Good job cleaning and explaining your steps and results by tidy data principle. Your write up was missing a clear headings including the introduction, discussion, and conclusion. 18/20

This UN dataset contains a total of nine sheets with six of them containing real data and the remaining sheets are classifications and explanations for people who want to work with this dataset.

This dataset provides the trends in international migrant and refugee stocks and across different regions and major areas from the year 1990 to 2015 by different sexes. Sheets containing data to work with are:

1. Table 1 - International migrant stock at mid-year by sex and by major area, region, country or area, 1990-2015
2. Table 2 - Total population at mid-year by sex and by major area, region, country or area, 1990-2015 (thousands)
3. Table 3 - International migrant stock as a percentage of the total population by sex and by major area, region, country or area, 1990-2015
4. Table 4 - Female migrants as a percentage of the international migrant stock by major area, region, country or area, 1990-2015
5. Table 5 - Annual rate of change of the migrant stock by sex and by major area, region, country or area, 1990-2015 (percentage)
6. Table 6 - Estimated refugee stock at mid-year by major area, region, country or area, 1990-2015

There are some common problems are identified for all six of the sheets, and these problems will be addressed while using tidy data principles in the data-wrangling process. Moreover, all excel tables contain a few rows of context information, these will be skipped when cleaning the dataset. All missing values will be replaced by NA. The data-wrangling process will use pandas as the main method and pandas is installed and ready to use.

Moreover, based on the context information, this excel sheet contains tables in three aspects with various measurements in each table, namely international migrants, total population and refuges stocks. There will be three main tables merged based the themes at the end.

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

This table contains 23 columns and 266 rows (skip the first 14 context rows). Before starting the wrangling process following tidy data principles, I found that *sort order* and *country code* columns have one decimal placed on each value in the cells after jupyter reads the table and this problem will be addressed and fixed in the following process.

```
#INF1340 DATA CELANING
import pandas as pd

undata = pd.ExcelFile("UN_MigrantStockTotal_2015.xlsx")

#there are multiple sheets in one excel, we should read them seperately
#Every sheets has few rows of data background information,for easy computational purpose,i choose skip those rows
ims = pd.read_excel(undata, 'Table 1',skiprows=14) #ims = International migrant stock
```

```
#check basic dataset feasture
ims.head(40)
#noticed that Sort\norder and country code columns has one decimal been placed on each value in the rows
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	International migrant stock at mid-year (both sexes)	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unname 1
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	2000	2005	2010	201
1	1.0	WORLD	NaN	900.0	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	12611543
2	2.0	Developed regions	(b)	901.0	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	6761861
3	3.0	Developing regions	(c)	902.0	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	5849681
4	4.0	Least developed countries	(d)	941.0	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	646321

Principle 1: Column names need to be informative, variable names and not values

```
#Priciple 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them imformative
ims=ims.rename(columns={"International migrant stock at mid-year (both sexes)":"b1990","Unnamed: 6":"b1995","Unnamed: 7
                        "International migrant stock at mid-year (male)":"M1990","Unnamed: 12":"M1995","Unnamed: 13":"M
                        "International migrant stock at mid-year (female)":"F1990","Unnamed: 18" : "F1995","Unnamed: 19
ims
```

After the pandas read the sheets, there are unnamed columns that are not informative to tell what information is contained in each column, I will rename them first. However, after I renamed these columns, there are variables in the title rows that are not informative as well and they will be fixed after styling in principle 2.

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	...	M2000	M2005	M2010	M2015
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	2000	2005	2010	2015
1	1.0	WORLD	NaN	900.0	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	126115435
2	2.0	Developed regions	(b)	901.0	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	67618619
3	3.0	Developing regions	(c)	902.0	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	58496816
4	4.0	Least developed countries	(d)	941.0	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	6463217
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	3357	4694	5998	5746	5122	...	3101	2940	2594	2469
262	262.0	Tokelau	NaN	772.0	B	270	266	262	258	429	...	144	133	206	233
263	263.0	Tonga	NaN	776.0	B	2911	3274	3684	4301	5022	...	1981	2328	2727	3127
264	264.0	Tuvalu	NaN	798.0	C	318	263	217	183	154	...	121	101	85	78
265	265.0	Wallis and Futuna Islands	NaN	876.0	B	1402	1680	2015	2365	2776	...	1018	1194	1401	1438

Principle 2: Each column needs to consist of one and only one variable

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged
#temporary column years_named is created to help solve this issue
ims1 = ims1.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","Ty
ims1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	International migrant stock at mid-year
0	1.0	WORLD	NaN	900.0	NaN	b1990	152563212
1	2.0	Developed regions	(b)	901.0	NaN	b1990	82378628
2	3.0	Developing regions	(c)	902.0	NaN	b1990	70184584
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	11075966
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	59105261
...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	F2015	2460
4766	262.0	Tokelau	NaN	772.0	B	F2015	254
4767	263.0	Tonga	NaN	776.0	B	F2015	2604
4768	264.0	Tuvalu	NaN	798.0	C	F2015	63
4769	265.0	Wallis and Futuna Islands	NaN	876.0	B	F2015	1411

4770 rows × 7 columns

The *years_renamed* column is created to bring columns into rows first and then split into two columns *gender* and *year* to make sure each column consists of one and only one variable. Since I have brought down some columns into rows, now the total number of rows is 4770 and the total column numbers reduced to 7. After that, I dropped one temporary row created by combining columns and the total row number was reduced to 4769.

```
#drop the first non-informative rows
ims1 = ims[1:]
ims1
```

After this step is done, I renamed new columns in a more informative way, b=Both, M=Male, F=Female.

```
#drop and temporary years_renamed column
#style the name of new columns to be informative
ims2=ims1.drop(['years_renamed'], axis=1,inplace=True)
ims2 = ims1.replace(to_replace=["b","F","M"],value=["Both","Male","Female"])
ims2
#now each columns consist one and only one variable
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	International migrant stock at mid-year	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	152563212	Both	1990
1	2.0	Developed regions	(b)	901.0	NaN	82378628	Both	1990
2	3.0	Developing regions	(c)	902.0	NaN	70184584	Both	1990
3	4.0	Least developed countries	(d)	941.0	NaN	11075966	Both	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	59105261	Both	1990
...	...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	2460	Male	2015
4766	262.0	Tokelau	NaN	772.0	B	254	Male	2015
4767	263.0	Tonga	NaN	776.0	B	2604	Male	2015
4768	264.0	Tuvalu	NaN	798.0	C	63	Male	2015
4769	265.0	Wallis and Futuna Islands	NaN	876.0	B	1411	Male	2015

Principle 3: Variables need to be on cells, not rows and columns

Variables are already stored in cells following the above process. (see above explanation)

Principle 4: Each table column needs to have a singular data type

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
ims2.info()
#there are 6 columns are object and 2 columns are float, split them into two tables by data type with unique combinatic
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4770 entries, 0 to 4769
Data columns (total 8 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Sort
order                                          4770 non-null   float64
 1   Major area, region, country or area of destination  4770 non-null   object
 2   Notes                                           468 non-null    object
 3   Country code                                    4770 non-null   float64
 4   Type of data (a)                                4176 non-null   object
 5   International migrant stock at mid-year         4770 non-null   object
 6   Gender                                          4770 non-null   object
 7   Year                                            4770 non-null   object
dtypes: float64(2), object(6)
memory usage: 298.2+ KB
```

Before creating new tables based on data type, I used info() function to check the original data type of this table. This table contains two data types: float64 and object. As mentioned above, jupyter add one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#clean decimal places to integer, back to original number presentation in the excel sheet
ims2["Sort\norder"]=ims2["Sort\norder"].astype("int64")
ims2["Country code"]=ims2["Country code"].astype("int64")
ims2.head(15)
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	International migrant stock at mid-year	Gender	Year
0	1	WORLD	NaN	900	NaN	152563212	Both	1990
1	2	Developed regions	(b)	901	NaN	82378628	Both	1990
2	3	Developing regions	(c)	902	NaN	70184584	Both	1990
3	4	Least developed countries	(d)	941	NaN	11075966	Both	1990
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	Both	1990
5	6	Sub-Saharan Africa	(e)	947	NaN	14690319	Both	1990
6	7	Africa	NaN	903	NaN	15690623	Both	1990
7	8	Eastern Africa	NaN	910	NaN	5964031	Both	1990
8	9	Burundi	NaN	108	B R	333110	Both	1990
9	10	Comoros	NaN	174	B	14079	Both	1990
10	11	Djibouti	NaN	262	B R	122221	Both	1990
11	12	Eritrea	NaN	232	I	11848	Both	1990
12	13	Ethiopia	NaN	231	B R	1155390	Both	1990
13	14	Kenya	NaN	404	B R	297292	Both	1990
14	15	Madagascar	NaN	450	C	23917	Both	1990

Then I double-checked the data type and make sure they are back to integer.

```
#double check data type, ready to split into tables by data type following pricinple 4
ims2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4770 entries, 0 to 4769
Data columns (total 8 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Sort
order                                          4770 non-null   int64
 1   Major area, region, country or area of destination  4770 non-null   object
 2   Notes                                           468 non-null    object
 3   Country code                                    4770 non-null   int64
 4   Type of data (a)                                4176 non-null   object
 5   International migrant stock at mid-year         4770 non-null   object
 6   Gender                                          4770 non-null   object
 7   Year                                            4770 non-null   object
dtypes: int64(2), object(6)
memory usage: 298.2+ KB
```

After that, I find the unique combination under the same data type and set them as index.

```
ims3 = ims2.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","International m
ims3
```

Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year	Sort\norder	Country code
WORLD	NaN	NaN	152563212	Both	1990	1	900
Developed regions	(b)	NaN	82378628	Both	1990	2	901
Developing regions	(c)	NaN	70184584	Both	1990	3	902
Least developed countries	(d)	NaN	11075966	Both	1990	4	941
Less developed regions excluding least developed countries	NaN	NaN	59105261	Both	1990	5	934
...	...	...	...	...	...	...	...
Samoa	NaN	B	2460	Male	2015	261	882
Tokelau	NaN	B	254	Male	2015	262	772
Tonga	NaN	B	2604	Male	2015	263	776
Tuvalu	NaN	C	63	Male	2015	264	798
Wallis and Futuna Islands	NaN	B	1411	Male	2015	265	876

4770 rows × 2 columns

The main table of this excel sheet would contain major informative and critical parts that provide unique combinations with population-region-year-gender-notes-type of data.

```
#setting up Maintable for table 1 excel sheet
Maintable = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","International migran
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(ims3.index.unique())
    ],
)
```

```
Maintable.head(20)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year
0	1	WORLD	NaN	NaN	152563212	Both	1990
1	2	Developed regions	(b)	NaN	82378628	Both	1990
2	3	Developing regions	(c)	NaN	70184584	Both	1990
3	4	Least developed countries	(d)	NaN	11075966	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	59105261	Both	1990
5	6	Sub-Saharan Africa	(e)	NaN	14690319	Both	1990
6	7	Africa	NaN	NaN	15690623	Both	1990
7	8	Eastern Africa	NaN	NaN	5964031	Both	1990
8	9	Burundi	NaN	B R	333110	Both	1990
9	10	Comoros	NaN	B	14079	Both	1990
10	11	Djibouti	NaN	B R	122221	Both	1990
11	12	Eritrea	NaN	I	11848	Both	1990

Then I go on to create the appendix table of this excel under the same data type (int64) and set the main table as an index with ID.

```
#setting up Appendix table for table 1 excel sheet
Appendix = ims3[["Sort\norder","Country code"]].copy()
Appendix["ID"] = Maintable.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","
```

```
Appendix = Appendix.reset_index(drop=True).set_index("ID")
Appendix.head(268)
```

ID	Sort\norder	Country code
1	1	900
2	2	901
3	3	902
4	4	941
5	5	934
...	...	...
264	264	798
265	265	876
266	1	900
267	2	901
268	3	902

268 rows × 2 columns

```
Maintable.set_index(["ID"])
```

ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year
1	WORLD	NaN	NaN	152563212	Both	1990
2	Developed regions	(b)	NaN	82378628	Both	1990
3	Developing regions	(c)	NaN	70184584	Both	1990
4	Least developed countries	(d)	NaN	11075966	Both	1990
5	Less developed regions excluding least develop...	NaN	NaN	59105261	Both	1990
...	...	...	...	...	...	...
4766	Samoa	NaN	B	2460	Male	2015
4767	Tokelau	NaN	B	254	Male	2015
4768	Tonga	NaN	B	2604	Male	2015
4769	Tuvalu	NaN	C	63	Male	2015
4770	Wallis and Futuna Islands	NaN	B	1411	Male	2015

4770 rows × 6 columns

There is a lot of missing values identified within the main table and for easy computational purpose, I replaced these values with NA.

```
#replace missing data with NA for future computational use
Maintable.replace(to_replace="..",value=pd.NA,inplace=True)
```

```
Maintable.head(45)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year
0	1	WORLD	NaN	NaN	152563212	Both	1990
1	2	Developed regions	(b)	NaN	82378628	Both	1990
2	3	Developing regions	(c)	NaN	70184584	Both	1990
3	4	Least developed countries	(d)	NaN	11075966	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	59105261	Both	1990
5	6	Sub-Saharan Africa	(e)	NaN	14690319	Both	1990
6	7	Africa	NaN	NaN	15690623	Both	1990
7	8	Eastern Africa	NaN	NaN	5964031	Both	1990
8	9	Burundi	NaN	B R	333110	Both	1990
9	10	Comoros	NaN	B	14079	Both	1990
10	11	Djibouti	NaN	B R	122221	Both	1990
11	12	Eritrea	NaN	I	11848	Both	1990
12	13	Ethiopia	NaN	B R	1155390	Both	1990
13	14	Kenya	NaN	B R	297292	Both	1990
14	15	Madagascar	NaN	C	23917	Both	1990
15	16	Malawi	NaN	B R	1127724	Both	1990
16	17	Mauritius	(1)	C	3613	Both	1990

Principle 5: A single observational unit must be in 1 table

Both the main and appendix table has already been set to a single observational based on the previous cleaning process.

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

This table contains 22 columns and 266 rows (skip the first 14 context rows) and this table has the same issue about decimal places of sort order and country code columns. Moreover, compared to table 1, the population column is presented in thousand units and every value is saved with three decimal places. For easy computational purposes, these values will be multiplied by 1000 to become integers.

