INF1340 Mid-term Project Write-up

Introduction:

I cleaned all 6 tables about Trends in International Migrant Stock by following all the five tidy data principles below:

Tidy data principle #1: Column names need to be informative, variable names and not values.

Tidy data principle #2: each column needs to consist of one and only one variable.

Tidy data principle #3: variables need to be in cells, not rows and columns.

Tidy data principle #4: each table column needs to have a singular data type.

Tidy data principle #5: a single observational units must be in 1 table.

My write-up contains all the code and specific descriptions for cleaning table1, and brief descriptions for table2 to table6 since there are some repetitive steps same to cleaning table1.

Data Cleaning for Table1:

1. First, I import some libraries we might use for data cleaning, and I read the table1 excel file into a pandas DataFrame by using the pandas.read_excel function. And I take a look at the head of the table realized that there are all NaN values since the table starts in row 15th in excel.

```
In [120]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns
          import plotly
          table1 = pd.read_excel(r'/Users/liangshuang/Desktop/table1.xlsx')

          table1.head()
```

Out[120]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnamed: 15	Un
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	United Nations	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	Population Division	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	

5 rows × 23 columns

2. So I drop those rows which are before the table starts. I use pandas.DataFrame.drop function. And I use the range(start, stop) function, the range never includes the stop number in its result. I put the range 0 to 15 since I want to delete the first 14 rows. Axis = 0 indicates dropping the rows.

```
In [121]: table1 = table1.drop(labels = range(0, 15), axis=0)
          table1
```

Out[121]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnam
15	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	...	87884839	97866674	11461371
16	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	...	50536796	57217777	6408107
17	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	...	37348043	40648897	5053263
18	4	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128.0	...	5361902	5383009	546271
19	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	56778501	59244124	64272611	79130668.0	...	31986141	35265888	4506992
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
275	261	Samoa	NaN	882	B	3357	4694	5998	5746	5122.0	...	3101	2940	259
276	262	Tokelau	NaN	772	B	270	266	262	258	429.0	...	144	133	20
277	263	Tonga	NaN	776	B	2911	3274	3684	4301	5022.0	...	1981	2328	272
278	264	Tuvalu	NaN	798	C	318	263	217	183	154.0	...	121	101	8
279	265	Wallis and Futuna Islands	NaN	876	B	1402	1680	2015	2365	2776.0	...	1018	1194	140

265 rows × 23 columns

3. Since the index starts from 15 and it is wrong, we want to change it starts from 0. I use pandas.DataFrame.reset_index function, the drop parameter is to avoid the old index being added as a column, so I set it to drop = True.

```
In [122]: table1 = table1.reset_index(drop = True)
          table1
```

Out[122]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnam
0	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	...	87884839	97866674	11461371
1	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	...	50536796	57217777	6408107
2	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	...	37348043	40648897	5053263
3	4	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128.0	...	5361902	5383009	546271
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	56778501	59244124	64272611	79130668.0	...	31986141	35265888	4506992
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
260	261	Samoa	NaN	882	B	3357	4694	5998	5746	5122.0	...	3101	2940	259
261	262	Tokelau	NaN	772	B	270	266	262	258	429.0	...	144	133	20
262	263	Tonga	NaN	776	B	2911	3274	3684	4301	5022.0	...	1981	2328	272
263	264	Tuvalu	NaN	798	C	318	263	217	183	154.0	...	121	101	8
264	265	Wallis and Futuna Islands	NaN	876	B	1402	1680	2015	2365	2776.0	...	1018	1194	140

265 rows × 23 columns

4. Now, the table is ready to follow the principles of cleaning.

Tidy data principle #1, column names need to be informative, it should be variable names. The column names are all unnamed, so it is definitely untidy. I will correct this problem after some steps. For now, I just want to define those unnamed columns to a simple name for manipulation.

```
In [123]: table1.columns = ["Sort order", "Major area, region, country or area of destination", "Notes", "Country code", "Type
          table1
```

```
"Type of data (a)", "1990B","1995B","2000B","2005B","2010B","2015B","1990M","1995M","2000M","2005M","2010M","2015M"
```

```
,"1990F","1995F", "2000F","2005F","2010F","2015F"]
```

Out[281]:

	Sort order	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	1990B	1995B	2000B	2005B	2010B	...	2000M	2005M	2010M	2015
0	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	...	87884839	97866674	114613714.0	126115435
1	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	...	50536796	57217777	64081077.0	67618619
2	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	...	37348043	40648897	50532637.0	58496816
3	4	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128.0	...	5361902	5383009	5462714.0	6463217
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	56778501	59244124	64272611	79130668.0	...	31986141	35265888	45069923.0	52033599
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
260	261	Samoa	NaN	882	B	3357	4694	5998	5746	5122.0	...	3101	2940	2594.0	2469
261	262	Tokelau	NaN	772	B	270	266	262	258	429.0	...	144	133	206.0	233
262	263	Tonga	NaN	776	B	2911	3274	3684	4301	5022.0	...	1981	2328	2727.0	3127
263	264	Tuvalu	NaN	798	C	318	263	217	183	154.0	...	121	101	85.0	78
264	265	Wallis and Futuna Islands	NaN	876	B	1402	1680	2015	2365	2776.0	...	1018	1194	1401.0	1438

265 rows × 23 columns

5. In the third column, “Notes” is just symbols representing comments and it should not include in the DataFrame, we have an information sheet called “NOTES” that would explain those symbols.

In [124]:
```
# "Notes" in the third column is just symbols represent for comments and it should not include in the dataframe, we
table1 = table1.drop(["Notes"], axis = 1)
table1
```

Out[124]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	1990B	1995B	2000B	2005B	2010B	2015B	...	2000M	2005M	2010M	2
0	1	WORLD	900	NaN	152563212	160801752	172703309	191269100	221714243.0	243700236.0	...	87884839	97866674	114613714.0	126115
1	2	Developed regions	901	NaN	82378628	92306854	103375363	117181109	132560325.0	140481955.0	...	50536796	57217777	64081077.0	67618
2	3	Developing regions	902	NaN	70184584	68494898	69327946	74087991	89153918.0	103218281.0	...	37348043	40648897	50532637.0	58496
3	4	Least developed countries	941	NaN	11075966	11711703	10077824	9809634	10018128.0	11951316.0	...	5361902	5383009	5462714.0	6463
4	5	Less developed regions excluding least develop...	934	NaN	59105261	56778501	59244124	64272611	79130668.0	91262036.0	...	31986141	35265888	45069923.0	52033
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
260	261	Samoa	882	B	3357	4694	5998	5746	5122.0	4929.0	...	3101	2940	2594.0	2
261	262	Tokelau	772	B	270	266	262	258	429.0	487.0	...	144	133	206.0	
262	263	Tonga	776	B	2911	3274	3684	4301	5022.0	5731.0	...	1981	2328	2727.0	3
263	264	Tuvalu	798	C	318	263	217	183	154.0	141.0	...	121	101	85.0	
264	265	Wallis and Futuna Islands	876	B	1402	1680	2015	2365	2776.0	2849.0	...	1018	1194	1401.0	1

265 rows × 22 columns

6. Tidy data principle #2: each column needs to consist of one and only one variable. I use pandas.DataFrame.columns and head function to check the columns and the table. And there are multiple variables stored in one column, such as columns “1990B”, “1995B” and so on. We have two variables, year and gender in one single column.

```
In [125]: #problem: there are multiple variables stored in 1 column
          #tidy data principle #2: each column needs to consist of one and only one variable
          table1.columns
```

```
Out[125]: Index(['Sort order', 'Major area, region, country or area of destination',
                 'Country code', 'Type of data (a)', '1990B', '1995B', '2000B', '2005B',
                 '2010B', '2015B', '1990M', '1995M', '2000M', '2005M', '2010M', '2015M',
                 '1990F', '1995F', '2000F', '2005F', '2010F', '2015F'],
                dtype='object')
```

