Submission Details - Anshuta R Kulkarni

1. Introduction

This dataset is about the trends in International Migrant Stock: 2015 revision, Department of Economic and Social Affairs, United Nations. The excel workbook contains 8 sheets - 6 tables, 1 'Annex', and 1 'Notes'. The objective is to clean the data efficiently and make it machine-readable for analysis using tidy data principles.

2. Approach

Since the data is split into multiple tables across sheets, it would be best to have all of this data combined into one or two tables based on their KPIs/variables. After studying the data, I decided to combine tables 1,2,3, and 4 as one data frame along with some data from table 6 and another data frame with values from table 5 and a section of table 6. The first dataframe would contain values of the KPIs with years as key while the second dataframe would contain values of the KPIs with the year ranges as key.

As most of the tables follow a similar data frame structure, I decided to put them in a for loop to carry out the processing of each table to reduce code redundancy. Starting with creating a for loop to run through each sheet/table in the excel file, the data is processed, cleaned and restructured based on its KPIs/variables using conditional if statements.

The code in the for loop and subsequently, the conditional if statements, is repeated to a certain extent across all the tables. In this phase, I focused on creating relevant column headings for each of the tables, so that I could split them efficiently based on the year information. This was done by pushing the first two rows of the data into a single list of column headings, followed by overwriting the original column names with this new, structured and cleaned column headings that include the year and sex as additional information. Using the tidy data principles, I melted the data frame and pivoted the new column containing year and sex information, in order to have that information as rows instead. Once they were in the row format, I split the variable into two columns - Year and Sex to contain their respective information.

This process is replicated across the remaining tables 2,3,4, and 5; with the additional change for year ranges for table 5. Table 6 was slightly different in structure than the rest, as it had 'year' information in both, individual years as well as year ranges, but with little information on 'sex' variable. I decided to split up this table into three sections based on their KPIs and their 'year' information. The first section contained singular years and sex information for that KPI, the second section contained singular year information for that KPI and no sex information and lastly, the third contained year range information for that KPI. This entire process was done in the for loop, using the conditional statements, giving the output of two lists. One list that contained five cleaned and structured data frames from Tables 1,2,3,4 and 5 respectively, and the other list contained the three split sections of table 6.

Keeping in line with the idea to create as few tables as possible as the final output, I decided to add the first two sections of table 6 to tables 1,2,3 and 4 as all of these tables used singular

year information - and combine all of them into one final data frame. Additionally, I created a second final data frame that merged the cleaned data frame from table 5 and the third section from table 6, as they shared the same structure for year ranges.

As the last phase of the wrangling process, I added information from the 'Annex' Sheet to both of these data frames in order to make them more structured and cleaner with the relevant information.

3. Methodology

I created a list with elements as numbers from 1 to 6, as the sheet numbers, and another list containing the KPIs or variable names for the values mentioned in the table.

```
## Creating lists for Sheet Number and KPIs in the excel sheet
Sheet_List = [1,2,3,4,5,6]
Value_List = ['International_Migrant_Stock_Value',
              'Total_Population_Value',
              'International_Migrant_Stock_Pcnt_Value',
              'Female_Migrant_Pnct_Value',
              'Annual_Rate_Change_Migrant_Value']
```

Using 'i' as my iterator variable, I initialized the iterator to start the loop with the value of 1. Initializing the value as 1 would make it easier for the sheet number to be selected in the group, rather than starting from a value of 0. Additionally, I created an empty list that would be populated with the cleaned data frames from each of the tables.
Starting with the for loop, the code begins with reading the excel file, keeping the 'sheet_name' as the iterator value in Sheet_List, using the pandas read_excel function.
Based on the tables in the excel file, it is confirmed that the actual table values start at row number 13. Therefore, to remove the empty space and irrelevant text, I used iloc to subset the data to contain all the rows from the 13th row to the end of the dataset, followed by resetting the index.
Moving on to restructure the headings, I decided to combine the first and second rows of the subsetted dataset. So the combination of these two rows would be the complete heading for the respective columns.
To push them into a single row of headings, I used the NaN in the first row to be populated by the equivalent values in the second row by index. I created a variable 'nan_index' to store the indices of the NaN values using is.null() function in the first row (index = 0). Using these indices, I extracted the values of the subheadings (years) from the second row (index = 1) and replace the NaN cells in the first row with these values.
To have structured column headings, I created a list of the first row, and convert all elements to string type. Additionally, some of the year values contain decimal values, so using replace function, I replaced all the decimal points with empty spaces, effectively converting the values