```
#INF1340 DATA CELANING
import pandas as pd

undata = pd.ExcelFile("UN_MigrantStockTotal_2015.xlsx")

#there are multiple sheets in one excel, we should read them seperately
#Every sheets has few rows of data background information,for easy computational purpose,i choose skip those rows
TP = pd.read_excel(undata, 'Table 2',skiprows=14) #TP=Total population
```

```
#check basic dataset feasture
TP.head()
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Total population of both sexes at mid-year (thousands)	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unnamed: 13	Unr
0	NaN	NaN	NaN	NaN	1990.000	1995.000	2000.000	2005.000	2010.000	2015.000	...	2000	2005	
1	1.0	WORLD	NaN	900.0	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	3493
2	2.0	Developed regions	(b)	901.0	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	599
3	3.0	Developing regions	(c)	902.0	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	2894
4	4.0	Least developed countries	(d)	941.0	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	...	331482.475	375757.715	422

Principle 1: Column names need to be informative, variable names and not values

There are many merged columns in table 2, after panadas read the table, lots of unnamed columns are created. I will first rename these columns after their original presentation, gender + year.
E.g. Both Sex 1990 = b1990

```
#Priciple 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
TP=TP.rename(columns={"Total population of both sexes at mid-year (thousands)":"b1990","Unnamed: 5":"b1995","Unnamed: 6
                      "Total male population at mid-year (thousands)":"M1990","Unnamed: 11":"M1995","Unnamed: 12":"M2
                      "Total female population at mid-year (thousands)":"F1990","Unnamed: 17" : "F1995","Unnamed: 18"
TP
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	b1990	b1995	b2000	b2005	b2010	b2015	...	M2000	M2005	
0	NaN	NaN	NaN	NaN	1990.000	1995.000	2000.000	2005.000	2010.000	2015.000	...	2000	2005	
1	1.0	WORLD	NaN	900.0	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	3
2	2.0	Developed regions	(b)	901.0	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	5
3	3.0	Developing regions	(c)	902.0	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	2
4	4.0	Least developed countries	(d)	941.0	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	...	331482.475	375757.715	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	

```
#drop the first non-informative rows
TP1 = TP[1:]
TP1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	b1990	b1995	b2000	b2005	b2010	b2015	...	M2000	M2005	
1	1.0	WORLD	NaN	900.0	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	349
2	2.0	Developed regions	(b)	901.0	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	59
3	3.0	Developing regions	(c)	902.0	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	289
4	4.0	Least developed countries	(d)	941.0	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	...	331482.475	375757.715	42
5	5.0	Less developed regions excluding least develop...	NaN	934.0	3655147.008	3980172.519	4273424.303	4557911.390	4849094.485	5143963.209	...	2175044.969	2321362.321	247

After renaming columns into an informative format, I dropped the extra rows created.

Principle 2: Each column needs to consist of one and only one variable

As shown in the table above, starting from column 6, each column has 2 variables and this needs to be split into gender and year columns. First, I created the *years_renamed* and *Total population at mid-year* column to melt those columns first, and then split *years_renamed* column into gender and year. After that, I dropped the temporary *years_renamed* column.

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
TP1 = TP1.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code"],var_
TP1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	years_renamed	Total population at mid-year
0	1.0	WORLD	NaN	900.0	b1990	5309667.699
1	2.0	Developed regions	(b)	901.0	b1990	1144463.062
2	3.0	Developing regions	(c)	902.0	b1990	4165204.637
3	4.0	Least developed countries	(d)	941.0	b1990	510057.629
4	5.0	Less developed regions excluding least develop...	NaN	934.0	b1990	3655147.008
...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	F2015	93.584
4766	262.0	Tokelau	NaN	772.0	F2015	..
4767	263.0	Tonga	NaN	776.0	F2015	52.931
4768	264.0	Tuvalu	NaN	798.0	F2015	..
4769	265.0	Wallis and Futuna Islands	NaN	876.0	F2015	..

4770 rows × 6 columns

```
#split temporary column to two different columns with one and only variable
TP1 = TP1.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
TP1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	years_renamed	Total population at mid-year	Gender	Year
0	1.0	WORLD	NaN	900.0	b1990	5309667.699	b	1990
1	2.0	Developed regions	(b)	901.0	b1990	1144463.062	b	1990
2	3.0	Developing regions	(c)	902.0	b1990	4165204.637	b	1990
3	4.0	Least developed countries	(d)	941.0	b1990	510057.629	b	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	b1990	3655147.008	b	1990
...	...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	F2015	93.584	F	2015
4766	262.0	Tokelau	NaN	772.0	F2015	..	F	2015
4767	263.0	Tonga	NaN	776.0	F2015	52.931	F	2015
4768	264.0	Tuvalu	NaN	798.0	F2015	..	F	2015
4769	265.0	Wallis and Futuna Islands	NaN	876.0	F2015	..	F	2015

4770 rows × 8 columns

```
#drop the temporary column years_renamed, rename other columns to be more informative
TP2=TP1.drop(['years_renamed'], axis=1,inplace=True)
TP2=TP1.replace(to_replace=["b","F","M"],value=["Both","Male","Female"])
TP2
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Total population at mid-year	Gender	Year
0	1.0	WORLD	NaN	900.0	5309667.699	Both	1990
1	2.0	Developed regions	(b)	901.0	1144463.062	Both	1990
2	3.0	Developing regions	(c)	902.0	4165204.637	Both	1990
3	4.0	Least developed countries	(d)	941.0	510057.629	Both	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	3655147.008	Both	1990
...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	93.584	Male	2015
4766	262.0	Tokelau	NaN	772.0	..	Male	2015
4767	263.0	Tonga	NaN	776.0	52.931	Male	2015
4768	264.0	Tuvalu	NaN	798.0	..	Male	2015
4769	265.0	Wallis and Futuna Islands	NaN	876.0	..	Male	2015

4770 rows × 7 columns

Then styled the columns to the right name given the current information.
E.g. F= Female

Principle 3: Variables need to be in cells, not rows and columns

Variables are already stored in cells following the above process.

Principle 4: Each table column needs to have a singular data type

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
TP2.info()
#there are 5 columns are object and 2 columns are float, split them into two tables by data type+unique combination
```

```
<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   float64
 1   Major area, region, country or area of destination  4770 non-null   object
 2   Notes                                           468 non-null    object
 3   Country code                                    4770 non-null   float64
 4   Total population at mid-year                    4770 non-null   object
 5   Gender                                          4770 non-null   object
 6   Year                                            4770 non-null   object
dtypes: float64(2), object(5)
memory usage: 261.0+ KB
```

I first check the data type of this table using info() function. This table contains two data types: float64 and object. As mentioned above, jupyter add one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

Having a singular data type does not necessarily mean the datatype of the file when it is read as a dataframe. Oftentimes when you read a file using pandas the file will record the feature as the wrong datatype and you need to fix it manually.

```
#clean decimal places to integer, back to original number presentation in the excel sheet
TP2["Sort\norder"]=TP2["Sort\norder"].astype("int64")
TP2["Country code"]=TP2["Country code"].astype("int64")
TP2.head(10)
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Total population at mid-year	Gender	Year
0	1	WORLD	NaN	900	5309667.699	Both	1990
1	2	Developed regions	(b)	901	1144463.062	Both	1990
2	3	Developing regions	(c)	902	4165204.637	Both	1990
3	4	Least developed countries	(d)	941	510057.629	Both	1990

```
#double check data type, ready to split into tables by data type following pricinple 4
TP2.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   int64
 1   Major area, region, country or area of destination  4770 non-null   object
 2   Notes                                           468 non-null    object
 3   Country code                                    4770 non-null   int64
 4   Total population at mid-year                    4770 non-null   object
 5   Gender                                          4770 non-null   object
 6   Year                                            4770 non-null   object
dtypes: int64(2), object(5)
memory usage: 261.0+ KB
```

After that, I find the unique combination within objective data type, set them as index and then create a new table called Main Table which contains major critical information on the total world population of the world by sex and by major area in years between 1990 to 2010.

```
TP3 = TP2.set_index(["Major area, region, country or area of destination","Notes","Total population at mid-year","Gende
TP3
```

Major area, region, country or area of destination	Notes	Total population at mid-year	Gender	Year	Sort\norder	Country code
WORLD	NaN	5309667.699	Both	1990	1	900
Developed regions	(b)	1144463.062	Both	1990	2	901
Developing regions	(c)	4165204.637	Both	1990	3	902
Least developed countries	(d)	510057.629	Both	1990	4	941
Less developed regions excluding least developed countries	NaN	3655147.008	Both	1990	5	934
...	...	...	...	...	...	...
Samoa	NaN	93.584	Male	2015	261	882
Tokelau	NaN	..	Male	2015	262	772
Tonga	NaN	52.931	Male	2015	263	776
Tuvalu	NaN	..	Male	2015	264	798
Wallis and Futuna Islands	NaN	..	Male	2015	265	876

4770 rows × 2 columns

```
#setting up Maintable for table 2 excel sheet
Maintable = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Total population at mid-year","Gender",
    data =[
        (a + 1,b,c,d,e,f,)
        for (a, (b,c,d,e,f,)) in enumerate(TP3.index.unique())
    ],
)
```

```
Maintable.head(20)
```

	ID	Major area, region, country or area of destination	Notes	Total population at mid-year	Gender	Year
0	1	WORLD	NaN	5309667.699	Both	1990
1	2	Developed regions	(b)	1144463.062	Both	1990
2	3	Developing regions	(c)	4165204.637	Both	1990
3	4	Least developed countries	(d)	510057.629	Both	1990
4	5	Less developed regions excluding least develop...	NaN	3655147.008	Both	1990
5	6	Sub-Saharan Africa	(e)	491497.691	Both	1990
6	7	Africa	NaN	631614.304	Both	1990
7	8	Eastern Africa	NaN	198231.687	Both	1990
8	9	Burundi	NaN	5613.141	Both	1990
9	10	Comoros	NaN	415.144	Both	1990

Following the same step, I create a new table with data type int64 called appendix table containing information on sort, order and country code. Then, set the new Main Table as the index.

```
#setting up Appendix table for table 2 excel sheet
Appendix = TP3[["Sort\norder","Country code"]].copy()
Appendix["ID"] = Maintable.set_index(["Major area, region, country or area of destination","Notes","Total population at
```

```
Appendix = Appendix.reset_index(drop=True).set_index("ID")
Appendix.head(100)
```

ID	Sort\norder	Country code
1	1	900
2	2	901
3	3	902
4	4	941
5	5	934

```
Maintable.set_index(["ID"])
```

ID	Major area, region, country or area of destination	Notes	Total population at mid-year	Gender	Year
1	WORLD	NaN	5309667.699	Both	1990
2	Developed regions	(b)	1144463.062	Both	1990
3	Developing regions	(c)	4165204.637	Both	1990
4	Least developed countries	(d)	510057.629	Both	1990
5	Less developed regions excluding least develop...	NaN	3655147.008	Both	1990
...	...	...	...	...	...
4766	Samoa	NaN	93.584	Male	2015

Same as table 1, there are many missing values in table 2 as well. I will fist replace these missing values with NA in pandas. Then, as mentioned in the previous section, for easy computational purposes, I will set the values in the population to integers by times 1000t, not in thousand units with decimals.

```
#replace missing data with NA for future computational use
Maintable.replace(to_replace="..",value=pd.NA,inplace=True)
```

```
Maintable.head(214)
```

	ID	Major area, region, country or area of destination	Notes	Total population at mid-year	Gender	Year
0	1	WORLD	NaN	5309667.699	Both	1990
1	2	Developed regions	(b)	1144463.062	Both	1990
2	3	Developing regions	(c)	4165204.637	Both	1990
3	4	Least developed countries	(d)	510057.629	Both	1990
4	5	Less developed regions excluding least develop...	NaN	3655147.008	Both	1990
...	...	...	...	...	...	...
209	210	Costa Rica	NaN	3095.994	Both	1990
210	211	El Salvador	NaN	5252.082	Both	1990
211	212	Guatemala	NaN	9158.547	Both	1990
212	213	Honduras	NaN	4903.363	Both	1990
213	214	Mexico	NaN	85609.404	Both	1990

```
#Noticed that in the original table the population numer are in thousands numbers
#for easy computaitonal purpose, will not times all of the numers by 1000
Maintable["Total population at mid-year"]=Maintable["Total population at mid-year"]*1000
Maintable.head(20)
```

	ID	Major area, region, country or area of destination	Notes	Total population at mid-year	Gender	Year
0	1	WORLD	NaN	5309667699.0	Both	1990
1	2	Developed regions	(b)	1144463062.0	Both	1990
2	3	Developing regions	(c)	4165204637.0	Both	1990
3	4	Least developed countries	(d)	510057629.0	Both	1990
4	5	Less developed regions excluding least develop...	NaN	3655147008.0	Both	1990
5	6	Sub-Saharan Africa	(e)	491497691.0	Both	1990
6	7	Africa	NaN	631614304.0	Both	1990
7	8	Eastern Africa	NaN	198231687.0	Both	1990
8	9	Burundi	NaN	5613141.0	Both	1990
9	10	Comoros	NaN	415144.0	Both	1990
10	11	Djibouti	NaN	588356.0	Both	1990
11	12	Eritrea	NaN	3139083.0	Both	1990

Principle 5: A single observational unit must be in 1 table

Both the main and appendix table has already been set to a single observational based on the previous cleaning process.