7. Tidy data principle #3: variables need to be in cells, not rows and columns.

As you can see below, variables are stored in columns. We have year and gender distributed across the columns.

```
In [126]: #problem: variables are stored in both rows and columns
          #tidy data principle #3: variables need to be in cells, not rows and columns
          table1.head()
```

Out[126]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	1990B	1995B	2000B	2005B	2010B	2015B	...	2000M	2005M	2010M	201
0	1	WORLD	900	NaN	152563212	160801752	172703309	191269100	221714243.0	243700236.0	...	87884839	97866674	114613714.0	12611543
1	2	Developed regions	901	NaN	82378628	92306854	103375363	117181109	132560325.0	140481955.0	...	50536796	57217777	64081077.0	6761861
2	3	Developing regions	902	NaN	70184584	68494898	69327946	74087991	89153918.0	103218281.0	...	37348043	40648897	50532637.0	5849681
3	4	Least developed countries	941	NaN	11075966	11711703	10077824	9809634	10018128.0	11951316.0	...	5361902	5383009	5462714.0	646321
4	5	Less developed regions excluding least develop...	934	NaN	59105261	56778501	59244124	64272611	79130668.0	91262036.0	...	31986141	35265888	45069923.0	5203359

5 rows × 22 columns

8. To correct these two issues, I use pandas.melt function to unpivot the DataFrame from wide to long format, and then I use pandas.DataFrame.assign function to assign a new column for splitting the two variables that are in one column.

```
In [128]: table1 = table1.melt(id_vars = ["Sort order", "Major area, region, country or area of destination", "Country code",
          table1
```

```
"Type of data (a)"], value_vars = ["1990B","1995B","2000B","2005B", "2010B","2015B","1990M","1995M", "2000M","2005M"
```

```
"2010M","2015M","1990F","1995F", "2000F","2005F","2010F","2015F"])
```

Out[128]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	variable	value
0	1	WORLD	900	NaN	1990B	152563212
1	2	Developed regions	901	NaN	1990B	82378628
2	3	Developing regions	902	NaN	1990B	70184584
3	4	Least developed countries	941	NaN	1990B	11075966
4	5	Less developed regions excluding least develop...	934	NaN	1990B	59105261
...	...	...	...	...	...	...
4765	261	Samoa	882	B	2015F	2460.0
4766	262	Tokelau	772	B	2015F	254.0
4767	263	Tonga	776	B	2015F	2604.0
4768	264	Tuvalu	798	C	2015F	63.0
4769	265	Wallis and Futuna Islands	876	B	2015F	1411.0

4770 rows × 6 columns

(1) In pandas.melt function above, id_vars is the columns we use as identifier variables, value_vars is the columns we need to unpivot. When those variables are unpivoted to the row axis, we have two non-identifier columns, "variable" and "value".

```
In [129]: table1 = (table1.assign(Year = lambda x: x.variable.str[0:4].astype(int), Gender = lambda x: x.variable.str[4].astyp
          table1.head(10)
```

```
.astype(str)).drop("variable", axis = 1))
```

Out[129]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	value	Year	Gender
0	1	WORLD	900	NaN	152563212	1990	B
1	2	Developed regions	901	NaN	82378628	1990	B
2	3	Developing regions	902	NaN	70184584	1990	B
3	4	Least developed countries	941	NaN	11075966	1990	B
4	5	Less developed regions excluding least develop...	934	NaN	59105261	1990	B
5	6	Sub-Saharan Africa	947	NaN	14690319	1990	B
6	7	Africa	903	NaN	15690623	1990	B
7	8	Eastern Africa	910	NaN	5964031	1990	B
8	9	Burundi	108	B R	333110	1990	B
9	10	Comoros	174	B	14079	1990	B

(2) In pandas.DataFrame.assign function above, I applied the lambda function to create two columns ("Year" and "Gender") for splitting the year and gender under column "variable". Also, I change the data type of column "Year" to an integer using the .astype() function.

Now, the problems by tidy data principles #2 and #3 are solved. Each column consists of only one variable, and variables are in cells, not columns.

9. We can style the dataset to make it clearer and more informative.
   First, I use pandas.DataFrame.replace function to replace "B" to "Both sexes" under column "Gender" and so on.

```
In [130]: # Styling the dataset
          table1 = table1.replace({"Gender": {"B": "Both sexes", "F": "Female", "M": "Male"}})
          table1.sample(10)
```

Out[130]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	value	Year	Gender
3854	145	Iceland	352	B	8685	2000	Female
2670	21	Rwanda	646	B R	218425.0	2010	Male
2164	45	Tunisia	788	C	18523	2000	Male
2879	230	Uruguay	858	B	34507.0	2010	Male
1095	36	Equatorial Guinea	226	C	8658.0	2010	Both sexes
2823	174	Germany	276	B	5519698.0	2010	Male
4045	71	Asia	935	NaN	23785993	2005	Female
1434	110	Azerbaijan	31	B R	264241.0	2015	Both sexes
4524	20	Réunion	638	B	62672.0	2015	Female
3505	61	Guinea-Bissau	624	B R	14128	1995	Female

Second, I use pandas.DataFrame.rename function to change the column name to

make it informative. Then the problem by tidy principle #1 is solved.

```
In [131]: table1 = table1.rename({"value": "International migrant stock at mid-year"}, axis = 1)
          table1
```

Out[131]:

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

Moreover, I think the column “international migrant stock at mid-year” looks better in the last column of the DataFrame since it indicates the most important values, so I use pandas.DataFrame.iloc function to change the position of columns.

```
In [132]: #change position of columns
          table1 = table1.iloc[:, [0,1,2,3,5,6,4]]
          table1
```

Out[132]:

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

10. After styling the dataset, we need to take a look at the summary of the DataFrame by using the pandas.DataFrame.info function.

```
In [133]: table1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4770 entries, 0 to 4769
Data columns (total 7 columns):
 #   Column                                             Non-Null Count  Dtype 
---  ------                                             --------------  ----- 
 0   Sort order                                         4770 non-null   object
 1   Major area, region, country or area of destination  4770 non-null   object
 2   Country code                                       4770 non-null   object
 3   Type of data (a)                                   4176 non-null   object
 4   Year                                               4770 non-null   int64 
 5   Gender                                             4770 non-null   object
 6   International migrant stock at mid-year            4770 non-null   object
dtypes: int64(1), object(6)
memory usage: 261.0+ KB
```

The data types are not correct since there are some missing values shown by “..” in the DataFrame. So we need to change them to “NaN” and drop those meaningless missing values.

```
In [134]: # There are some missing value but shows by "..", so we change to "NaN" that we can drop those in the next step
          table1 = table1.replace("..", "NaN")

          # The missing values are meaningless, so we drop those rows include missing values
          table1 = table1.drop(table1[table1["International migrant stock at mid-year"] == "NaN"].index)
```

```
In [135]: table1.dtypes
```

```
Out[135]: Sort order                                          int64
          Major area, region, country or area of destination    object
          Country code                                         int64
          Type of data (a)                                    object
          Year                                                 int64
          Gender                                              object
          International migrant stock at mid-year             object
          dtype: object
```

Only the data type of column “International migrant stock at mid-year” is not correct so we change it by using pandas.DataFrame.astype function.

```
In [136]: table1 = table1.astype({"International migrant stock at mid-year": "float"})
          table1.dtypes # The types are all correct now
```

```
Out[136]: Sort order                                          int64
          Major area, region, country or area of destination    object
          Country code                                         int64
          Type of data (a)                                    object
          Year                                                 int64
          Gender                                              object
          International migrant stock at mid-year            float64
          dtype: object
```

Now, the data types are all correct.