into integer values of years.

```
## Creating for loop
i = 1
DF_List = []
for i in Sheet_List:
    UN_Data = pd.read_excel("UN_MigrantStockTotal_2015.xlsx", sheet_name = i)
    # Subsetting for valid rows - removing first 12 rows
    UN_Data_Temp = UN_Data.iloc[13:,].reset_index()
    # Replacing NaN values in the first row from the second row year value
    nan_index = UN_Data_Temp.loc[0,:].isnull()
    UN_Data_Temp.loc[0, nan_index] = UN_Data_Temp.loc[1, nan_index]
    # Creating list for columns in dataframe
    List_Columns = UN_Data_Temp.iloc[0,:].tolist()
    # Converting all list elements to string
    List_Columns = [str(i) for i in List_Columns]
    # Replacing '.0' with empty '' for data structuring
    List_Columns= list(map(lambda x: x.replace('.0', ''), List_Columns))
```

This above process is common across all the tables, so I decided to keep it as the first phase in the loop.

Once this phase is completed, I started to move into cleaning each of the tables based on their sheet_name value. As mentioned earlier, the Excel workbook contains 6 tables - the first 5 of these tables follow a similar format, while the 6th table is slightly different in structure. To maintain that structure, I decided to create a conditional if statement based on the value of the iterator into 2 sections - one for i <= 5 and the other for i == 6.
Further, I added conditional if statements for each of the iterator values to process each of the tables separately, thereby creating a nested if statement within the for loop.
To maintain the structure of each of the tables, I created a block of code for processing the heading information. I then re-used this code across each of the conditional statements, with minor changes based on the table's KPI values.

For the 1st table -

```
if (i <= 5):
    if((i==1)):
        # Adding suffix to each section of international migrant stock information based on the sex information in
        # heading columns
        # For both sexes
        List_Columns[6] = '1990_BS'
        List_Columns[7:12] = [i + '_BS' for i in List_Columns[7:12]]
        # For male
        List_Columns[12] = '1990_M'
        List_Columns[13:18] = [i + '_M' for i in List_Columns[13:18]]
        # For female
        List_Columns[18] = '1990_F'
        List_Columns[19:24] = [i + '_F' for i in List_Columns[19:24]]

        # Renaming the columns with the new column list
        UN_Data_Temp.columns = List_Columns
        # Dropping first column and first and second row
        UN_Data_Temp = UN_Data_Temp.drop(columns = ['13'])
        UN_Data_Temp = UN_Data_Temp.drop([0,1])

        UN_Data_Temp = UN_Data_Temp.melt(id_vars = ['Sort\norder',
                                    'Major area, region, country or area of destination',
                                    'Notes',
                                    'Country code',
                                    'Type of data (a)'],
                         var_name = "Year_Sex_Var", value_name = Value_List[i-1])
```

Using tidy data principles, I focused on pushing the years from the column's headings into a separate column, which later could be used as an identifier or key variable. To maintain information about the sex in the heading for each of those columns, I added a suffix after the years based on the categorization of the columns under the main heading. This list now contains all the cleaned column headings. I used this list to overwrite the current column headings in the data frame. Then, I deleted the first column '13', and the first and second row, as they were irrelevant to the dataset.
Finally, using the 'melt' function, I pivoted these column years and sex headings into a separate column called 'Year_Sex_Var' and named the value column from the value list initialized in the first phase.

This was replicated for each of the tables based on their KPI value and year and sex information.

