Ryan Spencer
INF1340
Tidy Data Principals
UN Migrant Stock 2015

The United Nations, Population Division, Dept. of Economic and Social Affairs: Trends in International Migrant Stock: The 2015 Revision dataset presents several issues that conflict with Tidy Data principals. At first blush, this dataset seems more appropriate as an internal document that is physically printed and distributed in a meeting. The dataset contains six tables (and two reference tables) with varying information pertaining to migrant stock. Special care must be taken as this data reflects especially vulnerable people that have fled due to war, famine, natural disasters, and general economic strife. My approach applying Tidy Data principals is to transform this dataset into a machine-readable document for the purpose of visualizing the data. I do not intend for this data set to be printed and distributed internally like in its current tabular state. Rather, I will be cleaning and organizing the data in hopes it will be utilized by a visualization tool like Tableau or PowerBI. What I am concerned with this dataset is for the user to be able to group and sum totals of migratory patterns. This is the most important information being displayed by this dataset.

Furthermore, the Annex Table located in the dataset is meant as a codebook or guidebook for the users. It is up to the user to manipulate the data how they see fit utilizing said Annex. I will not be altering the Annex or Notes tables.

The Tidy Data principals that I will be applying are as follows:

1. Every column is a variable.
2. Every row is an observation.
3. Every cell is a single value.
4. Column headers are values, not variable names.
5. Multiple variables are NOT stored in one column.
6. Variables are NOT stored in both rows and columns.
7. Multiple types of observational units are NOT stored in the same table.
8. A single observational unit is NOT stored in multiple tables.

<u>Major Points of Contention with Data Set</u>

First, we have column headers that are values and multiple variables in columns. Across all the tables, the instances of years need to be shifted into a single column. Countries, Regions, et al. needs to be broken up into discrete columns, each being placed in their own column.

The use of gender is complex. On one hand, gender should not be used. Sex indicates a biological state rather than a construction like gender. However, rather than omitting completely, it is important to note that women are vulnerable persons that should be accounted for. And for this reason, I am keeping sex as a column to reflect this sensitive issue.

Developed or less developed countries will be omitted in this table. All these tables are trying to convey migratory patterns of human beings. As such, physical geographical boundaries are the major variables to be investigated and grouped. “Developed” and “Less Developed” countries indicate GDP values that have very little to do with migratory stocks. Furthermore, this information exists in the Annex which can be crossed referenced with the country codes by the user.

I am also omitting World from the rows as this is redundant. The sums of all countries will give us this total. Again, the user will have the ability to show the total World migrant sum by selecting all the countries in Tableau or PowerBI. It would also be silly to put World in its own column – is there another World variable that we are not aware of….? Mars? For those reasons, I will be omitting World.

I am omitting “Order”, “Notes”, and “Type of Data” from the dataset. These columns bring no unique values to the dataset and represent more policy issues for the UN. This information also exists in the Note and Annex tables.

In terms of the text, I will be changing everything to lower case with no spaces, using underscores instead. From my research, this is good practice for machine readable text. Again, my goal is to make this dataset for a program.

Table 6 needs to be split into three discrete tables (Table 6A, Table 6B, Table 6C). Currently, Table 6 has three different units of measurement for migrant stock. Following the tidy data principals of only having one unit per table means I need to separate this table.

<u>Table 1: International migrant stock at mid-year by sex and by major area, region, country or area, 1990-2015</u>

First, I would like to see the shape of this table. I will use this information shortly as a reference point to see if my code is changing this table from “wide format” to “long format”.

```
In [449]: #How many rows and columns are there in df1 = Table 1
          df1.shape

Out[449]: (280, 23)
```

At the top of this table is a vanity header that is not needed for our dataset. In a few steps, I will assign column names, so I will be removing this information from the excel spreadsheet.

```
In [450]: #Let's start off easy, Column 1 to Row 15 should be omitted. This is the vanity title of dataset and not necessary
          df1_clean = df1.drop(df1.index[:15])
          print(df1_clean)
```

Python does not recognize the originally assigned columns so that's why I am deleting it. It's a clean slate to work from and build a tidier dataset.

There is an issue with the column: Major area, region, country or area of destination and the Gender and Year columns. There are too many variables for these columns. We need to split them up.

But first, I assign names for all the columns. At this point, I do not want to delete any columns just yet. I'd rather wait and see how this table shapes up once I assign names to the columns. This ensures I place the right name to the right column. I have added a suffix of: S, M, F representing Both, Male, and Female to all the Year columns. This will help me to create a new column for Sex in a few more steps.

```
In [451]: #Since I cleaned up the vanity cells in the excel file, I know need to assign names to the columns.
          #I am using S= Same Sex, M = Male, F= Female. I will separate by gender in a big. I just want to melt the years right nc
          df1_clean.rename(columns={df1.columns[0]:"Order", df1.columns[1]:"Major area, region, country or area of destination",
                                    df1.columns[2]:"Notes",df1.columns[3]:"Country Code",df1.columns[4]:"Type of Data",
                                    df1.columns[5]:"1990S",df1.columns[6]:"1995S",df1.columns[7]:"2000S",df1.columns[8]:"2005S",df1.colu
                                    df1.columns[10]:"2015S",df1.columns[11]:"1990M",df1.columns[12]:"1995M",df1.columns[13]:"2000M",df1.
                                    df1.columns[17]:"1990F",df1.columns[18]:"1995F",df1.columns[19]:"2000F",df1.columns[20]:"2005F",df1.
```

Next, I want to check that I correctly assigned the columns. My favorite function during this project has been head(). It's a wonderful quick reference that helped me navigate all the moving parts and to check if my code is working. In this case, my code worked well.

```
#All right, let's see if it worked by looking at the column index.
#It did work. Good job, Ryan. You can do this!
df1_clean.columns
```

```
Index(['Order', 'Major area, region, country or area of destination', 'Notes',
       'Country Code', 'Type of Data', '1990S', '1995S', '2000S', '2005S',
       '2010S', '2015S', '1990M', '1995M', '2000M', '2005M', '2010M', '2015M',
       '1990F', '1995F', '2000F', '2005F', '2010F', '2015F'],
      dtype='object')
```

click to scroll output; double click to hide

```
#But let's just check the head and tails to see if the values match the excel spread sheet.
#World > 1990S > 152,363,212 is a match!
df1_clean.head()
```

Next, I am using the melt function. I have fixed my constant columns below and will 'melt' International migrant stock at mid-year.

```
dtype='object')
```

```
In [456]: # Order, Major are, et al. are what I want to keep constant.
          #I am not naming all of the years as those are the variables that I would like to melt.
          #Value_name: Int. Migrant Stock is the specific data that the UN is gathering in the dataset.
          #It's important to get this right!
          df1_melt =(df1_clean.melt(id_vars = ['Order', 'Major area, region, country or area of destination', 'Notes',
                 'Country Code', 'Type of Data'], value_name = "International migrant stock at mid-year"))
```

```
In [457]: # Y'all, did it work? It looks like it. I am happy with these column names.
          df1_melt.columns
```

```
Out[457]: Index(['Order', 'Major area, region, country or area of destination', 'Notes',
                 'Country Code', 'Type of Data', 'variable',
                 'International migrant stock at mid-year'],
                dtype='object')
```

Per below, the melt worked as the shape of this table has changed considerably by adding about 4,000 more rows.

```
In [459]: #But to be sure let's make sure this melt, melted properly.
          #Yup, it did! We went from a wide to a long dataset as displayed by the exponential growth of the rows.
          #I am tidying this data!
          df1_melt.shape
```

```
Out[459]: (4770, 7)
```

When I melted, the function created a new column called ‘variable’ that held all the different years. I renamed this column to ‘Year.”

```
df1_named
```

```
[461]:
```

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year
0	1	WORLD	NaN	900	NaN	1990S	152563212
1	2	Developed regions	(b)	901	NaN	1990S	82378628
2	3	Developing regions	(c)	902	NaN	1990S	70184584
3	4	Least developed countries	(d)	941	NaN	1990S	11075966
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990S	59105261
...	...	...	...	...	...	...	...
4765	261	Samoa	NaN	882	B	2015F	2460.0
4766	262	Tokelau	NaN	772	B	2015F	254.0
4767	263	Tonga	NaN	776	B	2015F	2604.0
4768	264	Tuvalu	NaN	798	C	2015F	63.0
4769	265	Wallis and Futuna Islands	NaN	876	B	2015F	1411.0

click to scroll output; double click to hide

4770 rows × 7 columns

The next step is me separating the suffixes that I added to the years. I used the lambda function to move the suffix into a new column called Gender.

4770 rows × 7 columns

```
#Using the lambda function and assigning by string index I am able to remove the S, M, or F variable from year.

df1_sex =(df1_named.assign(Gender = lambda x: x.Year.str[4].astype(str), Year = lambda x: x.Year.str[:4].astype(str)))
df1_sex.head()
```

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year	Gender
0	1	WORLD	NaN	900	NaN	1990	152563212	S
1	2	Developed regions	(b)	901	NaN	1990	82378628	S
2	3	Developing regions	(c)	902	NaN	1990	70184584	S
3	4	Least developed countries	(d)	941	NaN	1990	11075966	S
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990	59105261	S

I then renamed the suffixes to Both, Male, and Female.

```
#I then will take these Sex variables, rename them as S = Both, M = Male, F = Female
df1_sex_both = (df1_sex.replace(to_replace =["S","M","F"],value =["Both","Male","Female"]))
```

```
#Great! We have a specific column for sex and specific observation that is unique to every row.
#Say it with me, we are tidying this data!
df1_sex_both
```

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year	Gender
0	1	WORLD	NaN	900	NaN	1990	152563212	Both
1	2	Developed regions	(b)	901	NaN	1990	82378628	Both
2	3	Developing regions	(c)	902	NaN	1990	70184584	Both
3	4	Least developed countries	(d)	941	NaN	1990	11075966	Both
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990	59105261	Both
...	...	...	...	...	...	...	...	...