11. We need to reset the index after we drop the missing values since the table shows there are 4725 rows but the index is 4769. We should make it consistent by using pandas.DataFrame.reset_index function as I mentioned above.

```
In [239]: table1
```

Out[239]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	International migrant stock at mid-year
0	1	WORLD	900	NaN	1990	Both sexes	152563212.0
1	2	Developed regions	901	NaN	1990	Both sexes	82378628.0
2	3	Developing regions	902	NaN	1990	Both sexes	70184584.0
3	4	Least developed countries	941	NaN	1990	Both sexes	11075966.0
4	5	Less developed regions excluding least develop...	934	NaN	1990	Both sexes	59105261.0
...	...	...	...	...	...	...	...
4765	261	Samoa	882	B	2015	Female	2460.0
4766	262	Tokelau	772	B	2015	Female	254.0
4767	263	Tonga	776	B	2015	Female	2604.0
4768	264	Tuvalu	798	C	2015	Female	63.0
4769	265	Wallis and Futuna Islands	876	B	2015	Female	1411.0

4725 rows × 7 columns

```
In [240]: table1 = table1.reset_index(drop=True)
          table1
```

Out[240]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	International migrant stock at mid-year
0	1	WORLD	900	NaN	1990	Both sexes	152563212.0
1	2	Developed regions	901	NaN	1990	Both sexes	82378628.0
2	3	Developing regions	902	NaN	1990	Both sexes	70184584.0
3	4	Least developed countries	941	NaN	1990	Both sexes	11075966.0
4	5	Less developed regions excluding least develop...	934	NaN	1990	Both sexes	59105261.0
...	...	...	...	...	...	...	...
4720	261	Samoa	882	B	2015	Female	2460.0
4721	262	Tokelau	772	B	2015	Female	254.0
4722	263	Tonga	776	B	2015	Female	2604.0
4723	264	Tuvalu	798	C	2015	Female	63.0
4724	265	Wallis and Futuna Islands	876	B	2015	Female	1411.0

4725 rows × 7 columns

12. When I look at the table, I realize that there are 4725 rows but the “Sort order” is just 265, since we unpivot the DataFrame, those identifier variables are repeated many times.
Tidy data principle #4: each table column needs to have a singular data type.
There is a problem that multiple types of data are stored in one table, so we want to split the observational units into two separate tables.

First, I stored the unique major area related combinations in t1 by using pandas.DataFrame.set_index function.

```
In [298]: t1 = table1.set_index(["Sort order", "Major area, region, country or area of destination", "Country code", "Type of data (a)"])
          t1 = t1.sort_values("Sort order")
          t1
```

Out[298]:

				Year	Gender	International migrant stock at mid-year
Sort order	**Major area, region, country or area of destination**	**Country code**	**Type of data (a)**			
1	**WORLD**	**900**	**NaN**	1990	Both sexes	152563212.0
			NaN	2000	Both sexes	172703309.0
			NaN	2005	Both sexes	191269100.0
			NaN	2010	Both sexes	221714243.0
			NaN	2015	Both sexes	243700236.0
...	**...**	**...**	**...**	...	...	...
265	**Wallis and Futuna Islands**	**876**	**B**	2015	Both sexes	2849.0
			B	1995	Both sexes	1680.0
			B	2010	Male	1401.0
			B	2015	Male	1438.0
			B	2015	Female	1411.0

4725 rows × 3 columns

Second, I set up the table “order1” for the unique major area related combinations by using the pandas.DataFrame.from_records function with a built-in function enumerate().

```
In [299]: order1 = pd.DataFrame.from_records(
              columns = ["id", "Sort order", "Major area, region, country or area of destination", "Country code", "Type of data (a)"],
              data = [
                  (a+1,b,c,d,e)
                  for (a,(b,c,d,e)) in enumerate(t1.index.unique())
              ],
          )
```

```
In [300]: order1
```

Out[300]:

	id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
0	1	1	WORLD	900	NaN
1	2	2	Developed regions	901	NaN
2	3	3	Developing regions	902	NaN
3	4	4	Least developed countries	941	NaN
4	5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...	...
260	261	261	Samoa	882	B
261	262	262	Tokelau	772	B
262	263	263	Tonga	776	B
263	264	264	Tuvalu	798	C
264	265	265	Wallis and Futuna Islands	876	B

265 rows × 5 columns

Similarly, I stored the “Year”, “Gender” and “International migrant stock at mid-year” in stock1. And I set the “id” to match with order1.

```
In [301]: stock1 = t1[["Year", "Gender", "International migrant stock at mid-year"]].copy()
stock1["id"] = order1.set_index(["Sort order", "Major area, region, country or area of destination", "Country code",
stock1.head(20)
```

Out[301]:

				Year	Gender	International migrant stock at mid-year	id
Sort order	**Major area, region, country or area of destination**	**Country code**	**Type of data (a)**				
1	**WORLD**	**900**	**NaN**	1990	Both sexes	152563212.0	1
			NaN	2000	Both sexes	172703309.0	1
			NaN	2005	Both sexes	191269100.0	1
			NaN	2010	Both sexes	221714243.0	1
			NaN	2015	Both sexes	243700236.0	1
			NaN	1990	Male	77747510.0	1
			NaN	1995	Male	81737477.0	1
			NaN	2000	Male	87884839.0	1
			NaN	2005	Male	97866674.0	1
			NaN	2015	Male	126115435.0	1
			NaN	1990	Female	74815702.0	1
			NaN	1995	Female	79064275.0	1
			NaN	2000	Female	84818470.0	1
			NaN	2005	Female	93402426.0	1
			NaN	2010	Male	114613714.0	1
			NaN	1995	Both sexes	160801752.0	1
			NaN	2010	Female	107100529.0	1
			NaN	2015	Female	117584801.0	1
2	**Developed regions**	**901**	**NaN**	1990	Both sexes	82378628.0	2
			NaN	2000	Both sexes	103375363.0	2

Finally, I set the “id” as the index in order1_tidy and stock1_tidy.

In [303]:
```
order1_tidy = order1.set_index("id")
order1_tidy
```

Out[303]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
261	261	Samoa	882	B
262	262	Tokelau	772	B
263	263	Tonga	776	B
264	264	Tuvalu	798	C
265	265	Wallis and Futuna Islands	876	B

265 rows × 4 columns

In [302]:
```
stock1_tidy = stock1.reset_index(drop = True).set_index("id")
stock1_tidy
```

Out[302]:

id	Year	Gender	International migrant stock at mid-year
1	1990	Both sexes	152563212.0
1	2000	Both sexes	172703309.0
1	2005	Both sexes	191269100.0
1	2010	Both sexes	221714243.0
1	2015	Both sexes	243700236.0
...	...	...	...
265	2015	Both sexes	2849.0
265	1995	Both sexes	1680.0
265	2010	Male	1401.0
265	2015	Male	1438.0
265	2015	Female	1411.0

4725 rows × 3 columns

13. Tidy data principle #5, a single observational unit must be in one table. The two tidy tables are not violated tidy data principle #5 since each table contains only one single observational unit.
    Finally, we have two cleaned tables for table1 by applying the tidy data principles above.

Data Cleaning for Table2:

Table2 repeated the exact same steps as table1 above, except table2 does not have the column “Type of data(a)” in table1. So, after doing the 13 steps above, we get the two cleaned tables for table2 by applying those five tidy data principles.

```
In [30]: order2_tidy = order2.set_index("id")
         order2_tidy
```

Out[30]:

id	Sort order	Major area, region, country or area of destination	Country code
1	1	WORLD	900
2	2	Developed regions	901
3	3	Developing regions	902
4	4	Least developed countries	941
5	5	Less developed regions excluding least develop...	934
...	...	...	...
261	261	Samoa	882
262	262	Tokelau	772
263	263	Tonga	776
264	264	Tuvalu	798
265	265	Wallis and Futuna Islands	876

265 rows × 3 columns

```
In [29]: pop2_tidy = pop2.reset_index(drop = True).set_index("id")
         pop2_tidy
```

Out[29]:

id	Year	Gender	Total population at mid-year (thousands)
1	1990	Both sexes	5309667.699
1	2000	Both sexes	6126622.121
1	2010	Female	3435768.139
1	2005	Both sexes	6519635.850
1	2005	Female	3234553.601
...	...	...	...
265	2000	Both sexes	14.497
265	2005	Both sexes	14.246
265	1995	Both sexes	14.143
265	1990	Both sexes	13.880
265	2015	Both sexes	13.151

4386 rows × 3 columns

Data Cleaning for Table3:

Table3 repeated the exact same steps in table1 above, moreover, we need to add one more step after step#10 in table1(styling the dataset and correcting the data types).

