Table of contents

Cleaning of table 1 2

Cleaning of table 2 7

Cleaning of table 3 12

Cleaning of table 4 16

Cleaning of table 5 19

Cleaning of table 6 28

Cleaning of Annex 34

Cleaning of table 1

Step 1: Importing essential libraries

```
import numpy as np
import pandas as pd
```

These libraries help to build the data frame that support interchange protocol. It has also been imported to get high performance data structure and data analysis tools.

Step 2: Importing and reading the datasheet as a dataframe

```
[2] df1 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 1',skiprows = 14)
```

```
df1.head(5)
```

	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	Unnamed: 16	International migrant stock at mid-year (female)	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnam
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	2000	2005	2010	2015	1990	1995	2000	2005	2
1	1.0	WORLD	NaN	900.0	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	126115435	74815702	79064275	84818470	93402426	107100
2	2.0	Developed regions	(b)	901.0	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	67618619	42115231	47214055	52838567	59963332	68479
3	3.0	Developing regions	(c)	902.0	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	58496816	32700471	31850220	31979903	33439094	38621
4	4.0	Least developed countries	(d)	941.0	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	6463217	5236216	5573685	4721920	4432371	4560

5 rows × 23 columns

According to the tidy data principle, data cleaning requires importing of the dataset as a dataframe. In this regard, the 14 rows have been skipped as they contain headings of the dataset which are not included in the data. After importing the data, df1.head(5) function is used to read the data.

Step 3: Checking the information of the dataframe

```
[4] df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 23 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort
order                                                 265 non-null    float64
 1   Major area, region, country or area of destination    265 non-null    object
 2   Notes                                                  26 non-null     object
 3   Country code                                           265 non-null    float64
 4   Type of data (a)                                       232 non-null    object
 5   International migrant stock at mid-year (both sexes)   266 non-null    object
 6   Unnamed: 6                                             266 non-null    object
 7   Unnamed: 7                                             266 non-null    object
 8   Unnamed: 8                                             266 non-null    object
 9   Unnamed: 9                                             266 non-null    int64
 10  Unnamed: 10                                            266 non-null    int64
 11  International migrant stock at mid-year (male)         266 non-null    object
 12  Unnamed: 12                                            266 non-null    object
 13  Unnamed: 13                                            266 non-null    object
 14  Unnamed: 14                                            266 non-null    object
 15  Unnamed: 15                                            266 non-null    int64
 16  Unnamed: 16                                            266 non-null    int64
 17  International migrant stock at mid-year (female)       266 non-null    object
 18  Unnamed: 18                                            266 non-null    object
 19  Unnamed: 19                                            266 non-null    object
 20  Unnamed: 20                                            266 non-null    object
 21  Unnamed: 21                                            266 non-null    int64
 22  Unnamed: 22                                            266 non-null    int64
dtypes: float64(2), int64(6), object(15)
memory usage: 47.9+ KB
```

df1.info() function is used to check the information of the data type, non-null count and columns.

Step 4: Renaming the columns' name

```
[5] df1.rename(columns = {'International migrant stock at mid-year (both sexes)':'International migrant stock at mid-year (both sexes)(1990)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 6':'International migrant stock at mid-year (both sexes)(1995)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 7':'International migrant stock at mid-year (both sexes)(2000)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 8':'International migrant stock at mid-year (both sexes)(2005)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 9':'International migrant stock at mid-year (both sexes)(2010)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 10':'International migrant stock at mid-year (both sexes)(2015)'}, inplace = True)


    df1.rename(columns = {'International migrant stock at mid-year (male)':'International migrant stock at mid-year (male)(1990)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 12':'International migrant stock at mid-year (male)(1995)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 13':'International migrant stock at mid-year (male)(2000)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 14':'International migrant stock at mid-year (male)(2005)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 15':'International migrant stock at mid-year (male)(2010)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 16':'International migrant stock at mid-year (male)(2015)'}, inplace = True)


    df1.rename(columns = {'International migrant stock at mid-year (female)':'International migrant stock at mid-year (female)(1990)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 18':'International migrant stock at mid-year (female)(1995)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 19':'International migrant stock at mid-year (female)(2000)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 20':'International migrant stock at mid-year (female)(2005)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 21':'International migrant stock at mid-year (female)(2010)'}, inplace = True)
    df1.rename(columns = {'Unnamed: 22':'International migrant stock at mid-year (female)(2015)'}, inplace = True)
```

The names of some columns were unnamed 6, 7, 8 and etc. that do not signify the column appropriately due to which, it has been renamed with the help of giving an appropriate name to the columns.

Step 5: Reading the dataframe to check changed columns' name

```
[6] df1.head(5)
```

	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) (1990)	International migrant stock at mid-year (both sexes) (1995)	International migrant stock at mid-year (both sexes) (2000)	International migrant stock at mid-year (both sexes) (2005)	International migrant stock at mid-year (both sexes) (2010)	...	International migrant stock at mid-year (male)(2000)	International migrant stock at mid-year (male)(2005)	International migrant stock at mid-year (male)(2010)	International migrant stock at mid-year (male)(2015)	International migrant stock at mid-year (female) (1990)	In mi
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	2000	2005	2010	2015	1990	
1	1.0	WORLD	NaN	900.0	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	126115435	74815702	
2	2.0	Developed regions	(b)	901.0	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	67618619	42115231	
3	3.0	Developing regions	(c)	902.0	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	58496816	32700471	
4	4.0	Least developed countries	(d)	941.0	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	6463217	5236216	

5 rows × 23 columns

After renaming the columns' name, the information of the data frame has been checked.

Step 6: Creating new dataframe by dropping first row

```
[7] df2 = df1.drop(0)

[8] df2.head(5)
```

	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) (1990)	International migrant stock at mid-year (both sexes) (1995)	International migrant stock at mid-year (both sexes) (2000)	International migrant stock at mid-year (both sexes) (2005)	International migrant stock at mid-year (both sexes) (2010)	...	International migrant stock at mid-year (male)(2000)	International migrant stock at mid-year (male)(2005)	International migrant stock at mid-year (male)(2010)	International migrant stock at mid-year (male)(2015)	International migrant stock at mid-year (female) (1990)	In mi
1	1.0	WORLD	NaN	900.0	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	126115435	74815702	
2	2.0	Developed regions	(b)	901.0	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	67618619	42115231	
3	3.0	Developing regions	(c)	902.0	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	58496816	32700471	
4	4.0	Least developed countries	(d)	941.0	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	6463217	5236216	
		Less developed															

It is checked that the renamed columns itself contain the details of year, due to which the first row after the heading was not required. Hence, it is removed to get accuracy as some null values were also there. The df2.head(5) function is also used to check the data frame structure.

Step 7: Removing null values

```
[9] newdf2 = df2.dropna()
    newdf2
```

	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) (1990)	International migrant stock at mid-year (both sexes) (1995)	International migrant stock at mid-year (both sexes) (2000)	International migrant stock at mid-year (both sexes) (2005)	International migrant stock at mid-year (both sexes) (2010)	...	International migrant stock at mid-year (male)(2000)	International migrant stock at mid-year (male)(2005)	International migrant stock at mid-year (male)(2010)	International migrant stock at mid-year (male)(2015)	Internatic migrant st at mid-y (fema (19
17	17.0	Mauritius	(1)	480.0	C	3613	7493	15543	19647	24836	...	5705	8943	13188	15832	1
26	26.0	United Republic of Tanzania	(2)	834.0	B R	574025	1106043	928180	770846	308600	...	470962	486774	153984	130404	290
44	44.0	Sudan	(3)	729.0	B R	1402896	1053396	801883	541994	578363	...	403048	275915	294782	254530	706
67	67.0	Saint Helena	(4)	654.0	B	383	394	405	487	569	...	215	263	312	335	
79	79.0	China	(5)	156.0	C	376361	442198	508034	678947	849861	...	254082	379920	505758	600136	184
80	80.0	China, Hong Kong Special Administrative Region	(6)	344.0	B	2218473	2443798	2669122	2721235	2779950	...	1225629	1185121	1147539	1119957	1093
81	81.0	China, Macao Special Administrative Region	(7)	446.0	B	205047	224929	240791	279308	318506	...	109391	130387	146407	155692	108
91	91.0	Malaysia	(8)	458.0	C R	695920	937368	1277223	1722344	2406011	...	712912	995607	1437206	1529630	290
110	110.0	Azerbaijan	(9)	31.0	B R	360600	344070	327540	302220	276901	...	141161	135336	129512	126600	202
112	112.0	Cyprus	(10)	196.0	B	43805	61941	80076	117165	187923	...	34745	50421	82284	86980	23

This step consists of dropping all the null values from the dataframe that can improve the accuracy of the result.

Step8: ".." in the datasheet has been removed with 0 using regex function

```
[10] newdf2 = newdf2.replace('\..','0',regex=True)

[11] newdf2
```

	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) (1990)	International migrant stock at mid-year (both sexes) (1995)	International migrant stock at mid-year (both sexes) (2000)	International migrant stock at mid-year (both sexes) (2005)	International migrant stock at mid-year (both sexes) (2010)	...	International migrant stock at mid-year (male)(2000)	International migrant stock at mid-year (male)(2005)	International migrant stock at mid-year (male)(2010)	International migrant stock at mid-year (male)(2015)	Internationa migrant stoc at mid-yea (female (1990
17	17.0	Mauritius	(1)	480.0	C	3613	7493	15543	19647	24836	...	5705	8943	13188	15832	185
26	26.0	United Republic of Tanzania	(2)	834.0	B R	574025	1106043	928180	770846	308600	...	470962	486774	153984	130404	29063
44	44.0	Sudan	(3)	729.0	B R	1402896	1053396	801883	541994	578363	...	403048	275915	294782	254530	70651
67	67.0	Saint Helena	(4)	654.0	B	383	394	405	487	569	...	215	263	312	335	16
79	79.0	China	(5)	156.0	C	376361	442198	508034	678947	849861	...	254082	379920	505758	600136	18440

There are some fields in the dataframe that contain ".." values which do not signify anything and are not required for the data analytical part. Due to this reason, it is replaced with 0 value with the help of using regex function followed by reading the dataframe.

Step 9: Convert categorical variable into dummy/indicator variables

```
[12] table1 = pd.get_dummies(newdf2)

[13] table1
```

	Sort\norder	Country code	International migrant stock at mid-year (both sexes) (2010)	International migrant stock at mid-year (both sexes) (2015)	International migrant stock at mid-year (male)(2010)	International migrant stock at mid-year (male)(2015)	International migrant stock at mid-year (female) (2010)	International migrant stock at mid-year (female) (2015)	Major area, region, country or area of destination_Australia	Major area, region, country or area of destination_Azerbaijan	...	International migrant stock at mid-year (female) (2005)_166884	International migrant stock at mid-year (female) (2005)_184700	Internati migrant s at mid- (fem (2005)_26
17	17.0	480.0	24836	28585	13188	15832	11648	12753	0	0	...	0	0	
26	26.0	834.0	308600	261222	153984	130404	154616	130818	0	0	...	0	0	
44	44.0	729.0	578363	503477	294782	254530	283581	248947	0	0	...	0	0	
67	67.0	654.0	569	604	312	335	257	269	0	0	...	0	0	
79	79.0	156.0	849861	978046	505758	600136	344103	377910	0	0	...	0	0	
80	80.0	344.0	2779950	2838665	1147539	1119957	1632411	1718708	0	0	...	0	0	
81	81.0	446.0	318506	342703	146407	155692	172099	187011	0	0	...	0	0	
91	91.0	458.0	2406011	2514243	1437206	1529630	968805	984613	0	0	...	0	0	
110	110.0	31.0	276901	264241	129512	126600	147389	137641	0	1	...	1	0	

Get_dummies is used to convert categorical variables into dummy/indicator variables.

Step 10: Creating a series using pd.series

```
[14] ser = pd.Series(newdf2['Country code']).head(10)
```

```
pd.to_numeric(ser, downcast ='signed')
```

```
17     480
26     834
44     729
67     654
79     156
80     344
81     446
91     458
110     31
112    196
Name: Country code, dtype: int16
```

Pd.series is used to create a series of country code up to 10 in the list and reading the data. Pd.to_numeric is used to convert arguments to numeric types so that the data give accurate results.