I have decided to drop WORLD, developed regions, Developing regions, Least developed countries, Less developed regions excluding least developed countries, and Sub-Saharan Africa. The reason is because these are representing groups of countries and their corresponding sums. If Tableau or Powerbase is used to view this dataset, the user could simply group the corresponding countries themselves utilizing the Annex table as a reference. Furthermore, Sub-Saharan Africa is redundant as Northern Africa is already a region listed that I am keeping.

My goal is to make this dataset as simple as possible so that a user can manipulate the data how they see fit using a visualization tool.

In [469]:
```
#After some debate, I decided that Major Area is not necessary because the information is redundant.
#I rather add a new column that notes where each country is located i.e. what continent. This conveys the same informat
#that is more machine readable.
df1_dropped = df1_new_column.drop(df1_new_column.index[:8])
df1_dropped
```

Out[469]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year	Gender
8	9	Burundi	NaN	108	B R	1990	333110	Both
9	10	Comoros	NaN	174	B	1990	14079	Both
10	11	Djibouti	NaN	262	B R	1990	122221	Both
11	12	Eritrea	NaN	232	I	1990	11848	Both
12	13	Ethiopia	NaN	231	B R	1990	1155390	Both
...	...	...	...	...	...	...	...	...
4765	261	Samoa	NaN	882	B	2015	2460.0	Female
4766	262	Tokelau	NaN	772	B	2015	254.0	Female

Next, I have grouped all the countries to their corresponding continents. Again, this will help with data manipulation when visualizations are needed. The user will be able to quickly group countries, continents, or sums of migratory patterns via continents. This also fixes the problem having multiple variables in one column.

```
df1_dropped['continent'] = df1_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else '

df1_dropped.head(25)
```

t[470]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year	Gender	continent
8	9	Burundi	NaN	108	B R	1990	333110	Both	Africa
9	10	Comoros	NaN	174	B	1990	14079	Both	Africa
10	11	Djibouti	NaN	262	B R	1990	122221	Both	Africa
11	12	Eritrea	NaN	232	I	1990	11848	Both	Africa
12	13	Ethiopia	NaN	231	B R	1990	1155390	Both	Africa

I removed all the regions from the country_or_area_of_destination column. This contradicts the tidy data principal of keeping only one variable per column.

```
'Australia and New Zealand',
'Melanesia',
'Melanesia',
'Polynesia',
'Micronesia',
'WORLD',
'Developed regions',
'Developing regions',
'Least developed countries',
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df1_remove_area.head(25)
```

471]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year	Gender	continent
8	9	Burundi	NaN	108	B R	1990	333110	Both	Africa
9	10	Comoros	NaN	174	B	1990	14079	Both	Africa
10	11	Djibouti	NaN	262	B R	1990	122221	Both	Africa
11	12	Eritrea	NaN	232	I	1990	11848	Both	Africa
12	13	Ethiopia	NaN	231	B R	1990	1155390	Both	Africa

Using a lambda if else function, I then regrouped all the major_regions and placed them into their own column.

```
'Western Europe' if x in western_europe else
'Caribbean' if x in caribbean else
'Central America' if x in central_america else
'South America' if x in south_america else
'Northern America' if x in northern_america else
'Australia and New Zealand' if x in australia_and_new_zealand else
'Melanesia' if x in melanesia else
'Micronesia' if x in micronesia else
'Polynesia' if x in polynesia else
'none')

df1_region.head(25)
```

Out[472]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock at mid-year	Gender	continent	major_region
8	9	Burundi	NaN	108	B R	1990	333110	Both	Africa	Eastern Africa
9	10	Comoros	NaN	174	B	1990	14079	Both	Africa	Eastern Africa
10	11	Djibouti	NaN	262	B R	1990	122221	Both	Africa	Eastern Africa
11	12	Eritrea	NaN	232	I	1990	11848	Both	Africa	Eastern Africa
12	13	Ethiopia	NaN	231	B R	1990	1155390	Both	Africa	Eastern Africa

I cleaned up the column names to be more machine readable i.e. no spaces and all lower cases.

In [473]:
```
#I want to rename the columns into something more machine readable and change gender to sex as gender is a construct.

df1_new_name = df1_remove_area

df1_new_name.rename(columns={'Notes' : 'notes', 'Country Code' : 'country_code', 'Type of Data' : 'type_of_data',
                             'Year' : 'year', 'International migrant stock at mid-year' :'international_migrant_stoc
                             'Gender' : 'sex',}, inplace=True)
```

I reordered the columns to move continent and major_region closer to country. These variables influence one another when sorting. This will help the user navigate the information better when they import into a visualizing tool.

In [474]:
```
#I would like to move continent closer to country_area_of_destination

#Move last Column to 3rd Column
new_cols = ['order', 'country_or_area_of_destination','continent', 'major_region', 'notes', 'country_code', 'type_of_da

df1_new_order=df1_new_name.reindex(columns=new_cols)
df1_new_order.head()
```

Out[474]:

	order	country_or_area_of_destination	continent	major_region	notes	country_code	type_of_data	year	international_migrant_stock_at_mid_year	sex
8	NaN	Burundi	Africa	Eastern Africa	NaN	108	B R	1990	333110	Both
9	NaN	Comoros	Africa	Eastern Africa	NaN	174	B	1990	14079	Both
10	NaN	Djibouti	Africa	Eastern Africa	NaN	262	B R	1990	122221	Both

I decided to drop order, notes, type of data. Order is redundant since I have kept country_code. The user could look up a countries information via the Annex by looking up its corresponding country code. Type of data is extraneous and not necessary for visualization proposes. This deals mainly with policy issues that cannot be reflected in the dataset. Again, this is something that can be viewed by the user in the Annex table. Notes is the same issue. Extraneous information that doesn't support the integer values of the migrant sums.

In [475]:
```
#On second thought the order is arbitrary at this point. It's extraneous and not tidy.
df1_drop_order = df1_new_order.drop(['order', 'notes', 'type_of_data'], axis = 1)
df1_drop_order.head()
```

Out[475]:

	country_or_area_of_destination	continent	major_region	country_code	year	international_migrant_stock_at_mid_year	sex
8	Burundi	Africa	Eastern Africa	108	1990	333110	Both
9	Comoros	Africa	Eastern Africa	174	1990	14079	Both
10	Djibouti	Africa	Eastern Africa	262	1990	122221	Both

I reset the index since the pervious index was resorted and contained omissions when I removed rows. I then resorted the table based on country as this seems to be the most pertinent information to the table.

```
In [478]: #reset index
df1_sort_country = df1_sort_country.reset_index()
```

```
In [479]:  #df1_drop_index = df1_sort_country.drop("index", axis =1)
df1_drop_index = df1_sort_country.drop("index", axis = 1)

#print(df1_sort_country.columns)
```

```
In [480]: #1) Concise columns with one observed variable. Check!
#2) Each row shows a unique measurment. Check!
#3) We are clear of redundancies. Check!
df1_drop_index.head()
```

Out[480]:

	country_or_area_of_destination	continent	major_region	country_code	year	international_migrant_stock_at_mid_year	sex
0	Afghanistan	Asia	Southern Asia	4	1995	39105	Male
1	Afghanistan	Asia	Southern Asia	4	2015	193445.0	Male

Table 2 - Total population at mid-year by sex and by major area, region, country or area, 1990-2015 (thousands)

This table contains the same information as Table 1, however the units of measurement have changed to “thousands.” In keeping with the Tidy Data principals, I am keeping Table 2 discrete from Table 1. For the sake of brevity, I utilized the same code from Table 1 to Table 2. I will only show the different code I used for this specific Table.

I renamed the columns. This Table contained one less column than Table 1. Table 2 does not include Type of Data – this only supports my argument for omitting this column from Table 1 (and subsequent Tables).

```
In [643]: #Since I cleaned up the vanity cells in the excel file, I now need to assign names to the columns.
#I am using S= Same Sex, M = Male, F= Female. I will separate by gender in a big. I just want to melt the years right n
df2_new_columns = df2_clean.rename(columns= {df2.columns[0]:"order", df2.columns[1]:"Major area, region, country or are
                    df2.columns[2]:"Notes",df2.columns[3]:"Country Code",df2.columns[4]:"1990S",df2.columns[5]:"1995S",
                    df2.columns[9]:"2015S",df2.columns[10]:"1990M",df2.columns[11]:"1995M",df2.columns[12]:"2000M",df2.
                    df2.columns[16]:"1990F",df2.columns[17]:"1995F",df2.columns[18]:"2000F",df2.columns[19]:"2005F",df2

df2_new_columns.head()
```

Out[643]:

der	Major area, region, country or area of destination	Notes	Country Code	1990S	1995S	2000S	2005S	2010S	2015S	...	2000M	2005M	2010M	
1	WORLD	NaN	900	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	3493956.904	3707
2	Developed regions	(b)	901	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	599955.476	609
3	Developing regions	(c)	902	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	2894001.428	3097
	Least													

Per below, I melted Table 2 like Table 1 and created a Gender column so that every column has only one variable.

```
In [488]: #Let's use the rename function and assign by column
df2_named = df2_melt.rename(columns={df2_melt.columns[4]:"Year"})
df2_named.head()
```