#The reason we add the rounding step is that the International migrant stock column in the DataFrame is not matching the table3 in excel. The values in that column are not rounded to 1 decimal place.

```
In [78]: table3 # The values in the last column need to round to 1 decimal place
```

Out[78]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	International migrant stock as a percentage of the total population
0	1	WORLD	900	NaN	1990	Both sexes	2.873310
1	2	Developed regions	901	NaN	1990	Both sexes	7.198015
2	3	Developing regions	902	NaN	1990	Both sexes	1.685021
3	4	Least developed countries	941	NaN	1990	Both sexes	2.171513
4	5	Less developed regions excluding least develop...	934	NaN	1990	Both sexes	1.617042
...	...	...	...	...	...	...	...
4756	252	Micronesia (Federated States of)	583	B	2015	Female	2.518155
4760	256	Polynesia	957	NaN	2015	Female	9.925472
4763	259	French Polynesia	258	B	2015	Female	9.332120
4765	261	Samoa	882	B	2015	Female	2.628654
4767	263	Tonga	776	B	2015	Female	4.919612

4343 rows × 7 columns

After we correct the data types to float, we can round to 1 decimal place by using the

pandas.DataFrame.round function and set decimals = 1.

```
In [19]: # We can round the values now under the "International migrant stock as a percentage of the total population" column
table3["International migrant stock as a percentage of the total population"] = table3["International migrant stock
table3
```

```
since the types are all correct.
as a percentage of the total population"].round(decimals = 1)
```

Out[19]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	International migrant stock as a percentage of the total population
0	1	WORLD	900	NaN	1990	Both sexes	2.9
1	2	Developed regions	901	NaN	1990	Both sexes	7.2
2	3	Developing regions	902	NaN	1990	Both sexes	1.7
3	4	Least developed countries	941	NaN	1990	Both sexes	2.2
4	5	Less developed regions excluding least develop...	934	NaN	1990	Both sexes	1.6
...	...	...	...	...	...	...	...
4756	252	Micronesia (Federated States of)	583	B	2015	Female	2.5
4760	256	Polynesia	957	NaN	2015	Female	9.9
4763	259	French Polynesia	258	B	2015	Female	9.3
4765	261	Samoa	882	B	2015	Female	2.6
4767	263	Tonga	776	B	2015	Female	4.9

4343 rows × 7 columns

Repeatedly, reset the index after we drop the “NaN”.

```
In [80]: table3 = table3.reset_index(drop=True)
table3
```

Out[80]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	International migrant stock as a percentage of the total population
0	1	WORLD	900	NaN	1990	Both sexes	2.9
1	2	Developed regions	901	NaN	1990	Both sexes	7.2
2	3	Developing regions	902	NaN	1990	Both sexes	1.7
3	4	Least developed countries	941	NaN	1990	Both sexes	2.2
4	5	Less developed regions excluding least develop...	934	NaN	1990	Both sexes	1.6
...	...	...	...	...	...	...	...
4338	252	Micronesia (Federated States of)	583	B	2015	Female	2.5
4339	256	Polynesia	957	NaN	2015	Female	9.9
4340	259	French Polynesia	258	B	2015	Female	9.3
4341	261	Samoa	882	B	2015	Female	2.6
4342	263	Tonga	776	B	2015	Female	4.9

4343 rows × 7 columns

So, after doing all the steps, we get the two cleaned tables for table3 by applying those five tidy data principles.

In [27]:
```
order3_tidy = order3.set_index("id")
order3_tidy
```

Out[27]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
261	261	Samoa	882	B
262	262	Tokelau	772	B
263	263	Tonga	776	B
264	264	Tuvalu	798	C
265	265	Wallis and Futuna Islands	876	B

265 rows × 4 columns

In [26]:
```
stock3_tidy = stock3.reset_index(drop = True).set_index("id")
stock3_tidy
```

Out[26]:

id	Year	Gender	International migrant stock as a percentage of the total population
1	1990	Both sexes	2.9
1	2015	Female	3.2
1	2000	Both sexes	2.8
1	2010	Female	3.1
1	2005	Both sexes	2.9
...	...	...	...
265	2015	Both sexes	21.7
265	1990	Both sexes	10.1
265	2005	Both sexes	16.6
265	2010	Both sexes	20.5
265	2000	Both sexes	13.9

4343 rows × 3 columns

Data Cleaning for Table4:

Table4 is similar to table3, besides the repeated steps of table1, we also add the rounding step after step#10 in table1.

In [42]:
```
table4 # The values in the last column need to round to 1 decimal place
```

Out[42]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female migrants as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	49.039150
1	2	Developed regions	901	NaN	1990	51.123977
2	3	Developing regions	902	NaN	1990	46.592099
3	4	Least developed countries	941	NaN	1990	47.261155
4	5	Less developed regions excluding least develop...	934	NaN	1990	46.466684
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	49.908704
1586	262	Tokelau	772	B	2015	52.156057
1587	263	Tonga	776	B	2015	45.437096
1588	264	Tuvalu	798	C	2015	44.680851
1589	265	Wallis and Futuna Islands	876	B	2015	49.526150

1575 rows × 6 columns

After we correct the data types to float, we can round to 1 decimal place by using the pandas.DataFrame.round function and set decimals = 1.

In [16]:
```
# We can round the values now under the "Female migrants as a percentage of the international migrant stock" column
table4["Female migrants as a percentage of the international migrant stock"] = table4["Female migrants as a percenta
table4
```

```
 column since the types are all correct
percentage of the international migrant stock"].round(decimals = 1)
```

Out[16]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female migrants as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	49.0
1	2	Developed regions	901	NaN	1990	51.1
2	3	Developing regions	902	NaN	1990	46.6
3	4	Least developed countries	941	NaN	1990	47.3
4	5	Less developed regions excluding least develop...	934	NaN	1990	46.5
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	49.9
1586	262	Tokelau	772	B	2015	52.2
1587	263	Tonga	776	B	2015	45.4
1588	264	Tuvalu	798	C	2015	44.7
1589	265	Wallis and Futuna Islands	876	B	2015	49.5

1575 rows × 6 columns

Repeatedly, reset the index after we drop the “NaN”.

In [44]:
```
table4 = table4.reset_index(drop=True)
table4
```

Out[44]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female migrants as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	49.0
1	2	Developed regions	901	NaN	1990	51.1
2	3	Developing regions	902	NaN	1990	46.6
3	4	Least developed countries	941	NaN	1990	47.3
4	5	Less developed regions excluding least develop...	934	NaN	1990	46.5
...	...	...	...	...	...	...
1570	261	Samoa	882	B	2015	49.9
1571	262	Tokelau	772	B	2015	52.2
1572	263	Tonga	776	B	2015	45.4
1573	264	Tuvalu	798	C	2015	44.7
1574	265	Wallis and Futuna Islands	876	B	2015	49.5

1575 rows × 6 columns

So, after doing all the steps, we get the two cleaned tables for table4 by applying those five tidy data principles.