Table 2:

```
elif(i==2):
    # Adding suffix to each section of international migrant stock information based on the sex information in
    # heading columns
    # For both sexes
    List_Columns[5] = '1990_BS'
    List_Columns[6:11] = [i + '_BS' for i in List_Columns[6:11]]
    # For male
    List_Columns[11] = '1990_M'
    List_Columns[12:17] = [i + '_M' for i in List_Columns[12:17]]
    # For female
    List_Columns[17] = '1990_F'
    List_Columns[18:23] = [i + '_F' for i in List_Columns[18:23]]

    # Renaming the columns with the new column list
    UN_Data_Temp.columns = List_Columns
    # Dropping first column and first and second row
    UN_Data_Temp = UN_Data_Temp.drop(columns = ['13'])
    UN_Data_Temp = UN_Data_Temp.drop([0,1])

    UN_Data_Temp = UN_Data_Temp.melt(id_vars = ['Sort\norder',
                                'Major area, region, country or area of destination',
                                'Notes',
                                'Country code'],
                     var_name = "Year_Sex_Var", value_name = Value_List[i-1])
```

Table 3:

```
elif((i==3)):
    # Adding suffix to each section of international migrant stock information based on the sex information in
    # heading columns
    # For both sexes
    List_Columns[6] = '1990_BS'
    List_Columns[7:12] = [i + '_BS' for i in List_Columns[7:12]]
    # For male
    List_Columns[12] = '1990_M'
    List_Columns[13:18] = [i + '_M' for i in List_Columns[13:18]]
    # For female
    List_Columns[18] = '1990_F'
    List_Columns[19:24] = [i + '_F' for i in List_Columns[19:24]]

    # Renaming the columns with the new column list
    UN_Data_Temp.columns = List_Columns
    # Dropping first column and first and second row
    UN_Data_Temp = UN_Data_Temp.drop(columns = ['13'])
    UN_Data_Temp = UN_Data_Temp.drop([0,1])

    UN_Data_Temp = UN_Data_Temp.melt(id_vars = ['Sort\norder',
                                'Major area, region, country or area of destination',
                                'Notes',
                                'Country code',
                                'Type of data (a)'],
                     var_name = "Year_Sex_Var", value_name = Value_List[i-1])
```

Table 4:

```
elif(i==4):
    # Adding suffix to each section of international migrant stock information based on the sex information in
    # heading columns
    # For female
    List_Columns[6] = '1990_F'
    List_Columns[7:12] = [i + '_F' for i in List_Columns[7:12]]

    # Renaming the columns with the new column list
    UN_Data_Temp.columns = List_Columns
    # Dropping first column and first and second row
    UN_Data_Temp = UN_Data_Temp.drop(columns = ['13'])
    UN_Data_Temp = UN_Data_Temp.drop([0,1])

    UN_Data_Temp = UN_Data_Temp.melt(id_vars = ['Sort\norder',
                                                'Major area, region, country or area of destination',
                                                'Notes',
                                                'Country code',
                                                'Type of data (a)'],
                     var_name = "Year_Sex_Var", value_name = Value_List[i-1])
```

Table 5:

```
elif(i==5):
    # Adding suffix to each section of international migrant stock information based on the sex information in
    # heading columns
    # For both sexes
    List_Columns[6] = '1990-2000_BS'
    List_Columns[7:11] = [i + '_BS' for i in List_Columns[7:11]]
    # For male
    List_Columns[11] = '1990-2000_M'
    List_Columns[12:17] = [i + '_M' for i in List_Columns[12:17]]
    # For female
    List_Columns[16] = '1990-2000_F'
    List_Columns[17:23] = [i + '_F' for i in List_Columns[17:23]]

    # Renaming the columns with the new column list
    UN_Data_Temp.columns = List_Columns
    # Dropping first column and first and second row
    UN_Data_Temp = UN_Data_Temp.drop(columns = ['13'])
    UN_Data_Temp = UN_Data_Temp.drop([0,1])

    UN_Data_Temp = UN_Data_Temp.melt(id_vars = ['Sort\norder',
                                                'Major area, region, country or area of destination',
                                                'Notes',
                                                'Country code',
                                                'Type of data (a)'],
                     var_name = "Year_Sex_Var", value_name = Value_List[i-1])
```