Out[488]:

	order	Major area, region, country or area of destination	Notes	Country Code	Year	International migrant stock at mid-year (thousands)
0	1	WORLD	NaN	900	1990S	5309667.699
1	2	Developed regions	(b)	901	1990S	1144463.062
2	3	Developing regions	(c)	902	1990S	4165204.637
3	4	Least developed countries	(d)	941	1990S	510057.629
4	5	Less developed regions excluding least develop...	NaN	934	1990S	3655147.008

click to scroll output; double click to hide

```
In [489]: #Using the lambda function and assigning by string index I am able to remove the S, M, or F variable from year.

df2_sex =(df2_named.assign(Sex = lambda x: x.Year.str[4].astype(str), Year = lambda x: x.Year.str[:4].astype(str)))
df2_sex.head()
```

Out[489]:

	order	Major area, region, country or area of destination	Notes	Country Code	Year	International migrant stock at mid-year (thousands)	Sex
0	1	WORLD	NaN	900	1990	5309667.699	S
1	2	Developed regions	(b)	901	1990	1144463.062	S
2	3	Developing regions	(c)	902	1990	4165204.637	S

Created a column called continent to keep every column to one variable.

```
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df2_remove_area.head(25)
```

[495]:

	order	country_or_area_of_destination	Notes	Country Code	Year	International migrant stock at mid-year (thousands)	Sex	continent
8	9	Burundi	NaN	108	1990	5613.141	Both	Africa
9	10	Comoros	NaN	174	1990	415.144	Both	Africa
10	11	Djibouti	NaN	262	1990	588.356	Both	Africa

I removed all the major regions from country_or_area_of_destination. This major region will be placed in their own column.

```
df2_dropped['continent'] = df2_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa el

df2_dropped.head(25)
```

493]:

	order	country_or_area_of_destination	Notes	Country Code	Year	International migrant stock at mid-year (thousands)	Sex	continent
8	9	Burundi	NaN	108	1990	5613.141	Both	Africa
9	10	Comoros	NaN	174	1990	415.144	Both	Africa
10	11	Djibouti	NaN	262	1990	588.356	Both	Africa
11	12	Eritrea	NaN	232	1990	3139.083	Both	Africa

click to scroll output; double click to hide

Using the lambda function with an if/else, I placed all the countries to their corresponding major regions.

```
eastern_africa else
```

t[496]:

	order	country_or_area_of_destination	Notes	Country Code	Year	International migrant stock at mid-year (thousands)	Sex	continent	major_region
8	9	Burundi	NaN	108	1990	5613.141	Both	Africa	Eastern Africa
9	10	Comoros	NaN	174	1990	415.144	Both	Africa	Eastern Africa
10	11	Djibouti	NaN	262	1990	588.356	Both	Africa	Eastern Africa
11	12	Eritrea	NaN	232	1990	3139.083	Both	Africa	Eastern Africa
12	13	Ethiopia	NaN	231	1990	48057.094	Both	Africa	Eastern Africa
13	14	Kenya	NaN	404	1990	23446.229	Both	Africa	Eastern Africa
14	15	Madagascar	NaN	450	1990	11545.782	Both	Africa	Eastern Africa
15	16	Malawi	NaN	454	1990	9408.998	Both	Africa	Eastern Africa
16	17	Mauritius	(1)	480	1990	1055.865	Both	Africa	Eastern Africa
17	18	Mayotte	NaN	175	1990	94.78	Both	Africa	Eastern Africa

I dropped Notes and order. They were both extraneous and provided no insight or integer value to the migrant stock. I reordered the columns to better reflecting the grouping of country, continent, and major region. I also changed the gender column to sex as this better represents people's identity since gender can be interpreted as a construct.

4	4603	Afghanistan	Asia	Southern Asia	4	2015	15752.858	Female

In [504]:
```
df2_drop_index = df2_reset_index.drop("index", axis = 1)
df2_drop_index.head()
```

Out[504]:

	country_or_area_of_destination	continent	major_region	country_code	year	international_migrant_stock_at_mid_year_as_thousands	sex
0	Afghanistan	Asia	Southern Asia	4	1995	8682.442	Male
1	Afghanistan	Asia	Southern Asia	4	2015	16773.704	Male
2	Afghanistan	Asia	Southern Asia	4	2000	9555.403	Female
3	Afghanistan	Asia	Southern Asia	4	2005	11783.622	Female
4	Afghanistan	Asia	Southern Asia	4	2015	15752.858	Female

<u>Table 3 - International migrant stock as a percentage of the total population (both sexes)</u>

Like Table 2, the information is the same to Table 1, however the values are different. Table 3 gives percentages of migrant stock. Again, abiding by Tidy Data principals, I will be keeping Table 3 discrete from Table 1 and Table 2 since the values are different.

I assigned names to the columns. I used the suffix of S, M, and F again to create a new sex column.

In [511]:
```
#Since I cleaned up the vanity cells in the excel file, I now need to assign names to the columns.
#I am using S= Both Sexes, M = Male, F= Female. I will separate by gender in a bit. I just want to melt the years right
df3_clean.rename(columns={df3.columns[0]:"Order", df3.columns[1]:"Major area, region, country or area of destination",
                          df3.columns[2]:"Notes",df3.columns[3]:"Country Code",df3.columns[4]:"Type of Data",
                          df3.columns[5]:"1990S",df3.columns[6]:"1995S",df3.columns[7]:"2000S",df3.columns[8]:"2005S",df3.col
                          df3.columns[10]:"2015S",df3.columns[11]:"1990M",df3.columns[12]:"1995M",df3.columns[13]:"2000M",df3
                          df3.columns[17]:"1990F",df3.columns[18]:"1995F",df3.columns[19]:"2000F",df3.columns[20]:"2005F",df3
```

In [512]:
```
#All right, let's see if it worked by looking at the column index.
#It did work. Good job, Ryan. You can do this!
df3_clean.head()
```

Out[512]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	1990S	1995S	2000S	2005S	2010S	...	2000M	2005M	2010M	2015M	1990F	199

I melted the table, keeping Major Areas, Region constant.

```
5]: # Order, Major are, et al. are what I want to keep constant.
    #I am not naming all of the years as those are the variables that I would like to melt.
    #Value_name: Int. Migrant Stock is the specific data that the UN is gathering in the dataset. It's important to get thi
    df3_melt =(df3_clean.melt(id_vars = ['Order', 'Major area, region, country or area of destination', 'Notes',
           'Country Code', 'Type of Data'], value_name = "International migrant stock as a percentage of total"))

6]: df3_melt.head()
```

6]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	variable	International migrant stock as a percentage of total
0	1	WORLD	NaN	900	NaN	1990S	2.87331
1	2	Developed regions	(b)	901	NaN	1990S	7.198015
2	3	Developing regions	(c)	902	NaN	1990S	1.685021

I renamed the variable column to Year as this is the actual variable in this column.

```
In [518]: #Let's use the rename function and assign by column
          df3_named = df3_melt.rename(columns={df3_melt.columns[5]:"Year"})

In [519]: df3_named.head()
```

Out[519]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	International migrant stock as a percentage of total
0	1	WORLD	NaN	900	NaN	1990S	2.87331
1	2	Developed regions	(b)	901	NaN	1990S	7.198015
2	3	Developing regions	(c)	902	NaN	1990S	1.685021

I separated the suffixes from the year (S, M, F) and created their own gender column.

```
In [520]: df3_sex =(df3_named.assign(Gender = lambda x: x.Year.str[4].astype(str), Year = lambda x: x.Year.str[:4].astype(str)))
          df3_sex.head()
```

Out[520]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	International migrant stock as a percentage of total	Gender
0	1	WORLD	NaN	900	NaN	1990	2.87331	S
1	2	Developed regions	(b)	901	NaN	1990	7.198015	S
2	3	Developing regions	(c)	902	NaN	1990	1.685021	S
3	4	Least developed countries	(d)	941	NaN	1990	2.171513	S
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990	1.617042	S

```
In [521]: #I then will take these Sex variables, rename them as S = Both, M = Male, F = Female
          df3_sex_both = (df3_sex.replace(to_replace =["S","M","F"],value =["Both","Male","Female"]))

In [522]: df3_sex_both.head()
```

Out[522]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	International migrant stock as a percentage of total	Gender
0	1	WORLD	NaN	900	NaN	1990	2.87331	Both
1	2	Developed regions	(b)	901	NaN	1990	7.198015	Both

Using the lambda function and if else, I assigned all of the countries to their respective continents and created a new continent column.

```
df3_dropped['continent'] = df3_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else '
df3_dropped.head(25)
```

t[526]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock as a percentage of total	Gender	continent
8	9	Burundi	NaN	108	B R	1990	5.934467	Both	Africa
9	10	Comoros	NaN	174	B	1990	3.391353	Both	Africa
10	11	Djibouti	NaN	262	B R	1990	20.773307	Both	Africa
11	12	Eritrea	NaN	232	I	1990	0.377435	Both	Africa
12	13	Ethiopia	NaN	231	B R	1990	2.404203	Both	Africa
13	14	Kenya	NaN	404	B R	1990	1.267974	Both	Africa
14	15	Madagascar	NaN	450	C	1990	0.207149	Both	Africa

I removed all major areas from country or area of destination. I will be creating a new column for major areas. Again, this is applying the tidy data principal of having only one variable per column.

```
'Least developed countries',
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df3_remove_area.head(25)
```

Out[527]:

click to scroll output; double click to hide

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock as a percentage of total	Gender	continent
8	9	Burundi	NaN	108	B R	1990	5.934467	Both	Africa
9	10	Comoros	NaN	174	B	1990	3.391353	Both	Africa
10	11	Djibouti	NaN	262	B R	1990	20.773307	Both	Africa
11	12	Eritrea	NaN	232	I	1990	0.377435	Both	Africa
12	13	Ethiopia	NaN	231	B R	1990	2.404203	Both	Africa