Step 11: Hashing the value of a column

```
[16] hash(newdf2['International migrant stock at mid-year (female)(2015)'].values.tobytes())
```

```
2463136387318800110
```

```
[17] newdf3 = newdf2.iloc[:, 0:6]
     newdf3
```

	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)(1990)
17	17.0	Mauritius	(1)	480.0	C	3613
26	26.0	United Republic of Tanzania	(2)	834.0	B R	574025
44	44.0	Sudan	(3)	729.0	B R	1402896
67	67.0	Saint Helena	(4)	654.0	B	383
79	79.0	China	(5)	156.0	C	376361
80	80.0	China, Hong Kong Special Administrative Region	(6)	344.0	B	2218473

Hash function is used to return the hash value of the chosen column that returns an array of deterministic integers. The iloc function is used to select specific columns from the dataset.

Step 12: Renaming column name

```
newdf3.rename(columns = {'Sort\norder':'sort_order'}, inplace = True)
newdf3.rename(columns = {'Country code':'Country_code'}, inplace = True)
newdf3.rename(columns = {'International migrant stock at mid-year (both sexes)(1990)':'International_migrant_stock_at_mid_year_(both_sexes)_(1990)'}, inplace = True)
```

```
[19] newdf3.head(2)
```

	sort_order	Major area, region, country or area of destination	Notes	Country_code	Type of data (a)	International_migrant_stock_at_mid_year_(both_sexes)_(1990)
17	17.0	Mauritius	(1)	480.0	C	3613
26	26.0	United Republic of Tanzania	(2)	834.0	B R	574025

Columns' names are renamed to signify appropriate meaning.

Step 13: .eval function usage

```
[20] print(newdf3.eval("sort_order + Country_code"))

17     497.0
26     860.0
44     773.0
67     721.0
79     235.0
80     424.0
81     527.0
91     549.0
110    141.0
112    308.0
113    381.0
122    397.0
134    632.0
140    970.0
144    390.0
150    728.0
160    496.0
166    854.0
168    892.0
```

Newdf3.eval function is used to evaluate the python expression as a string using various backbends.

Cleaning of table 2

Step 1: Importing the datasheet table 2

```
[73] df21 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 2',skiprows = 14)
```

```
df21.head(5)
```

	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	Unnamed: 14	Unnamed: 15	Total female population at mid-year (thousands)	Unnamed: 17	Unna
0	NaN	NaN	NaN	NaN	1990.000	1995.000	2000.000	2005.000	2010.000	2015.000	...	2000	2005	2010	2015	1990	1995	
1	1.0	WORLD	NaN	900.0	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	3493956.904	3707205.753	2639243.998	2848487.191	304208
2	2.0	Developed regions	(b)	901.0	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	599955.476	609297.148	589207.436	601492.755	61080
3	3.0	Developing regions	(c)	902.0	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	2894001.428	3097908.605	2050036.562	2246994.436	243128
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	422397.532	476031.179	256015.073	293162.612	33290

5 rows × 22 columns

Import the datasheet as a dataframe df21 by skipping 14 rows of the above part that contains the heading of the sheet and reading the data frame.

Step 2: Getting the information of the dataframe

```
df21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 22 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   Sort
order                                                 265 non-null    float64
 1   Major area, region, country or area of destination   265 non-null    object
 2   Notes                                                 26 non-null     object
 3   Country code                                          265 non-null    float64
 4   Total population of both sexes at mid-year (thousands) 266 non-null   float64
 5   Unnamed: 5                                            266 non-null    float64
 6   Unnamed: 6                                            266 non-null    float64
 7   Unnamed: 7                                            266 non-null    float64
 8   Unnamed: 8                                            266 non-null    float64
 9   Unnamed: 9                                            266 non-null    float64
 10  Total male population at mid-year (thousands)         266 non-null    object
 11  Unnamed: 11                                           266 non-null    object
 12  Unnamed: 12                                           266 non-null    object
 13  Unnamed: 13                                           266 non-null    object
 14  Unnamed: 14                                           266 non-null    object
 15  Unnamed: 15                                           266 non-null    object
 16  Total female population at mid-year (thousands)       266 non-null    object
 17  Unnamed: 17                                           266 non-null    object
 18  Unnamed: 18                                           266 non-null    object
 19  Unnamed: 19                                           266 non-null    object
 20  Unnamed: 20                                           266 non-null    object
 21  Unnamed: 21                                           266 non-null    object
dtypes: float64(8), object(14)
memory usage: 45.8+ KB
```

The detailed information of the data frame has been identified using df21.info() function.

Step 3: Renaming all the columns

```
df21.rename(columns = {'Total population of both sexes at mid-year (thousands)':'Total population of both sexes at mid-year (thousands)(1990)'}, inplace = True)
df21.rename(columns = {'Unnamed: 5':'Total population of both sexes at mid-year (thousands)(1995)'}, inplace = True)
df21.rename(columns = {'Unnamed: 6':'Total population of both sexes at mid-year (thousands)(2000)'}, inplace = True)
df21.rename(columns = {'Unnamed: 7':'Total population of both sexes at mid-year (thousands)(2005)'}, inplace = True)
df21.rename(columns = {'Unnamed: 8':'Total population of both sexes at mid-year (thousands)(2010)'}, inplace = True)
df21.rename(columns = {'Unnamed: 9':'Total population of both sexes at mid-year (thousands)(2015)'}, inplace = True)


df21.rename(columns = {'Total male population at mid-year (thousands)':'Total male population at mid-year (thousands)(1990)'}, inplace = True)
df21.rename(columns = {'Unnamed: 11':'Total male population at mid-year (thousands)(1995)'}, inplace = True)
df21.rename(columns = {'Unnamed: 12':'Total male population at mid-year (thousands)(2000)'}, inplace = True)
df21.rename(columns = {'Unnamed: 13':'Total male population at mid-year (thousands)(2005)'}, inplace = True)
df21.rename(columns = {'Unnamed: 14':'Total male population at mid-year (thousands)(2010)'}, inplace = True)
df21.rename(columns = {'Unnamed: 15':'Total male population at mid-year (thousands)(2015)'}, inplace = True)


df21.rename(columns = {'Total female population at mid-year (thousands)':'Total female population at mid-year (thousands)(1990)'}, inplace = True)
df21.rename(columns = {'Unnamed: 17':'Total female population at mid-year (thousands)(1995)'}, inplace = True)
df21.rename(columns = {'Unnamed: 18':'Total female population at mid-year (thousands)(2000)'}, inplace = True)
df21.rename(columns = {'Unnamed: 19':'Total female population at mid-year (thousands)(2005)'}, inplace = True)
df21.rename(columns = {'Unnamed: 20':'Total female population at mid-year (thousands)(2010)'}, inplace = True)
df21.rename(columns = {'Unnamed: 21':'Total female population at mid-year (thousands)(2015)'}, inplace = True)


df21.rename(columns = {'Sort\norder':'Sort_order'}, inplace = True)
```

The renaming of columns has been done to get accurate insights of the column's name.

Step 4: Checking the data frame information

```
df21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort_order                                              265 non-null    float64
 1   Major area, region, country or area of destination      265 non-null    object
 2   Notes                                                   26 non-null     object
 3   Country code                                            265 non-null    float64
 4   Total population of both sexes at mid-year (thousands)(1990)  266 non-null    float64
 5   Total population of both sexes at mid-year (thousands)(1995)  266 non-null    float64
 6   Total population of both sexes at mid-year (thousands)(2000)  266 non-null    float64
 7   Total population of both sexes at mid-year (thousands)(2005)  266 non-null    float64
 8   Total population of both sexes at mid-year (thousands)(2010)  266 non-null    float64
 9   Total population of both sexes at mid-year (thousands)(2015)  266 non-null    float64
 10  Total male population at mid-year (thousands)(1990)     266 non-null    object
 11  Total male population at mid-year (thousands)(1995)     266 non-null    object
 12  Total male population at mid-year (thousands)(2000)     266 non-null    object
 13  Total male population at mid-year (thousands)(2005)     266 non-null    object
 14  Total male population at mid-year (thousands)(2010)     266 non-null    object
 15  Total male population at mid-year (thousands)(2015)     266 non-null    object
 16  Total female population at mid-year (thousands)(1990)   266 non-null    object
 17  Total female population at mid-year (thousands)(1995)   266 non-null    object
 18  Total female population at mid-year (thousands)(2000)   266 non-null    object
 19  Total female population at mid-year (thousands)(2005)   266 non-null    object
 20  Total female population at mid-year (thousands)(2010)   266 non-null    object
 21  Total female population at mid-year (thousands)(2015)   266 non-null    object
dtypes: float64(8), object(14)
memory usage: 45.8+ KB
```

The data frame information has been obtained after the renaming of the data frame columns to check its functionality.

Step 5: Reading the dataframe

```
df21.head(5)
```

	Sort_order	Major area, region, country or area of destination	Notes	Country code	Total population of both sexes at mid-year (thousands) (1990)	Total population of both sexes at mid-year (thousands) (1995)	Total population of both sexes at mid-year (thousands) (2000)	Total population of both sexes at mid-year (thousands) (2005)	Total population of both sexes at mid-year (thousands) (2010)	Total population of both sexes at mid-year (thousands) (2015)	...	Total male population at mid-year (thousands) (2000)	Total male population at mid-year (thousands) (2005)	Total male population at mid-year (thousands) (2010)	Total male population at mid-year (thousands) (2015)	Total female population at mid-year (thousands) (1990)	Total female population at mid-year (thousands) (1995)	po at (th
0	NaN	NaN	NaN	NaN	1990.000	1995.000	2000.000	2005.000	2010.000	2015.000	...	2000	2005	2010	2015	1990	1995	
1	1.0	WORLD	NaN	900.0	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	3493956.904	3707205.753	2639243.998	2848487.191	304
2	2.0	Developed regions	(b)	901.0	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	599955.476	609297.148	589207.436	601492.755	61
3	3.0	Developing regions	(c)	902.0	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	2894001.428	3097908.605	2050036.562	2246994.436	243
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	422397.532	476031.179	256015.073	293162.612	33

5 rows × 22 columns

The data is read to visualize it appropriately.

Step 6: Dropping extra row and reading the data frame

```
df22 = df21.drop(0)
```

```
[80] df22.head(5)
```

	Sort_order	Major area, region, country or area of destination	Notes	Country code	Total population of both sexes at mid-year (thousands) (1990)	Total population of both sexes at mid-year (thousands) (1995)	Total population of both sexes at mid-year (thousands) (2000)	Total population of both sexes at mid-year (thousands) (2005)	Total population of both sexes at mid-year (thousands) (2010)	Total population of both sexes at mid-year (thousands) (2015)	...	Total male population at mid-year (thousands) (2000)	Total male population at mid-year (thousands) (2005)	Total male population at mid-year (thousands) (2010)	Total male population at mid-year (thousands) (2015)	Total female population at mid-year (thousands) (1990)	Total female population at mid-year (thousands) (1995)	po at (th
1	1.0	WORLD	NaN	900.0	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	3493956.904	3707205.753	2639243.998	2848487.191	304
2	2.0	Developed regions	(b)	901.0	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	599955.476	609297.148	589207.436	601492.755	61
3	3.0	Developing regions	(c)	902.0	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	2894001.428	3097908.605	2050036.562	2246994.436	243
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	422397.532	476031.179	256015.073	293162.612	33
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	2471603.896	2621877.426	1794021.489	1953831.824	209

5 rows × 22 columns

Extra row of year is dropped from the dataframe as it has no utilization.

Step 7: Checking null values in the dataframe

```
df22.isna()
```

	Sort_order	Major area, region, country or area of destination	Notes	Country code	Total population of both sexes at mid-year (thousands) (1990)	Total population of both sexes at mid-year (thousands) (1995)	Total population of both sexes at mid-year (thousands) (2000)	Total population of both sexes at mid-year (thousands) (2005)	Total population of both sexes at mid-year (thousands) (2010)	Total population of both sexes at mid-year (thousands) (2015)	...	Total male population at mid-year (thousands) (2000)	Total male population at mid-year (thousands) (2005)	Total male population at mid-year (thousands) (2010)	Total male population at mid-year (thousands) (2015)	Total female population at mid-year (thousands) (1990)	Total female population at mid-year (thousands) (1995)	at (t
1	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
5	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
261	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
262	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
263	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
264	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	
265	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	

265 rows × 22 columns

.isna() function is used to detect missing values of the dataframe.