Table 3 - International migrant stock as a percentage of the total population by sex and by major area, region, country or area, 1990-2015

This table contains 23 columns and 266 rows (skip the first 14 context rows). S*ort order* and *country code* columns have the same decimal places issue with the previous two tables and this problem will be addressed and fixed in the following process. This table contains information on the International migrant stock as a percentage and values in the cells contain many decimal places. To keep the accuracy of this dataset for future use, all values will keep the decimal places except values in S*ort order* and *country code.*

```
#INF1340 DATA CELANING
import pandas as pd

undata = pd.ExcelFile("UN_MigrantStockTotal_2015.xlsx")

#there are multiple sheets in one excel, we should read them seperately
#Every sheets has few rows of data background information,for easy computational purpose,i choose skip those rows
imsp = pd.read_excel(undata, 'Table 3',skiprows=14) #imsp=International migrant stock as a percentage of the total popu
```

```
#check basic dataset feasture
imsp.head()
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	International migrant stock as a percentage of the total population (both sexes)	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	...	2000	2005	2010	2015
1	1.0	WORLD	NaN	900.0	NaN	2.87331	2.803806	2.818899	2.933739	3.199467	...	2.849206	2.979124	3.280341	3.4019
2	2.0	Developed regions	(b)	901.0	NaN	7.198015	7.891085	8.695688	9.693045	10.747765	...	8.743236	9.73154	10.680972	11.097807
3	3.0	Developing regions	(c)	902.0	NaN	1.685021	1.500317	1.404022	1.395066	1.565106	...	1.490031	1.507122	1.746117	1.888268
4	4.0	Least developed countries	(d)	941.0	NaN	2.171513	2.001353	1.516863	1.303078	1.182422	...	1.617552	1.432574	1.293264	1.35773

Principle 1: Column names need to be informative, variable names and not values

```
#Principle 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
imsp=imsp.rename(columns={"International migrant stock as a percentage of the total population (both sexes)":"b1990","U
                          "International migrant stock as a percentage of the total population (male)":"M1990","Unnamed:
                          "International migrant stock as a percentage of the total population (female)":"F1990","Unnamed
imsp
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	...	M2000	M2005	M2010	M2015	
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	...	2000	2005	2010	2015	
1	1.0	WORLD	NaN	900.0	NaN	2.87331	2.803806	2.818899	2.933739	3.199467	...	2.849206	2.979124	3.280341	3.4019	2.
2	2.0	Developed regions	(b)	901.0	NaN	7.198015	7.891085	8.695688	9.693045	10.747765	...	8.743236	9.73154	10.680972	11.097807	7.1
3	3.0	Developing regions	(c)	902.0	NaN	1.685021	1.500317	1.404022	1.395066	1.565106	...	1.490031	1.507122	1.746117	1.888268	1.5
4	4.0	Least developed countries	(d)	941.0	NaN	2.171513	2.001353	1.516863	1.303078	1.182422	...	1.617552	1.432574	1.293264	1.35773	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
261	261.0	Samoa	NaN	882.0	B	2.061216	2.758613	3.435005	3.1935	2.753334	...	3.410241	3.155014	2.703491	2.477821	2.0
262	262.0	Tokelau	NaN	772.0	B	16.780609	17.5	16.881443	21.322314	37.797357	...	..	..	..	..	

There are merged columns as the other tables and after panadas read the table, lots of unnamed columns are created. I will first rename these columns after their original presentation, gender + year.
E.g. Male 1990 = M1990

```
#drop the first non-informative rows
imsp= imsp[1:]
imsp
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	...	M2000	M2005	M2010	M2015	F1
1	1.0	WORLD	NaN	900.0	NaN	2.87331	2.803806	2.818899	2.933739	3.199467	...	2.849206	2.979124	3.280341	3.4019	2.83
2	2.0	Developed regions	(b)	901.0	NaN	7.198015	7.891085	8.695688	9.693045	10.747765	...	8.743236	9.73154	10.680972	11.097807	7.147
3	3.0	Developing regions	(c)	902.0	NaN	1.685021	1.500317	1.404022	1.395066	1.565106	...	1.490031	1.507122	1.746117	1.888268	1.595
4	4.0	Least developed countries	(d)	941.0	NaN	2.171513	2.001353	1.516863	1.303078	1.182422	...	1.617552	1.432574	1.293264	1.35773	2.045

Drop the temporary non-informative rows after splitting the merged columns.

Principle 2: Each column needs to consist of one and only one variable

As shown in the table above, starting from column 6, each column has 2 variables and this needs to be split. First, I created the *years_renamed* which contains both sex and year information. International migrant stock as a percentage of the total population(%) is also created to contains the percentage information that been combined. Afterwards, I split years_renamed column into gender and year and drop the temporary years_renamed column.

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
imsp1 = imsp.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","T
imsp1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	International migrant stock as a percentage of the total population(%)
0	1.0	WORLD	NaN	900.0	NaN	b1990	2.87331
1	2.0	Developed regions	(b)	901.0	NaN	b1990	7.198015
2	3.0	Developing regions	(c)	902.0	NaN	b1990	1.685021
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	2.171513
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	1.617042
...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	F2015	2.628654

```
#split temporary column to two different columns with one and only variable
imsp1 = imsp1.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
imsp1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	International migrant stock as a percentage of the total population(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	b1990	2.87331	b	1990
1	2.0	Developed regions	(b)	901.0	NaN	b1990	7.198015	b	1990
2	3.0	Developing regions	(c)	902.0	NaN	b1990	1.685021	b	1990
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	2.171513	b	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	1.617042	b	1990
...	...	...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	F2015	2.628654	F	2015

Then styled the new columns with the right names to provide the current information.
E.g. F= Female

Principle 3: Variables need to be on cells, not rows and columns

Variables are already stored in cells while doing principle 2 following the above process.

Principle 4: Each table column needs to have a singular data type

I first check the data type of this table using info() function. This table contains two data types: float64 and object. As mentioned above, jupyter add one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
imsp2.info()
#there are 6 columns are object and 2 columns are float,split them into two tables by data type+unique combination
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4770 entries, 0 to 4769
Data columns (total 8 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   Sort
order                                                                 4770 non-null   float64
 1   Major area, region, country or area of destination                     4770 non-null   object
 2   Notes                                                                   468 non-null    object
 3   Country code                                                            4770 non-null   float64
 4   Type of data (a)                                                        4176 non-null   object
 5   International migrant stock as a percentage of the total population(%)  4770 non-null   object
 6   Gender                                                                  4770 non-null   object
 7   Year                                                                    4770 non-null   object
dtypes: float64(2), object(6)
memory usage: 298.2+ KB
```

```
#clean decimal places to integer, back to original number presentation in the excel sheet
imsp2["Sort\norder"]=imsp2["Sort\norder"].astype("int64")
imsp2["Country code"]=imsp2["Country code"].astype("int64")
imsp2.head(10)
```

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

```
#double check data type, ready to split into tables by data type following pricinple 4
imsp2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4770 entries, 0 to 4769
Data columns (total 8 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   Sort
order                                                                 4770 non-null   int64
 1   Major area, region, country or area of destination                     4770 non-null   object
 2   Notes                                                                   468 non-null    object
 3   Country code                                                            4770 non-null   int64
 4   Type of data (a)                                                        4176 non-null   object
 5   International migrant stock as a percentage of the total population(%)  4770 non-null   object
 6   Gender                                                                  4770 non-null   object
 7   Year                                                                    4770 non-null   object
dtypes: int64(2), object(6)
memory usage: 298.2+ KB
```

After that, I find the unique combination within objective data type, set them as index and then create a new table called Main Table which contains major critical information on the International migrant stock as a percentage of the total population(%)

```
imsp3 = imsp2.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","International
imsp3
```

Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock as a percentage of the total population(%)	Gender	Year	Sort\norder	Country code
WORLD	NaN	NaN	2.873309981879527	Both	1990	1	900
Developed regions	(b)	NaN	7.19801544805122	Both	1990	2	901
Developing regions	(c)	NaN	1.6850212682599583	Both	1990	3	902
Least developed countries	(d)	NaN	2.171512662542687	Both	1990	4	941
Less developed regions excluding least developed countries	NaN	NaN	1.6170419649507022	Both	1990	5	934

```
#setting up Maintable for table 3 excel sheet
Maintable = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","International migrar
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(imsp3.index.unique())
    ],
)
```

```
Maintable.head(30)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock as a percentage of the total population(%)	Gender	Year
0	1	WORLD	NaN	NaN	2.87331	Both	1990
1	2	Developed regions	(b)	NaN	7.198015	Both	1990
2	3	Developing regions	(c)	NaN	1.685021	Both	1990
3	4	Least developed countries	(d)	NaN	2.171513	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	1.617042	Both	1990
5	6	Sub-Saharan Africa	(e)	NaN	2.988889	Both	1990

Following the same step, I create a new table with data type int64 called appendix table containing information on sort, order and country code. Then, set the new Main Table as the index.

```
#setting up Appendix table for table 3 excel sheet
Appendix = imsp3[["Sort\norder","Country code"]].copy()
Appendix["ID"] = Maintable.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","
```

```
Appendix = Appendix.reset_index(drop=True).set_index("ID")
Appendix.head(50)
```

ID	Sort\norder	Country code
1	1	900
2	2	901
3	3	902
4	4	941
5	5	934

```
Maintable.set_index(["ID"])
```

ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock as a percentage of the total population(%)	Gender	Year
1	WORLD	NaN	NaN	2.87331	Both	1990
2	Developed regions	(b)	NaN	7.198015	Both	1990
3	Developing regions	(c)	NaN	1.685021	Both	1990
4	Least developed countries	(d)	NaN	2.171513	Both	1990
5	Less developed regions excluding least develop...	NaN	NaN	1.617042	Both	1990
...	...	...	...	...	...	...
4766	Samoa	NaN	B	2.628654	Male	2015
4767	Tokelau	NaN	B	..	Male	2015

Same as the previous table, there are many missing values in table 3 as well. I will replace these missing values with NA in pandas.

```
#replace missing data with NA for future computational use
Maintable.replace(to_replace="..",value=pd.NA,inplace=True)
```

```
Maintable.head(25)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock as a percentage of the total population(%)	Gender	Year
0	1	WORLD	NaN	NaN	2.87331	Both	1990
1	2	Developed regions	(b)	NaN	7.198015	Both	1990
2	3	Developing regions	(c)	NaN	1.685021	Both	1990
3	4	Least developed countries	(d)	NaN	2.171513	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	1.617042	Both	1990
5	6	Sub-Saharan Africa	(e)	NaN	2.988889	Both	1990
6	7	Africa	NaN	NaN	2.48421	Both	1990
7	8	Eastern Africa	NaN	NaN	3.008616	Both	1990
8	9	Burundi	NaN	B R	5.934467	Both	1990

Principle 5: A single observational unit must be in 1 table

Both the main and appendix table has already been set to a single observational based on the previous cleaning process.

Table 4 - Female migrants as a percentage of the international migrant stock by major area, region, country or area, 1990-2015

This table is also very similar to the others but this time it is just about female migrants and for accuracy purpose, all values in the cells will keep as what it is, with no rounded numbers. The first 14 context rows will skipped as usual. S*ort order* and *country code* columns still have the same decimal places issue with the previous tables and this problem will be addressed and fixed in the following process.