Out[528]:

click to scroll output; double click to hide

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	International migrant stock as a percentage of total	Gender	continent	major_region
8	9	Burundi	NaN	108	B R	1990	5.934467	Both	Africa	Eastern Africa
9	10	Comoros	NaN	174	B	1990	3.391353	Both	Africa	Eastern Africa
10	11	Djibouti	NaN	262	B R	1990	20.773307	Both	Africa	Eastern Africa
11	12	Eritrea	NaN	232	I	1990	0.377435	Both	Africa	Eastern Africa
12	13	Ethiopia	NaN	231	B R	1990	2.404203	Both	Africa	Eastern Africa
13	14	Kenya	NaN	404	B R	1990	1.267974	Both	Africa	Eastern Africa
14	15	Madagascar	NaN	450	C	1990	0.207149	Both	Africa	Eastern Africa
15	16	Malawi	NaN	454	B R	1990	11.985591	Both	Africa	Eastern Africa
16	17	Mauritius	(1)	480	C	1990	0.342184	Both	Africa	Eastern Africa

I dropped order, notes, and type of data as they are extraneous and serve no real value to either grouping the data or the sum of the migrant stock. I also reordered the columns to show country, continent, and major region as these three columns influence one another and show a progression of going from smaller to bigger grouping.

```
In [533]: #On second thought the order is arbitrary at this point. It's extraneous and not tidy.
df3_drop_order = df3_new_order.drop(['Order', 'notes','type_of_data'], axis = 1)
df3_drop_order.head()
```

Out[533]:

click to scroll output; double click to hide

	nation	continent	major_region	country_code	year	international_migrant_stock_as_a_percentage_of_total	sex
8	Burundi	Africa	Eastern Africa	108	1990	5.934467	Both
9	Comoros	Africa	Eastern Africa	174	1990	3.391353	Both
10	Djibouti	Africa	Eastern Africa	262	1990	20.773307	Both

I resorted the index and dropped the order column as that was no longer necessary.

```
In [536]: #df1_drop_index = df1_sort_country.drop("index", axis =1)
df3_drop_index = df3_sort_country.drop("index", axis = 1)

In [537]: df3_drop_index.head()
```

Out[537]:

	country_or_area_of_destination	continent	major_region	country_code	year	international_migrant_stock_as_a_percentage_of_total	sex
0	Afghanistan	Asia	Southern Asia	4	1995	0.450392	Male
1	Afghanistan	Asia	Southern Asia	4	2015	1.153263	Male
2	Afghanistan	Asia	Southern Asia	4	2000	0.346076	Female

<u>Table 4 – Female Migrants as a percentage of the international migrant stock by major area, region, country or area, 1990-2015</u>

At first glance, this table seems to be showing similar data as Table 3 – percentage of female migrant stock. However, the values do not correspond which leads me to think this is based on different information. Regardless, I will be keeping this table discrete.

I assigned names to the column. This time around I will not be adding suffixes to the years since we are only dealing with one sex for this table – Female.

```
In [543]: #Since I cleaned up the vanity cells in the excel file, I nnow need to assign names to the columns.
#I will not add gender suffixes to the year as this table only displays one sex, female.
#I will separate by gender in a big. I just want to melt the years right now. Keeping it simple.
df4_clean.rename(columns={df4.columns[0]:"Order", df4.columns[1]:"Major area, region, country or area of destination",
                          df4.columns[2]:"Notes",df4.columns[3]:"Country Code",df4.columns[4]:"Type of Data",
                          df4.columns[5]:"1990",df4.columns[6]:"1995",df4.columns[7]:"2000",df4.columns[8]:"2005",df4.columns
                          df4.columns[10]:"2015"}, inplace = True)

In [544]: df4_clean.head()
```

Out[544]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	1990	1995	2000	2005	2010	2015
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

I kept major area, region, et al. constant and melted the years.

In [545]:
```
# Order, Major are, et al. are what I want to keep constant.
#I am not naming all of the years as those are the variables that I would like to melt.
#Value_name: Female Migrants as a percenatge of the int migrant stock is the specific data that the UN is gathering in
df4_melt =(df4_clean.melt(id_vars = ['Order', 'Major area, region, country or area of destination', 'Notes',
       'Country Code', 'Type of Data'], value_name = "Female migrants as a percentage of the international migrant stoc
```

In [546]:
```
df4_melt.head()
```

Out[546]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	variable	Female migrants as a percentage of the international migrant stock
0	1	WORLD	NaN	900	NaN	1990	49.03915
1	2	Developed regions	(b)	901	NaN	1990	51.123977
2	3	Developing regions	(c)	902	NaN	1990	46.592099
3	4	Least developed countries	(d)	941	NaN	1990	47.261155
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990	46.466684

I renamed the variable column to years.

n [548]:
```
#But I still need to rename that "variable" column.
#Let's use the rename function and assign by column
df4_named = df4_melt.rename(columns={df4_melt.columns[5]:"Year"})
```

n [549]:
```
df4_named
```

ut[549]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	Female migrants as a percentage of the international migrant stock
0	1	WORLD	NaN	900	NaN	1990	49.03915
1	2	Developed regions	(b)	901	NaN	1990	51.123977
2	3	Developing regions	(c)	902	NaN	1990	46.592099
3	4	Least developed countries	(d)	941	NaN	1990	47.261155

I created a column for continent to their corresponding countries. Again, keeping every column to one variable.

```
df4_dropped['continent'] = df4_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else '

df4_dropped.head(25)
```

ut[552]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	Female migrants as a percentage of the international migrant stock	continent
8	9	Burundi	NaN	108	B R	1990	50.987061	Africa
9	10	Comoros	NaN	174	B	1990	52.290646	Africa
10	11	Djibouti	NaN	262	B R	1990	47.437838	Africa
11	12	Eritrea	NaN	232	I	1990	47.434166	Africa
12	13	Ethiopia	NaN	231	B R	1990	47.439047	Africa
13	14	Kenya	NaN	404	B R	1990	45.894272	Africa
14	15	Madagascar	NaN	450	C	1990	44.190325	Africa

I removed the major regions from the country or area of destination. I created a new column for major regions.

```
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df4_remove_area.head(25)
```

53]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	Female migrants as a percentage of the international migrant stock	continent
8	9	Burundi	NaN	108	B R	1990	50.987061	Africa
9	10	Comoros	NaN	174	B	1990	52.290646	Africa
10	11	Djibouti	NaN	262	B R	1990	47.437838	Africa
11	12	Eritrea	NaN	232	I	1990	47.434166	Africa
12	13	Ethiopia	NaN	231	B R	1990	47.439047	Africa
13	14	Kenya	NaN	404	B R	1990	45.894272	Africa
14	15	Madagascar	NaN	450	C	1990	44.190325	Africa
15	16	Malawi	NaN	454	B R	1990	51.537788	Africa
16	17	Mauritius	(1)	480	C	1990	51.203986	Africa

```
df4_region = df4_remove_area

df4_region['major_region'] = df4_region["country_or_area_of_destination"].apply(lambda x: 'Eastern Africa' if x in east
                                                                                  'Middle Africa' if x in middle_africa
                                                                                  'Northern Africa' if x in northern_af
                                                                                  'Southern Africa' if x in southern_af
                                                                                  'Western Africa' if x in western_afri
                                                                                  'Central Asia' if x in central_asia e
                                                'Western Asia' if x in western_asia else
                                                'Eastern Asia' if x in eastern_asia else
                                                'South-Eastern Asia' if x in south_eastern_asia else
                                                'Southern Asia' if x in southern_asia else
                                                'Eastern Europe' if x in eastern_europe else
                                                'Northern Europe' if x in northern_europe else
                                                'Southern Europe' if x in southern_europe else
                                                'Western Europe' if x in western_europe else
                                                'Caribbean' if x in caribbean else
                                                'Central America' if x in central_america else
                                                'South America' if x in south_america else
                                                'Northern America' if x in northern_america else
                                                'Australia and New Zealand' if x in australia_and_new_zealand else
                                                'Melanesia' if x in melanesia else
                                                'Micronesia' if x in micronesia else
                                                'Polynesia' if x in polynesia else
```

Out[554]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	Female migrants as a percentage of the international migrant stock	continent	major_region
8	9	Burundi	NaN	108	B R	1990	50.987061	Africa	Eastern Africa
9	10	Comoros	NaN	174	B	1990	52.290646	Africa	Eastern Africa
10	11	Djibouti	NaN	262	B R	1990	47.437838	Africa	Eastern Africa
11	12	Eritrea	NaN	232	I	1990	47.434166	Africa	Eastern Africa
12	13	Ethiopia	NaN	231	B R	1990	47.439047	Africa	Eastern Africa
13	14	Kenya	NaN	404	B R	1990	45.894272	Africa	Eastern Africa

I dropped order, notes, and type of data as I explained for the pervious tables. I then resorted the tables.

```
In [557]: #On second thought the order is arbitrary at this point. It's extraneous and not tidy.
          df4_drop_order = df4_new_name.drop(['Order', 'notes','type_of_data'], axis = 1)
          df4_drop_order.head()
```