Step 8: Dropping Null values from the dataframe and reading it

```
df23 = df22.dropna()
df23
```

17	17.0	Mauritius	(1)	480.0	1055.865	1128.676	1185.143	1222.006	1247.951	1273.212	...	587.689	605.618	618.578	628.841	529.321	566.024
26	26.0	United Republic of Tanzania	(2)	834.0	25458.208	29903.329	33991.590	39065.600	45648.525	53470.420	...	16910.902	19394.517	22665.982	26574.231	12849.925	15068.098
44	44.0	Sudan	(3)	729.0	20008.804	24691.970	28079.664	31990.003	36114.885	40234.882	...	14093.864	16050.47	18119.951	20197.019	9962.092	12294.025
67	67.0	Saint Helena	(4)	654.0	5.531	5.287	5.113	4.274	4.173	3.961	...	..	..	..	..	..	..
79	79.0	China	(5)	156.0	1154605.773	1227841.281	1269974.572	1305600.630	1340968.737	1376048.943	...	652374.453	671409.793	690256.33	708977.116	562099.374	597280.78
80	80.0	China, Hong Kong Special Administrative Region	(6)	344.0	5794.034	6144.498	6783.502	6842.465	6993.570	7287.983	...	3304.376	3270.159	3280.775	3421.641	2831.572	3070.301
81	81.0	China, Macao Special Administrative Region	(7)	446.0	359.735	398.459	431.907	468.149	534.626	587.606	...	208.612	224.338	256.532	283.011	184.676	205.453
91	91.0	Malaysia	(8)	458.0	18211.097	20725.374	23420.751	25796.124	28119.500	30331.007	...	11909.394	12995.419	13956.403	15025.676	8966.626	10195.405
110	110.0	Azerbaijan	(9)	31.0	7216.503	7770.806	8117.742	8563.398	9099.893	9753.968	...	3966.493	4218.381	4510.161	4855.546	3681.397	3967.359
112	112.0	Cyprus	(10)	196.0	766.611	855.389	943.287	1032.586	1103.685	1165.300	...	475.425	522.198	563.329	594.492	377.982	423.596
113	113.0	Georgia	(11)	268.0	5460.309	5067.143	4743.591	4475.273	4250.132	3999.812	...	2244.958	2110.993	2017.821	1908.151	2865.126	2661.038
122	122.0	State of Palestine	(12)	275.0	2101.156	2618.272	3223.781	3579.462	4068.780	4668.466	...	1634.363	1815.186	2063.007	2366.75	1037.806	1291.77
134	134.0	Republic of Moldova	(13)	498.0	4364.114	4339.082	4201.088	4157.707	4084.483	4068.897	...	2010.77	1991.248	1963.724	1955.859	2284.052	2266.578

.dropna() is used to drop null values of the data frame followed by reading the data frame.

Step 9: Replacing ".." with 0 in the dataframe and read

```
df24 = df23.replace('\..','0',regex=True)
```

```
[84] df24
```

17	17.0	Mauritius	(1)	480.0	1055.865	1128.676	1185.143	1222.006	1247.951	1273.212	...	587.689	605.618	618.578	628.841	529.321	566.024
26	26.0	United Republic of Tanzania	(2)	834.0	25458.208	29903.329	33991.590	39065.600	45648.525	53470.420	...	16910.902	19394.517	22665.982	26574.231	12849.925	15068.098
44	44.0	Sudan	(3)	729.0	20008.804	24691.970	28079.664	31990.003	36114.885	40234.882	...	14093.864	16050.47	18119.951	20197.019	9962.092	12294.025
67	67.0	Saint Helena	(4)	654.0	5.531	5.287	5.113	4.274	4.173	3.961	...	0	0	0	0	0	0
79	79.0	China	(5)	156.0	1154605.773	1227841.281	1269974.572	1305600.630	1340968.737	1376048.943	...	652374.453	671409.793	690256.33	708977.116	562099.374	597280.78
80	80.0	China, Hong Kong Special Administrative Region	(6)	344.0	5794.034	6144.498	6783.502	6842.465	6993.570	7287.983	...	3304.376	3270.159	3280.775	3421.641	2831.572	3070.301
81	81.0	China, Macao Special Administrative Region	(7)	446.0	359.735	398.459	431.907	468.149	534.626	587.606	...	208.612	224.338	256.532	283.011	184.676	205.453
91	91.0	Malaysia	(8)	458.0	18211.097	20725.374	23420.751	25796.124	28119.500	30331.007	...	11909.394	12995.419	13956.403	15025.676	8966.626	10195.405
110	110.0	Azerbaijan	(9)	31.0	7216.503	7770.806	8117.742	8563.398	9099.893	9753.968	...	3966.493	4218.381	4510.161	4855.546	3681.397	3967.359
112	112.0	Cyprus	(10)	196.0	766.611	855.389	943.287	1032.586	1103.685	1165.300	...	475.425	522.198	563.329	594.492	377.982	423.596
113	113.0	Georgia	(11)	268.0	5460.309	5067.143	4743.591	4475.273	4250.132	3999.812	...	2244.958	2110.993	2017.821	1908.151	2865.126	2661.038
122	122.0	State of Palestine	(12)	275.0	2101.156	2618.272	3223.781	3579.462	4068.780	4668.466	...	1634.363	1815.186	2063.007	2366.75	1037.806	1291.77

".." values were there in the dataset that have been replaced with 0 using regex function so that accuracy can be improved followed by reading the dataframe.

Step 10: Converting numeric value to get accurate results

```
[85] ser21 = pd.Series(df24['Total male population at mid-year (thousands)(2010)']).head(10)
```

```
pd.to_numeric(ser21, downcast ='signed')
```

```
2      599955.476
3     2894001.428
4      422397.532
6      419507.414
17        618.578
26      22665.982
44      18119.951
67          0.000
79     690256.330
80       3280.775
Name: Total male population at mid-year (thousands)(2010), dtype: float64
```

pd.Series function is used to take a series of 10 values of the total male population at mid-year (thousands) (2010) followed by converting the value to numeric to increase the accuracy of the data.

Step 11: Hashing the dataframe value

```
hash(df24['Total male population at mid-year (thousands)(2000)'].values.tobytes())

-4456051440570328561
```

Hash function is used to get the hash value of the column in the dataframe.

Step 12: Column rename

```
[88] df24.rename(columns = {'Country code':'Country_code'}, inplace = True)
     df24.rename(columns = {'Total population of both sexes at mid-year (thousands)(2005)':'Total_population_of_both_sexes_at_mid_year_(thousands)_(2005)'}, inplace = True)
```

Column names have been renamed to get the correct understanding.

Step 13: Used melt function

```
[89] df25 = pd.melt(df24, id_vars=['Sort_order'], value_vars=list(df24.columns)[1:],
                    var_name='Country_code', value_name='Total_population_of_both_sexes_at_mid_year_(thousands)_(2005)')

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: This dataframe has a column name that matches the 'value_name' column name of the resulting Dataframe. In the fut
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Melt function is used to manipulate the data frame in which Unpivot of the DataFrame has been done from wide to long format, optionally leaving identifiers set.

Step 14: Reading first 10 row data

```
[90] df25.head(10)
```

	Sort_order	Country_code	Total_population_of_both_sexes_at_mid_year_(thousands)_(2005)
0	2.0	Major area, region, country or area of destina...	Developed regions
1	3.0	Major area, region, country or area of destina...	Developing regions
2	4.0	Major area, region, country or area of destina...	Least developed countries
3	6.0	Major area, region, country or area of destina...	Sub-Saharan Africa
4	17.0	Major area, region, country or area of destina...	Mauritius
5	26.0	Major area, region, country or area of destina...	United Republic of Tanzania
6	44.0	Major area, region, country or area of destina...	Sudan
7	67.0	Major area, region, country or area of destina...	Saint Helena
8	79.0	Major area, region, country or area of destina...	China
9	80.0	Major area, region, country or area of destina...	China, Hong Kong Special Administrative Region

The data frame is read to get the accurate value of the data.

Cleaning of table 3

Step 1: reading the table three

```
df31 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 3',skiprows = 14)
```

```
[40] df31.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 (both sexes)	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16	International migrant stock as a percentage of the total population (female)	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnamed: 21
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	...	2000	2005	2010	2015	1990	1995	2000	2005	2010
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	2.83474	2.775658	2.788169	2.887645	3.117222
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	7.147777	7.84948	8.650694	9.656595	10.811029
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	1.595116	1.417459	1.315351	1.279428	1.378176
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	2.045276	1.901226	1.418405	1.175548	1.073428
5	5.0	Less developed regions excluding	NaN	934.0	NaN	1.617042	1.426534	1.386338	1.410133	1.631865	...	1.470597	1.519189	1.823509	1.984593	1.530877	1.344872	1.299002	1.296941	1.432634

The table 3 has been imported by skipping 14 rows and it has read.

Step 2: Information of the data is checked

```
[41] df31.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 23 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   Sort
order                                                                     265 non-null    float64
 1   Major area, region, country or area of destination                     265 non-null    object
 2   Notes                                                                   26 non-null     object
 3   Country code                                                            265 non-null    float64
 4   Type of data (a)                                                        232 non-null    object
 5   International migrant stock as a percentage of the total population (both sexes)  266 non-null    object
 6   Unnamed: 6                                                              266 non-null    object
 7   Unnamed: 7                                                              266 non-null    object
 8   Unnamed: 8                                                              266 non-null    object
 9   Unnamed: 9                                                              266 non-null    float64
 10  Unnamed: 10                                                             266 non-null    float64
 11  International migrant stock as a percentage of the total population (male)  266 non-null    object
 12  Unnamed: 12                                                             266 non-null    object
 13  Unnamed: 13                                                             266 non-null    object
 14  Unnamed: 14                                                             266 non-null    object
 15  Unnamed: 15                                                             266 non-null    object
 16  Unnamed: 16                                                             266 non-null    object
 17  International migrant stock as a percentage of the total population (female)  266 non-null    object
 18  Unnamed: 18                                                             266 non-null    object
 19  Unnamed: 19                                                             266 non-null    object
 20  Unnamed: 20                                                             266 non-null    object
 21  Unnamed: 21                                                             266 non-null    object
 22  Unnamed: 22                                                             266 non-null    object
dtypes: float64(4), object(19)
memory usage: 47.9+ KB
```

The information of the data is checked in the system to get proper information about data type non-null count and columns.

Step 3: columns are renamed

```
[42] df31.rename(columns = {'International migrant stock as a percentage of the total population (both sexes)':'International migrant stock as a percentage of the total population (both sexes)(1990)'}, inplace
     df31.rename(columns = {'Unnamed: 6':'International migrant stock as a percentage of the total population (both sexes)(1995)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 7':'International migrant stock as a percentage of the total population (both sexes)(2000)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 8':'International migrant stock as a percentage of the total population (both sexes)(2005)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 9':'International migrant stock as a percentage of the total population (both sexes)(2010)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 10':'International migrant stock as a percentage of the total population (both sexes)(2015)'}, inplace = True)


     df31.rename(columns = {'International migrant stock as a percentage of the total population (male)':'International migrant stock as a percentage of the total population (male)(1990)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 12':'International migrant stock as a percentage of the total population (male)(1995)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 13':'International migrant stock as a percentage of the total population (male)(2000)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 14':'International migrant stock as a percentage of the total population (male)(2005)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 15':'International migrant stock as a percentage of the total population (male)(2010)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 16':'International migrant stock as a percentage of the total population (male)(2015)'}, inplace = True)


     df31.rename(columns = {'International migrant stock as a percentage of the total population (female)':'International migrant stock as a percentage of the total population (female)(1990)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 18':'International migrant stock as a percentage of the total population (female)(1995)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 19':'International migrant stock as a percentage of the total population (female)(2000)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 20':'International migrant stock as a percentage of the total population (female)(2005)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 21':'International migrant stock as a percentage of the total population (female)(2010)'}, inplace = True)
     df31.rename(columns = {'Unnamed: 22':'International migrant stock as a percentage of the total population (female)(2015)'}, inplace = True)


     df31.rename(columns = {'Sort\norder':'Sort_order'}, inplace = True)
     df31.rename(columns = {'Country code':'Country_code'}, inplace = True)
```

This step consists of renaming the columns to get accurate insights of the data.