```
#INF1340 DATA CELANING
import pandas as pd

undata = pd.ExcelFile("UN_MigrantStockTotal_2015.xlsx")

#there are multiple sheets in one excel, we should read them seperately
#Every sheets has few rows of data background information,for easy computational purpose,i choose skip those rows
fmsp = pd.read_excel(undata, 'Table 4',skiprows=14) #Female migrants as a percentage
```

```
#check basic dataset feasture
fmsp.head()
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Female migrants as a percentage of the international migrant stock	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	2015.000000
1	1.0	WORLD	NaN	900.0	NaN	49.03915	49.16879	49.112244	48.832993	48.305660	48.249769
2	2.0	Developed regions	(b)	901.0	NaN	51.123977	51.149024	51.113307	51.171501	51.658932	51.866687
3	3.0	Developing regions	(c)	902.0	NaN	46.592099	46.500135	46.128444	45.134297	43.319780	43.327078
4	4.0	Least developed countries	(d)	941.0	NaN	47.261155	47.571664	46.826689	45.157406	45.499573	45.942752

Principle 1: Column names need to be informative, variable names and not values

```
#Principle 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
fmsp=fmsp.rename(columns={"Female migrants as a percentage of the international migrant stock":"F1990","Unnamed: 6":"F1
fmsp
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	F1990	F1995	F2000	F2005	F2010	F2015
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	2015.000000
1	1.0	WORLD	NaN	900.0	NaN	49.03915	49.16879	49.112244	48.832993	48.305660	48.249769
2	2.0	Developed regions	(b)	901.0	NaN	51.123977	51.149024	51.113307	51.171501	51.658932	51.866687
3	3.0	Developing regions	(c)	902.0	NaN	46.592099	46.500135	46.128444	45.134297	43.319780	43.327078
4	4.0	Least developed countries	(d)	941.0	NaN	47.261155	47.571664	46.826689	45.157406	45.499573	45.942752
...	...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	47.244564	47.784406	48.299433	48.833971	49.355720	49.908704
262	262.0	Tokelau	NaN	772.0	B	44.444444	44.736842	45.038168	48.449612	51.981352	52.156057
263	263.0	Tonga	NaN	776.0	B	48.883545	47.525962	46.226927	45.873053	45.698925	45.437096
264	264.0	Tuvalu	NaN	798.0	C	43.396226	43.726236	44.239631	44.808743	44.805195	44.680851
265	265.0	Wallis and Futuna Islands	NaN	876.0	B	48.216833	48.869048	49.478908	49.513742	49.531700	49.526150

There are merged columns same as the other tables and after panadas read the table, lots of unnamed columns are created. I will first rename these columns after their original presentation, gender + year, for this table, it would be only female in the gender column.

```
#drop the first non-informative rows
fmsp= fmsp[1:]
fmsp
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	F1990	F1995	F2000	F2005	F2010	F2015
1	1.0	WORLD	NaN	900.0	NaN	49.03915	49.16879	49.112244	48.832993	48.305660	48.249769
2	2.0	Developed regions	(b)	901.0	NaN	51.123977	51.149024	51.113307	51.171501	51.658932	51.866687
3	3.0	Developing regions	(c)	902.0	NaN	46.592099	46.500135	46.128444	45.134297	43.319780	43.327078
4	4.0	Least developed countries	(d)	941.0	NaN	47.261155	47.571664	46.826689	45.157406	45.499573	45.942752
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	46.466684	46.279022	46.009598	45.130768	43.043672	42.984398

Principle 2: Each column needs to consist of one and only one variable

This table has the same issue as previous tables are that some columns have 2 variables and this need to be split. I will use the same way to clean this table as before, see the below pictures for a detailed process.

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
fmsp1 = fmsp.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","I
fmsp1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Female migrants as a percentage of the international migrant stock(%)
0	1.0	WORLD	NaN	900.0	NaN	F1990	49.03915
1	2.0	Developed regions	(b)	901.0	NaN	F1990	51.123977
2	3.0	Developing regions	(c)	902.0	NaN	F1990	46.592099
3	4.0	Least developed countries	(d)	941.0	NaN	F1990	47.261155
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	F1990	46.466684

```
#split temporary column to two different columns with one and only variable
fmsp1 = fmsp1.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
fmsp1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Female migrants as a percentage of the international migrant stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	F1990	49.03915	F	1990
1	2.0	Developed regions	(b)	901.0	NaN	F1990	51.123977	F	1990
2	3.0	Developing regions	(c)	902.0	NaN	F1990	46.592099	F	1990
3	4.0	Least developed countries	(d)	941.0	NaN	F1990	47.261155	F	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	F1990	46.466684	F	1990

```
#drop the temporary column years_renamed, rename other columns to be more informative
fmsp2=fmsp1.drop(['years_renamed'], axis=1,inplace=True)
fmsp2=fmsp1.replace(to_replace=["F"],value=["Female"])
fmsp2
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Female migrants as a percentage of the international migrant stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	49.03915	Female	1990
1	2.0	Developed regions	(b)	901.0	NaN	51.123977	Female	1990
2	3.0	Developing regions	(c)	902.0	NaN	46.592099	Female	1990
3	4.0	Least developed countries	(d)	941.0	NaN	47.261155	Female	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	46.466684	Female	1990

Styled the new columns with right names to provide the current information. (F= Female)

Principle 3: Variables need to be on cells, not rows and columns

Variables in this table are already in cells as went through the previous cleaning process.

Principle 4: Each table column needs to have a singular data type

I first check the data type of this table using info() function as usual. This table has the same data type as table 3. As mentioned above, jupyter add one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
fmsp2.info()
#there are 6 columns are object and 2 columns are float,split them into two tables by data type+unique combination
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1590 entries, 0 to 1589
Data columns (total 8 columns):
 #   Column                                                                Non-Null Count  Dtype
---  ------                                                                --------------  -----
 0   Sort
order                                                                 1590 non-null   float64
 1   Major area, region, country or area of destination                   1590 non-null   object
 2   Notes                                                                 156 non-null    object
 3   Country code                                                          1590 non-null   float64
 4   Type of data (a)                                                      1392 non-null   object
 5   Female migrants as a percentage of the international migrant stock(%)  1590 non-null   object
 6   Gender                                                                1590 non-null   object
 7   Year                                                                  1590 non-null   object
dtypes: float64(2), object(6)
memory usage: 99.5+ KB
```

```
#clean decimal places to integer, back to original number presentation in the excel sheet
fmsp2["Sort\norder"]=fmsp2["Sort\norder"].astype("int64")
fmsp2["Country code"]=fmsp2["Country code"].astype("int64")
fmsp2.head(10)
```

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

```
fmsp3 = fmsp2.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","Female migran
fmsp3
```

						Sort\norder	Country code
Major area, region, country or area of destination	Notes	Type of data (a)	Female migrants as a percentage of the international migrant stock(%)	Gender	Year		
WORLD	NaN	NaN	49.039150	Female	1990	1	900
Developed regions	(b)	NaN	51.123977	Female	1990	2	901
Developing regions	(c)	NaN	46.592099	Female	1990	3	902
Least developed countries	(d)	NaN	47.261155	Female	1990	4	941
Less developed regions excluding least developed countries	NaN	NaN	46.466684	Female	1990	5	934

```
#setting up Maintable for table 4 excel sheet
Maintable = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","Female migrants as a
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(fmsp3.index.unique())
    ],
)
```

```
Maintable.head(30)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Female migrants as a percentage of the international migrant stock(%)	Gender	Year
0	1	WORLD	NaN	NaN	49.03915	Female	1990
1	2	Developed regions	(b)	NaN	51.123977	Female	1990
2	3	Developing regions	(c)	NaN	46.592099	Female	1990
3	4	Least developed countries	(d)	NaN	47.261155	Female	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	46.466684	Female	1990
5	6	Sub-Saharan Africa	(e)	NaN	47.276121	Female	1990
6	7	Africa	NaN	NaN	47.232408	Female	1990

Use the same way to set appendix table.

```
#setting up Appendix table for table 3 excel sheet
Appendix = fmsp3[["Sort\norder","Country code"]].copy()
Appendix["ID"] = Maintable.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","
```

```
Appendix = Appendix.reset_index(drop=True).set_index("ID")
Appendix.head(20)
```

ID	Sort\norder	Country code
1	1	900
2	2	901
3	3	902
4	4	941

Same as in the previous table, there are many missing values in table 4 as well. I will replace these missing values with NA in pandas.

```
Maintable.set_index(["ID"])
```

ID	Major area, region, country or area of destination	Notes	Type of data (a)	Female migrants as a percentage of the international migrant stock(%)	Gender	Year
1	WORLD	NaN	NaN	49.03915	Female	1990
2	Developed regions	(b)	NaN	51.123977	Female	1990
3	Developing regions	(c)	NaN	46.592099	Female	1990
4	Least developed countries	(d)	NaN	47.261155	Female	1990
5	Less developed regions excluding least develop...	NaN	NaN	46.466684	Female	1990
...	...	...	...	...	...	...
1586	Samoa	NaN	B	49.908704	Female	2015
1587	Tokelau	NaN	B	52.156057	Female	2015
1588	Tonga	NaN	B	45.437096	Female	2015
1589	Tuvalu	NaN	C	44.680851	Female	2015
1590	Wallis and Futuna Islands	NaN	B	49.52615	Female	2015

1590 rows × 6 columns

```
#replace missing data with NA for future computational use
Maintable.replace(to_replace="..",value=pd.NA,inplace=True)
Maintable.head(25)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Female migrants as a percentage of the international migrant stock(%)	Gender	Year
0	1	WORLD	NaN	NaN	49.03915	Female	1990
1	2	Developed regions	(b)	NaN	51.123977	Female	1990
2	3	Developing regions	(c)	NaN	46.592099	Female	1990
3	4	Least developed countries	(d)	NaN	47.261155	Female	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	46.466684	Female	1990

Principle 5: A single observational unit must be in 1 table

Both the main and appendix table has already been set to a single observational based on the previous cleaning process.

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

This table is also very similar to the other tables. However, this is an annual rate of change table, years are counted differently than the other table since each column is represented a period of time. The other table have one unit for each five-year period from 1995 to 2015 and six counts in total. This table will have only 5 counts within 1990-2015 because the year 1990 does not have a base year to compare to create an annual rate of change. Thus, I will create dummy columns, and values in cells will be NA in order to easily merge tables together later on for putting the same type of observation in one table (principle 5). For accuracy purpose, all values presented in percentage in the cells will keep as what it is, with no rounded numbers. S*ort order* and *country code* columns still have the same decimal places issue with the previous tables and this problem will be addressed and fixed in the following process.

```
#INF1340 DATA CELANING
import pandas as pd

undata = pd.ExcelFile("UN_MigrantStockTotal_2015.xlsx")

#there are multiple sheets in one excel, we should read them seperately
#Every sheets has few rows of data background information,for easy computational purpose,i choose skip those rows
aroc = pd.read_excel(undata, 'Table 5',skiprows=14) #aroc=Annual rate of change of the migrant stock
```

```
#check basic dataset feasture
aroc.head()
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Annual rate of change of the migrant stock (both sexes)	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Annual rate of change of the migrant stock (male)	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unname
0	NaN	NaN	NaN	NaN	NaN	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015	1990-1995	1995-2000	2000-2005	2005-2010	2010-201
1	1.0	WORLD	NaN	900.0	NaN	1.051865	1.428058	2.042124	2.95416	1.890991	1.000922	1.450294	2.151575	3.159228	1.91260
2	2.0	Developed regions	(b)	901.0	NaN	2.275847	2.264965	2.50708	2.466343	1.160824	2.265595	2.279583	2.483259	2.265689	1.07468
3	3.0	Developing regions	(c)	902.0	NaN	-0.487389	0.241777	1.328107	3.702217	2.929634	-0.45298	0.380246	1.693824	4.352954	2.92705
4	4.0	Least developed countries	(d)	941.0	NaN	1.118175	-3.001139	-0.539636	0.419137	3.526927	1.000073	-2.718952	0.078575	0.293964	3.36362