Out[557]:

	country_or_area_of_destination	country_code	year	female_migrants_as_a_percentage_of_the_international_migrant_stock	continent	major_region
8	Burundi	108	1990	50.987061	Africa	Eastern Africa
9	Comoros	174	1990	52.290646	Africa	Eastern Africa
10	Djibouti	262	1990	47.437838	Africa	Eastern Africa
11	Eritrea	232	1990	47.434166	Africa	Eastern Africa
12	Ethiopia	231	1990	47.439047	Africa	Eastern Africa

```
In [558]: df4_new_sort = df4_drop_order
          df4_sort_country = df4_new_sort.sort_values(by=['country_or_area_of_destination'])
          df4_sort_country
```

Out[558]:

	country_or_area_of_destination	country_code	year	female_migrants_as_a_percentage_of_the_international_migrant_stock	continent	major_region

I reordered the columns as explained in previous tables.

```
In [562]: #I would like to move the columns to better reflect the variables (continents and regions) against the countries.

          new_cols = ['country_or_area_of_destination','continent', 'major_region', 'country_code', 'year', 'female_migrants_as_a
          df4_new_order  = df4_drop_index.reindex(columns=new_cols)
          df4_new_order.head()
```

Out[562]:

	country_or_area_of_destination	continent	major_region	country_code	year	female_migrants_as_a_percentage_of_the_international_migrant_stock
0	Afghanistan	Asia	Southern Asia	4	2005	43.557847
1	Afghanistan	Asia	Southern Asia	4	2015	49.408288
2	Afghanistan	Asia	Southern Asia	4	2000	43.559414
3	Afghanistan	Asia	Southern Asia	4	1990	43.559963
4	Afghanistan	Asia	Southern Asia	4	2010	43.558672

```
In [563]: #changing the objects name to keep all of my outputs consistent
          df4_drop_index = df4_new_order
          df4_drop_index.head()
```

<u>Table 5 - Annual rate of change of the migrant stock by sex and by major area, region, country or area, 1990-2015 (percentage)</u>

First, I assigned the names to the columns. Since we are back to dealing with both sexes, I added the suffixes to the year again.

```
In [568]: #Since I cleaned up the vanity cells in the excel file, I know need to assign names to the columns.
          #I am using S= Same Sex, M = Male, F= Female. I will separate by gender in a big. I just want to melt the years right n
          df5_clean.rename(columns={df5.columns[0]:"Order", df5.columns[1]:"Major area, region, country or area of destination",
                                    df5.columns[2]:"Notes",df5.columns[3]:"Country Code",df5.columns[4]:"Type of Data",
                                    df5.columns[5]:"1990-1995S",df5.columns[6]:"1995-2000S",df5.columns[7]:"2000-2005S",df5.columns[8]:
                                    df5.columns[10]:"1990-1995M",df5.columns[11]:"1995-2000M",df5.columns[12]:"2000-2005M",df5.columns[
                                    df5.columns[17]:"2000-2005F",df5.columns[18]:"2005-2010F",df5.columns[19]:"2010-2015F"}, inplace =
```

```
In [569]: df5_clean.head()
```

Out[569]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	1990-1995S	1995-2000S	2000-2005S	2005-2010S	2010-2015S	1990-1995M	1995-2000M	2000-2005M	2005-2010M	2010-2015M	1990-1995F
15	1	WORLD	NaN	900	NaN	1.051865	1.428058	2.042124	2.95416	1.890991	1.000922	1.450294	2.151575	3.159228	1.912603	1.104667
16	2	Developed regions	(b)	901	NaN	2.275847	2.264965	2.50708	2.466343	1.160824	2.265595	2.279583	2.483259	2.265689	1.074685	2.285643
17	3	Developing regions	(c)	902	NaN	-0.487389	0.241777	1.328107	3.702217	2.929634	-0.45298	0.380246	1.693824	4.352954	2.927058	-0.526904

I melted the table from wide format to long format, keeping major area, region, et al. as the constant.

```
[572]: # Order, Major are, et al. are what I want to keep constant.
       #I am not naming all of the years as those are the variables that I would like to melt.
       #Value_name: Int. Migrant Stock is the specific data that the UN is gathering in the dataset. It's important to get thi
       df5_melt =(df5_clean.melt(id_vars = ['Order', 'Major area, region, country or area of destination', 'Notes',
              'Country Code', 'Type of Data'], value_name = "Annual rate of change of the migrant stock"))

[573]: #Melt worked. We are now long table from a wide table
       df5_melt.shape

[573]: (3975, 7)

[574]: df5_melt.head()
```

[574]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	variable	Annual rate of change of the migrant stock
0	1	WORLD	NaN	900	NaN	1990-1995S	1.051865
1	2	Developed regions	(b)	901	NaN	1990-1995S	2.275847
2	3	Developing regions	(c)	902	NaN	1990-1995S	-0.487389
3	4	Least developed countries	(d)	941	NaN	1990-1995S	1.118175

I renamed the variable column to Year.

```
In [574]: df5_melt.head()
```

Out[574]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	variable	Annual rate of change of the migrant stock
0	1	WORLD	NaN	900	NaN	1990-1995S	1.051865
1	2	Developed regions	(b)	901	NaN	1990-1995S	2.275847
2	3	Developing regions	(c)	902	NaN	1990-1995S	-0.487389
3	4	Least developed countries	(d)	941	NaN	1990-1995S	1.118175
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990-1995S	-0.803244

```
In [575]: df5_named = df5_melt.rename(columns={df5_melt.columns[5]:"Year"})
          df5_named.head()
```

Out[575]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	Annual rate of change of the migrant stock
0	1	WORLD	NaN	900	NaN	1990-1995S	1.051865
1	2	Developed regions	(b)	901	NaN	1990-1995S	2.275847
2	3	Developing regions	(c)	902	NaN	1990-1995S	-0.487389

Using the lambda function, I separated the suffixes from the year and created a Gender column (I will rename this column to sex shortly).

```
[576]: df5_sex =(df5_named.assign(Gender = lambda x: x.Year.str[9].astype(str), Year = lambda x: x.Year.str[:9].astype(str)))
       df5_sex.head()
```

[576]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	Annual rate of change of the migrant stock	Gender
0	1	WORLD	NaN	900	NaN	1990-1995	1.051865	S
1	2	Developed regions	(b)	901	NaN	1990-1995	2.275847	S
2	3	Developing regions	(c)	902	NaN	1990-1995	-0.487389	S
3	4	Least developed countries	(d)	941	NaN	1990-1995	1.118175	S
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990-1995	-0.803244	S

```
[577]: df5_sex_both = (df5_sex.replace(to_replace =["S","M","F"],value =["Both","Male","Female"]))
       df5_sex_both.head()
```

[577]:

I replaced the instance of S, M, F to Both, Male, and Female respectively in the Gender column.

```
n [577]: df5_sex_both = (df5_sex.replace(to_replace =["S","M","F"],value =["Both","Male","Female"]))
         df5_sex_both.head()
```

ut[577]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Year	Annual rate of change of the migrant stock	Gender
0	1	WORLD	NaN	900	NaN	1990-1995	1.051865	Both
1	2	Developed regions	(b)	901	NaN	1990-1995	2.275847	Both
2	3	Developing regions	(c)	902	NaN	1990-1995	-0.487389	Both
3	4	Least developed countries	(d)	941	NaN	1990-1995	1.118175	Both
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990-1995	-0.803244	Both

I created a continent column and assigned the countries to their respective continent.

```
df5_dropped['continent'] = df5_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else '

df5_dropped.head(25)
```

ut[580]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	Annual rate of change of the migrant stock	Gender	continent
8	9	Burundi	NaN	108	B R	1990-1995	-5.355717	Both	Africa
9	10	Comoros	NaN	174	B	1990-1995	-0.199873	Both	Africa
10	11	Djibouti	NaN	262	B R	1990-1995	-4.058465	Both	Africa
11	12	Eritrea	NaN	232	I	1990-1995	0.910748	Both	Africa
12	13	Ethiopia	NaN	231	B R	1990-1995	-7.179771	Both	Africa
13	14	Kenya	NaN	404	B R	1990-1995	14.659568	Both	Africa
14	15	Madagascar	NaN	450	C	1990-1995	-2.433476	Both	Africa
15	16	Malawi	NaN	454	B R	1990-1995	-30.811478	Both	Africa

I removed major regions from the country of area of destination. Again, in keeping with Tidy Data principals, I want to keep only one variable per column. I will now make a major regions column.

```
'WORLD',
'Developed regions',
'Developing regions',
'Least developed countries',
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df5_remove_area.head(25)
```

[581]:

	Order	country_or_area_of_destination	Notes	Country Code	Type of Data	Year	Annual rate of change of the migrant stock	Gender	continent
8	9	Burundi	NaN	108	B R	1990-1995	-5.355717	Both	Africa
9	10	Comoros	NaN	174	B	1990-1995	-0.199873	Both	Africa
10	11	Djibouti	NaN	262	B R	1990-1995	-4.058465	Both	Africa
11	12	Eritrea	NaN	232	I	1990-1995	0.910748	Both	Africa
12	13	Ethiopia	NaN	231	B R	1990-1995	-7.179771	Both	Africa
13	14	Kenya	NaN	404	B R	1990-1995	14.659568	Both	Africa