Step 4: Reading the data

```
[43] df31.head(5)
```

	Sort_order	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) (1990)	International migrant stock as a percentage of the total population (both sexes) (1995)	International migrant stock as a percentage of the total population (both sexes) (2000)	International migrant stock as a percentage of the total population (both sexes) (2005)	International migrant stock as a percentage of the total population (both sexes) (2010)	...	International migrant stock as a percentage of the total population (male)(2000)	International migrant stock as a percentage of the total population (male)(2005)	International migrant stock as a percentage of the total population (male)(2010)	International migrant stock as a percentage of the total population (male)(2015)	International migrant stock as a percentage of the total population (female) (1990)
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.000000	...	2000	2005	2010	2015	1990
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	2.83474
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	7.147777
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	1.595116
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	2.045276

5 rows × 23 columns

The data frame has been read to analyze the change columns name

Step 6: checking null values

```
[44] df31.isna()
```

	Sort_order	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) (1990)	International migrant stock as a percentage of the total population (both sexes) (1995)	International migrant stock as a percentage of the total population (both sexes) (2000)	International migrant stock as a percentage of the total population (both sexes) (2005)	International migrant stock as a percentage of the total population (both sexes) (2010)	...	International migrant stock as a percentage of the total population (male)(2000)	International migrant stock as a percentage of the total population (male)(2005)	International migrant stock as a percentage of the total population (male)(2010)	International migrant stock as a percentage of the total population (male)(2015)	Internatior migrant stc as percentage the tot populati (femal (195
0	True	True	True	True	True	False	False	False	False	False	...	False	False	False	False	Fa
1	False	False	True	False	True	False	False	False	False	False	...	False	False	False	False	Fa
2	False	False	False	False	True	False	False	False	False	False	...	False	False	False	False	Fa
3	False	False	False	False	True	False	False	False	False	False	...	False	False	False	False	Fa
4	False	False	False	False	True	False	False	False	False	False	...	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
261	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	Fa
262	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	Fa
263	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	Fa
264	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	Fa
265	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	Fa

266 rows × 23 columns

This step consist of checking the null values in the data frame

Step 7: dropping null values

```
[45] df32 = df31.dropna()
```

```
[46] df32.head(5)
```

	Sort_order	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) (1990)	International migrant stock as a percentage of the total population (both sexes) (1995)	International migrant stock as a percentage of the total population (both sexes) (2000)	International migrant stock as a percentage of the total population (both sexes) (2005)	International migrant stock as a percentage of the total population (both sexes) (2010)	...	International migrant stock as a percentage of the total population (male)(2000)	International migrant stock as a percentage of the total population (male)(2005)	International migrant stock as a percentage of the total population (male)(2010)	International migrant stock as a percentage of the total population (male)(2015)	I m p
17	17.0	Mauritius	(1)	480.0	C	0.342184	0.663875	1.311487	1.607766	1.990142	...	0.970752	1.476673	2.131987	2.517648	
26	26.0	United Republic of Tanzania	(2)	834.0	B R	2.254774	3.698729	2.730617	1.973209	0.676035	...	2.784961	2.509854	0.679362	0.490716	
44	44.0	Sudan	(3)	729.0	B R	7.011394	4.266148	2.855743	1.694261	1.601453	...	2.859741	1.719046	1.626837	1.260235	
67	67.0	Saint Helena	(4)	654.0	B	6.924607	7.452241	7.920986	11.394478	13.635274	...	..	..	..	..	
79	79.0	China	(5)	156.0	C	0.032596	0.036014	0.040003	0.052003	0.063377	...	0.038947	0.056585	0.073271	0.084648	

5 rows × 23 columns

This is consist of dropping the null values from the data frame

Step 8: replacing '..' with 0 using regex function

```
df33 = df32.replace('\..','0',regex=True)
```

```
[48] df33.head(10)
```

	Sort_order	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) (1990)	International migrant stock as a percentage of the total population (both sexes) (1995)	International migrant stock as a percentage of the total population (both sexes) (2000)	International migrant stock as a percentage of the total population (both sexes) (2005)	International migrant stock as a percentage of the total population (both sexes) (2010)	...	International migrant stock as a percentage of the total population (male)(2000)	International migrant stock as a percentage of the total population (male)(2005)	International migrant stock as a percentage of the total population (male)(2010)
17	17.0	Mauritius	(1)	480.0	C	0.342184	0.663875	1.311487	1.607766	1.990142	...	0.970752	1.476673	2.131987
26	26.0	United Republic of Tanzania	(2)	834.0	B R	2.254774	3.698729	2.730617	1.973209	0.676035	...	2.784961	2.509854	0.679362
44	44.0	Sudan	(3)	729.0	B R	7.011394	4.266148	2.855743	1.694261	1.601453	...	2.859741	1.719046	1.626837
67	67.0	Saint Helena	(4)	654.0	B	6.924607	7.452241	7.920986	11.394478	13.635274	...	0	0	0
79	79.0	China	(5)	156.0	C	0.032596	0.036014	0.040003	0.052003	0.063377	...	0.038947	0.056585	0.073271
80	80.0	China, Hong Kong Special Administrative Region	(6)	344.0	B	38.288919	39.772134	39.347258	39.769805	39.750085	...	37.091088	36.24047	34.977681
81	81.0	China, Macao Special Administrative Region	(7)	446.0	B	56.999458	56.449723	55.750659	59.662202	59.575479	...	52.43754	58.120782	57.071632
91	91.0	Malaysia	(8)	458.0	C R	3.821406	4.522804	5.453382	6.676755	8.556379	...	5.986132	7.661215	10.297825

"."is replaced with zero using regex function since the row has no functionality in the data frame.

Step 9: hashing the value

```
hash(df33['International migrant stock as a percentage of the total population (male)(2015)'].values.tobytes())

5140964633953251780
```

```
[50] df33.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 23 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   Sort_order                                                              22 non-null     float64
 1   Major area, region, country or area of destination                      22 non-null     object
 2   Notes                                                                   22 non-null     object
 3   Country_code                                                            22 non-null     float64
 4   Type of data (a)                                                        22 non-null     object
 5   International migrant stock as a percentage of the total population (both sexes)(1990)  22 non-null  object
 6   International migrant stock as a percentage of the total population (both sexes)(1995)  22 non-null  object
 7   International migrant stock as a percentage of the total population (both sexes)(2000)  22 non-null  object
 8   International migrant stock as a percentage of the total population (both sexes)(2005)  22 non-null  object
 9   International migrant stock as a percentage of the total population (both sexes)(2010)  22 non-null  float64
 10  International migrant stock as a percentage of the total population (both sexes)(2015)  22 non-null  float64
 11  International migrant stock as a percentage of the total population (male)(1990)  22 non-null  object
 12  International migrant stock as a percentage of the total population (male)(1995)  22 non-null  object
 13  International migrant stock as a percentage of the total population (male)(2000)  22 non-null  object
 14  International migrant stock as a percentage of the total population (male)(2005)  22 non-null  object
 15  International migrant stock as a percentage of the total population (male)(2010)  22 non-null  object
 16  International migrant stock as a percentage of the total population (male)(2015)  22 non-null  object
 17  International migrant stock as a percentage of the total population (female)(1990)  22 non-null  object
 18  International migrant stock as a percentage of the total population (female)(1995)  22 non-null  object
 19  International migrant stock as a percentage of the total population (female)(2000)  22 non-null  object
 20  International migrant stock as a percentage of the total population (female)(2005)  22 non-null  object
 21  International migrant stock as a percentage of the total population (female)(2010)  22 non-null  object
 22  International migrant stock as a percentage of the total population (female)(2015)  22 non-null  object
dtypes: float64(4), object(19)
```

This is step consist of hashing the value to buy and getting information of the data frame

Step 10: Renaming the columns

```
[51] df33.rename(columns = {'International migrant stock as a percentage of the total population (both sexes)(2015)':'International_migrant_stock_as_a_percentage_of_the_total_population_both_sexes_2015'}, inp

print(df33.eval("Country_code + International_migrant_stock_as_a_percentage_of_the_total_population_both_sexes_2015"))

17     482.245109
26     834.488536
44     730.251345
67     669.248675
79     156.071076
80     382.949940
81     504.321903
91     466.289349
110     33.709062
112    212.834034
113    272.220248
122    280.473040
134    501.512107
140    880.281626
144    251.739683
150    592.235611
160    436.000000
166    697.122622
168    736.690237
187    587.298781
195    333.028285
240     64.218410
dtype: float64
```

This step consists of renaming the columns 9f the data frame to get an accurate understanding.

Step 11: melt function usage

```
df34 = pd.melt(df33, id_vars=['Sort_order'], value_vars=list(df33.columns)[1:],
               var_name='Country_code', value_name='International_migrant_stock_as_a_percentage_of_the_total_population_both_sexes_2015')

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: This dataframe has a column name that matches the 'value_name' column name of the resulting Dataframe. I
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Melt function is used in the data frame to manipulate the data with the help of unpivot a data frame from white to long format.

Step 12: reading the data frame

```
[56] df34.head(5)
```

	Sort_order	Country_code	International_migrant_stock_as_a_percentage_of_the_total_population_both_sexes_2015
0	17.0	Major area, region, country or area of destina...	Mauritius
1	26.0	Major area, region, country or area of destina...	United Republic of Tanzania
2	44.0	Major area, region, country or area of destina...	Sudan
3	67.0	Major area, region, country or area of destina...	Saint Helena
4	79.0	Major area, region, country or area of destina...	China

Data manipulation is done using the .melt() function in which Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

Cleaning of table 4

Step 1: Importing the data set in Google colab

```
[57] df41 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 4',skiprows = 14)
```

```
df41.head(5)
```

	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	Unna
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010
1	1.0	WORLD	NaN	900.0	NaN	49.03915	49.16879	49.112244	48.832993	48.
2	2.0	Developed regions	(b)	901.0	NaN	51.123977	51.149024	51.113307	51.171501	51.
3	3.0	Developing regions	(c)	902.0	NaN	46.592099	46.500135	46.128444	45.134297	43.
4	4.0	Least developed countries	(d)	941.0	NaN	47.261155	47.571664	46.826689	45.157406	45.

This stage consists of importing and reading the data set to the Google colab using pd.read and df.head() functions.

Step 2: Checking the data information

```
[59] df41.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 11 columns):
 #   Column                                                        Non-Null Count  Dtype
---  ------                                                        --------------  -----
 0   Sort
order                                                        265 non-null    float64
 1   Major area, region, country or area of destination           265 non-null    object
 2   Notes                                                         26 non-null     object
 3   Country code                                                  265 non-null    float64
 4   Type of data (a)                                              232 non-null    object
 5   Female migrants as a percentage of the international migrant stock  266 non-null    object
 6   Unnamed: 6                                                    266 non-null    object
 7   Unnamed: 7                                                    266 non-null    object
 8   Unnamed: 8                                                    266 non-null    object
 9   Unnamed: 9                                                    266 non-null    float64
 10  Unnamed: 10                                                   266 non-null    float64
dtypes: float64(4), object(7)
memory usage: 23.0+ KB
```

In this stage df41.info() is used to check the data information from the dataframe.

Step 3: renaming the columns and reading the data frame

```
[60] df41.rename(columns = {'Female migrants as a percentage of the international migrant stock':'Female_migrants_as_a_percentage_of_the_international_migrant_stock_1990'}, inplace = True)
     df41.rename(columns = {'Unnamed: 6':'Female_migrants_as_a_percentage_of_the_international_migrant_stock_1995'}, inplace = True)
     df41.rename(columns = {'Unnamed: 7':'Female_migrants_as_a_percentage_of_the_international_migrant_stock_2000'}, inplace = True)
     df41.rename(columns = {'Unnamed: 8':'Female_migrants_as_a_percentage_of_the_international_migrant_stock_2005'}, inplace = True)
     df41.rename(columns = {'Unnamed: 9':'Female_migrants_as_a_percentage_of_the_international_migrant_stock_2010'}, inplace = True)
     df41.rename(columns = {'Unnamed: 10':'Female_migrants_as_a_percentage_of_the_international_migrant_stock_2015'}, inplace = True)

     df41.rename(columns = {'Sort\norder':'Sort_order'}, inplace = True)
     df41.rename(columns = {'Country code':'Country_code'}, inplace = True)
```

```
df41.head(5)
```

	Sort_order	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_1990	Female_migrants_as_a_percentage_of_the_international_migrant_stock_1995	Femal
0	NaN	NaN	NaN	NaN	NaN	1990	1995	
1	1.0	WORLD	NaN	900.0	NaN	49.03915	49.16879	
2	2.0	Developed regions	(b)	901.0	NaN	51.123977	51.149024	
3	3.0	Developing regions	(c)	902.0	NaN	46.592099	46.500135	
4	4.0	Least developed countries	(d)	941.0	NaN	47.261155	47.571664	

Renaming of the columns has been done along with reading the data frame.

Step 4: Dropping null values