In [24]:
```
order4_tidy = order4.set_index("id")
order4_tidy
```

Out[24]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
261	261	Samoa	882	B
262	262	Tokelau	772	B
263	263	Tonga	776	B
264	264	Tuvalu	798	C
265	265	Wallis and Futuna Islands	876	B

265 rows × 4 columns

In [23]:
```
stock4_tidy = stock4.reset_index(drop = True).set_index("id")
stock4_tidy
```

Out[23]:

id	Year	Female migrants as a percentage of the international migrant stock
1	1990	49.0
1	2000	49.1
1	2005	48.8
1	2015	48.2
1	2010	48.3
...	...	...
265	1995	48.9
265	2005	49.5
265	2010	49.5
265	2000	49.5
265	2015	49.5

1575 rows × 2 columns

Data Cleaning for Table5:

Table5 is similar to the first four tables, but a slightly different is that the “Year” of

the table5 represents time periods, not a specific year in the first table. So I just screenshot those important steps with a little change.

In [4]:
```
table5.columns = ["Sort order", "Major area, region, country or area of destination", "Notes", "Country code", "Type
table5
```

```
"Type of data (a)", "1990-1995B","1995-2000B","2000-2005B","2005-2010B","2010-2015B","1990-1995M","1995-2000M","200
```

```
"2000-2005M","2005-2010M","2010-2015M","1990-1995F","1995-2000F","2000-2005F","2005-2010F","2010-2015F"]
```

Out[4]:

	Sort order	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	1990-1995B	1995-2000B	2000-2005B	2005-2010B	2010-2015B	1990-1995M	1995-2000M	2000-2005M	2005-2010M	2010-2015M	
0	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
1	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
2	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
3	4	Least developed countries	(d)	941	NaN	1.118175	-3.001139	-0.539636	0.419137	3.526927	1.000073	-2.718952	0.078575	0.293964	3.363629	1
4	5	Less developed regions excluding least develop...	NaN	934	NaN	-0.803244	0.850177	1.62934	4.159339	2.852687	-0.733256	0.950231	1.952269	4.90598	2.87349	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
260	261	Samoa	NaN	882	B	6.704748	4.90282	-0.858442	-2.299179	-0.768177	6.499035	4.704571	-1.066301	-2.50417	-0.987758	6
261	262	Tokelau	NaN	772	B	-0.298513	-0.303036	-0.307698	10.169947	2.536144	-0.404054	-0.412386	-1.589283	8.750541	2.463246	-0
262	263	Tonga	NaN	776	B	2.350316	2.359733	3.096969	3.099614	2.641235	2.874558	2.848819	3.228155	3.163851	2.737439	1
263	264	Tuvalu	NaN	798	C	-3.797947	-3.845134	-3.408224	-3.450671	-1.763854	-3.914892	-4.028435	-3.613401	-3.449385	-1.718849	-3
264	265	Wallis and Futuna Islands	NaN	876	B	3.61788	3.636508	3.203177	3.20466	0.51914	3.364378	3.396526	3.189382	3.197545	0.52134	3

265 rows × 20 columns

In [9]:
```
table5 = table5.melt(id_vars = ["Sort order", "Major area, region, country or area of destination", "Country code",
table5
```

```
"Type of data (a)"], value_vars = ["1990-1995B","1995-2000B","2000-2005B","2005-2010B","2010-2015B","1990-1995M",
```

```
,"1995-2000M","2000-2005M","2005-2010M","2010-2015M","1990-1995F","1995-2000F","2000-2005F","2005-2010F",
```

```
"2010-2015F"])
```

Out[9]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	variable	value
0	1	WORLD	900	NaN	1990-1995B	1.051865
1	2	Developed regions	901	NaN	1990-1995B	2.275847
2	3	Developing regions	902	NaN	1990-1995B	-0.487389
3	4	Least developed countries	941	NaN	1990-1995B	1.118175
4	5	Less developed regions excluding least develop...	934	NaN	1990-1995B	-0.803244
...	...	...	...	...	...	...
3970	261	Samoa	882	B	2010-2015F	-0.545343
3971	262	Tokelau	772	B	2010-2015F	2.60325
3972	263	Tonga	776	B	2010-2015F	2.526318
3973	264	Tuvalu	798	C	2010-2015F	-1.819436
3974	265	Wallis and Futuna Islands	876	B	2010-2015F	0.516899

3975 rows × 6 columns

In [10]:
```
table5 = (table5.assign(Year = lambda x: x.variable.str[0:9].astype(str), Gender = lambda x: x.variable.str[9].astyp
table5.head(10)
```

```
.astype(str), Gender = lambda x: x.variable.str[9].astype(str)).drop("variable", axis = 1))
```

Out[10]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	value	Year	Gender
0	1	WORLD	900	NaN	1.051865	1990-1995	B
1	2	Developed regions	901	NaN	2.275847	1990-1995	B
2	3	Developing regions	902	NaN	-0.487389	1990-1995	B
3	4	Least developed countries	941	NaN	1.118175	1990-1995	B
4	5	Less developed regions excluding least develop...	934	NaN	-0.803244	1990-1995	B
5	6	Sub-Saharan Africa	947	NaN	0.845374	1990-1995	B
6	7	Africa	903	NaN	0.826734	1990-1995	B
7	8	Eastern Africa	910	NaN	-3.435412	1990-1995	B
8	9	Burundi	108	B R	-5.355717	1990-1995	B
9	10	Comoros	174	B	-0.199873	1990-1995	B

#After Styling the dataset and changing the data types, we do the rounding.

In [48]:
```
table5 # The values in the last column need to round to 2 decimal places
```

Out[48]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	Annual rate of change of the migrant stock
0	1	WORLD	900	NaN	1990-1995	Both sexes	1.051865
1	2	Developed regions	901	NaN	1990-1995	Both sexes	2.275847
2	3	Developing regions	902	NaN	1990-1995	Both sexes	-0.487389
3	4	Least developed countries	941	NaN	1990-1995	Both sexes	1.118175
4	5	Less developed regions excluding least develop...	934	NaN	1990-1995	Both sexes	-0.803244
...	...	...	...	...	...	...	...
3970	261	Samoa	882	B	2010-2015	Female	-0.545343
3971	262	Tokelau	772	B	2010-2015	Female	2.603250
3972	263	Tonga	776	B	2010-2015	Female	2.526318
3973	264	Tuvalu	798	C	2010-2015	Female	-1.819436
3974	265	Wallis and Futuna Islands	876	B	2010-2015	Female	0.516899

3930 rows × 7 columns

After we correct the data types to float, we can round to 2 decimal place by using the pandas.DataFrame.round function and set decimals = 2.

```
In [49]: # We can round the values now under the "Annual rate of change of the migrant stock" column since the types are a
         table5["Annual rate of change of the migrant stock"] = table5["Annual rate of change of the migrant stock"].round
         table5
```

```
ll correct
(decimals = 2)
```

Out[49]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	Annual rate of change of the migrant stock
0	1	WORLD	900	NaN	1990-1995	Both sexes	1.05
1	2	Developed regions	901	NaN	1990-1995	Both sexes	2.28
2	3	Developing regions	902	NaN	1990-1995	Both sexes	-0.49
3	4	Least developed countries	941	NaN	1990-1995	Both sexes	1.12
4	5	Less developed regions excluding least develop...	934	NaN	1990-1995	Both sexes	-0.80
...	...	...	...	...	...	...	...
3970	261	Samoa	882	B	2010-2015	Female	-0.55
3971	262	Tokelau	772	B	2010-2015	Female	2.60
3972	263	Tonga	776	B	2010-2015	Female	2.53
3973	264	Tuvalu	798	C	2010-2015	Female	-1.82
3974	265	Wallis and Futuna Islands	876	B	2010-2015	Female	0.52

3930 rows × 7 columns

Repeatedly, reset the index after we drop the “NaN”.

```
In [50]: table5 = table5.reset_index(drop=True)
         table5
```

Out[50]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Gender	Annual rate of change of the migrant stock
0	1	WORLD	900	NaN	1990-1995	Both sexes	1.05
1	2	Developed regions	901	NaN	1990-1995	Both sexes	2.28
2	3	Developing regions	902	NaN	1990-1995	Both sexes	-0.49
3	4	Least developed countries	941	NaN	1990-1995	Both sexes	1.12
4	5	Less developed regions excluding least develop...	934	NaN	1990-1995	Both sexes	-0.80
...	...	...	...	...	...	...	...
3925	261	Samoa	882	B	2010-2015	Female	-0.55
3926	262	Tokelau	772	B	2010-2015	Female	2.60
3927	263	Tonga	776	B	2010-2015	Female	2.53
3928	264	Tuvalu	798	C	2010-2015	Female	-1.82
3929	265	Wallis and Futuna Islands	876	B	2010-2015	Female	0.52

3930 rows × 7 columns

So, after doing all the steps, we get the two cleaned tables for table5 by applying those five tidy data principles.