I have now created a major regions column and assigned all countries to their respective major regions. I used the lambda function if else statement to do this.

```
'Uganda',
'United Republic of Tanzania',
'Zambia',
'Zimbabwe']

df5_region = df5_remove_area

df5_region['major_region'] = df5_region["country_or_area_of_destination"].apply(lambda x: 'Eastern Africa' if x in east
                                                                                  'Middle Africa' if x in middle_africa
                                                                                  'Northern Africa' if x in northern_af
                                                                                  'Southern Africa' if x in southern_af
                                                                                  'Western Africa' if x in western_afri
                                                                                  'Central Asia' if x in central_asia e
                                                    'Western Asia' if x in western_asia else
                                                    'Eastern Asia' if x in eastern_asia else
                                                    'South-Eastern Asia' if x in south_eastern_asia else
                                                    'Southern Asia' if x in southern_asia else
                                                    'Eastern Europe' if x in eastern_europe else
                                                    'Northern Europe' if x in northern_europe else
                                                    'Southern Europe' if x in southern_europe else
                                                    'Western Europe' if x in western_europe else
                                                    'Caribbean' if x in caribbean else
                                                    'Central America' if x in central_america else
                                                    'South America' if x in south_america else
                                                    'Northern America' if x in northern_america else
                                                    'Australia and New Zealand' if x in australia_and_new_zealand else
                                                    'Melanesia' if x in melanesia else
                                                    'Micronesia' if x in micronesia else
                                                    'Polynesia' if x in polynesia else
                                                    'none')
```

I dropped order, notes, and type of data. Again, these columns are not necessary and do not influence the sums of migrant stock nor do they influence the grouping.

```
In [ ]: click to scroll output; double click to hide
        he order is arbitrary at this point. It's extraneous and not tidy.
        df5_drop_order = df5_new_order.drop(['Order', 'notes','type_of_data'], axis = 1)
        df5_drop_order.head(10)
```

Out[586]:

	country_or_area_of_destination	continent	major_region	country_code	year	annual_rate_of_change_of_the_migrant_stock	sex
8	Burundi	Africa	Eastern Africa	108	1990-1995	-5.355717	Both
9	Comoros	Africa	Eastern Africa	174	1990-1995	-0.199873	Both
10	Djibouti	Africa	Eastern Africa	262	1990-1995	-4.058465	Both
11	Eritrea	Africa	Eastern Africa	232	1990-1995	0.910748	Both
12	Ethiopia	Africa	Eastern Africa	231	1990-1995	-7.179771	Both
13	Kenya	Africa	Eastern Africa	404	1990-1995	14.659568	Both

I reset the index and dropped the former index as it is no longer necessary.

```
In [588]: #reset index
df5_sort_country = df5_sort_country.reset_index()
df5_sort_country.head()
```

Out[588]:

	index	country_or_area_of_destination	continent	major_region	country_code	year	annual_rate_of_change_of_the_migrant_stock	sex
0	98	Afghanistan	Asia	Southern Asia	4	1990-1995	4.299812	Both
1	1423	Afghanistan	Asia	Southern Asia	4	1990-1995	3.664544	Male
2	2748	Afghanistan	Asia	Southern Asia	4	1990-1995	5.094004	Female
3	363	Afghanistan	Asia	Southern Asia	4	1995-2000	1.192711	Both
4	1688	Afghanistan	Asia	Southern Asia	4	1995-2000	1.828173	Male

```
In [589]: #Drop index since we no longer need it. It's redundant.
df5_drop_index = df5_sort_country.drop("index", axis = 1)
df5_drop_index.head()
```

Out[589]:

	country_or_area_of_destination	continent	major_region	country_code	year	annual_rate_of_change_of_the_migrant_stock	sex
0	Afghanistan	Asia	Southern Asia	4	1990-1995	4.299812	Both
1	Afghanistan	Asia	Southern Asia	4	1990-1995	3.664544	Male
2	Afghanistan	Asia	Southern Asia	4	1990-1995	5.094004	Female
3	Afghanistan	Asia	Southern Asia	4	1995-2000	1.192711	Both
4	Afghanistan	Asia	Southern Asia	4	1995-2000	1.828173	Male

Table 6 - Estimated refugee stock at mid-year by major area, region, country or area, 1990-2015

Table 6 will be split into three discrete Tables as it contains three different values. In keeping with Tidy Data principals, this seems to be the best method to proceed.

Table 6A - Estimated refugee stock at mid-year (both sexes)

Table 6B - Refugees as a percentage of the international migrant stock

Table 6C - Annual rate of change of the refugee stock

Table 6A - Estimated refugee stock at mid-year (both sexes)

I assign names to relevant columns for this table. I will drop the extraneous columns shortly.

```
In [644]: #Rename column headers - This will be Table df6a - Estimated refugee stock at mid-year (both sexes)
df6A.rename(columns={df6A.columns[0]:"Order", df6A.columns[1]:"Major area, region, country or area of destination",
                     df6A.columns[2]:"Notes",df6A.columns[3]:"Country Code",df6A.columns[4]:"Type of Data",
                     df6A.columns[5]:"1990", df6A.columns[6]:"1995",  df6A.columns[7]: "2000",  df6A.columns[8]:"2005",
                     df6A.columns[10]: "2015"}, inplace = True)

In [645]: df6A.head()
```

Out[645]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	1990	1995	2000	2005	2010	...	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16
15	1	WORLD	NaN	900	NaN	18836571	17853840	15827803	13276733	15370755.0	...	11.103013	9.164736	6.941389	6.932687	8.033424
16	2	Developed regions	(b)	901	NaN	2014564	3609670	2997256	2361229	2046917.0	...	3.910511	2.899391	2.015025	1.544140	1.391085

I will now drop the extraneous columns that are different values. These other columns will be discretely placed in their own tables.

```
646]: #I will now drop columns 12 - 21. I will create two separate tables once I finish this table.
df6A_drop = df6A.drop(df6A.columns[[0,2,4,11,12,13,14,15,16,17,18,19,20,21]], axis=1)
df6A_drop.head()
```

646]:

	Major area, region, country or area of destination	Country Code	1990	1995	2000	2005	2010	2015
15	WORLD	900	18836571	17853840	15827803	13276733	15370755.0	19577474.0
16	Developed regions	901	2014564	3609670	2997256	2361229	2046917.0	1954224.0
17	Developing regions	902	16822007	14244170	12830547	10915504	13323838.0	17623250.0
18	Least developed countries	941	5048391	5160131	3047488	2363782	1957884.0	3443582.0
19	Less developed regions excluding least develop...	934	11773616	9084039	9783059	8551722	11365954.0	14179668.0

I will now melt the table, keeping major area, region, country, et al. as the constant. I will then rename the variable column to Year.

```
n [647]: df6A_melt =(df6A_drop.melt(id_vars = ['Major area, region, country or area of destination',
       'Country Code'], value_name = "Estimated refugee stock at mid-year (both sexes)"))

n [648]: df6A_rename = df6A_melt.rename(columns={df6A_melt.columns[2]:"Year"})
df6A_rename.head()
```

ut[648]:

	Major area, region, country or area of destination	Country Code	Year	Estimated refugee stock at mid-year (both sexes)
0	WORLD	900	1990	18836571
1	Developed regions	901	1990	2014564
2	Developing regions	902	1990	16822007
3	Least developed countries	941	1990	5048391
4	Less developed regions excluding least develop...	934	1990	11773616

Using the lambda if else function, I will add a continent column and assign the corresponding countries.

```
df6A_dropped['continent'] = df6A_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else

df6A_dropped.head(25)
```

[605]:

	country_or_area_of_destination	country_code	year	estimated_refugee_stock_at_mid_year	sex	continent
8	Burundi	108	1990	267929	Both	Africa
9	Comoros	174	1990	0	Both	Africa
10	Djibouti	262	1990	54508	Both	Africa
11	Eritrea	232	1990	0	Both	Africa
12	Ethiopia	231	1990	741965	Both	Africa
13	Kenya	404	1990	13452	Both	Africa

I will now remove major regions from the country or area of destination. Again, I need to keep all columns to one variable.

```
'WORLD',
'Developed regions',
'Developing regions',
'Least developed countries',
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df6A_remove.head(25)
```

Out[606]:

	country_or_area_of_destination	country_code	year	estimated_refugee_stock_at_mid_year	sex	continent
8	Burundi	108	1990	267929	Both	Africa
9	Comoros	174	1990	0	Both	Africa
10	Djibouti	262	1990	54508	Both	Africa
11	Eritrea	232	1990	0	Both	Africa
12	Ethiopia	231	1990	741965	Both	Africa

I will now assign the countries to their corresponding Major Regions and create a new column for this variable.

```
df6A_region = df6A_remove

df6A_region['major_region'] = df6A_region["country_or_area_of_destination"].apply(lambda x: 'Eastern Africa' if x in ea
                                                                                    'Middle Africa' if x in middle_africa
                                                                                    'Northern Africa' if x in northern_af
                                                                                    'Southern Africa' if x in southern_af
                                                                                    'Western Africa' if x in western_afri
                                                                                    'Central Asia' if x in central_asia e
                                                          'Western Asia' if x in western_asia else
                                                          'Eastern Asia' if x in eastern_asia else
                                                          'South-Eastern Asia' if x in south_eastern_asia else
                                                          'Southern Asia' if x in southern_asia else
                                                          'Eastern Europe' if x in eastern_europe else
                                                          'Northern Europe' if x in northern_europe else
                                                          'Southern Europe' if x in southern_europe else
                                                          'Western Europe' if x in western_europe else
                                                          'Caribbean' if x in caribbean else
                                                          'Central America' if x in central_america else
                                                          'South America' if x in south_america else
                                                          'Northern America' if x in northern_america else
                                                          'Australia and New Zealand' if x in australia_and_new_zealand else
                                                          'Melanesia' if x in melanesia else
                                                          'Micronesia' if x in micronesia else
```

I will reorder the columns to better reflect the country, continent, and major region since they influence one another in terms of grouping.