Principle 1: Column names need to be informative, variable names and not values

The annual rate of change represents a period of time and for this table, I only save the last year in the name for easy merging table purposes later on. E.g. 1990 ~ 1995 = 1995

```
#Principle 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
#the annual rate of change represent a period of time and for this table I only save the last year period as name
#for easy merging table purpose
aroc=aroc.rename(columns={"Annual rate of change of the migrant stock (both sexes)":"b1995","Unnamed: 6":"b2000","Unnam
                          "Annual rate of change of the migrant stock (male)":"M1995","Unnamed: 11":"M2000","Unnamed: 12"
                          "Annual rate of change of the migrant stock (female)":"F1995","Unnamed: 16" : "F2000","Unnamed:
aroc
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1995	b2000	b2005	b2010	b2015	M1995	M2000	M2005	M2010	M2015
0	NaN	NaN	NaN	NaN	NaN	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015
1	1.0	WORLD	NaN	900.0	NaN	1.051865	1.428058	2.042124	2.95416	1.890991	1.000922	1.450294	2.151575	3.159228	1.912603
2	2.0	Developed regions	(b)	901.0	NaN	2.275847	2.264965	2.50708	2.466343	1.160824	2.265595	2.279583	2.483259	2.265689	1.074685
3	3.0	Developing regions	(c)	902.0	NaN	-0.487389	0.241777	1.328107	3.702217	2.929634	-0.45298	0.380246	1.693824	4.352954	2.927058
4	4.0	Least developed countries	(d)	941.0	NaN	1.118175	-3.001139	-0.539636	0.419137	3.526927	1.000073	-2.718952	0.078575	0.293964	3.363629
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	6.704748	4.90282	-0.858442	-2.299179	-0.768177	6.499035	4.704571	-1.066301	-2.50417	-0.987758
262	262.0	Tokelau	NaN	772.0	B	-0.298513	-0.303036	-0.307698	10.169947	2.536144	-0.404054	-0.412386	-1.589283	8.750541	2.463246

```
#drop the first non-informative rows
aroc= aroc[1:]
aroc
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1995	b2000	b2005	b2010	b2015	M1995	M2000	M2005	M2010	M2015
1	1.0	WORLD	NaN	900.0	NaN	1.051865	1.428058	2.042124	2.95416	1.890991	1.000922	1.450294	2.151575	3.159228	1.912603
2	2.0	Developed regions	(b)	901.0	NaN	2.275847	2.264965	2.50708	2.466343	1.160824	2.265595	2.279583	2.483259	2.265689	1.074685
3	3.0	Developing regions	(c)	902.0	NaN	-0.487389	0.241777	1.328107	3.702217	2.929634	-0.45298	0.380246	1.693824	4.352954	2.927058
4	4.0	Least developed countries	(d)	941.0	NaN	1.118175	-3.001139	-0.539636	0.419137	3.526927	1.000073	-2.718952	0.078575	0.293964	3.363629

```
#because this is the annual rage of change status,we are missing base year value
#for easy computational purpose, I asign three base year columns according to gender(both,male,female)
aroc.insert(5,"b1990",pd.NA)
aroc.insert(11,"M1990",pd.NA)
aroc.insert(17,"F1990",pd.NA)
aroc
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	...	M2000	M2005	M2010	M2015	F1990	
1	1.0	WORLD	NaN	900.0	NaN	<NA>	1.051865	1.428058	2.042124	2.95416	...	1.450294	2.151575	3.159228	1.912603	<NA>	1.1
2	2.0	Developed regions	(b)	901.0	NaN	<NA>	2.275847	2.264965	2.50708	2.466343	...	2.279583	2.483259	2.265689	1.074685	<NA>	2.2
3	3.0	Developing regions	(c)	902.0	NaN	<NA>	-0.487389	0.241777	1.328107	3.702217	...	0.380246	1.693824	4.352954	2.927058	<NA>	-0.5
4	4.0	Least developed countries	(d)	941.0	NaN	<NA>	1.118175	-3.001139	-0.539636	0.419137	...	-2.718952	0.078575	0.293964	3.363629	<NA>	1.2
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	<NA>	-0.803244	0.850177	1.62934	4.159339	...	0.950231	1.952269	4.90598	2.87349	<NA>	-0

As mentioned before, the table will be merged with other tables and for combining purposes (same amount of columns), dummy columns are created.

Principle 2: Each column needs to consist of one and only one variable

This table has the same issue as previous tables in that some columns has 2 variables and this need to be split. I will use the same way to clean this table as before, see the below pictures for a detailed process.

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
aroc1 = aroc.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","T
aroc1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Annual rate of change of the migrant stock(%)
0	1.0	WORLD	NaN	900.0	NaN	b1990	<NA>
1	2.0	Developed regions	(b)	901.0	NaN	b1990	<NA>
2	3.0	Developing regions	(c)	902.0	NaN	b1990	<NA>
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	<NA>
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	<NA>
...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	F2015	-0.545343

```
#split temporary column to two different columns with one and only variable
aroc1 = aroc1.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
aroc1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Annual rate of change of the migrant stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	b1990	<NA>	b	1990
1	2.0	Developed regions	(b)	901.0	NaN	b1990	<NA>	b	1990
2	3.0	Developing regions	(c)	902.0	NaN	b1990	<NA>	b	1990
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	<NA>	b	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	<NA>	b	1990
...	...	...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	F2015	-0.545343	F	2015

```
#drop the temporary column years_renamed, rename other columns to be more informative
aroc2=aroc1.drop(['years_renamed'], axis=1,inplace=True)
aroc2=aroc1.replace(to_replace=["b","F","M"],value=["Both","Male","Female"])
aroc2
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Annual rate of change of the migrant stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	<NA>	Both	1990
1	2.0	Developed regions	(b)	901.0	NaN	<NA>	Both	1990
2	3.0	Developing regions	(c)	902.0	NaN	<NA>	Both	1990
3	4.0	Least developed countries	(d)	941.0	NaN	<NA>	Both	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	<NA>	Both	1990
...	...	...	...	...	...	...	...	...
4765	261.0	Samoa	NaN	882.0	B	-0.545343	Male	2015
4766	262.0	Tokelau	NaN	772.0	B	2.60325	Male	2015
4767	263.0	Tonga	NaN	776.0	B	2.526318	Male	2015
4768	264.0	Tuvalu	NaN	798.0	C	-1.819436	Male	2015
4769	265.0	Wallis and Futuna Islands	NaN	876.0	B	0.516899	Male	2015

4770 rows × 8 columns

Styled the names to the correct presentation.

Principle 3: Variables need to be on cells, not rows and columns

Variables in this table are already in cells as went through the previous cleaning process.

Principle 4: Each table column needs to have a singular data type

Same as what I have done for other tables, I will first check the data type of this table using info() function as usual. Once again, Jupyter adds one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
aroc2.info()
#there are 6 columns are object and 2 columns are float,split them into two tables by data type+unique combination
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4770 entries, 0 to 4769
Data columns (total 8 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Sort
order                                          4770 non-null   float64
 1   Major area, region, country or area of destination  4770 non-null   object
 2   Notes                                           468 non-null    object
 3   Country code                                    4770 non-null   float64
 4   Type of data (a)                                4176 non-null   object
 5   Annual rate of change of the migrant stock(%)   3975 non-null   object
 6   Gender                                          4770 non-null   object
 7   Year                                            4770 non-null   object
dtypes: float64(2), object(6)
memory usage: 298.2+ KB
```

```
#clean decimal places to integer, back to original number presentation in the excel sheet
aroc2["Sort\norder"]=aroc2["Sort\norder"].astype("int64")
aroc2["Country code"]=aroc2["Country code"].astype("int64")
aroc2.head(10)
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Annual rate of change of the migrant stock(%)	Gender	Year
0	1	WORLD	NaN	900	NaN	<NA>	Both	1990
1	2	Developed regions	(b)	901	NaN	<NA>	Both	1990
2	3	Developing regions	(c)	902	NaN	<NA>	Both	1990
3	4	Least developed countries	(d)	941	NaN	<NA>	Both	1990
4	5	Less developed regions excluding least develop...	NaN	934	NaN	<NA>	Both	1990
5	6	Sub-Saharan Africa	(e)	947	NaN	<NA>	Both	1990

After that, I find the unique combination within the objective data type, set them as index and then create a new table called Main Table.

```
aroc3 = aroc2.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","Annual rate c
aroc3
```

Major area, region, country or area of destination	Notes	Type of data (a)	Annual rate of change of the migrant stock(%)	Gender	Year	Sort\norder	Country code
WORLD	NaN	NaN	NaN	Both	1990	1	900
Developed regions	(b)	NaN	NaN	Both	1990	2	901
Developing regions	(c)	NaN	NaN	Both	1990	3	902
Least developed countries	(d)	NaN	NaN	Both	1990	4	941
Less developed regions excluding least developed countries	NaN	NaN	NaN	Both	1990	5	934

```
#setting up Maintable for table 5 excel sheet
Maintable = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","Annual rate of chang
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(aroc3.index.unique())
    ],
)
```

```
Maintable.head(10)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Annual rate of change of the migrant stock(%)	Gender	Year
0	1	WORLD	NaN	NaN	NaN	Both	1990
1	2	Developed regions	(b)	NaN	NaN	Both	1990
2	3	Developing regions	(c)	NaN	NaN	Both	1990
3	4	Least developed countries	(d)	NaN	NaN	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	NaN	Both	1990
5	6	Sub-Saharan Africa	(e)	NaN	NaN	Both	1990
6	7	Africa	NaN	NaN	NaN	Both	1990
7	8	Eastern Africa	NaN	NaN	NaN	Both	1990

Will do the same for the appendix table within int64 data type.

```
#setting up Appendix table for table 5 excel sheet
Appendix = aroc3[["Sort\norder","Country code"]].copy()
Appendix["ID"] = Maintable.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","
```

```
Appendix = Appendix.reset_index(drop=True).set_index("ID")
Appendix.head(20)
```

ID	Sort\norder	Country code
1	1	900
2	2	901
3	3	902
4	4	941
5	5	934
6	6	947
7	7	903
8	8	910
9	9	108

```
Maintable.set_index(["ID"])
```

ID	Major area, region, country or area of destination	Notes	Type of data (a)	Annual rate of change of the migrant stock(%)	Gender	Year
1	WORLD	NaN	NaN	NaN	Both	1990
2	Developed regions	(b)	NaN	NaN	Both	1990
3	Developing regions	(c)	NaN	NaN	Both	1990
4	Least developed countries	(d)	NaN	NaN	Both	1990
5	Less developed regions excluding least develop...	NaN	NaN	NaN	Both	1990
...	...	...	...	...	...	...
4766	Samoa	NaN	B	-0.545343	Male	2015
4767	Tokelau	NaN	B	2.60325	Male	2015
4768	Tonga	NaN	B	2.526318	Male	2015

Replace missing values in the dataset.

```
#replace missing data with NA for future computational use
Maintable.replace(to_replace="..",value=pd.NA,inplace=True)
```

```
Maintable.head(15)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Annual rate of change of the migrant stock(%)	Gender	Year
0	1	WORLD	NaN	NaN	NaN	Both	1990
1	2	Developed regions	(b)	NaN	NaN	Both	1990
2	3	Developing regions	(c)	NaN	NaN	Both	1990
3	4	Least developed countries	(d)	NaN	NaN	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	NaN	Both	1990
5	6	Sub-Saharan Africa	(e)	NaN	NaN	Both	1990
6	7	Africa	NaN	NaN	NaN	Both	1990
7	8	Eastern Africa	NaN	NaN	NaN	Both	1990
8	9	Burundi	NaN	B R	NaN	Both	1990
9	10	Comoros	NaN	B	NaN	Both	1990

Principle 5: A single observational unit must be in 1 table

Both the main and appendix table has already been set to a single observational based on the previous cleaning process.

Merged Main Table: Trends in International Migrants Stock 2015 Revision Main Table

According to Principle 5, a single observation unit must be in 1 table and based on the previous cleaning process. The first five tables have each been cleaned on their own. But on a holistic level, there are some of these tables can be grouped into the same observation because they have the same columns of information and they all measured the same things. Therefore, I used the merge function in pandas and combined the main table of table 1, table 3, table 4 and table 5 together to form a new table with observations on Trends in International Migrants Stock 2015 Revision.

I first export these cleaned tables using Maintable.to_excel function and make sure they are ready to use.

Table1_Maintable.xlsx

Table2_Maintable.xlsx

Table3_Maintable.xlsx

Table4_Maintable.xlsx

Table5_Maintable.xlsx

```
import pandas as pd

data1 = pd.read_excel("Table1_Maintable.xlsx")
data3 = pd.read_excel("Table3_Maintable.xlsx")
data4 = pd.read_excel("Table4_Maintable.xlsx")
data4.pop("ID")
#because table 4 only have single sex which is female, and I will drop ID columns to avoid ID conflicts for new merged
data5 = pd.read_excel("Table5_Maintable.xlsx")
```

```
mergel = data1.merge(data3,how="outer")
mergel
```

	Unnamed: 0	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year	International migrant stock as a percentage of the total population(%)
0	0	1	WORLD	NaN	NaN	152563212.0	Both	1990	2.873310
1	1	2	Developed regions	(b)	NaN	82378628.0	Both	1990	7.198015
2	2	3	Developing regions	(c)	NaN	70184584.0	Both	1990	1.685021
3	3	4	Least developed countries	(d)	NaN	11075966.0	Both	1990	2.171513
4	4	5	Less developed regions excluding least develop...	NaN	NaN	59105261.0	Both	1990	1.617042

```
#using pop function drop the temporary columns that creaed by defult
mergel.pop("Unnamed: 0")
```

```
0          0
1          1
2          2
3          3
4          4
       ...
4765    4765
4766    4766
4767    4767
4768    4768
4769    4769
Name: Unnamed: 0, Length: 4770, dtype: int64
```

Pandas has created some dummy unamed columns by default and I used the pop function to drop these columns. To make sure no column conflicts, each time merge a table I need to drop this type of dummy column. Then I used merge function with the outer option. It will use the union of keys from both frames make full outer join.