As these tables have a common structure, I added a block of code to recode the new ‘Year_Sex_Var’ into a more readable format

```
     # Replacing .. values with NaN
    UN_Data_Temp = UN_Data_Temp.replace('..' , 'NaN')
    # Adding two new columns 'Year' and 'Sex' for the combined variable Year_Sex_Var
    UN_Data_Temp[['Year','Sex']] = UN_Data_Temp.Year_Sex_Var.str.split('_', expand = True)
    # Dropping the combined variable
    UN_Data_Temp = UN_Data_Temp.drop(columns = ['Year_Sex_Var'])
    # Recoding the values of sexes
    UN_Data_Temp['Sex'] = UN_Data_Temp['Sex'].map({'BS': 'Both_Sexes', 'F': 'Female', 'M': 'Male'})
```

For table 6:

For table 6, since it has multiple structures, I decided to take a slightly different approach by splitting the data into three sections based on their KPI.

```
elif(i == 6):
    # Adding suffix to each section of international migrant stock information based on the sex information in the
    # heading columns
    # For both sexes
    List_Columns[6] = '1990_BS'
    List_Columns[7:12] = [i + '_BS' for i in List_Columns[7:12]]
    # For male
    List_Columns[12] = '1990_R'
    List_Columns[13:18] = [i + '_R' for i in List_Columns[13:18]]
    # For female
    List_Columns[18] = '1990-1995_A'
    List_Columns[19:24] = [i + '_A' for i in List_Columns[19:24]]

     # Renaming the columns with the new column list
    UN_Data_Temp.columns = List_Columns
            # Dropping first column and first and second row
    UN_Data_Temp = UN_Data_Temp.drop(columns = ['13'])
    UN_Data_Temp = UN_Data_Temp.drop([0,1])

    UN_Data_Temp = UN_Data_Temp.replace('..' , 'NaN')

    ## Splitting the DF into three DFs

    ### 1
    Data_6_1 = UN_Data_Temp.iloc[:,0:11]


    ## Melt Data
    Data_6_1 = Data_6_1.melt(id_vars = ['Sort\norder',
                                        'Major area, region, country or area of destination',
                                        'Notes',
                                        'Country code',
                                        'Type of data (a)'],
                             var_name = "Year_Sex_Var", value_name = "Estimated_Refugee_Stock_Value_Both_Sexe

    # Adding two new columns 'Year' and 'Sex' for the combined variable Year_Sex_Var
    Data_6_1[['Year','Sex']] = Data_6_1.Year_Sex_Var.str.split('_', expand = True)
    # Dropping the combined variable
    Data_6_1 = Data_6_1.drop(columns = ['Year_Sex_Var'])
    Data_6_1['Sex'] = Data_6_1['Sex'].map({'BS': 'Both_Sexes'})


    ### 2
    Data_6_2 = UN_Data_Temp.iloc[:,np.r_[0:5,11:17]]

    ## Melt Data
    Data_6_2 = Data_6_2.melt(id_vars = ['Sort\norder',
                                        'Major area, region, country or area of destination',
                                        'Notes',
                                        'Country code',
                                        'Type of data (a)'],
                             var_name = "Year_Sex_Var", value_name = "Refugee_Pnct_InternationalStock_Value_(

    # Adding two new columns 'Year' and 'Sex' for the combined variable Year_Sex_Var
    Data_6_2[['Year','Sex']] = Data_6_2.Year_Sex_Var.str.split('_', expand = True)
    # Dropping the combined variable
    Data_6_2 = Data_6_2.drop(columns = ['Year_Sex_Var', 'Sex'])

    #### 3
    Data_6_3 = UN_Data_Temp.iloc[:,np.r_[0:5,17:22]]
    Data_6_3

    ## Melt Data
    Data_6_3 = Data_6_3.melt(id_vars = ['Sort\norder',
                                        'Major area, region, country or area of destination',
                                        'Notes',
                                        'Country code',
                                        'Type of data (a)'],
                             var_name = "Year_Sex_Var", value_name = "Annual_RateChange_Refugee_Value_General

    # Adding two new columns 'Year' and 'Sex' for the combined variable Year_Sex_Var
    Data_6_3[['Year','Sex']] = Data_6_3.Year_Sex_Var.str.split('_', expand = True)
    # Dropping the combined variable
    Data_6_3 = Data_6_3.drop(columns = ['Year_Sex_Var', 'Sex'])

    DF_Table_6 = [Data_6_1, Data_6_2, Data_6_3]
```

The result of the above for loop is a list containing 5 data frames (for tables 1 to 5) and another list containing 3 data frames (for table 6 - split into 3 data frames).