```
df42 = df41.dropna()
```

```
[64] df42.head(5)
```

	Sort_order	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_1990	Female_migrants_as_a_percentage_of_the_international_migrant_stock_1995	F
17	17.0	Mauritius	(1)	480.0	C	51.203986	56.919792	
26	26.0	United Republic of Tanzania	(2)	834.0	B R	50.63107	50.536372	
44	44.0	Sudan	(3)	729.0	B R	50.360896	50.094646	
67	67.0	Saint Helena	(4)	654.0	B	42.036554	44.670051	
79	79.0	China	(5)	156.0	C	48.997372	49.566032	

.dropna() is used to drop the null values.

Step 5: Checking the data frame information

```
df42.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 11 columns):
 #   Column                                                    Non-Null Count  Dtype
---  ------                                                    --------------  -----
 0   Sort_order                                                22 non-null     float64
 1   Major area, region, country or area of destination       22 non-null     object
 2   Notes                                                     22 non-null     object
 3   Country_code                                              22 non-null     float64
 4   Type of data (a)                                          22 non-null     object
 5   Female_migrants_as_a_percentage_of_the_international_migrant_stock_1990  22 non-null  object
 6   Female_migrants_as_a_percentage_of_the_international_migrant_stock_1995  22 non-null  object
 7   Female_migrants_as_a_percentage_of_the_international_migrant_stock_2000  22 non-null  object
 8   Female_migrants_as_a_percentage_of_the_international_migrant_stock_2005  22 non-null  object
 9   Female_migrants_as_a_percentage_of_the_international_migrant_stock_2010  22 non-null  float64
 10  Female_migrants_as_a_percentage_of_the_international_migrant_stock_2015  22 non-null  float64
dtypes: float64(4), object(7)
memory usage: 2.1+ KB
```

Data frame information is checked using .info() function

Step 6: Replacing unnecessary value with zero

```
[66] df43 = df42.replace('\..','0',regex=True)
```

```
df43.head(5)
```

	Sort_order	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_1990	Female_migrants_as_a_percentage_of_the_international_migrant_stock_1995	Femal
17	17.0	Mauritius	(1)	480.0	C	51.203986	56.919792	
26	26.0	United Republic of Tanzania	(2)	834.0	B R	50.63107	50.536372	
44	44.0	Sudan	(3)	729.0	B R	50.360896	50.094646	
67	67.0	Saint Helena	(4)	654.0	B	42.036554	44.670051	
79	79.0	China	(5)	156.0	C	48.997372	49.566032	

".." value is replaced with 0 using the regex function.

Step 7: Creating a series of 10 values in the concerned column of data frame and its numeric conversation

```
[68] ser43 = pd.Series(df43['Female_migrants_as_a_percentage_of_the_international_migrant_stock_2015']).head(10)
```

```
[69] pd.to_numeric(ser43, downcast ='signed')
```

```
17     44.614308
26     50.079243
44     49.445556
67     44.536424
79     38.639287
80     60.546348
81     54.569408
91     39.161410
110    52.089191
112    55.660228
Name: Female_migrants_as_a_percentage_of_the_international_migrant_stock_2015, dtype: float64
```

A series of 10 values have been created and it is converted to numeric values.

Step 8: Returning unique values based on hash table and reading the data frame

```
[70] df44 = df43['Female_migrants_as_a_percentage_of_the_international_migrant_stock_2015'].unique()
```

```
[72] print(df44)
```

```
[44.6143082  50.07924294 49.44555561 44.53642384 38.6392869  60.54634837
 54.5694085  39.16140962 52.08919131 55.66022827 56.83641189 55.68144904
 64.63010133 52.20576622 49.14224027 47.81730706 53.25       56.01164667
 51.21814578 51.60744501 55.47727573 50.65425347]
```

Unique values of the column are returned using the .unique() function and the value is printed to check the values.

Cleaning of table 5

Step 1: Reading the table five by importing it as a data frame

```
df51 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 5',skiprows = 14)
```

```
[70] df51.head(5)
```

	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	Unnamed: 14	Annual rate of change of the migrant stock (female)	Unnamed: 16	Unnamed: 17	Unnamed: 18	Unnamed 1
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	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	1.912603	1.104667	1.405044	1.92808	2.737012	1.86783
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	2.285643	2.250995	2.529838	2.65595	1.24109
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	-0.526904	0.081268	0.89236	2.881555	2.93300
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	1.249146	-3.316818	-1.265617	0.57011	3.7207

Table 5 is imported to google colab as a dataframe by skipping 14 rows and it has read.

Step 2: Checking the data frame information

```
[71] df51.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 20 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort
order                                                 265 non-null    float64
 1   Major area, region, country or area of destination     265 non-null    object
 2   Notes                                                   26 non-null     object
 3   Country code                                            265 non-null    float64
 4   Type of data (a)                                        232 non-null    object
 5   Annual rate of change of the migrant stock (both sexes) 266 non-null    object
 6   Unnamed: 6                                              266 non-null    object
 7   Unnamed: 7                                              266 non-null    object
 8   Unnamed: 8                                              266 non-null    object
 9   Unnamed: 9                                              266 non-null    object
 10  Annual rate of change of the migrant stock (male)       266 non-null    object
 11  Unnamed: 11                                             266 non-null    object
 12  Unnamed: 12                                             266 non-null    object
 13  Unnamed: 13                                             266 non-null    object
 14  Unnamed: 14                                             266 non-null    object
 15  Annual rate of change of the migrant stock (female)     266 non-null    object
 16  Unnamed: 16                                             266 non-null    object
 17  Unnamed: 17                                             266 non-null    object
 18  Unnamed: 18                                             266 non-null    object
 19  Unnamed: 19                                             266 non-null    object
dtypes: float64(2), object(18)
memory usage: 41.7+ KB
```

The information of the dataframe is checked using .info() function

Step 3: Renaming the columns name

```
[72] df51.rename(columns = {'Annual rate of change of the migrant stock (both sexes)':'Annual rate of change of the migrant stock (both sexes)(1990-1995)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 6':'Annual rate of change of the migrant stock (both sexes)(1995-2000)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 7':'Annual rate of change of the migrant stock (both sexes)(2000-2005)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 8':'Annual rate of change of the migrant stock (both sexes)(2005-2010)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 9':'Annual rate of change of the migrant stock (both sexes)(2010-2015)'}, inplace = True)


     df51.rename(columns = {'Annual rate of change of the migrant stock (male)':'Annual rate of change of the migrant stock (male)(1990-1995)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 11':'Annual rate of change of the migrant stock (male)(1995-2000)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 12':'Annual rate of change of the migrant stock (male)(2000-2005)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 13':'Annual rate of change of the migrant stock (male)(2005-2010)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 14':'Annual rate of change of the migrant stock (male)(2010-2015)'}, inplace = True)


     df51.rename(columns = {'Annual rate of change of the migrant stock (female)':'Annual rate of change of the migrant stock (female)(1990-1995)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 16':'Annual rate of change of the migrant stock (female)(1995-2000)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 17':'Annual rate of change of the migrant stock (female)(2000-2005)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 18':'Annual rate of change of the migrant stock (female)(2005-2010)'}, inplace = True)
     df51.rename(columns = {'Unnamed: 19':'Annual rate of change of the migrant stock (female)(2010-2015)'}, inplace = True)


     df51.rename(columns = {'Sort\norder':'Sort_order'}, inplace = True)
     df51.rename(columns = {'Country code':'Country_code'}, inplace = True)
```

Column names are renamed to get accurate insights of the data.

Step 4: Reading the data frame after the renaming of columns

```
[73] df51.head(5)
```

	Sort_order	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) (1990-1995)	Annual rate of change of the migrant stock (both sexes) (1995-2000)	Annual rate of change of the migrant stock (both sexes) (2000-2005)	Annual rate of change of the migrant stock (both sexes) (2005-2010)	Annual rate of change of the migrant stock (both sexes) (2010-2015)	Annual rate of change of the migrant stock (male) (1990-1995)	Annual rate of change of the migrant stock (male) (1995-2000)	Annual rate of change of the migrant stock (male) (2000-2005)	Annual rate of change of the migrant stock (male) (2005-2010)	Annual rate of change of the migrant stock (male) (2010-2015)	Annual rate of change of the migrant stock (female) (1990-1995)	Annual rate of change of the migrant stock (female) (1995-2000)	Annual rate of change of the migrant stock (female) (2000-2005)	Annual rate of change of the migrant stock (female) (2005-2010)
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	1990-1995	1995-2000	2000-2005	2005-2010
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	1.104667	1.405044	1.92808	2.737012
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	2.285643	2.250995	2.529838	2.65595
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	-0.526904	0.081268	0.89236	2.881555
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	1.249146	-3.316818	-1.265617	0.57011

.head() function is used to read the data and check if the column names are renamed.

Step 5: Dropping null values and reading the data

```
[74] df52 = df51.dropna()

[75] df52.head(5)
```

	Sort_order	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) (1990-1995)	Annual rate of change of the migrant stock (both sexes) (1995-2000)	Annual rate of change of the migrant stock (both sexes) (2000-2005)	Annual rate of change of the migrant stock (both sexes) (2005-2010)	Annual rate of change of the migrant stock (both sexes) (2010-2015)	Annual rate of change of the migrant stock (male) (1990-1995)	Annual rate of change of the migrant stock (male) (1995-2000)	Annual rate of change of the migrant stock (male) (2000-2005)	Annual rate of change of the migrant stock (male) (2005-2010)	Annual rate of change of the migrant stock (male) (2010-2015)	Annual rate of change of the migrant stock (female) (1990-1995)	Annual rate of change of the migrant stock (female) (1995-2000)	Annual rate of change of the migrant stock (female) (2000-2005)	Annual rate of change of the migrant stock (female) (2005-2010)
17	17.0	Mauritius	(1)	480.0	C	14.588616	14.592823	4.686286	4.687391	2.811758	12.096917	11.389605	8.990562	7.768724	3.654518	16.705131	16.716205	1.687301	1.69034
26	26.0	United Republic of Tanzania	(2)	834.0	B R	13.117422	-3.506368	-3.714741	-18.308853	-3.333506	13.155749	-2.996682	0.660451	-23.019025	-3.324228	13.07998	-4.018138	-9.518651	-12.165662
44	44.0	Sudan	(3)	729.0	B R	-5.730389	-5.456236	-7.834156	1.298935	-2.773273	-5.623401	-5.313539	-7.579256	1.322865	-2.936348	-5.836406	-5.599413	-8.095091	1.274091
67	67.0	Saint Helena	(4)	654.0	B	0.566318	0.550723	3.687541	3.112326	1.193875	-0.363646	-0.277141	4.03032	3.416983	1.422547	1.781593	1.530802	3.29244	2.748601
79	79.0	China	(5)	156.0	C	3.224179	2.775813	5.799894	4.490595	2.809678	2.999935	2.608091	8.046073	5.721952	3.42196	3.454961	2.945039	3.267772	2.808143

Null values of dropped from the dataframe using .dropna() and it has read using .head()

Step 6: Replacing unwanted value with zero using regex function and reading the data

```
df53 = df52.replace('\..','0',regex=True)
df53
```

	Sort_order	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)(1990-1995)	Annual rate of change of the migrant stock (both sexes)(1995-2000)	Annual rate of change of the migrant stock (both sexes)(2000-2005)	Annual rate of change of the migrant stock (both sexes)(2005-2010)	Annual rate of change of the migrant stock (both sexes)(2010-2015)	Annual rate of change of the migrant stock (male)(1990-1995)	Annual rate of change of the migrant stock (male)(1995-2000)	Annual rate of change of the migrant stock (male)(2000-2005)	Annual rate of change of the migrant stock (male)(2005-2010)	Annual rate of change of the migrant stock (male)(2010-2015)	Annual rate of change of the migrant stock (female)(1990-1995)	Annual rate of change of the migrant stock (female)(1995-2000)	Annual rate of change of the migrant stock (female)(2000-2005)	Annual rate of change of the migrant stock (female)(2005-2010)
17	17.0	Mauritius	(1)	480.0	C	14.588616	14.592823	4.686286	4.687391	2.811758	12.096917	11.389605	8.990562	7.768724	3.654518	16.705131	16.716205	1.687301	1.69034
26	26.0	United Republic of Tanzania	(2)	834.0	B R	13.117422	-3.506368	-3.714741	-18.308853	-3.333506	13.155749	-2.996682	0.660451	-23.019025	-3.324228	13.07998	-4.018138	-9.518651	-12.165662
44	44.0	Sudan	(3)	729.0	B R	-5.730389	-5.456236	-7.834156	1.298935	-2.773273	-5.623401	-5.313539	-7.579256	1.322865	-2.936348	-5.836406	-5.599413	-8.095091	1.274091
67	67.0	Saint Helena	(4)	654.0	B	0.566318	0.550723	3.687541	3.112326	1.193875	-0.363646	-0.277141	4.03032	3.416983	1.422547	1.781593	1.530802	3.29244	2.748601
79	79.0	China	(5)	156.0	C	3.224179	2.775813	5.799894	4.490595	2.809678	2.999935	2.608091	8.046073	5.721952	3.42196	3.454961	2.945039	3.267772	2.808143
80	80.0	China, Hong Kong Special Administrative Region	(6)	344.0	B	1.934685	1.763924	0.386725	0.426942	0.418019	0.879291	0.84224	-0.672186	-0.644505	-0.486587	2.964512	2.581289	1.2438	1.216044
81	81.0	China, Macao Special Administrative Region	(7)	446.0	B	1.850912	1.362891	2.967716	2.626524	1.464457	1.45656	1.076246	3.511567	2.317669	1.229786	2.194277	1.604698	2.503397	2.893079
91	91.0	Malaysia	(8)	458.0	C R	5.956825	6.18735	5.979959	6.685681	0.880031	5.526386	5.743171	6.679892	7.342073	1.246499	6.54379	6.762979	5.059183	5.749974

"..” value from the dataframe is replaced with 0 using the regex function and the data is read using the .head() function.

Step 7: Checking data information