```
merge2 = merge1.merge(data4,how="outer")
merge2
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year	International migrant stock as a percentage of the total population(%)	Unnamed: 0	Female migrants as a percentage of the international migrant stock(%)
0	1	WORLD	NaN	NaN	152563212.0	Both	1990	2.873310	NaN	NaN
1	2	Developed regions	(b)	NaN	82378628.0	Both	1990	7.198015	NaN	NaN
2	3	Developing regions	(c)	NaN	70184584.0	Both	1990	1.685021	NaN	NaN
3	4	Least developed countries	(d)	NaN	11075966.0	Both	1990	2.171513	NaN	NaN
4	5	Less developed regions excluding least develop...	NaN	NaN	59105261.0	Both	1990	1.617042	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
4765	4766	Samoa	NaN	B	2460.0	Male	2015	2.628654	NaN	NaN
4766	4767	Tokelau	NaN	B	254.0	Male	2015	NaN	NaN	NaN

```
#using pop function drop the temporary columns that creaed by deful
merge2.pop("Unnamed: 0")
```

```
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
        ..
4765    NaN
4766    NaN
4767    NaN
4768    NaN
4769    NaN
Name: Unnamed: 0, Length: 4770, dtype: float64
```

```
Maintable = merge2.merge(data5,how="outer")
Maintable
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year	International migrant stock as a percentage of the total population(%)	Female migrants as a percentage of the international migrant stock(%)	Unnamed: 0	Annual rate of change of the migrant stock(%)
0	1	WORLD	NaN	NaN	152563212.0	Both	1990	2.873310	NaN	0	NaN
1	2	Developed regions	(b)	NaN	82378628.0	Both	1990	7.198015	NaN	1	NaN
2	3	Developing regions	(c)	NaN	70184584.0	Both	1990	1.685021	NaN	2	NaN
3	4	Least developed countries	(d)	NaN	11075966.0	Both	1990	2.171513	NaN	3	NaN
4	5	Less developed regions excluding least develop...	NaN	NaN	59105261.0	Both	1990	1.617042	NaN	4	NaN
...	...	...	...	...	...	...	...	...	...	...	...
4765	4766	Samoa	NaN	B	2460.0	Male	2015	2.628654	NaN	4765	-0.545343
4766	4767	Tokelau	NaN	B	254.0	Male	2015	NaN	NaN	4766	2.603250
4767	4768	Tonga	NaN	B	2604.0	Male	2015	4.919612	NaN	4767	2.526318
4768	4769	Tuvalu	NaN	C	63.0	Male	2015	NaN	NaN	4768	-1.819436
4769	4770	Wallis and Futuna Islands	NaN	B	1411.0	Male	2015	NaN	NaN	4769	0.516899

4770 rows × 11 columns

```
#using pop function drop the temporary columns that creaed by deful
Maintable.pop("Unnamed: 0")
```

```
0          0
1          1
2          2
3          3
4          4
        ...
4765    4765
4766    4766
4767    4767
4768    4768
4769    4769
Name: Unnamed: 0, Length: 4770, dtype: int64
```

```
Maintable.head()
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock at mid-year	Gender	Year	International migrant stock as a percentage of the total population(%)	Female migrants as a percentage of the international migrant stock(%)	Annual rate of change of the migrant stock(%)
0	1	WORLD	NaN	NaN	152563212.0	Both	1990	2.873310	NaN	NaN
1	2	Developed regions	(b)	NaN	82378628.0	Both	1990	7.198015	NaN	NaN
2	3	Developing regions	(c)	NaN	70184584.0	Both	1990	1.685021	NaN	NaN
3	4	Least developed countries	(d)	NaN	11075966.0	Both	1990	2.171513	NaN	NaN
4	5	Less developed regions excluding least develop...	NaN	NaN	59105261.0	Both	1990	1.617042	NaN	NaN

```
#Rearrange the column order to make this new merged table more informative
Maintable=Maintable.iloc[:,[0,1,6,5,4,7,8,9,2,3]]
```

```
Maintable
```

	ID	Major area, region, country or area of destination	Year	Gender	International migrant stock at mid-year	International migrant stock as a percentage of the total population(%)	Female migrants as a percentage of the international migrant stock(%)	Annual rate of change of the migrant stock(%)	Notes	Type of data (a)
0	1	WORLD	1990	Both	152563212.0	2.873310	NaN	NaN	NaN	NaN
1	2	Developed regions	1990	Both	82378628.0	7.198015	NaN	NaN	(b)	NaN
2	3	Developing regions	1990	Both	70184584.0	1.685021	NaN	NaN	(c)	NaN
3	4	Least developed countries	1990	Both	11075966.0	2.171513	NaN	NaN	(d)	NaN
4	5	Less developed regions excluding least develop...	1990	Both	59105261.0	1.617042	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
4765	4766	Samoa	2015	Male	2460.0	2.628654	NaN	-0.545343	NaN	B

After merging these tables, we can see these columns are all about numbers and trends of international migrant stocks which each row can be considered as 1 single observation. This table does not include table 2 and table 6 because these two tables are not directly falling into the international migrant observation. Though, total population and refugee numbers do associate with international migrants to some extent.

```
Maintable.to_excel("Trends in International Migrants Stock 2015 Revision_maintable.xlsx")
```

At the end, I used the export function to export this new table and stored in the direction.

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

Different from the previous table, some Table 6 columns have single variable and some have multiple variables and those columns contain values from different aspects. For a clear and easy understanding approach, I will read and clean those columns separately and at the end use the merge function to create the main table and appendix table for this sheet.

```
#INF1340 DATA CELANING
import pandas as pd

undata = pd.ExcelFile("UN_MigrantStockTotal_2015.xlsx")

#there are multiple sheets in one excel, we should read them seperately
#Every sheets has few rows of data background information,for easy computational purpose,i choose skip those rows
ersdata = pd.read_excel(undata, 'Table 6',skiprows=14) #Estimated refugee stock at mid-year
```

```
#check basic dataset feasture
ersdata.head()
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	1995	2000	2005	2010.000000
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	...	11.103013	9.164736	6.941389	6.932687
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	...	3.910511	2.899391	2.015025	1.544140
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	...	20.795958	18.507035	14.733162	14.944759
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	...	44.041961	30.221557	24.08243	19.533425

After using pandas to read this excel, I will split it into 4 tables. There are three tables containing different measures namely, Estimated refugee stock at mid-year (both sexes), Refugees as a percentage of the international migrant stock and Annual rate of change of the refugee stock. There is also an appendix table containing all the common columns including sort order, major area, region, country or area of destination, notes, country code and type of data(a).

```
#some of this table's columns have one variables and some have multiple variables
#these columns contain values from diffent aspects under same observation
#for a more clear and easy understanding apporch, I will read those columns separately
```

```
#First seperated table:Estimated refugee stock at mid-year (both sexes)
ers1=ersdata.iloc[:,0:11]
ers1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	2015
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	19577474
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	1954224
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	17623250
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	3443582
...	...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	0	0	0	0	0	0
262	262.0	Tokelau	NaN	772.0	B	0	0	0	0	0	0
263	263.0	Tonga	NaN	776.0	B	0	0	0	0	0	0
264	264.0	Tuvalu	NaN	798.0	C	0	0	0	0	0	0
265	265.0	Wallis and Futuna Islands	NaN	876.0	B	0	0	0	0	0	0

266 rows × 11 columns

The first table is separated :Estimated refugee stock at mid-year (both sexes) named ers1.

```
#Second seperated table:Refugees as a percentage of the international migrant stock
ers2=ersdata.iloc[:,11:17]
ers2
```

	Refugees as a percentage of the international migrant stock	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16
0	1990	1995	2000	2005	2010.000000	2015.000000
1	12.346732	11.103013	9.164736	6.941389	6.932687	8.033424
2	2.445494	3.910511	2.899391	2.015025	1.544140	1.391085
3	23.968236	20.795958	18.507035	14.733162	14.944759	17.073768
4	45.56588	44.041961	30.221557	24.08243	19.533425	28.801534
...	...	...	...	...	...	...
261	0	0	0	0	0.000000	0.000000
262	0	0	0	0	0.000000	0.000000
263	0	0	0	0	0.000000	0.000000
264	0	0	0	0	0.000000	0.000000
265	0	0	0	0	0.000000	0.000000

Second separated table: Refugees as a percentage of the international migrant stock named ers2.

```
#Third seperated table:Annual rate of change of the refugee stock
ers3=ersdata.iloc[:,17:]
ers3
```

	Annual rate of change of the refugee stock	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnamed: 21
0	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015
1	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
2	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
3	-2.839417	-2.332154	-4.561	0.285195	2.663652
4	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
...	...	...	...	...	...
261	..	..	..	..	..
262	..	..	..	..	..
263	..	..	..	..	..
264	..	..	..	..	..
265	..	..	..	..	..

Third separated table: Annual rate of change of the refugee stock named ers3.

```
#Fourth seperated table:appendix code
ersappendix=ersdata.iloc[:,0:5]
ersappendix
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)
0	NaN	NaN	NaN	NaN	NaN
1	1.0	WORLD	NaN	900.0	NaN
2	2.0	Developed regions	(b)	901.0	NaN
3	3.0	Developing regions	(c)	902.0	NaN
4	4.0	Least developed countries	(d)	941.0	NaN
...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B
262	262.0	Tokelau	NaN	772.0	B
263	263.0	Tonga	NaN	776.0	B
264	264.0	Tuvalu	NaN	798.0	C
265	265.0	Wallis and Futuna Islands	NaN	876.0	B

Fourth separated table: appendix, named ersappendix.

After making those separated tables, I will clean then use tidy data principles separately. The process will be similar as previous cleaning process.

First Table

Principle 1: Column names need to be informative, variable names and not values

```
#Clean first table
#Principle 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
ers1=ers1.rename(columns={"Estimated refugee stock at mid-year (both sexes)":"b1990","Unnamed: 6":"b1995","Unnamed: 7":
                          })
ers1
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	b2015
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	2015
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	19577474
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	1954224
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	17623250
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	3443582
...	...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	0	0	0	0	0	0
262	262.0	Tokelau	NaN	772.0	B	0	0	0	0	0	0
263	263.0	Tonga	NaN	776.0	B	0	0	0	0	0	0
264	264.0	Tuvalu	NaN	798.0	C	0	0	0	0	0	0
265	265.0	Wallis and Futuna Islands	NaN	876.0	B	0	0	0	0	0	0

```
#drop the first non-informative rows
ers1= ers1[1:]
ers1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	b2015
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	19577474
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	1954224
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	17623250
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	3443582
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	11773616	9084039	9783059	8551722	11365954	14179668

Rename unnamed columns and drop the non- informative rows.

Principle 2: Each column needs to consist of one and only one variable

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
ers1 = ers1.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","Ty
ers1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Estimated refugee stock at mid-year (both sexes)
0	1.0	WORLD	NaN	900.0	NaN	b1990	18836571
1	2.0	Developed regions	(b)	901.0	NaN	b1990	2014564
2	3.0	Developing regions	(c)	902.0	NaN	b1990	16822007
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	5048391
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	11773616
...	...	...	...	...	...	...	...
1585	261.0	Samoa	NaN	882.0	B	b2015	0
1586	262.0	Tokelau	NaN	772.0	B	b2015	0
1587	263.0	Tonga	NaN	776.0	B	b2015	0
1588	264.0	Tuvalu	NaN	798.0	C	b2015	0
1589	265.0	Wallis and Futuna Islands	NaN	876.0	B	b2015	0

Created temporary columns *years_renamed*, now this column contains two variables and be able to be separated into two columns.