Since the first four data frames follow the same structure for the year column, I combined them using pandas merge (with 'outer') wrapped in a reduce function from functools.

```
## Reading the dfs for the first four dataframes and merging

DF_List_1_4 = DF_List[0:4]

import functools as ft
DF_Final = ft.reduce(lambda left, right: pd.merge(left, right, on=['Sort\norder',
       'Major area, region, country or area of destination', 'Notes',
       'Country code', 'Year', 'Sex'], how = "outer"), DF_List_1_4)
```

I choose to keep 6 of the common variables as key for merging the 4 data frames. As variable 'type of data(a)' is not present in Table 2, I chose not to use it as a key variable and process it separately. Since this variable is only common for the other tables, the output of the above merge showed duplicates of these variables.

After confirming if they were essentially columns with the same values, I decided to drop one of the duplicates from the merged data frame and rename the other variables based on their KPI. So, the merged data frame contained two columns for 'type of data(a)' - one as the general one (denoted by _G) and the other one for table 4: female migrant information (denoted by_FM).

```
## Checking if type of data variable is same
DF_Final['Type of data (a)_x'].equals(DF_Final['Type of data (a)_y'])
```

```
True
```

```
## Dropping column
DF_Final = DF_Final.drop(columns = ['Type of data (a)_x'])
# Renaming
DF_Final = DF_Final.rename(columns = {'Type of data (a)_y' : 'Type of data (a)_G', # For general variables
                                      'Type of data (a)' : 'Type of data (a)_FM'}) # For the female migrant table
# Renaming back to original
DF_Final = DF_Final.rename(columns = {'Type of data (a)_G' : 'Type of data (a)'})

# Listing all column headings
cols = DF_Final.columns.tolist()
```

```
cols
```

```
## Renaming column names to reorder Year and Sex
cols = ['Sort\norder',
 'Major area, region, country or area of destination',
 'Notes',
 'Country code',
 'Year',
 'Sex',
 'Type of data (a)',
 'International_Migrant_Stock_Value',
 'Total_Population_Value',
 'International_Migrant_Stock_Pcnt_Value',
 'Type of data (a)_FM',
 'Female_Migrant_Pnct_Value']
```

```
# Renaming columns
DF_Final = DF_Final[cols]
```

Following this, I renamed the data frame to bring Year and Sex as key columns for easier interpretability.

In the second last phase, I merged the data frames as mentioned above using the key columns as the common variables.

```
## Merging DF with the table 6 data that matches the year headings
Data_Temp_1 = pd.merge(DF_Final, DF_Table_6[0], on = ['Sort\norder',
 'Major area, region, country or area of destination',
 'Notes',
 'Country code',
 'Year',
 'Sex', 'Type of data (a)'], how = "outer")
```

```
## Merging DF with the table 6 data that matches the year headings
Data_Temp_New_1 = pd.merge(Data_Temp_1, DF_Table_6[1], on = ['Sort\norder',
 'Major area, region, country or area of destination',
 'Notes',
 'Country code',
 'Year', 'Type of data (a)'], how = "outer")
```

```
## Merging DF with the table 6 data that matches the year headings based on year bins
Data_Temp_2 = pd.merge(DF_List[4], DF_Table_6[2], on = ['Sort\norder',
 'Major area, region, country or area of destination',
 'Notes',
 'Country code',
 'Year', 'Type of data (a)'])
```