```
[77] df53.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 20 columns):
 #   Column                                                      Non-Null Count  Dtype
---  ------                                                      --------------  -----
 0   Sort_order                                                  22 non-null     float64
 1   Major area, region, country or area of destination          22 non-null     object
 2   Notes                                                       22 non-null     object
 3   Country_code                                                22 non-null     float64
 4   Type of data (a)                                            22 non-null     object
 5   Annual rate of change of the migrant stock (both sexes)(1990-1995)  22 non-null  object
 6   Annual rate of change of the migrant stock (both sexes)(1995-2000)  22 non-null  object
 7   Annual rate of change of the migrant stock (both sexes)(2000-2005)  22 non-null  object
 8   Annual rate of change of the migrant stock (both sexes)(2005-2010)  22 non-null  object
 9   Annual rate of change of the migrant stock (both sexes)(2010-2015)  22 non-null  object
 10  Annual rate of change of the migrant stock (male)(1990-1995)  22 non-null  object
 11  Annual rate of change of the migrant stock (male)(1995-2000)  22 non-null  object
 12  Annual rate of change of the migrant stock (male)(2000-2005)  22 non-null  object
 13  Annual rate of change of the migrant stock (male)(2005-2010)  22 non-null  object
 14  Annual rate of change of the migrant stock (male)(2010-2015)  22 non-null  object
 15  Annual rate of change of the migrant stock (female)(1990-1995)  22 non-null  object
 16  Annual rate of change of the migrant stock (female)(1995-2000)  22 non-null  object
 17  Annual rate of change of the migrant stock (female)(2000-2005)  22 non-null  object
 18  Annual rate of change of the migrant stock (female)(2005-2010)  22 non-null  object
 19  Annual rate of change of the migrant stock (female)(2010-2015)  22 non-null  object
dtypes: float64(2), object(18)
memory usage: 3.6+ KB
```

Data information is obtained using .info()

Step 8: Evaluate python expression as string using various back ends for top level evaluation

```
[78] print(df53.eval("Country_code + Sort_order"))

17     497.0
26     860.0
44     773.0
67     721.0
79     235.0
80     424.0
81     527.0
91     549.0
110    141.0
112    308.0
113    381.0
122    397.0
134    632.0
140    970.0
144    390.0
150    728.0
160    496.0
166    854.0
168    892.0
187    722.0
195    507.0
240    276.0
dtype: float64
```

.eval() is used to get top level evaluation of the dataframe.

Step 9: Returning unique values based on hash table

```
[79] df54 = df53['Country_code'].unique()

[80] print(df54)

[480. 834. 729. 654. 156. 344. 446. 458.  31. 196. 268. 275. 498. 830.
 246. 578. 336. 688. 724. 535. 312.  36.]
```

Unique values of the column are returned based on the hash table and the value is printed using print function to check the values.

Cleaning of table 6

Step 1: Importing the table 6 in the form of data frame by skipping 14 rows and reading it

```
df61 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 6',skiprows = 14)
```

```
df61.head(6)
```

	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	Unnamed: 16	Annual rate of change of the refugee stock	Unnamed: 18
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	1995	2000	2005	2010.000000	2015.000000	1990-1995	1995-2000
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	...	11.103013	9.164736	6.941389	6.932687	8.033424	-2.123497	-3.837069
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	...	3.910511	2.899391	2.015025	1.544140	1.391085	9.388424	-5.983348
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	...	20.795958	18.507035	14.733162	14.944759	17.073768	-2.839417	-2.332154
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	...	44.041961	30.221557	24.08243	19.533425	28.801534	-0.680327	-7.531747
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	11773616	9084039	9783059	8551722	11365954	...	15.999082	16.51313	13.305391	14.363526	15.537313	-4.3836	0.632489

6 rows × 22 columns

Table 6 is imported to google colab and its data is read using .head() function.

Step 2: Getting the information of the data frame

```
[83] df61.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort
order                                                   265 non-null    float64
 1   Major area, region, country or area of destination     265 non-null    object
 2   Notes                                                   26 non-null     object
 3   Country code                                            265 non-null    float64
 4   Type of data (a)                                        232 non-null    object
 5   Estimated refugee stock at mid-year (both sexes)        266 non-null    object
 6   Unnamed: 6                                              266 non-null    object
 7   Unnamed: 7                                              266 non-null    object
 8   Unnamed: 8                                              266 non-null    object
 9   Unnamed: 9                                              266 non-null    int64
 10  Unnamed: 10                                             266 non-null    int64
 11  Refugees as a percentage of the international migrant stock  266 non-null    object
 12  Unnamed: 12                                             266 non-null    object
 13  Unnamed: 13                                             266 non-null    object
 14  Unnamed: 14                                             266 non-null    object
 15  Unnamed: 15                                             266 non-null    float64
 16  Unnamed: 16                                             266 non-null    float64
 17  Annual rate of change of the refugee stock              266 non-null    object
 18  Unnamed: 18                                             266 non-null    object
 19  Unnamed: 19                                             266 non-null    object
 20  Unnamed: 20                                             266 non-null    object
 21  Unnamed: 21                                             266 non-null    object
dtypes: float64(4), int64(2), object(16)
memory usage: 45.8+ KB
```

Dataframe information is obtained using .info()

Step 3: Renaming the column name

```
[84] df61.rename(columns = {'Estimated refugee stock at mid-year (both sexes)':'Estimated refugee stock at mid-year (both sexes)(1990)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 6':'Estimated refugee stock at mid-year (both sexes)(1995)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 7':'Estimated refugee stock at mid-year (both sexes)(2000)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 8':'Estimated refugee stock at mid-year (both sexes)(2005)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 9':'Estimated refugee stock at mid-year (both sexes)(2010)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 10':'Estimated refugee stock at mid-year (both sexes)(2015)'}, inplace = True)


     df61.rename(columns = {'Refugees as a percentage of the international migrant stock':'Refugees as a percentage of the international migrant stock(1990)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 12':'Refugees as a percentage of the international migrant stock(1995)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 13':'Refugees as a percentage of the international migrant stock(2000)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 14':'Refugees as a percentage of the international migrant stock(2005)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 15':'Refugees as a percentage of the international migrant stock(2010)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 16':'Refugees as a percentage of the international migrant stock(2015)'}, inplace = True)


     df61.rename(columns = {'Annual rate of change of the refugee stock':'Annual rate of change of the refugee stock(1990-1995)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 18':'Annual rate of change of the refugee stock(1995-2000)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 19':'Annual rate of change of the refugee stock(2000-2005)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 20':'Annual rate of change of the refugee stock(2005-2010)'}, inplace = True)
     df61.rename(columns = {'Unnamed: 21':'Annual rate of change of the refugee stock(2010-2015)'}, inplace = True)


     df61.rename(columns = {'Sort\norder':'Sort_order'}, inplace = True)
     df61.rename(columns = {'Country code':'Country_code'}, inplace = True)
```

Column names are renamed to get accurate insights of the column values.

Step 4: Reading the data frame

```
[85] df61.head(5)
```

	Sort_order	Major area, region, country or area of destination	Notes	Country_code	Type of data (a)	Estimated refugee stock at mid-year (both sexes) (1990)	Estimated refugee stock at mid-year (both sexes) (1995)	Estimated refugee stock at mid-year (both sexes) (2000)	Estimated refugee stock at mid-year (both sexes) (2005)	Estimated refugee stock at mid-year (both sexes) (2010)	...	Refugees as a percentage of the international migrant stock(1995)	Refugees as a percentage of the international migrant stock(2000)	Refugees as a percentage of the international migrant stock(2005)	Refugees as a percentage of the international migrant stock(2010)	Refugees as a percentage of the international migrant stock(2015)
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	1995	2000	2005	2010.000000	2015.000000
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	...	11.103013	9.164736	6.941389	6.932687	8.033424
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	...	3.910511	2.899391	2.015025	1.544140	1.391085
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	...	20.795958	18.507035	14.733162	14.944759	17.073768
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	...	44.041961	30.221557	24.08243	19.533425	28.801534

5 rows × 22 columns

The data frame is read using .head() function

Step 5: Dropping null values

```
[86] df62 = df61.dropna()
```

Null values of the dataframe are dropped using .dropna()

Step 6: Replacing unwanted value with zero using regex and checking data

```
[87] df63 = df62.replace('\..','0',regex=True)
     df63.head(5)
```

	Sort_order	Major area, region, country or area of destination	Notes	Country_code	Type of data (a)	Estimated refugee stock at mid-year (both sexes) (1990)	Estimated refugee stock at mid-year (both sexes) (1995)	Estimated refugee stock at mid-year (both sexes) (2000)	Estimated refugee stock at mid-year (both sexes) (2005)
17	17.0	Mauritius	(1)	480.0	C	0	0	0	0
26	26.0	United Republic of Tanzania	(2)	834.0	B R	265184	829671	680862	548824
44	44.0	Sudan	(3)	729.0	B R	1031050	674071	414928	147256
67	67.0	Saint Helena	(4)	654.0	B	0	0	0	0
79	79.0	China	(5)	156.0	C	285788	289747	293705	297346

5 rows × 22 columns

".." value is replaced with 0 using the regex function.

Step 7: Creating dummies data to convert categorical variable into dummy

```
table61 = pd.get_dummies(df63)
table61
```

	Sort_order	Country_code	Estimated refugee stock at mid-year (both sexes) (2010)	Estimated refugee stock at mid-year (both sexes) (2015)	Refugees as a percentage of the international migrant stock(2010)	Refugees as a percentage of the international migrant stock(2015)	Major area, region, country or area of destination_Australia	Major area, region, country or area of destination_Azerbaijan	Major area, region, country or area of destination_Bonaire, Sint Eustatius and Saba	Major area, region, country or area of destination_Channel Islands	...
17	17.0	480.0	0	0	0.000000	0.000000	0	0	0	0	...
26	26.0	834.0	109286	90650	35.413480	34.702284	0	0	0	0	...
44	44.0	729.0	144008	205174	24.899241	40.751415	0	0	0	0	...
67	67.0	654.0	0	0	0.000000	0.000000	0	0	0	0	...
79	79.0	156.0	300986	301052	35.415909	30.780965	0	0	0	0	...
80	80.0	344.0	154	184	0.005540	0.006482	0	0	0	0	...
81	81.0	446.0	0	0	0.000000	0.000000	0	0	0	0	...
91	91.0	458.0	81516	98207	3.388014	3.906027	0	0	0	0	...
110	110.0	31.0	1908	1380	0.689055	0.522251	0	1	0	0	...
112	112.0	196.0	3394	4281	1.806059	2.182324	0	0	0	0	...
113	113.0	268.0	639	857	0.350710	0.507695	0	0	0	0	...
122	122.0	275.0	1955469	2051096	757.839725	802.755306	0	0	0	0	...
134	134.0	498.0	148	283	0.093868	0.198035	0	0	0	0	...

Dummy data is created to convert categorical variables into dummies using .get_dummies().

Step 8: Checking the data frame information