In [57]:
```
order5_tidy = order5.set_index("id")
order5_tidy
```

Out[57]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
261	261	Samoa	882	B
262	262	Tokelau	772	B
263	263	Tonga	776	B
264	264	Tuvalu	798	C
265	265	Wallis and Futuna Islands	876	B

265 rows × 4 columns

In [56]:
```
stock5_tidy = stock5.reset_index(drop = True).set_index("id")
stock5_tidy
```

Out[56]:

id	Year	Gender	Annual rate of change of the migrant stock
1	1990-1995	Both sexes	1.05
1	2000-2005	Male	2.15
1	2010-2015	Male	1.91
1	1995-2000	Male	1.45
1	1990-1995	Female	1.10
...	...	...	...
265	1995-2000	Both sexes	3.64
265	1990-1995	Male	3.36
265	2005-2010	Male	3.20
265	1990-1995	Female	3.89
265	2010-2015	Female	0.52

3930 rows × 3 columns

Data Cleaning for Table6:

In table6, it is a bit different than the first 5 tables since there are three different variables in one table, we would like to split it into several tables.

There are also some repetitive steps, so I just screenshot those steps are different.

1. First, I would like to split into two tables (table6_1 and table6_2) by using pandas.DataFrame.drop function, table6_1 is about the “Estimated refugee stock at mid-year (both sexes)” and “Refugees as a percentage of the international migrant stock”, table6_2 is about “Annual rate of change of the refugee stock”. Since the first table would contain the specific year under “Year” column, and the second table would contain the time periods under “Year” column.

```
In [578]: # We want to split table6 into two tables(table6_1 and table6_2) since some columns are related to specific years bu
          table6_1 = table6.drop(table6.columns[[17, 18, 19, 20, 21]],axis = 1)
          table6_1
```

Out[578]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	U
0	1	WORLD	NaN	900	NaN	18836571	17853840	15827803	13276733	15370755.0	19577474.0	12.346732	11.103013	9
1	2	Developed regions	(b)	901	NaN	2014564	3609670	2997256	2361229	2046917.0	1954224.0	2.445494	3.910511	2
2	3	Developing regions	(c)	902	NaN	16822007	14244170	12830547	10915504	13323838.0	17623250.0	23.968236	20.795958	18
3	4	Least developed countries	(d)	941	NaN	5048391	5160131	3047488	2363782	1957884.0	3443582.0	45.56588	44.041961	3(
4	5	Less developed regions excluding least develop...	NaN	934	NaN	11773616	9084039	9783059	8551722	11365954.0	14179668.0	19.919743	15.999082	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
260	261	Samoa	NaN	882	B	0	0	0	0	0.0	0.0	0	0	
261	262	Tokelau	NaN	772	B	0	0	0	0	0.0	0.0	0	0	
262	263	Tonga	NaN	776	B	0	0	0	0	0.0	0.0	0	0	
263	264	Tuvalu	NaN	798	C	0	0	0	0	0.0	0.0	0	0	
264	265	Wallis and Futuna Islands	NaN	876	B	0	0	0	0	0.0	0.0	0	0	

265 rows × 17 columns

2. # *Define column names.* I will correct column names problem following principle #1 after some steps.

```
In [579]: table6_1.columns = ["Sort order", "Major area, region, country or area of destination", "Notes", "Country code", "Ty
          table6_1
```

```
"Type of data (a)", "1990E","1995E","2000E","2005E","2010E","2015E","1990R","1995R","2000R","2005R","2010R","2015R"]
```

Out[579]:

	Sort order	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	1990E	1995E	2000E	2005E	2010E	2015E	1990R	1995R	2000R	2005R
0	1	WORLD	NaN	900	NaN	18836571	17853840	15827803	13276733	15370755.0	19577474.0	12.346732	11.103013	9.164736	6.941389
1	2	Developed regions	(b)	901	NaN	2014564	3609670	2997256	2361229	2046917.0	1954224.0	2.445494	3.910511	2.899391	2.015025
2	3	Developing regions	(c)	902	NaN	16822007	14244170	12830547	10915504	13323838.0	17623250.0	23.968236	20.795958	18.507035	14.733162
3	4	Least developed countries	(d)	941	NaN	5048391	5160131	3047488	2363782	1957884.0	3443582.0	45.56588	44.041961	30.221557	24.08243
4	5	Less developed regions excluding least develop...	NaN	934	NaN	11773616	9084039	9783059	8551722	11365954.0	14179668.0	19.919743	15.999082	16.51313	13.305391
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
260	261	Samoa	NaN	882	B	0	0	0	0	0.0	0.0	0	0	0	0
261	262	Tokelau	NaN	772	B	0	0	0	0	0.0	0.0	0	0	0	0
262	263	Tonga	NaN	776	B	0	0	0	0	0.0	0.0	0	0	0	0
263	264	Tuvalu	NaN	798	C	0	0	0	0	0.0	0.0	0	0	0	0
264	265	Wallis and Futuna Islands	NaN	876	B	0	0	0	0	0.0	0.0	0	0	0	0

265 rows × 17 columns

3.

Tidy data principle #2: each column needs to consist of one and only one variable.
Tidy data principle #3: variables need to be in cells, not rows and columns.
To resolve the issues by these principles, I also use the use the same methods of cleaning table1. But for table6_1, I did melt twice individually for the two columns.

(1) Melt for column “Estimated refugee stock at mid-year (both sexes)”.
Table “est” represents Estimated refugee stock at mid-year (both sexes).

In [584]:
```
est = table6_1.melt(id_vars = ["Sort order", "Major area, region, country or area of destination", "Country code",
"Type of data (a)"], value_vars = ["1990E","1995E","2000E","2005E","2010E","2015E"], value_name = "Estimated refugee
                                                                                                                    "Estimated refugee stock at mid-year (both sexes)")
est
```

Out[584]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	variable	Estimated refugee stock at mid-year (both sexes)
0	1	WORLD	900	NaN	1990E	18836571
1	2	Developed regions	901	NaN	1990E	2014564
2	3	Developing regions	902	NaN	1990E	16822007
3	4	Least developed countries	941	NaN	1990E	5048391
4	5	Less developed regions excluding least develop...	934	NaN	1990E	11773616
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015E	0.0
1586	262	Tokelau	772	B	2015E	0.0
1587	263	Tonga	776	B	2015E	0.0
1588	264	Tuvalu	798	C	2015E	0.0
1589	265	Wallis and Futuna Islands	876	B	2015E	0.0

1590 rows × 6 columns

 # Get rid of the letter “E” under “variable” column

In [585]:
```
est["variable"] = est["variable"].str.replace("E", "").astype(int)
est
```

Out[585]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	variable	Estimated refugee stock at mid-year (both sexes)
0	1	WORLD	900	NaN	1990	18836571
1	2	Developed regions	901	NaN	1990	2014564
2	3	Developing regions	902	NaN	1990	16822007
3	4	Least developed countries	941	NaN	1990	5048391
4	5	Less developed regions excluding least develop...	934	NaN	1990	11773616
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	0.0
1586	262	Tokelau	772	B	2015	0.0
1587	263	Tonga	776	B	2015	0.0
1588	264	Tuvalu	798	C	2015	0.0
1589	265	Wallis and Futuna Islands	876	B	2015	0.0

1590 rows × 6 columns

 # Rename the "variable" column to "Year".

Now, the problem by tidy principle #1, #2, #3 are all solved for table "est".