In [661]:
```
#I would like to move the columns to better reflect the variables (continents and regions) against the countries.

new_cols = ['country_or_area_of_destination','continent', 'major_region', 'country_code', 'year', 'estimated_refugee_st
df6A_new_order  = df6A_region.reindex(columns=new_cols)
df6A_new_order.head()
```

Out[661]:

	country_or_area_of_destination	continent	major_region	country_code	year	estimated_refugee_stock_at_mid_year
8	Burundi	Africa	Eastern Africa	108	1990	267929
9	Comoros	Africa	Eastern Africa	174	1990	0
10	Djibouti	Africa	Eastern Africa	262	1990	54508
11	Eritrea	Africa	Eastern Africa	232	1990	0

I will re-sort the table and drop the old index since it is out of sync from the rows.

```
In [661]: #I would like to move the columns to better reflect the variables (continents and regions) against the countries.

new_cols = ['country_or_area_of_destination','continent', 'major_region', 'country_code', 'year', 'estimated_refugee_st
df6A_new_order  = df6A_region.reindex(columns=new_cols)
df6A_new_order.head()
```

Out[661]:

	country_or_area_of_destination	continent	major_region	country_code	year	estimated_refugee_stock_at_mid_year
8	Burundi	Africa	Eastern Africa	108	1990	267929
9	Comoros	Africa	Eastern Africa	174	1990	0
10	Djibouti	Africa	Eastern Africa	262	1990	54508
11	Eritrea	Africa	Eastern Africa	232	1990	0
12	Ethiopia	Africa	Eastern Africa	231	1990	741965

click to scroll output; double click to hide

```
In [609]: #I would like to see how the table 6A looks
df6A_new_order.to_csv("table6A.csv")
```

```
In [662]: #reset index
df6A_sort_country = df6A_new_order.reset_index()
df6A_sort_country.head()
```

Out[662]:

	index	country_or_area_of_destination	continent	major_region	country_code	year	estimated_refugee_stock_at_mid_year
0	8	Burundi	Africa	Eastern Africa	108	1990	267929
1	9	Comoros	Africa	Eastern Africa	174	1990	0
2	10	Djibouti	Africa	Eastern Africa	262	1990	54508

Table 6B - Refugees as a percentage of the international migrant stock

I will assign names to only the columns that reflect the percentage of international migrant stock. No sex or gender is given so I will not add suffixes to the years.

```
In [614]: #Rename column headers - This will be Table df6a - Estimated refugee stock at mid-year (both sexes)
df6B_drop.rename(columns={df6B_drop.columns[0]:"Order", df6B_drop.columns[1]:"Major area, region, country or area of de
                          df6B_drop.columns[2]:"Notes",df6B_drop.columns[3]:"Country Code",df6B_drop.columns[4]:"Type of Data
                          df6B_drop.columns[11]:"1990", df6B_drop.columns[12]:"1995", df6B_drop.columns[13]: "2000", df6B_drc
                          df6B_drop.columns[16]: "2015"}, inplace = True)
df6B_drop.head()
```

Out[614]:

	Order	Major area, region, country or area of destination	Notes	Country Code	Type of Data	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	1995	2000	2005	2010	20
15	1	WORLD	NaN	900	NaN	18836571	17853840	15827803	13276733	15370755.0	...	11.103013	9.164736	6.941389	6.932687	8.0334
16	2	Developed regions	(b)	901	NaN	2014564	3609670	2997256	2361229	2046917.0	...	3.910511	2.899391	2.015025	1.544140	1.3910
17	3	Developing regions	(c)	902	NaN	16822007	14244170	12830547	10915504	13323838.0	...	20.795958	18.507035	14.733162	14.944759	17.0737

I will now drop the other columns that contain other values. I will now melt only the columns with percentage values, keeping major area, region, country et al. as the constant.

```
In [615]: #I will now drop columns 5 - 10. I will create two separate tables once I finish this table.
df6B_drop_columns = df6B_drop.drop(df6B_drop.columns[[0,2,4,5,6,7,8,9,10,17,18,19,20,21]], axis=1)
df6B_drop_columns.head()
```

Out[615]:

	Major area, region, country or area of destination	Country Code	1990	1995	2000	2005	2010	2015
15	WORLD	900	12.346732	11.103013	9.164736	6.941389	6.932687	8.033424
16	Developed regions	901	2.445494	3.910511	2.899391	2.015025	1.544140	1.391085
17	Developing regions	902	23.968236	20.795958	18.507035	14.733162	14.944759	17.073768
18	Least developed countries	941	45.56588	44.041961	30.221557	24.08243	19.533425	28.801534
19	Less developed regions excluding least develop...	934	19.919743	15.999082	16.51313	13.305391	14.363526	15.537313

```
In [616]: #Melt Refugees as a percentage of the international migrant stock to Major area, region, et al.
df6B_melt =(df6B_drop_columns.melt(id_vars = ['Major area, region, country or area of destination',
       'Country Code'], value_name = "Refugees as a percentage of the international migrant stock"))
df6B_melt.head()
```

Out[616]:

	Major area, region, country or area of destination	Country Code	variable	Refugees as a percentage of the international migrant stock
0	WORLD	900	1990	12.346732
1	Developed regions	901	1990	2.445494

I will now rename the variable column to year.

```
In [617]: #rename the variable column to year
df6B_rename = df6B_melt.rename(columns={df6B_melt.columns[2]:"Year"})
df6B_rename.head()
```

Out[617]:

	Major area, region, country or area of destination	Country Code	Year	Refugees as a percentage of the international migrant stock
0	WORLD	900	1990	12.346732
1	Developed regions	901	1990	2.445494
2	Developing regions	902	1990	23.968236
3	Least developed countries	941	1990	45.56588
4	Less developed regions excluding least develop...	934	1990	19.919743

I will now add a continent column and assign the respective countries.

```
'Niue',
'Samoa',
'Tokelau',
'Tonga',
'Tuvalu',
'Wallis and Futuna Islands']



df6B_dropped['continent'] = df6B_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else

df6B_dropped.head(25)
```

[620]:

	country_or_area_of_destination	country_code	year	refugees_as_a_percentage_of_the_international_migrant_stock	continent
8	Burundi	108	1990	80.43259	Africa
9	Comoros	174	1990	0	Africa
10	Djibouti	262	1990	44.597901	Africa
11	Eritrea	232	1990	0	Africa
12	Ethiopia	231	1990	64.21771	Africa

I will now remove major regions from the country or area of destination column and make a new column for major regions.

```
'Least developed countries',
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df6B_remove.head(25)
```

Out[621]:

	country_or_area_of_destination	country_code	year	refugees_as_a_percentage_of_the_international_migrant_stock	continent
8	Burundi	108	1990	80.43259	Africa
9	Comoros	174	1990	0	Africa
10	Djibouti	262	1990	44.597901	Africa
11	Eritrea	232	1990	0	Africa
12	Ethiopia	231	1990	64.21771	Africa
13	Kenya	404	1990	4.524844	Africa
14	Madagascar	450	1990	0	Africa
15	Malawi	454	1990	77.555679	Africa

click to scroll output; double click to hide

I will now assign countries to a new major regions column.

```
df6B_region = df6B_remove

df6B_region['major_region'] = df6B_region["country_or_area_of_destination"].apply(lambda x: 'Eastern Africa' if x in ea
                                                                                    'Middle Africa' if x in middle_africa
                                                                                    'Northern Africa' if x in northern_af
                                                                                    'Southern Africa' if x in southern_af
                                                                                    'Western Africa' if x in western_afri
                                                                                    'Central Asia' if x in central_asia e
                                                    'Western Asia' if x in western_asia else
                                                    'Eastern Asia' if x in eastern_asia else
                                                    'South-Eastern Asia' if x in south_eastern_asia else
                                                    'Southern Asia' if x in southern_asia else
                                                    'Eastern Europe' if x in eastern_europe else
                                                    'Northern Europe' if x in northern_europe else
                                                    'Southern Europe' if x in southern_europe else
                                                    'Western Europe' if x in western_europe else
                                                    'Caribbean' if x in caribbean else
                                                    'Central America' if x in central_america else
                                                    'South America' if x in south_america else
                                                    'Northern America' if x in northern_america else
                                                    'Australia and New Zealand' if x in australia_and_new_zealand else
                                                    'Melanesia' if x in melanesia else
                                                    'Micronesia' if x in micronesia else
                                                    'Polynesia' if x in polynesia else
                                                    'none')

df6B_region.head(25)
```

I will sort and drop the index column. Index is out of sync with table and not necessary.