```
[89] df63.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 22 columns):
 #   Column                                                          Non-Null Count  Dtype
---  ------                                                          --------------  -----
 0   Sort_order                                                      22 non-null     float64
 1   Major area, region, country or area of destination              22 non-null     object
 2   Notes                                                           22 non-null     object
 3   Country_code                                                    22 non-null     float64
 4   Type of data (a)                                                22 non-null     object
 5   Estimated refugee stock at mid-year (both sexes)(1990)          22 non-null     object
 6   Estimated refugee stock at mid-year (both sexes)(1995)          22 non-null     object
 7   Estimated refugee stock at mid-year (both sexes)(2000)          22 non-null     object
 8   Estimated refugee stock at mid-year (both sexes)(2005)          22 non-null     object
 9   Estimated refugee stock at mid-year (both sexes)(2010)          22 non-null     int64
 10  Estimated refugee stock at mid-year (both sexes)(2015)          22 non-null     int64
 11  Refugees as a percentage of the international migrant stock(1990)  22 non-null  object
 12  Refugees as a percentage of the international migrant stock(1995)  22 non-null  object
 13  Refugees as a percentage of the international migrant stock(2000)  22 non-null  object
 14  Refugees as a percentage of the international migrant stock(2005)  22 non-null  object
 15  Refugees as a percentage of the international migrant stock(2010)  22 non-null  float64
 16  Refugees as a percentage of the international migrant stock(2015)  22 non-null  float64
 17  Annual rate of change of the refugee stock(1990-1995)           22 non-null     object
 18  Annual rate of change of the refugee stock(1995-2000)           22 non-null     object
 19  Annual rate of change of the refugee stock(2000-2005)           22 non-null     object
 20  Annual rate of change of the refugee stock(2005-2010)           22 non-null     object
 21  Annual rate of change of the refugee stock(2010-2015)           22 non-null     object
dtypes: float64(4), int64(2), object(16)
memory usage: 4.0+ KB
```

The dataframe information is checked using .info()

Step 9: Rename in the column and checking the data information

```
df63.rename(columns = {'Estimated refugee stock at mid-year (both sexes)(2010)':'Estimated_refugee_stock_at_mid_year_both_sexes_2010'}, inplace = True)
```

```
[91] df63.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 22 columns):
 #   Column                                                          Non-Null Count  Dtype
---  ------                                                          --------------  -----
 0   Sort_order                                                      22 non-null     float64
 1   Major area, region, country or area of destination              22 non-null     object
 2   Notes                                                           22 non-null     object
 3   Country_code                                                    22 non-null     float64
 4   Type of data (a)                                                22 non-null     object
 5   Estimated refugee stock at mid-year (both sexes)(1990)          22 non-null     object
 6   Estimated refugee stock at mid-year (both sexes)(1995)          22 non-null     object
 7   Estimated refugee stock at mid-year (both sexes)(2000)          22 non-null     object
 8   Estimated refugee stock at mid-year (both sexes)(2005)          22 non-null     object
 9   Estimated_refugee_stock_at_mid_year_both_sexes_2010             22 non-null     int64
 10  Estimated refugee stock at mid-year (both sexes)(2015)          22 non-null     int64
 11  Refugees as a percentage of the international migrant stock(1990)  22 non-null  object
 12  Refugees as a percentage of the international migrant stock(1995)  22 non-null  object
 13  Refugees as a percentage of the international migrant stock(2000)  22 non-null  object
 14  Refugees as a percentage of the international migrant stock(2005)  22 non-null  object
 15  Refugees as a percentage of the international migrant stock(2010)  22 non-null  float64
 16  Refugees as a percentage of the international migrant stock(2015)  22 non-null  float64
 17  Annual rate of change of the refugee stock(1990-1995)           22 non-null     object
 18  Annual rate of change of the refugee stock(1995-2000)           22 non-null     object
 19  Annual rate of change of the refugee stock(2000-2005)           22 non-null     object
 20  Annual rate of change of the refugee stock(2005-2010)           22 non-null     object
 21  Annual rate of change of the refugee stock(2010-2015)           22 non-null     object
dtypes: float64(4), int64(2), object(16)
memory usage: 4.0+ KB
```

Renaming the column and checking the data information has been done in this stage using .rename() and .info() functions

Step 10: Creating a series of a column and converting the data to numeric

```
ser61 = pd.Series(df63['Estimated_refugee_stock_at_mid_year_both_sexes_2010']).head(5)
```

```
[93] pd.to_numeric(ser61, downcast ='signed')
```

```
17         0
26    109286
44    144008
67         0
79    300986
Name: Estimated_refugee_stock_at_mid_year_both_sexes_2010, dtype: int32
```

A series of 5 values of the column has been created and its values are converted to numeric values.

Step 11: Top level evaluation

```
[94] print(df63.eval("Sort_order + Estimated_refugee_stock_at_mid_year_both_sexes_2010"))
```

```
17          17.0
26      109312.0
44      144052.0
67          67.0
79      301065.0
80         234.0
81          81.0
91       81607.0
110       2018.0
112       3506.0
113        752.0
122    1955591.0
134        282.0
140        140.0
144       8868.0
150      40410.0
160        160.0
166      73774.0
168       3988.0
187        187.0
195        195.0
240      25745.0
dtype: float64
```

.eval () is used to get top level evaluation and its value is printed.

Step 12: Manipulating the data

```
[95] df64 = pd.melt(df63, id_vars=['Sort_order'], value_vars=list(df63.columns)[1:],
                    var_name='Country_code', value_name='Estimated_refugee_stock_at_mid_year_both_sexes_2010')

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: This dataframe has a column name that matches the 'value_name' column name of the resulting Dataframe.
  exec(code_obj, self.user_global_ns, self.user_ns)

[96] df64.head(5)
```

	Sort_order	Country_code	Estimated_refugee_stock_at_mid_year_both_sexes_2010
0	17.0	Major area, region, country or area of destina...	Mauritius
1	26.0	Major area, region, country or area of destina...	United Republic of Tanzania
2	44.0	Major area, region, country or area of destina...	Sudan
3	67.0	Major area, region, country or area of destina...	Saint Helena
4	79.0	Major area, region, country or area of destina...	China

Manipulation of the data is done using .melt() and the data is read.

Cleaning of table 6

Step 1: Importing the table 6 in the form of a data frame and reading it.

```
df61 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'Table 6',skiprows = 14)

df61.head(6)
```

	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	Unnamed: 16	Annual rate of change of the refugee stock	Unnamed: 18	Unnamed: 19	Unnamed: 20
0	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010	...	1995	2000	2005	2010.000000	2015.000000	1990-1995	1995-2000	2000-2005	2005-2010
1	1.0	WORLD	NaN	900.0	NaN	18836571	17853840	15827803	13276733	15370755	...	11.103013	9.164736	6.941389	6.932687	8.033424	-2.123497	-3.837069	-5.557223	-0.025089
2	2.0	Developed regions	(b)	901.0	NaN	2014564	3609670	2997256	2361229	2046917	...	3.910511	2.899391	2.015025	1.544140	1.391085	9.388424	-5.983348	-7.277379	-5.323293
3	3.0	Developing regions	(c)	902.0	NaN	16822007	14244170	12830547	10915504	13323838	...	20.795958	18.507035	14.733162	14.944759	17.073768	-2.839417	-2.332154	-4.561	0.285195
4	4.0	Least developed countries	(d)	941.0	NaN	5048391	5160131	3047488	2363782	1957884	...	44.041961	30.221557	24.08243	19.533425	28.801534	-0.680327	-7.531747	-4.541459	-4.187109
5	5.0	Less developed regions excluding least develop...	NaN	934.0	NaN	11773616	9084039	9783059	8551722	11365954	...	15.999082	16.51313	13.305391	14.363526	15.537313	-4.3836	0.632489	-4.319731	1.530456

6 rows × 22 columns

Table 6 is imported to google colab by skipping 14 rows and the data frame is read.

Step 2: Checking the data frame information

```
df61.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Sort
order                                                   265 non-null    float64
 1   Major area, region, country or area of destination     265 non-null    object
 2   Notes                                                   26 non-null     object
 3   Country code                                            265 non-null    float64
 4   Type of data (a)                                        232 non-null    object
 5   Estimated refugee stock at mid-year (both sexes)        266 non-null    object
 6   Unnamed: 6                                              266 non-null    object
 7   Unnamed: 7                                              266 non-null    object
 8   Unnamed: 8                                              266 non-null    object
 9   Unnamed: 9                                              266 non-null    int64
 10  Unnamed: 10                                             266 non-null    int64
 11  Refugees as a percentage of the international migrant stock  266 non-null  object
 12  Unnamed: 12                                             266 non-null    object
 13  Unnamed: 13                                             266 non-null    object
 14  Unnamed: 14                                             266 non-null    object
 15  Unnamed: 15                                             266 non-null    float64
 16  Unnamed: 16                                             266 non-null    float64
 17  Annual rate of change of the refugee stock              266 non-null    object
 18  Unnamed: 18                                             266 non-null    object
 19  Unnamed: 19                                             266 non-null    object
 20  Unnamed: 20                                             266 non-null    object
 21  Unnamed: 21                                             266 non-null    object
dtypes: float64(4), int64(2), object(16)
memory usage: 45.8+ KB
```

Dataframe info is checked using .info()

Step 3: Renaming the column name

```
[ ] df61.rename(columns = {'Estimated refugee stock at mid-year (both sexes)':'Estimated refugee stock at mid-year (both sexes)(1990)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 6':'Estimated refugee stock at mid-year (both sexes)(1995)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 7':'Estimated refugee stock at mid-year (both sexes)(2000)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 8':'Estimated refugee stock at mid-year (both sexes)(2005)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 9':'Estimated refugee stock at mid-year (both sexes)(2010)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 10':'Estimated refugee stock at mid-year (both sexes)(2015)'}, inplace = True)


    df61.rename(columns = {'Refugees as a percentage of the international migrant stock':'Refugees as a percentage of the international migrant stock(1990)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 12':'Refugees as a percentage of the international migrant stock(1995)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 13':'Refugees as a percentage of the international migrant stock(2000)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 14':'Refugees as a percentage of the international migrant stock(2005)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 15':'Refugees as a percentage of the international migrant stock(2010)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 16':'Refugees as a percentage of the international migrant stock(2015)'}, inplace = True)


    df61.rename(columns = {'Annual rate of change of the refugee stock':'Annual rate of change of the refugee stock(1990-1995)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 18':'Annual rate of change of the refugee stock(1995-2000)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 19':'Annual rate of change of the refugee stock(2000-2005)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 20':'Annual rate of change of the refugee stock(2005-2010)'}, inplace = True)
    df61.rename(columns = {'Unnamed: 21':'Annual rate of change of the refugee stock(2010-2015)'}, inplace = True)


    df61.rename(columns = {'Sort\norder':'Sort_order'}, inplace = True)
    df61.rename(columns = {'Country code':'Country_code'}, inplace = True)
```

Renaming of the columns has been done

Step 4: Reading the data

```
[ ] df61.head(5)
```

	Sort_order	Major area, region, country or area of destination	Notes	Country_code	Type of data (a)	Estimated refugee stock at mid-year (both sexes) (1990)	Estimated refugee stock at mid-year (both sexes) (1995)	Estimated refugee stock at mid-year (both sexes) (2000)	Estimated refugee stock at mid-year (both sexes) (2005)	Estimated refugee stock at mid-year (both sexes) (2010)	...	Refugees as a percentage of the international migrant stock(1995)	Refugees as a percentage of the international migrant stock(2000)	Refugees as a percentage of the international migrant stock(2005)	Refugees as a percentage of the international migrant stock(2010)	F p i
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	

5 rows × 22 columns

The data are read using .head()

Step 5: Dropping null values and unwanted value

```
[ ] df62 = df61.dropna()
```

```
[ ] df63 = df62.replace('\..','0',regex=True)
    df63.head(5)
```

	Sort_order	Major area, region, country or area of destination	Notes	Country_code	Type of data (a)	Estimated refugee stock at mid-year (both sexes) (1990)	Estimated refugee stock at mid-year (both sexes) (1995)	Estimated refugee stock at mid-year (both sexes) (2000)	Estimated refugee stock at mid-year (both sexes) (2005)	Estimated refugee stock at mid-year (both sexes) (2010)	...	Refugees as a percentage of the international migrant stock(1995)	Refugees as a percentage of the international migrant stock(2000)	Refugees as a percentage of the international migrant stock(2005)	Refugees as a percentage of the international migrant stock(2010)
17	17.0	Mauritius	(1)	480.0	C	0	0	0	0	0	...	0	0	0	0.000000
26	26.0	United Republic of Tanzania	(2)	834.0	B R	265184	829671	680862	548824	109286	...	75.012545	73.354522	71.197619	35.413480
44	44.0	Sudan	(3)	729.0	B R	1031050	674071	414928	147256	144008	...	63.990275	51.744207	27.169304	24.899241
67	67.0	Saint Helena	(4)	654.0	B	0	0	0	0	0	...	0	0	0	0.000000
79	79.0	China	(5)	156.0	C	285788	289747	293705	297346	300986	...	65.524267	57.812076	43.795171	35.415909

5 rows × 22 columns

Null values are dropped using .dropna() and "..” value are replaced with 0 using regex() function

Step 6: Converting categorical variable into dummy