In [586]:
```
est = est.rename({"variable": "Year"}, axis = 1)
est
```

Out[586]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Estimated refugee stock at mid-year (both sexes)
0	1	WORLD	900	NaN	1990	18836571
1	2	Developed regions	901	NaN	1990	2014564
2	3	Developing regions	902	NaN	1990	16822007
3	4	Least developed countries	941	NaN	1990	5048391
4	5	Less developed regions excluding least develop...	934	NaN	1990	11773616
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	0.0
1586	262	Tokelau	772	B	2015	0.0
1587	263	Tonga	776	B	2015	0.0
1588	264	Tuvalu	798	C	2015	0.0
1589	265	Wallis and Futuna Islands	876	B	2015	0.0

1590 rows × 6 columns

(2) Similarly, we melt for column "Refugees as a percentage of the international migrant stock".

Table "refu" represents Refugees as a percentage of the international migrant stock.

In [587]:
```
refu = table6_1.melt(id_vars = ["Sort order", "Major area, region, country or area of destination", "Country code",
"Type of data (a)"], value_vars = ["1990R","1995R","2000R","2005R","2010R","2015R"], value_name = "Refugees as a
"Refugees as a percentage of the international migrant stock")
refu
```

Out[587]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	variable	Refugees as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990R	12.346732
1	2	Developed regions	901	NaN	1990R	2.445494
2	3	Developing regions	902	NaN	1990R	23.968236
3	4	Least developed countries	941	NaN	1990R	45.56588
4	5	Less developed regions excluding least develop...	934	NaN	1990R	19.919743
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015R	0.0
1586	262	Tokelau	772	B	2015R	0.0
1587	263	Tonga	776	B	2015R	0.0
1588	264	Tuvalu	798	C	2015R	0.0
1589	265	Wallis and Futuna Islands	876	B	2015R	0.0

1590 rows × 6 columns

 # Get rid of the letter "R" under "variable" column

In [588]:
```
refu["variable"] = refu["variable"].str.replace("R", "").astype(int)
refu
```

Out[588]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	variable	Refugees as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	12.346732
1	2	Developed regions	901	NaN	1990	2.445494
2	3	Developing regions	902	NaN	1990	23.968236
3	4	Least developed countries	941	NaN	1990	45.56588
4	5	Less developed regions excluding least develop...	934	NaN	1990	19.919743
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	0.0
1586	262	Tokelau	772	B	2015	0.0
1587	263	Tonga	776	B	2015	0.0
1588	264	Tuvalu	798	C	2015	0.0
1589	265	Wallis and Futuna Islands	876	B	2015	0.0

1590 rows × 6 columns

 # Rename the "variable" column to "Year".

Now, the problem by tidy principle #1, #2, #3 are all solved for table "rufu".

In [589]:
```
refu = refu.rename({"variable": "Year"}, axis = 1)
refu
```

Out[589]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Refugees as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	12.346732
1	2	Developed regions	901	NaN	1990	2.445494
2	3	Developing regions	902	NaN	1990	23.968236
3	4	Least developed countries	941	NaN	1990	45.56588
4	5	Less developed regions excluding least develop...	934	NaN	1990	19.919743
...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	0.0
1586	262	Tokelau	772	B	2015	0.0
1587	263	Tonga	776	B	2015	0.0
1588	264	Tuvalu	798	C	2015	0.0
1589	265	Wallis and Futuna Islands	876	B	2015	0.0

1590 rows × 6 columns

4. Tidy data principle 5: a single observational units must be in 1 table. So I merge the table “est” and “refu” since one observational unit stored in two tables.

```
In [590]: # tidy data principle 5: a single observational units must be in 1 table
          table6_1 = pd.merge(est, refu)
          table6_1
```

Out[590]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Estimated refugee stock at mid-year (both sexes)	Refugees as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	18836571	12.346732
1	2	Developed regions	901	NaN	1990	2014564	2.445494
2	3	Developing regions	902	NaN	1990	16822007	23.968236
3	4	Least developed countries	941	NaN	1990	5048391	45.56588
4	5	Less developed regions excluding least develop...	934	NaN	1990	11773616	19.919743
...	...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	0.0	0.0
1586	262	Tokelau	772	B	2015	0.0	0.0
1587	263	Tonga	776	B	2015	0.0	0.0
1588	264	Tuvalu	798	C	2015	0.0	0.0
1589	265	Wallis and Futuna Islands	876	B	2015	0.0	0.0

1590 rows × 7 columns

5. we need to take a look at the summary of the DataFrame by using the pandas.DataFrame.info function.

```
In [591]: table6_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1590 entries, 0 to 1589
Data columns (total 7 columns):
 #   Column                                                  Non-Null Count  Dtype 
---  ------                                                  --------------  ----- 
 0   Sort order                                              1590 non-null   object
 1   Major area, region, country or area of destination     1590 non-null   object
 2   Country code                                            1590 non-null   object
 3   Type of data (a)                                        1392 non-null   object
 4   Year                                                    1590 non-null   int64 
 5   Estimated refugee stock at mid-year (both sexes)        1590 non-null   object
 6   Refugees as a percentage of the international migrant stock  1590 non-null   object
dtypes: int64(1), object(6)
memory usage: 99.4+ KB
```

6. The data types are not correct since there are some missing values shown by “..” in the DataFrame. So we need to change them to “NaN” and drop those meaningless missing values. *We only drop the rows that the values of both last two columns are “NaN”.

```
In [592]: # There are some missing value but shows by "..", so we change to "NaN" to make it more clear
          table6_1 = table6_1.replace("..", "NaN")

          # We only drop the rows that the values of both last two columns are “NaN”.
          NaN_values = table6_1[(table6_1['Estimated refugee stock at mid-year (both sexes)'] == 'NaN') & (table6_1['Refugees
          table6_1.drop(NaN_values , inplace=True)

          table6_1
```

```
'Refugees as a percentage of the international migrant stock'] == "NaN")].index
```

```
In [653]: table6_1.dtypes

Out[653]: Sort order                                                  int64
          Major area, region, country or area of destination         object
          Country code                                                int64
          Type of data (a)                                           object
          Year                                                        int64
          Estimated refugee stock at mid-year (both sexes)           object
          Refugees as a percentage of the international migrant stock object
          dtype: object
```

Since there is no “NaN” in the column “Estimated refugee stock at mid-year (both sexes)”, we change the data type to float.

Since we need to round to 1 decimal place for the values of column “Refugees as a percentage of the international migrant stock” in the next step, we change the data type to float.

```
In [733]: table6_1 = table6_1.astype({"Estimated refugee stock at mid-year (both sexes)": "float"})
          table6_1 = table6_1.astype({"Refugees as a percentage of the international migrant stock": "float"})

In [734]: table6_1.dtypes # The types are all correct now

Out[734]: Sort order                                                  int64
          Major area, region, country or area of destination         object
          Country code                                                int64
          Type of data (a)                                           object
          Year                                                        int64
          Estimated refugee stock at mid-year (both sexes)           float64
          Refugees as a percentage of the international migrant stock float64
          dtype: object
```

After we correct the data types to float, we can round to 1 decimal place by using the pandas.DataFrame.round function and set decimals = 1.