Finally, I added information from the ‘Annex’ sheet to the two newly cleaned data frames.

```
## Adding Annex info
Annex_Data = pd.read_excel("UN_MigrantStockTotal_2015.xlsx", sheet_name = 'ANNEX')
Annex_Data.head(15)

## Subsetting the data
Annex_Data = Annex_Data.iloc[13:,].reset_index()
Annex_Data.head(15)

# Pushing first row as column headers
Annex_Data.columns = Annex_Data.iloc[0]
Annex_Data = Annex_Data.drop([0])
Annex_Data = Annex_Data.drop(columns = [13])
# mapping = {Annex_Data.columns[2]: 'Sort_Order_Country'}
Annex_Data = Annex_Data.rename(columns = {Annex_Data.columns[2]: 'Sort_Order_Country'})
```

```
## First frame
# Merging with annex
Data_Final_1 = pd.merge(Annex_Data, Data_Temp_New_1,
                        on = ['Country code'])
```

```
## Second frame
# Merging with annex
Data_Final_2 = pd.merge(Annex_Data, Data_Temp_2,
                        on = ['Country code'])
```

The final output is two cleaned data frames - Data_Final_1 and Data_Final_2.

Data_Final_1 -

```
Data_Final_1.head(15)
```

	Country code	Country or area	Sort_Order_Country	Major area	Code	Sort_Order_Country	Region	Code	Sort_Order_Country	Developed region	...	Year	Sex	Type of data (a)	Ir
0	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	1990	Both_Sexes	B	
1	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	1990	Male	B	
2	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	1990	Female	B	
3	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	1995	Both_Sexes	B	
4	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	1995	Male	B	
5	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	1995	Female	B	
6	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	2000	Both_Sexes	B	
7	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	2000	Male	B	
8	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	...	2000	Female	B	

/pe of ata (a)	International_Migrant_Stock_Value	Total_Population_Value	International_Migrant_Stock_Pcnt_Value	Type of data (a)_FM	Female_Migrant_Pnct_Value	Estimated_Refugee_Stocl
B	57686	12067.57	0.478025	NaN	NaN	
B	32558	6179.834	0.526843	NaN	NaN	
B	25128	5887.736	0.426785	B	43.559963	
B	71522	16772.522	0.426424	NaN	NaN	
B	39105	8682.442	0.450392	NaN	NaN	
B	32417	8090.08	0.400701	B	45.324516	
B	75917	19701.94	0.385328	NaN	NaN	
B	42848	10146.537	0.422292	NaN	NaN	
B	33069	9555.403	0.346076	B	43.559414	
B	87300	24399.948	0.357788	NaN	NaN	
B	49274	12616.326	0.390557	NaN	NaN	
B	38026	11783.622	0.322702	B	43.557847	
B	102246.0	27962.207	0.365658	NaN	NaN	
B	57709.0	14367.633	0.40166	NaN	NaN	
B	44537.0	13594.574	0.327609	B	43.558672	

national_Migrant_Stock_Pcnt_Value	Type of data (a)_FM	Female_Migrant_Pnct_Value	Estimated_Refugee_Stock_Value_Both_Sexes	Refugee_Pnct_InternationalStock_Value_General
0.478025	NaN	NaN	25	0.043338
0.526843	NaN	NaN	NaN	0.043338
0.426785	B	43.559963	NaN	0.043338
0.426424	NaN	NaN	19605	27.411146
0.450392	NaN	NaN	NaN	27.411146
0.400701	B	45.324516	NaN	27.411146
0.385328	NaN	NaN	0	0
0.422292	NaN	NaN	NaN	0
0.346076	B	43.559414	NaN	0
0.357788	NaN	NaN	32	0.036655
0.390557	NaN	NaN	NaN	0.036655
0.322702	B	43.557847	NaN	0.036655
0.365658	NaN	NaN	6434.0	6.292667
0.40166	NaN	NaN	NaN	6.292667
0.327609	B	43.558672	NaN	6.292667