```
#split temporary column to two different columns with one and only variable
ers1 = ers1.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
ers1
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Estimated refugee stock at mid-year (both sexes)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	b1990	18836571	b	1990
1	2.0	Developed regions	(b)	901.0	NaN	b1990	2014564	b	1990
2	3.0	Developing regions	(c)	902.0	NaN	b1990	16822007	b	1990
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	5048391	b	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	11773616	b	1990
...	...	...	...	...	...	...	...	...	...
1585	261.0	Samoa	NaN	882.0	B	b2015	0	b	2015
1586	262.0	Tokelau	NaN	772.0	B	b2015	0	b	2015
1587	263.0	Tonga	NaN	776.0	B	b2015	0	b	2015
1588	264.0	Tuvalu	NaN	798.0	C	b2015	0	b	2015
1589	265.0	Wallis and Futuna Islands	NaN	876.0	B	b2015	0	b	2015

Separated columns with one viable gender and year are created, then I dropped the temporary column *years_renamed* and styled the values into the correct presentation.

```
#drop the temporary column years_renamed, rename other columns to be more informative
ers11=ers1.drop(['years_renamed'], axis=1,inplace=True)
ers11=ers1.replace(to_replace=["b"],value=["Both"])
ers11
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	18836571	Both	1990
1	2.0	Developed regions	(b)	901.0	NaN	2014564	Both	1990
2	3.0	Developing regions	(c)	902.0	NaN	16822007	Both	1990
3	4.0	Least developed countries	(d)	941.0	NaN	5048391	Both	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	11773616	Both	1990

Principle 3: Variables need to be on cells, not rows and columns
Variables in this table are already in cells.

Principle 4: Each table column needs to have a singular data type

Same as what I have done for other tables, I will first check the data type of this table using info() function as usual. Once again, Jupyter adds one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
ers11.info()
#there are 6 columns are object and 2 columns are float,split them into two tables by data type+unique combination
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1590 entries, 0 to 1589
Data columns (total 8 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Sort
order                                          1590 non-null   float64
 1   Major area, region, country or area of destination  1590 non-null   object
 2   Notes                                           156 non-null    object
 3   Country code                                    1590 non-null   float64
 4   Type of data (a)                                1392 non-null   object
 5   Estimated refugee stock at mid-year (both sexes)    1590 non-null   object
 6   Gender                                          1590 non-null   object
 7   Year                                            1590 non-null   object
dtypes: float64(2), object(6)
memory usage: 99.5+ KB
```

```
#clean decimal places to integer, back to original number presentation in the excel sheet
ers11["Sort\norder"]=ers11["Sort\norder"].astype("int64")
ers11["Country code"]=ers11["Country code"].astype("int64")
ers11.head(10)
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Gender	Year
0	1	WORLD	NaN	900	NaN	18836571	Both	1990
1	2	Developed regions	(b)	901	NaN	2014564	Both	1990
2	3	Developing regions	(c)	902	NaN	16822007	Both	1990
3	4	Least developed countries	(d)	941	NaN	5048391	Both	1990
4	5	Less developed regions excluding least develop...	NaN	934	NaN	11773616	Both	1990
5	6	Sub-Saharan Africa	(e)	947	NaN	5516042	Both	1990

```
ers11 = ers11.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","Estimated ref
ers11
```

						Sort\norder	Country code
Major area, region, country or area of destination	**Notes**	**Type of data (a)**	**Estimated refugee stock at mid-year (both sexes)**	**Gender**	**Year**		
WORLD	**NaN**	**NaN**	**18836571**	**Both**	**1990**	1	900
Developed regions	**(b)**	**NaN**	**2014564**	**Both**	**1990**	2	901
Developing regions	**(c)**	**NaN**	**16822007**	**Both**	**1990**	3	902
Least developed countries	**(d)**	**NaN**	**5048391**	**Both**	**1990**	4	941
Less developed regions excluding least developed countries	**NaN**	**NaN**	**11773616**	**Both**	**1990**	5	934

```
#setting up Maintable for sperated table 1
Maintable1 = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","Estimated refugee st
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(ers11.index.unique())
    ],
)
```

```
Maintable1.head()
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Gender	Year
0	1	WORLD	NaN	NaN	18836571	Both	1990
1	2	Developed regions	(b)	NaN	2014564	Both	1990
2	3	Developing regions	(c)	NaN	16822007	Both	1990
3	4	Least developed countries	(d)	NaN	5048391	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	11773616	Both	1990

The main table is created based on a singular data type with unique combination.

Second Table

Principle 1: Column names need to be informative, variable names and not values
The process of cleaning the second separated table would be similar to the previous process. However, since the stock number is indicated that it is for both sex, thus the percentage will be assumed for both sex as well.

I used concat function here to combine the second table with the appendix table to make sure though tables are clean separately but still with the same appendix. It will be easier to combine them back together later on.

```
#start cleaning second table
ers22=pd.concat([ersappendix,ers2],axis=1)
ers22
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Refugees as a percentage of the international migrant stock	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	2015.000000
1	1.0	WORLD	NaN	900.0	NaN	12.346732	11.103013	9.164736	6.941389	6.932687	8.033424
2	2.0	Developed regions	(b)	901.0	NaN	2.445494	3.910511	2.899391	2.015025	1.544140	1.391085
3	3.0	Developing regions	(c)	902.0	NaN	23.968236	20.795958	18.507035	14.733162	14.944759	17.073768
4	4.0	Least developed countries	(d)	941.0	NaN	45.56588	44.041961	30.221557	24.08243	19.533425	28.801534

```
#Principle 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
#As indicated in the excel sheet table 6, i will assume all values in columns are refer to both sex
ers22=ers22.rename(columns={"Refugees as a percentage of the international migrant stock":"b1990","Unnamed: 12":"b1995"
                            })
ers22
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	b2015
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	2015.000000
1	1.0	WORLD	NaN	900.0	NaN	12.346732	11.103013	9.164736	6.941389	6.932687	8.033424
2	2.0	Developed regions	(b)	901.0	NaN	2.445494	3.910511	2.899391	2.015025	1.544140	1.391085
3	3.0	Developing regions	(c)	902.0	NaN	23.968236	20.795958	18.507035	14.733162	14.944759	17.073768
4	4.0	Least developed countries	(d)	941.0	NaN	45.56588	44.041961	30.221557	24.08243	19.533425	28.801534

```
#drop the first non-informative rows
ers22= ers22[1:]
ers22
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	b2015
1	1.0	WORLD	NaN	900.0	NaN	12.346732	11.103013	9.164736	6.941389	6.932687	8.033424
2	2.0	Developed regions	(b)	901.0	NaN	2.445494	3.910511	2.899391	2.015025	1.544140	1.391085
3	3.0	Developing regions	(c)	902.0	NaN	23.968236	20.795958	18.507035	14.733162	14.944759	17.073768
4	4.0	Least developed countries	(d)	941.0	NaN	45.56588	44.041961	30.221557	24.08243	19.533425	28.801534
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	19.919743	15.999082	16.51313	13.305391	14.363526	15.537313

Rename unnamed columns and drop the non- informative rows.

Principle 2: Each column needs to consist of one and only one variable

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
ers22 = ers22.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","
ers22
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Refugees as a percentage of the international migrant stock(%)
0	1.0	WORLD	NaN	900.0	NaN	b1990	12.346732
1	2.0	Developed regions	(b)	901.0	NaN	b1990	2.445494
2	3.0	Developing regions	(c)	902.0	NaN	b1990	23.968236
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	45.56588
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	19.919743

Created temporary columns *years_renamed*, now this column contains two variables and be able to be separated into two columns.

```
#split temporary column to two different columns with one and only variable
ers22 = ers22.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
ers22
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Refugees as a percentage of the international migrant stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	b1990	12.346732	b	1990
1	2.0	Developed regions	(b)	901.0	NaN	b1990	2.445494	b	1990
2	3.0	Developing regions	(c)	902.0	NaN	b1990	23.968236	b	1990
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	45.56588	b	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	19.919743	b	1990

```
#drop the temporary column years_renamed, rename other columns to be more informative
ers23=ers22.drop(['years_renamed'], axis=1,inplace=True)
ers23=ers22.replace(to_replace=["b"],value=["Both"])
ers23
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Refugees as a percentage of the international migrant stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	12.346732	Both	1990
1	2.0	Developed regions	(b)	901.0	NaN	2.445494	Both	1990
2	3.0	Developing regions	(c)	902.0	NaN	23.968236	Both	1990
3	4.0	Least developed countries	(d)	941.0	NaN	45.56588	Both	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	19.919743	Both	1990
...	...	...	...	...	...	...	...	...
1585	261.0	Samoa	NaN	882.0	B	0.0	Both	2015
1586	262.0	Tokelau	NaN	772.0	B	0.0	Both	2015

Separated two new columns, year and gender and styled the columns to be more informative.
E.g. b1990= Both 1990

Principle 3: Variables need to be on cells, not rows and columns
Variables in this table are already in cells.

Principle 4: Each table column needs to have a singular data type

Same as what I have done for other tables, I will first check the data type of this table using info() function as usual. Once again, Jupyter adds one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
ers23.info()
#there are 6 columns are object and 2 columns are float,split them into two tables by data type+unique combination
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1590 entries, 0 to 1589
Data columns (total 8 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort
order                                                 1590 non-null   float64
 1   Major area, region, country or area of destination             1590 non-null   object
 2   Notes                                                           156 non-null    object
 3   Country code                                                    1590 non-null   float64
 4   Type of data (a)                                                1392 non-null   object
 5   Refugees as a percentage of the international migrant stock(%)  1590 non-null   object
 6   Gender                                                          1590 non-null   object
 7   Year                                                            1590 non-null   object
dtypes: float64(2), object(6)
memory usage: 99.5+ KB
```

```
#clean decimal places to integer, back to original number presentation in the excel sheet
ers23["Sort\norder"]=ers23["Sort\norder"].astype("int64")
ers23["Country code"]=ers23["Country code"].astype("int64")
ers23.head(10)
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Refugees as a percentage of the international migrant stock(%)	Gender	Year
0	1	WORLD	NaN	900	NaN	12.346732	Both	1990
1	2	Developed regions	(b)	901	NaN	2.445494	Both	1990
2	3	Developing regions	(c)	902	NaN	23.968236	Both	1990
3	4	Least developed countries	(d)	941	NaN	45.56588	Both	1990
4	5	Less developed regions excluding least develop...	NaN	934	NaN	19.919743	Both	1990

```
#double check data type, ready to split into tables by data type following pricinple 4
ers23.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1590 entries, 0 to 1589
Data columns (total 8 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort
order                                                 1590 non-null   int64
 1   Major area, region, country or area of destination             1590 non-null   object
 2   Notes                                                           156 non-null    object
 3   Country code                                                    1590 non-null   int64
 4   Type of data (a)                                                1392 non-null   object
 5   Refugees as a percentage of the international migrant stock(%)  1590 non-null   object
 6   Gender                                                          1590 non-null   object
 7   Year                                                            1590 non-null   object
dtypes: int64(2), object(6)
memory usage: 99.5+ KB
```

Double-check the data type.

```
ers23 = ers23.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","Refugees as a
ers23
```

						Sort\norder	Country code
Major area, region, country or area of destination	**Notes**	**Type of data (a)**	**Refugees as a percentage of the international migrant stock(%)**	**Gender**	**Year**		
WORLD	**NaN**	**NaN**	**12.346732**	**Both**	**1990**	1	900
Developed regions	**(b)**	**NaN**	**2.445494**	**Both**	**1990**	2	901
Developing regions	**(c)**	**NaN**	**23.968236**	**Both**	**1990**	3	902
Least developed countries	**(d)**	**NaN**	**45.565880**	**Both**	**1990**	4	941
Less developed regions excluding least developed countries	**NaN**	**NaN**	**19.919743**	**Both**	**1990**	5	934

```
#setting up Maintable for sperated table 2
Maintable2 = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","International migran
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(ers23.index.unique())
    ],
)
```

```
Maintable2.head()
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	International migrant stock as a percentage of the total population(%)	Gender	Year
0	1	WORLD	NaN	NaN	12.346732	Both	1990
1	2	Developed regions	(b)	NaN	2.445494	Both	1990
2	3	Developing regions	(c)	NaN	23.968236	Both	1990
3	4	Least developed countries	(d)	NaN	45.56588	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	19.919743	Both	1990

The main table is created based on a singular data type with unique combination.

Third Table

The process of cleaning the third separated table would be similar to the previous process. However, since the stock number is indicated that it is for both sex, thus the annual rate of change will be assumed for both sex as well.
I used the concat function here to combine the third table with the appendix table to make sure though tables are clean separately but still with the same appendix. It will be easier to combine them back together later on.

Principle 1: Column names need to be informative, variable names and not values
Rename unnamed columns and drop the non- informative rows.

```
#start cleaning third table
ers33=pd.concat([ersappendix,ers3],axis=1)
ers33
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Annual rate of change of the refugee stock	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnamed: 21
0	NaN	NaN	NaN	NaN	NaN	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015
1	1.0	WORLD	NaN	900.0	NaN	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
2	2.0	Developed regions	(b)	901.0	NaN	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
3	3.0	Developing regions	(c)	902.0	NaN	-2.839417	-2.332154	-4.561	0.285195	2.663652
4	4.0	Least developed countries	(d)	941.0	NaN	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	..	..	..	..	..