In [625]:
```
#reset index
df6B_sort_country = df6B_new_order.reset_index()
df6B_sort_country.head()
```

Out[625]:

	index	country_or_area_of_destination	continent	major_region	country_code	year	refugees_as_a_percentage_of_the_international_migrant_stock
0	8	Burundi	Africa	Eastern Africa	108	1990	80.43259
1	9	Comoros	Africa	Eastern Africa	174	1990	0
2	10	Djibouti	Africa	Eastern Africa	262	1990	44.597901
3	11	Eritrea	Africa	Eastern Africa	232	1990	0
4	12	Ethiopia	Africa	Eastern Africa	231	1990	64.21771

In [626]:
```
#drop index because it's not necessary
df6B_drop_index = df6B_sort_country.drop("index", axis = 1)
df6B_drop_index.head()
```

Out[626]:

	country_or_area_of_destination	continent	major_region	country_code	year	refugees_as_a_percentage_of_the_international_migrant_stock
0	Burundi	Africa	Eastern Africa	108	1990	80.43259
1	Comoros	Africa	Eastern Africa	174	1990	0
2	Djibouti	Africa	Eastern Africa	262	1990	44.597901
3	Eritrea	Africa	Eastern Africa	232	1990	0
4	Ethiopia	Africa	Eastern Africa	231	1990	64.21771

Table 6C - Annual rate of change of the refugee stock

I removed the vanity header and assigned names to the relevant columns for this table. We are only interested in the annual rate of change of refugee stock. I will not add gender suffixes to the year as this table does not include gender or sex.

```
In [628]: #Remove vanity cells at the top
          df6C_drop= df6C.drop(df6C.index[:15])
          df6C_drop.head()
```

Out[628]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unnamed: 13	Unnamed: 14	Un
click to scroll output; double click to hide			NaN	900	NaN	18836571	17853840	15827803	13276733	15370755.0	...	11.103013	9.164736	6.941389	6
16	2	Developed regions	(b)	901	NaN	2014564	3609670	2997256	2361229	2046917.0	...	3.910511	2.899391	2.015025	1
17	3	Developing regions	(c)	902	NaN	16822007	14244170	12830547	10915504	13323838.0	...	20.795958	18.507035	14.733162	14
18	4	Least developed countries	(d)	941	NaN	5048391	5160131	3047488	2363782	1957884.0	...	44.041961	30.221557	24.08243	19
19	5	Less developed regions excluding least develop...	NaN	934	NaN	11773616	9084039	9783059	8551722	11365954.0	...	15.999082	16.51313	13.305391	14

5 rows × 22 columns

```
In [629]: #Rename column headers - This will be Table df6a - Estimated refugee stock at mid-year (both sexes)
          df6C_drop.rename(columns={df6C_drop.columns[0]:"Order", df6C_drop.columns[1]:"Major area, region, country or area of de
                           df6C_drop.columns[2]:"Notes",df6C_drop.columns[3]:"Country Code",df6C_drop.columns[4]:"Type of Data
                           df6C_drop.columns[17]:"1990-1995", df6C_drop.columns[18]:"1995-2000", df6C_drop.columns[19]: "2000-
                           }, inplace = True)
```

I will now drop all other columns that have different values from annual rate of change of refugee stock. And then I will melt this table, keeping Major Area, region, country et al. as the constant.

```
[30]: df6C_drop_columns = df6C_drop.drop(df6C_drop.columns[[0,2,4,5,6,7,8,9,10,11,12,13,14,15,16]], axis=1)
      df6C_drop_columns.head()
```

[30]:

	Major area, region, country or area of destination	Country Code	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015
15	WORLD	900	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
16	Developed regions	901	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
17	Developing regions	902	-2.839417	-2.332154	-4.561	0.285195	2.663652
18	Least developed countries	941	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
19	Less developed regions excluding least develop...	934	-4.3836	0.632489	-4.319731	1.530456	1.571047

```
[31]: #Melt Refugees as a percentage of the international migrant stock to Major area, region, et al.
      df6C_melt =(df6C_drop_columns.melt(id_vars = ['Major area, region, country or area of destination',
            'Country Code'], value_name = "Annual rate of change of the refugee stock"))
      df6C_melt.head()
```

[31]:

	Major area, region, country or area of destination	Country Code	variable	Annual rate of change of the refugee stock
0	WORLD	900	1990-1995	-2.123497

I will now add a new continent column and assign their respective countries.

```
'Tuvalu',
'Wallis and Futuna Islands']



df6C_dropped['continent'] = df6C_dropped["country_or_area_of_destination"].apply(lambda x: 'Africa' if x in africa else

df6C_dropped.head(25)
```

Out[635]:

	country_or_area_of_destination	country_code	year	annual_rate_of_change_of_the_refugee_stock	continent
8	Burundi	108	1990-1995	-3.390926	Africa
9	Comoros	174	1990-1995	..	Africa
10	Djibouti	262	1990-1995	-9.763426	Africa
11	Eritrea	232	1990-1995	..	Africa
12	Ethiopia	231	1990-1995	-5.505717	Africa
13	Kenya	404	1990-1995	42.521055	Africa

I will now remove major areas from country or area of destination. Again, only one variable per column.

```
'Melanesia',
'Micronesia',
'Polynesia',
'WORLD',
'Developed regions',
'Developing regions',
'Least developed countries',
'Less developed regions excluding least developed countries',
'Sub-Saharan Africa',
'Africa',
'Eastern Africa'])]
df6C_remove.head(25)
```

Out[636]:

	country_or_area_of_destination	country_code	year	annual_rate_of_change_of_the_refugee_stock	continent
8	Burundi	108	1990-1995	-3.390926	Africa
9	Comoros	174	1990-1995	..	Africa
10	Djibouti	262	1990-1995	-9.763426	Africa
11	Eritrea	232	1990-1995	..	Africa
12	Ethiopia	231	1990-1995	-5.505717	Africa
13	Kenya	404	1990-1995	42.521055	Africa
14	Madagascar	450	1990-1995		Africa

I will now add a new column for major areas and assign the corresponding countries.

```
'Zimbabwe']

df6C_region = df6C_remove

df6C_region['major_region'] = df6C_region["country_or_area_of_destination"].apply(lambda x: 'Eastern Africa' if x in ea
                                                                                   'Middle Africa' if x in middle_africa
                                                                                   'Northern Africa' if x in northern_af
                                                                                   'Southern Africa' if x in southern_af
                                                                                   'Western Africa' if x in western_afri
                                                                                   'Central Asia' if x in central_asia e
                                                 'Western Asia' if x in western_asia else
                                                 'Eastern Asia' if x in eastern_asia else
                                                 'South-Eastern Asia' if x in south_eastern_asia else
                                                 'Southern Asia' if x in southern_asia else
                                                 'Eastern Europe' if x in eastern_europe else
                                                 'Northern Europe' if x in northern_europe else
                                                 'Southern Europe' if x in southern_europe else
                                                 'Western Europe' if x in western_europe else
                                                 'Caribbean' if x in caribbean else
                                                 'Central America' if x in central_america else
                                                 'South America' if x in south_america else
                                                 'Northern America' if x in northern_america else
                                                 'Australia and New Zealand' if x in australia_and_new_zealand else
                                                 'Melanesia' if x in melanesia else
                                                 'Micronesia' if x in micronesia else
                                                 'Polynesia' if x in polynesia else
                                                 'none')
```

I will reorder the columns to better reflect the variable of continent, region, and countries.

```
[638]: #I would like to move the columns to better reflect the variables (continents and regions) against the countries.

new_cols = ['country_or_area_of_destination','continent', 'major_region', 'country_code', 'year', 'annual_rate_of_chang
df6C_new_order  = df6C_region.reindex(columns=new_cols)
df6C_new_order.head()
```

Out[638]:

	country_or_area_of_destination	continent	major_region	country_code	year	annual_rate_of_change_of_the_refugee_stock
8	Burundi	Africa	Eastern Africa	108	1990-1995	-3.390926
9	Comoros	Africa	Eastern Africa	174	1990-1995	..
10	Djibouti	Africa	Eastern Africa	262	1990-1995	-9.763426
11	Eritrea	Africa	Eastern Africa	232	1990-1995	..
12	Ethiopia	Africa	Eastern Africa	231	1990-1995	-5.505717

I will rest the index and remove the older index since it is now out of sync with the table.

```
[640]: #reset index
df6C_sort_country = df6C_new_order.reset_index()
df6C_sort_country.head()
```

Out[640]:

	index	country_or_area_of_destination	continent	major_region	country_code	year	annual_rate_of_change_of_the_refugee_stock
0	8	Burundi	Africa	Eastern Africa	108	1990-1995	-3.390926
1	9	Comoros	Africa	Eastern Africa	174	1990-1995	..
2	10	Djibouti	Africa	Eastern Africa	262	1990-1995	-9.763426
3	11	Eritrea	Africa	Eastern Africa	232	1990-1995	..
4	12	Ethiopia	Africa	Eastern Africa	231	1990-1995	-5.505717

```
[641]: #drop index because it's not necessary
df6C_drop_index = df6C_sort_country.drop("index", axis = 1)
df6C_drop_index.head()
```

Out[641]:

	country_or_area_of_destination	continent	major_region	country_code	year	annual_rate_of_change_of_the_refugee_stock
0	Burundi	Africa	Eastern Africa	108	1990-1995	-3.390926
1	Comoros	Africa	Eastern Africa	174	1990-1995	..
2	Djibouti	Africa	Eastern Africa	262	1990-1995	-9.763426
3	Eritrea	Africa	Eastern Africa	232	1990-1995	..
4	Ethiopia	Africa	Eastern Africa	231	1990-1995	-5.505717

<u>What have I learned from this project?</u>

The head and tail function are my friends, and the lambda function is wonderful to get you out of coding pinches. This is my first-time using Python and while I would have liked to use more loops, my learning curve isn't quite there. The code that I have used is a bit clunky however it works. If I had more time, I would like to further investigate loops and extracting more information from the Annex and incorporating it into the tables.

I appreciated reusing code for the tables as this helped me refined the code as I reincorporated it for the tables. I understood how to be more concise and "elegant." And I know my code is far from elegant at the moment, I know where I can improve i.e. loops and more if/else coding.

Conclusion

What I find confounding about this data set is the fact that certain sums do not add up. Table 1, if you add up all the Eastern African countries the amounts are different. This is the case for every table. The same issue arises with the percentages. There is no base sum to factor these percentages. While I understand that I do not have the full picture for this dataset, there should be some basic principles that are upheld to allow any laymen to understand these values.

As I stated at the beginning, my goal with this dataset is to prepare it for a machine like Tableau or PowerBI for manipulation and visualization. The information is quite important to track migrant patterns therefore grouping of countries and regions is very important since migrants are a sovereign entity that pass over borders. My hope is this dataset is tidy and conveys the complex issues in the most simplistic way for the user to manipulate and investigate the data.