Data_Final_2 -

```
Data_Final_2.head(15)
```

	Country code	Country or area	Sort_Order_Country	Major area	Code	Sort_Order_Country	Region	Code	Sort_Order_Country	Developed region	Least developed country	Sub-Saharan Africa	Sort\nor
0	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
1	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
2	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
3	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
4	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
5	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
6	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
7	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
8	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
9	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
10	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
11	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No	
12	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	No	No	
13	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	No	No	
14	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	No	No	

```
Data_Final_2.head(15)
```

ed on	Least developed country	Sub-Saharan Africa	Sort\norder	Major area, region, country or area of destination	Notes	Type of data (a)	Annual_Rate_Change_Migrant_Value	Year	Sex	Annual_RateChange_Refugee_Value_General
No	Yes	No	99	Afghanistan	NaN	B	1.192711	1995-2000	Both_Sexes	NaN
No	Yes	No	99	Afghanistan	NaN	B	1.828173	1995-2000	Male	NaN
No	Yes	No	99	Afghanistan	NaN	B	0.398266	1995-2000	Female	NaN
No	Yes	No	99	Afghanistan	NaN	B	2.794196	2000-2005	Both_Sexes	NaN
No	Yes	No	99	Afghanistan	NaN	B	2.794752	2000-2005	Male	NaN
No	Yes	No	99	Afghanistan	NaN	B	2.793477	2000-2005	Female	NaN
No	Yes	No	99	Afghanistan	NaN	B	3.160624	2005-2010	Both_Sexes	102.911692
No	Yes	No	99	Afghanistan	NaN	B	3.160332	2005-2010	Male	102.911692
No	Yes	No	99	Afghanistan	NaN	B	3.161003	2005-2010	Female	102.911692
No	Yes	No	99	Afghanistan	NaN	B	26.37988	2010-2015	Both_Sexes	50.501739
No	Yes	No	99	Afghanistan	NaN	B	24.191602	2010-2015	Male	50.501739
No	Yes	No	99	Afghanistan	NaN	B	28.900067	2010-2015	Female	50.501739
es	No	No	154	Albania	NaN	B	1.443662	1995-2000	Both_Sexes	-45.253018
es	No	No	154	Albania	NaN	B	1.563286	1995-2000	Male	-45.253018
es	No	No	154	Albania	NaN	B	1.338487	1995-2000	Female	-45.253018

4. Tidy Data Principles

Principle 1: Each cell must be unique
This principle is applied across the data set

Principle 2: Each column needs to consist of one and only one variable
This principle is applied in the first phase of the for loop for the ‘year’ and ‘sex’ information

Principle 3: Variables need to be in cells, not rows and columns
This principle is also applied in the initial phases, where I pivot the columns ‘year_sex_var’ into row format, in order to split them up and use them as separate columns.

Principle 4: Each table column needs to have a singular data type
The principle is followed while merging the tables based on the ‘year’ information, to maintain structure across the 2 final data frames.

Principle 5: Single observational unit must be in 1 table

This principle is applied in the last phases of the data wrangling process when I finally merge all the tables based on their ‘year’ values and add information from the ‘Annex’ sheet.

5. Discussion and Suggestions

My takeaway from this project was learning about how important it is to clean data efficiently, in order to do the analysis effectively. With the help of tidy data principles, I brought information from essentially 7 tables (6 tables + Annex) and restructured them into 2 data frames, ready to use for analysis. This helps overall with the analysis as the cleaned data frame is standardized in a certain manner that is machine-readable and ready for further study. In this way, the final data frame contains clear values for each column, row, and cell.

One of my suggestions would be to look further into trying to make the column ‘Type of data(a)_FM’ as a part of one main column ‘Type of data(a)’. I think it might be better to reduce the redundancy of that column in the long run but since the data is so vastly different between the two columns - I decided to keep it as it is till I understand more about data wrangling.

Additionally, I would like to know more about the second and third KPI in table 6, iit contains no information regarding ‘sex’ so I would like to understand how to efficiently categorize such data.

6. Conclusion

I believe that this exercise has made me understand the importance of this phase of data wrangling more efficiently. Before heading directly into analysis, cleaning and restructuring your data is key. If I have to compare working with the original file with 7 tables vs working with 2 cleaned data frames, analyzing the latter would be so much more easier and efficient, keeping in mind machine-readability and interpretability. The final output of this project is 2 data frames, ready for further study and analysis.