```
#Principle 1:Column names need to be informative, variable names and not values
#rename unnamed columns to make them as imformative as the rest
#As indicated in the excel sheet table 6, i will assume all values in columns are refer to both sex
#the annual rate of change represent a period of time and for this table I only save the last year period as name
#for easy merging table purpose
ers33=ers33.rename(columns={"Annual rate of change of the refugee stock":"b1995","Unnamed: 18":"b2000","Unnamed: 19":"b
ers33
#Principle 3:Variables need to be on cells, not rows and columns
#after rename, columns have some variables in the title row and these are not informative as well
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1995	b2000	b2005	b2010	b2015
0	NaN	NaN	NaN	NaN	NaN	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015
1	1.0	WORLD	NaN	900.0	NaN	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
2	2.0	Developed regions	(b)	901.0	NaN	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
3	3.0	Developing regions	(c)	902.0	NaN	-2.839417	-2.332154	-4.561	0.285195	2.663652
4	4.0	Least developed countries	(d)	941.0	NaN	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
...	...	...	...	...	...	...	...	...	...	...

```
#drop the first non-informative rows
ers33= ers33[1:]
ers33
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1995	b2000	b2005	b2010	b2015
1	1.0	WORLD	NaN	900.0	NaN	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
2	2.0	Developed regions	(b)	901.0	NaN	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
3	3.0	Developing regions	(c)	902.0	NaN	-2.839417	-2.332154	-4.561	0.285195	2.663652
4	4.0	Least developed countries	(d)	941.0	NaN	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	-4.3836	0.632489	-4.319731	1.530456	1.571047
...	...	...	...	...	...	...	...	...	...	...

Rename unnamed columns and drop the non- informative rows.

```
#because this is the annual rage of change status,we are missing base year value
#for easy computational purpose, I create a dummy columns - base year according to gender(both)
ers33.insert(5,"b1990",pd.NA)
ers33
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	b1990	b1995	b2000	b2005	b2010	b2015
1	1.0	WORLD	NaN	900.0	NaN	<NA>	-2.123497	-3.837069	-5.557223	-0.025089	2.947267
2	2.0	Developed regions	(b)	901.0	NaN	<NA>	9.388424	-5.983348	-7.277379	-5.323293	-2.087656
3	3.0	Developing regions	(c)	902.0	NaN	<NA>	-2.839417	-2.332154	-4.561	0.285195	2.663652
4	4.0	Least developed countries	(d)	941.0	NaN	<NA>	-0.680327	-7.531747	-4.541459	-4.187109	7.766031
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	<NA>	-4.3836	0.632489	-4.319731	1.530456	1.571047
...	...	...	...	...	...	...	...	...	...	...	...
261	261.0	Samoa	NaN	882.0	B	<NA>	..	..	..	..	..

Similar to what I did for international migrant's annual rate of change, I will create dummy columns, and values in cells will be NA in order to easily merge tables together for refugees observation (principle 5).

Principle 2: Each column needs to consist of one and only one variable

```
#Principle 2:Each column needs to consist of one and only one variable
#columns contains multiple variables need to be melted and rearranged,create temporary columns
ers33 = ers33.melt(id_vars=["Sort\norder","Major area, region, country or area of destination","Notes","Country code","
ers33
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Annual rate of change of the refugee stock(%)
0	1.0	WORLD	NaN	900.0	NaN	b1990	<NA>
1	2.0	Developed regions	(b)	901.0	NaN	b1990	<NA>
2	3.0	Developing regions	(c)	902.0	NaN	b1990	<NA>
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	<NA>
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	<NA>
...	...	...	...	...	...	...	...
1585	261.0	Samoa	NaN	882.0	B	b2015	..

Created temporary columns *years_renamed*, now this column contains two variables and be able to be separated into two columns.

```
#split temporary column to two different columns with one and only variable
ers33 = ers33.assign(Gender = lambda x : x.years_renamed.str[0], Year = lambda x : x.years_renamed.str[1:])
ers33
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	years_renamed	Annual rate of change of the refugee stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	b1990	<NA>	b	1990
1	2.0	Developed regions	(b)	901.0	NaN	b1990	<NA>	b	1990
2	3.0	Developing regions	(c)	902.0	NaN	b1990	<NA>	b	1990
3	4.0	Least developed countries	(d)	941.0	NaN	b1990	<NA>	b	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	b1990	<NA>	b	1990
...	...	...	...	...	...	...	...	...	...
1585	261.0	Samoa	NaN	882.0	B	b2015	..	b	2015

```
#drop the temporary column years_renamed, rename other columns to be more informative
ers34=ers33.drop(['years_renamed'], axis=1,inplace=True)
ers34=ers33.replace(to_replace=["b"],value=["Both"])
ers34
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Annual rate of change of the refugee stock(%)	Gender	Year
0	1.0	WORLD	NaN	900.0	NaN	<NA>	Both	1990
1	2.0	Developed regions	(b)	901.0	NaN	<NA>	Both	1990
2	3.0	Developing regions	(c)	902.0	NaN	<NA>	Both	1990
3	4.0	Least developed countries	(d)	941.0	NaN	<NA>	Both	1990
4	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	<NA>	Both	1990
...	...	...	...	...	...	...	...	...
1585	261.0	Samoa	NaN	882.0	B	..	Both	2015

Separated two new columns, year and gender and styled the columns to be more informative.
E.g. b1995= Both 1990

Principle 3: Variables need to be on cells, not rows and columns
Variables in this table are already in cells.

Principle 4: Each table column needs to have a singular data type

Same as what I have done for other tables, I will first check the data type of this table using info() function as usual. Once again, Jupyter adds one decimal place automatically to the sort order and country code columns and the data type has changed. I will change this back to the original presentation as integer.

```
#Principle 4:each table column needs to have a singular data type
#check the current data stye
ers34.info()
#there are 6 columns are object and 2 columns are float,split them into two tables by data type+unique combination
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1590 entries, 0 to 1589
Data columns (total 8 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Sort
order                                       1590 non-null   float64
 1   Major area, region, country or area of destination  1590 non-null   object
 2   Notes                                           156 non-null    object
 3   Country code                                    1590 non-null   float64
 4   Type of data (a)                                1392 non-null   object
 5   Annual rate of change of the refugee stock(%)   1325 non-null   object
 6   Gender                                          1590 non-null   object
 7   Year                                            1590 non-null   object
dtypes: float64(2), object(6)
memory usage: 99.5+ KB
```

```
#clean decimal places to integer, back to original number presentation in the excel sheet
ers34["Sort\norder"]=ers34["Sort\norder"].astype("int64")
ers34["Country code"]=ers34["Country code"].astype("int64")
ers34.head(10)
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Annual rate of change of the refugee stock(%)	Gender	Year
0	1	WORLD	NaN	900	NaN	<NA>	Both	1990
1	2	Developed regions	(b)	901	NaN	<NA>	Both	1990
2	3	Developing regions	(c)	902	NaN	<NA>	Both	1990
3	4	Least developed countries	(d)	941	NaN	<NA>	Both	1990
4	5	Less developed regions excluding least develop...	NaN	934	NaN	<NA>	Both	1990
5	6	Sub-Saharan Africa	(e)	947	NaN	<NA>	Both	1990
6	7	Africa	NaN	903	NaN	<NA>	Both	1990

```
ers34 = ers34.set_index(["Major area, region, country or area of destination","Notes","Type of data (a)","Annual rate c
ers34
```

							Sort\norder	Country code
Major area, region, country or area of destination	Notes	Type of data (a)	Annual rate of change of the refugee stock(%)	Gender	Year			
WORLD	NaN	NaN	NaN	Both	1990		1	900
Developed regions	(b)	NaN	NaN	Both	1990		2	901
Developing regions	(c)	NaN	NaN	Both	1990		3	902
Least developed countries	(d)	NaN	NaN	Both	1990		4	941
Less developed regions excluding least developed countries	NaN	NaN	NaN	Both	1990		5	934
...	...	...	...	...	...		...	...
Samoa	NaN	B	..	Both	2015		261	882

```
#setting up Maintable for table 6 excel sheet
Maintable3 = pd.DataFrame.from_records(
    columns=["ID","Major area, region, country or area of destination","Notes","Type of data (a)","Annual rate of chang
    data =[
        (a + 1,b,c,d,e,f,g)
        for (a, (b,c,d,e,f,g)) in enumerate(ers34.index.unique())
    ],
)
```

```
Maintable3.head()
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Annual rate of change of the refugee stock(%)	Gender	Year
0	1	WORLD	NaN	NaN	NaN	Both	1990
1	2	Developed regions	(b)	NaN	NaN	Both	1990
2	3	Developing regions	(c)	NaN	NaN	Both	1990
3	4	Least developed countries	(d)	NaN	NaN	Both	1990
4	5	Less developed regions excluding least develop...	NaN	NaN	NaN	Both	1990

The main table is created based on a singular data type with unique combination.

Use same way to set up appendix table and ready to combine all of tables together.

```
Appendix = Appendix.reset_index(drop=True).set_index("ID")
Appendix.head()
```

	Sort\norder	Country code
ID		
1	1	900
2	2	901
3	3	902
4	4	941
5	5	934

Principle 5: A single observational unit must be in 1 table

According to Principle 5, a single observation unit must be in 1 table. Since I separated this refuges stock related observation into small tables for easy computational purposes. Now, I will merge them back together to make sure it follows the tidy data principle 5. I used the merge function and outer option again.

```
merge1 = Maintable1.merge(Maintable2,how="outer")
merge1
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Gender	Year	International migrant stock as a percentage of the total population(%)
0	1	WORLD	NaN	NaN	18836571	Both	1990	12.346732
1	2	Developed regions	(b)	NaN	2014564	Both	1990	2.445494
2	3	Developing regions	(c)	NaN	16822007	Both	1990	23.968236
3	4	Least developed countries	(d)	NaN	5048391	Both	1990	45.56588
4	5	Less developed regions excluding least develop...	NaN	NaN	11773616	Both	1990	19.919743
...	...	...	...	...	...	...	...	...
1585	1586	Samoa	NaN	B	0	Both	2015	0
1586	1587	Tokelau	NaN	B	0	Both	2015	0

```
table6Maintable = merge1.merge(Maintable3,how="outer")
table6Maintable
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Gender	Year	International migrant stock as a percentage of the total population(%)	Annual rate of change of the refugee stock(%)
0	1	WORLD	NaN	NaN	18836571	Both	1990	12.346732	NaN
1	2	Developed regions	(b)	NaN	2014564	Both	1990	2.445494	NaN
2	3	Developing regions	(c)	NaN	16822007	Both	1990	23.968236	NaN
3	4	Least developed countries	(d)	NaN	5048391	Both	1990	45.56588	NaN
4	5	Less developed regions excluding least develop...	NaN	NaN	11773616	Both	1990	19.919743	NaN
...	...	...	...	...	...	...	...	...	...
1585	1586	Samoa	NaN	B	0	Both	2015	0	..

The tables have been merged. There are missing values in this table as well. I replaced them with NA.

```
#replace missing data with NA for future computational use
table6Maintable.replace(to_replace="..",value=pd.NA,inplace=True)
```

```
table6Maintable.head(1340)
```

	ID	Major area, region, country or area of destination	Notes	Type of data (a)	Estimated refugee stock at mid-year (both sexes)	Gender	Year	International migrant stock as a percentage of the total population(%)	Annual rate of change of the refugee stock(%)
0	1	WORLD	NaN	NaN	18836571	Both	1990	12.346732	NaN
1	2	Developed regions	(b)	NaN	2014564	Both	1990	2.445494	NaN
2	3	Developing regions	(c)	NaN	16822007	Both	1990	23.968236	NaN
3	4	Least developed countries	(d)	NaN	5048391	Both	1990	45.56588	NaN
4	5	Less developed regions excluding least develop...	NaN	NaN	11773616	Both	1990	19.919743	NaN
...	...	...	...	...	...	...	...	...	...
1335	1336	Djibouti	NaN	B R	20530	Both	2015	18.273091	4.121943

Then I rearranged the sort of these columns, and try to make this new table more informative.

```
#Rearrange the column order to make this new merged table more informative
table6Maintable=table6Maintable.iloc[:,[0,1,6,5,4,7,8,2,3]]
```

```
table6Maintable.head()
```

	ID	Major area, region, country or area of destination	Year	Gender	Estimated refugee stock at mid-year (both sexes)	International migrant stock as a percentage of the total population(%)	Annual rate of change of the refugee stock(%)	Notes	Type of data (a)
0	1	WORLD	1990	Both	18836571	12.346732	NaN	NaN	NaN
1	2	Developed regions	1990	Both	2014564	2.445494	NaN	(b)	NaN
2	3	Developing regions	1990	Both	16822007	23.968236	NaN	(c)	NaN
3	4	Least developed countries	1990	Both	5048391	45.56588	NaN	(d)	NaN
4	5	Less developed regions excluding least develop...	1990	Both	11773616	19.919743	NaN	NaN	NaN

```
table6Maintable.to_excel("Estimated refugee stock at mid-year.xlsx")
```

To conclude, this write-up provides comprehensive explanations of the data-cleaning process of all six tables by using tidy data principles with a focus on easy computational purposes. There are three cleaned main tables created in total. The first table is about total population stocks, the second table is about international migrant stocks merged by cleaned tables 1,3,4 and 5. The last table is about estimated guess stocks.