I use the pandas.DataFrame.round function and set decimals = 1.

```
In [735]: # We can round the values now under the "Refugees as a percentage of the international migrant stock" column since t
          table6_1["Refugees as a percentage of the international migrant stock"] = table6_1["Refugees as a percentage of the
          table6_1
```

```
since the types are all correct.
of the international migrant stock"].round(decimals = 1)
```

Out[735]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Estimated refugee stock at mid-year (both sexes)	Refugees as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	18836571.0	12.3
1	2	Developed regions	901	NaN	1990	2014564.0	2.4
2	3	Developing regions	902	NaN	1990	16822007.0	24.0
3	4	Least developed countries	941	NaN	1990	5048391.0	45.6
4	5	Less developed regions excluding least develop...	934	NaN	1990	11773616.0	19.9
...	...	...	...	...	...	...	...
1585	261	Samoa	882	B	2015	0.0	0.0
1586	262	Tokelau	772	B	2015	0.0	0.0
1587	263	Tonga	776	B	2015	0.0	0.0
1588	264	Tuvalu	798	C	2015	0.0	0.0
1589	265	Wallis and Futuna Islands	876	B	2015	0.0	0.0

1579 rows × 7 columns

Repeatedly, reset the index after we drop the “NaN”.

```
In [736]: table6_1 = table6_1.reset_index(drop = True)
          table6_1
```

Out[736]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Estimated refugee stock at mid-year (both sexes)	Refugees as a percentage of the international migrant stock
0	1	WORLD	900	NaN	1990	18836571.0	12.3
1	2	Developed regions	901	NaN	1990	2014564.0	2.4
2	3	Developing regions	902	NaN	1990	16822007.0	24.0
3	4	Least developed countries	941	NaN	1990	5048391.0	45.6
4	5	Less developed regions excluding least develop...	934	NaN	1990	11773616.0	19.9
...	...	...	...	...	...	...	...
1574	261	Samoa	882	B	2015	0.0	0.0
1575	262	Tokelau	772	B	2015	0.0	0.0
1576	263	Tonga	776	B	2015	0.0	0.0
1577	264	Tuvalu	798	C	2015	0.0	0.0
1578	265	Wallis and Futuna Islands	876	B	2015	0.0	0.0

1579 rows × 7 columns

7. Tidy data principle #4: each table column needs to have a singular data type.
We have the same exact steps(starts from #12) in table1.

Tidy data principle #5, a single observational unit must be in one table. The following two tidy tables are not violated tidy data principle #5 since each table contains only one single observational unit.

After doing all the steps, we get the two cleaned tables for table6_1 by applying those five tidy data principles.

```
In [743]: order6_1_tidy = order6_1.set_index("id")
          order6_1_tidy
```

Out[743]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
261	261	Samoa	882	B
262	262	Tokelau	772	B
263	263	Tonga	776	B
264	264	Tuvalu	798	C
265	265	Wallis and Futuna Islands	876	B

265 rows × 4 columns

```
In [742]: refugee6_1_tidy = refugee6_1.reset_index(drop = True).set_index("id")
          refugee6_1_tidy
```

Out[742]:

id	Year	Estimated refugee stock at mid-year (both sexes)	Refugees as a percentage of the international migrant stock
1	1990	18836571.0	12.3
1	2000	15827803.0	9.2
1	2005	13276733.0	6.9
1	2015	19577474.0	8.0
1	2010	15370755.0	6.9
...	...	...	...
265	2010	0.0	0.0
265	1995	0.0	0.0
265	2005	0.0	0.0
265	1990	0.0	0.0
265	2015	0.0	0.0

1579 rows × 3 columns

8. Table6_2 is the final table that we need to clean. Similarly, we can use the same exact steps in table1.

```
In [746]: table6_2 = table6.drop(table6.iloc[:, 5:17],axis = 1)
          table6_2
```

Out[746]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 17	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnamed: 21
0	1	WORLD	NaN	900	NaN	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
1	2	Developed regions	(b)	901	NaN	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
2	3	Developing regions	(c)	902	NaN	-2.839417	-2.332154	-4.561	0.285195	2.663652
3	4	Least developed countries	(d)	941	NaN	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
4	5	Less developed regions excluding least develop...	NaN	934	NaN	-4.3836	0.632489	-4.319731	1.530456	1.571047
...	...	...	...	...	...	...	...	...	...	...
260	261	Samoa	NaN	882	B	..	..	..	..	..
261	262	Tokelau	NaN	772	B	..	..	..	..	..
262	263	Tonga	NaN	776	B	..	..	..	..	..
263	264	Tuvalu	NaN	798	C	..	..	..	..	..
264	265	Wallis and Futuna Islands	NaN	876	B	..	..	..	..	..

265 rows × 10 columns

9. After doing all the steps, we get the two cleaned tables for table6_2 by applying those five tidy data principles.

In [767]:
```
order6_2_tidy = order6_2.set_index("id")
order6_2_tidy
```

Out[767]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
195	241	New Zealand	554	B
196	242	Melanesia	928	NaN
197	243	Fiji	242	B
198	245	Papua New Guinea	598	C R
199	247	Vanuatu	548	B

199 rows × 4 columns

In [766]:
```
rate6_2_tidy = rate6_2.reset_index(drop = True).set_index("id")
rate6_2_tidy
```

Out[766]:

id	Year	Annual rate of change of the refugee stock
1	1990-1995	-2.12
1	2000-2005	-5.56
1	1995-2000	-3.84
1	2010-2015	2.95
1	2005-2010	-0.03
...	...	...
198	2000-2005	7.13
198	1995-2000	-4.08
198	2010-2015	0.14
198	2005-2010	-11.82
199	2010-2015	-15.13

890 rows × 2 columns

10. Since order6_1_tidy and order6_2_tidy are the same, we can just keep only one of them.

In [768]:
```
# Since order6_1_tidy and order6_2_tidy are the same, we can just keep only one of them.
order6_tidy = order6_2_tidy
order6_tidy
```

Out[768]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
195	241	New Zealand	554	B
196	242	Melanesia	928	NaN
197	243	Fiji	242	B
198	245	Papua New Guinea	598	C R
199	247	Vanuatu	548	B

199 rows × 4 columns

11. In conclusion for table6, we have three separate cleaned tables, order6_tidy, refugee6_1_tidy and rate6_2_tidy by applying all five principles.

In [829]:
```
order6_tidy
```

Out[829]:

id	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)
1	1	WORLD	900	NaN
2	2	Developed regions	901	NaN
3	3	Developing regions	902	NaN
4	4	Least developed countries	941	NaN
5	5	Less developed regions excluding least develop...	934	NaN
...	...	...	...	...
195	241	New Zealand	554	B
196	242	Melanesia	928	NaN
197	243	Fiji	242	B
198	245	Papua New Guinea	598	C R
199	247	Vanuatu	548	B

199 rows × 4 columns

In [830]:
```
refugee6_1_tidy
```

Out[830]:

id	Year	Estimated refugee stock at mid-year (both sexes)	Refugees as a percentage of the international migrant stock
1	1990	18836571.0	12.3
1	2000	15827803.0	9.2
1	2005	13276733.0	6.9
1	2015	19577474.0	8.0
1	2010	15370755.0	6.9
...	...	...	...
265	2010	0.0	0.0
265	1995	0.0	0.0
265	2005	0.0	0.0
265	1990	0.0	0.0
265	2015	0.0	0.0

1579 rows × 3 columns

In [831]:
```
rate6_2_tidy
```

Out[831]:

id	Year	Annual rate of change of the refugee stock
1	1990-1995	-2.12
1	2000-2005	-5.56
1	1995-2000	-3.84
1	2010-2015	2.95
1	2005-2010	-0.03
...	...	...
198	2000-2005	7.13
198	1995-2000	-4.08
198	2010-2015	0.14
198	2005-2010	-11.82
199	2010-2015	-15.13

890 rows × 2 columns

Conclusion:

After I cleaned all these datasets, I found out that data cleaning is not as easy as I imagined. I need to diagnose which principle the dataset violates and how to make it right. Through this process, I learned how to narrower the dataset and make it easier to compute in future analysis. In the cleaning process, the most interesting part and the trickiest part that confused me until now is how to balance the

convenience of coding and the cleanness of the visualization. I enjoy this cleaning process and hope we can learn more in the future.