1.685021
3	4	Least developed countries	Least developed countries	941	1990	Both Sexes	11075966.0	510057.629	2.171513
4	5	Less developed regions excluding least develop...	Less developed regions excluding least develop...	934	1990	Both Sexes	59105261.0	3655147.008	1.617042

Joined_table: combination of Table1, Table2, and Table3

	Order	Destination	Country Code	Year	Female Migrants Percentage	Refugee Percentage
0	1	WORLD	900	1990	49.03915	12.346732
1	2	Developed regions	901	1990	51.123977	2.445494
2	3	Developing regions	902	1990	46.592099	23.968236
3	4	Least developed countries	941	1990	47.261155	45.565880
4	5	Less developed regions excluding least develop...	934	1990	46.466684	19.919743

Table4: combination of Table 4 and the refugee percentage columns in Table 6

	Order	Destination	Country Code	Year	Both Sexes	Female	Male	Refugees
0	1	WORLD	900	1990-1995	1.051865	1.104667	1.000922	-2.123497
1	1	WORLD	900	1995-2000	1.428058	1.405044	1.450294	-3.837069
2	1	WORLD	900	2000-2005	2.042124	1.928080	2.151575	-5.557223
3	1	WORLD	900	2005-2010	2.954160	2.737012	3.159228	-0.025089
4	1	WORLD	900	2010-2015	1.890991	1.867837	1.912603	2.947267

Table5: combination of Table 5 and the refugee annual rate of change in Table 6

	Order	Destination	Country Code	Year	Refugee Stock
0	1	WORLD	900	1990	18836571.0
1	2	Developed regions	901	1990	2014564.0
2	3	Developing regions	902	1990	16822007.0
3	4	Least developed countries	941	1990	5048391.0
4	5	Less developed regions excluding least develop...	934	1990	11773616.0

Table 6: original Table 6 only lefting the refugee stock column.

For the future uses, we can export the result into an excel file by using the ExcelWriter as below:

```
with pd.ExcelWriter('~/Desktop/tidy_tables.xlsx') as writer:
    joined_table.to_excel(writer, sheet_name='Migrant Stock and Population')
    table4.to_excel(writer, sheet_name='Female and Refugee Percentage')
    table5.to_excel(writer, sheet_name='Annual Rate of Change')
    table6.to_excel(writer, sheet_name='Estimate Refugee Stock')
writer.save()
```