```
table61 = pd.get_dummies(df63)
table61
```

	Sort_order	Country_code	Estimated refugee stock at mid-year (both sexes) (2010)	Estimated refugee stock at mid-year (both sexes) (2015)	Refugees as a percentage of the international migrant stock(2010)	Refugees as a percentage of the international migrant stock(2015)	Major area, region, country or area of destination_Australia	Major area, region, country or area of destination_Azerbaijan	Major area, region, country or area of destination_Bonaire, Sint Eustatius and Saba	Major area, region, country or area of destination_Channel Islands
17	17.0	480.0	0	0	0.000000	0.000000	0	0	0	0
26	26.0	834.0	109286	90650	35.413480	34.702284	0	0	0	0
44	44.0	729.0	144008	205174	24.899241	40.751415	0	0	0	0
67	67.0	654.0	0	0	0.000000	0.000000	0	0	0	0
79	79.0	156.0	300986	301052	35.415909	30.780965	0	0	0	0
80	80.0	344.0	154	184	0.005540	0.006482	0	0	0	0
81	81.0	446.0	0	0	0.000000	0.000000	0	0	0	0
91	91.0	458.0	81516	98207	3.388014	3.906027	0	0	0	0
110	110.0	31.0	1908	1380	0.689055	0.522251	0	1	0	0
112	112.0	196.0	3394	4281	1.806059	2.182324	0	0	0	0
113	113.0	268.0	639	857	0.350710	0.507695	0	0	0	0
122	122.0	275.0	1955469	2051096	757.839725	802.755306	0	0	0	0
134	134.0	498.0	148	283	0.093868	0.198035	0	0	0	0
140	140.0	830.0	0	0	0.000000	0.000000	0	0	0	1

Dummy data is created using .get_dummies() to converting categorical variable into dummy

Step 7: Checking the data frame information

```
[ ] df63.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 22 columns):
 #   Column                                                          Non-Null Count  Dtype
---  ------                                                          --------------  -----
 0   Sort_order                                                      22 non-null     float64
 1   Major area, region, country or area of destination              22 non-null     object
 2   Notes                                                           22 non-null     object
 3   Country_code                                                    22 non-null     float64
 4   Type of data (a)                                                22 non-null     object
 5   Estimated refugee stock at mid-year (both sexes)(1990)          22 non-null     object
 6   Estimated refugee stock at mid-year (both sexes)(1995)          22 non-null     object
 7   Estimated refugee stock at mid-year (both sexes)(2000)          22 non-null     object
 8   Estimated refugee stock at mid-year (both sexes)(2005)          22 non-null     object
 9   Estimated refugee stock at mid-year (both sexes)(2010)          22 non-null     int64
 10  Estimated refugee stock at mid-year (both sexes)(2015)          22 non-null     int64
 11  Refugees as a percentage of the international migrant stock(1990)  22 non-null  object
 12  Refugees as a percentage of the international migrant stock(1995)  22 non-null  object
 13  Refugees as a percentage of the international migrant stock(2000)  22 non-null  object
 14  Refugees as a percentage of the international migrant stock(2005)  22 non-null  object
 15  Refugees as a percentage of the international migrant stock(2010)  22 non-null  float64
 16  Refugees as a percentage of the international migrant stock(2015)  22 non-null  float64
 17  Annual rate of change of the refugee stock(1990-1995)           22 non-null     object
 18  Annual rate of change of the refugee stock(1995-2000)           22 non-null     object
 19  Annual rate of change of the refugee stock(2000-2005)           22 non-null     object
 20  Annual rate of change of the refugee stock(2005-2010)           22 non-null     object
 21  Annual rate of change of the refugee stock(2010-2015)           22 non-null     object
dtypes: float64(4), int64(2), object(16)
memory usage: 4.0+ KB
```

The dataframe information is checked to understand the renaming of the columns

Step 8: Renaming the column name

```
df63.rename(columns = {'Estimated refugee stock at mid-year (both sexes)(2010)':'Estimated_refugee_stock_at_mid_year_both_sexes_2010'}, inplace = True)

[ ] df63.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 17 to 240
Data columns (total 22 columns):
 #   Column                                                          Non-Null Count  Dtype
---  ------                                                          --------------  -----
 0   Sort_order                                                      22 non-null     float64
 1   Major area, region, country or area of destination              22 non-null     object
 2   Notes                                                           22 non-null     object
 3   Country_code                                                    22 non-null     float64
 4   Type of data (a)                                                22 non-null     object
 5   Estimated refugee stock at mid-year (both sexes)(1990)          22 non-null     object
 6   Estimated refugee stock at mid-year (both sexes)(1995)          22 non-null     object
 7   Estimated refugee stock at mid-year (both sexes)(2000)          22 non-null     object
 8   Estimated refugee stock at mid-year (both sexes)(2005)          22 non-null     object
 9   Estimated_refugee_stock_at_mid_year_both_sexes_2010             22 non-null     int64
 10  Estimated refugee stock at mid-year (both sexes)(2015)          22 non-null     int64
 11  Refugees as a percentage of the international migrant stock(1990)  22 non-null  object
 12  Refugees as a percentage of the international migrant stock(1995)  22 non-null  object
 13  Refugees as a percentage of the international migrant stock(2000)  22 non-null  object
 14  Refugees as a percentage of the international migrant stock(2005)  22 non-null  object
 15  Refugees as a percentage of the international migrant stock(2010)  22 non-null  float64
 16  Refugees as a percentage of the international migrant stock(2015)  22 non-null  float64
 17  Annual rate of change of the refugee stock(1990-1995)           22 non-null     object
 18  Annual rate of change of the refugee stock(1995-2000)           22 non-null     object
 19  Annual rate of change of the refugee stock(2000-2005)           22 non-null     object
 20  Annual rate of change of the refugee stock(2005-2010)           22 non-null     object
 21  Annual rate of change of the refugee stock(2010-2015)           22 non-null     object
dtypes: float64(4), int64(2), object(16)
memory usage: 4.0+ KB
```

The column name is renamed and its information is checked.

Step 9: Creating a series of 10 value of the column and converting it to numeric

```
[ ] ser61 = pd.Series(df63['Estimated_refugee_stock_at_mid_year_both_sexes_2010']).head(5)
```

```
[ ] pd.to_numeric(ser61, downcast ='signed')
```

```
17         0
26    109286
44    144008
67         0
79    300986
Name: Estimated_refugee_stock_at_mid_year_both_sexes_2010, dtype: int32
```

A series has been created using 10 values of the columns and its values are converted to numeric to get accurate results with the data frame.

Step 10: Top level evaluation

```
[ ] print(df63.eval("Sort_order + Estimated_refugee_stock_at_mid_year_both_sexes_2010"))
```

```
17          17.0
26      109312.0
44      144052.0
67          67.0
79      301065.0
80         234.0
81          81.0
91       81607.0
110       2018.0
112       3506.0
113        752.0
122    1955591.0
134        282.0
140        140.0
144       8868.0
150      40410.0
160        160.0
166      73774.0
168       3988.0
187        187.0
195        195.0
240      25745.0
dtype: float64
```

.eval() is used to get the top level evaluation of the dataset.

Step 11: Manipulating the data

```
[ ] df64 = pd.melt(df63, id_vars=['Sort_order'], value_vars=list(df63.columns)[1:],
                   var_name='Country_code', value_name='Estimated_refugee_stock_at_mid_year_both_sexes_2010')
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: This dataframe has a column name that matches the 'value_name' column name of the resulting Dataframe.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
[ ] df64.head(5)
```

	Sort_order	Country_code	Estimated_refugee_stock_at_mid_year_both_sexes_2010
0	17.0	Major area, region, country or area of destina...	Mauritius
1	26.0	Major area, region, country or area of destina...	United Republic of Tanzania
2	44.0	Major area, region, country or area of destina...	Sudan
3	67.0	Major area, region, country or area of destina...	Saint Helena
4	79.0	Major area, region, country or area of destina...	China

The data is manipulated using the .melt() function in which Unpivot a DataFrame from wide to long format, optionally leaving identifiers are set along with reading the dataframe.

Cleaning of Annex

Step 1: Importing Annex table by skipping 14 rows and reading.

```
[ ] df71 = pd.read_excel("UN_MigrantStockTotal_2015.xlsx",'ANNEX',skiprows = 14)
```

```
[ ] df71.head(5)
```

	Country code	Country or area	Sort order	Major area	Code	Sort order.1	Region	Code.1	Sort order.2	Developed region	Least developed country	Sub-Saharan Africa
0	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No
1	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	No	No
2	12	Algeria	40	Africa	903	7	Northern Africa	912	39	No	No	No
3	16	American Samoa	257	Oceania	909	238	Polynesia	957	256	No	No	No
4	20	Andorra	155	Europe	908	127	Southern Europe	925	153	Yes	No	No

Annex table is imported by skipping 14 rows and it is read using the .head() function.

Step 2: Checking the data information

```
df71.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232 entries, 0 to 231
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Country code            232 non-null    int64
 1   Country or area         232 non-null    object
 2   Sort order              232 non-null    int64
 3   Major area              232 non-null    object
 4   Code                    232 non-null    int64
 5   Sort order.1            232 non-null    int64
 6   Region                  232 non-null    object
 7   Code.1                  232 non-null    int64
 8   Sort order.2            232 non-null    int64
 9   Developed region        232 non-null    object
 10  Least developed country 232 non-null    object
 11  Sub-Saharan Africa      232 non-null    object
dtypes: int64(6), object(6)
memory usage: 21.9+ KB
```

Dataframe information is checked to understand the columns to be renamed

Step 3: Renaming the column name

```
df71.rename(columns = {'Country code':'Country_code'}, inplace = True)
df71.rename(columns = {'Sort order':'Sort_order'}, inplace = True)
df71.rename(columns = {'Sort order.1':'Sort_order_1'}, inplace = True)
df71.rename(columns = {'Code.1':'Code_1'}, inplace = True)
df71.rename(columns = {'Sort order.2':'Sort_order_2'}, inplace = True)
```

Renaming of the columns has been done to get accurate insights of the columns.

Step 4: checking the data information

```
[ ] df71.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232 entries, 0 to 231
Data columns (total 12 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Country_code             232 non-null    int64
 1   Country or area          232 non-null    object
 2   Sort_order               232 non-null    int64
 3   Major area               232 non-null    object
 4   Code                     232 non-null    int64
 5   Sort_order_1             232 non-null    int64
 6   Region                   232 non-null    object
 7   Code_1                   232 non-null    int64
 8   Sort_order_2             232 non-null    int64
 9   Developed region         232 non-null    object
 10  Least developed country  232 non-null    object
 11  Sub-Saharan Africa       232 non-null    object
dtypes: int64(6), object(6)
memory usage: 21.9+ KB
```

Information of the data frame has been checked in this stage.

Step 5: Checking null values in the data frame

```
[ ] df71.isna()
```

	Country_code	Country or area	Sort_order	Major area	Code	Sort_order_1	Region	Code_1	Sort_order_2	Developed region	Least developed country	Sub-Saharan Africa
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
227	False	False	False	False	False	False	False	False	False	False	False	False
228	False	False	False	False	False	False	False	False	False	False	False	False
229	False	False	False	False	False	False	False	False	False	False	False	False
230	False	False	False	False	False	False	False	False	False	False	False	False
231	False	False	False	False	False	False	False	False	False	False	False	False

232 rows × 12 columns

Null value in the dataset has been checked so that it can be dropped.

Step 6: Dropping null values from the data frame and reading the data

```
[ ] df72 = df71.dropna()
```

```
[ ] df72
```

	Country_code	Country or area	Sort_order	Major area	Code	Sort_order_1	Region	Code_1	Sort_order_2	Developed region	Least developed country	Sub-Saharan Africa
0	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No
1	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	No	No
2	12	Algeria	40	Africa	903	7	Northern Africa	912	39	No	No	No
3	16	American Samoa	257	Oceania	909	238	Polynesia	957	256	No	No	No
4	20	Andorra	155	Europe	908	127	Southern Europe	925	153	Yes	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...
227	876	Wallis and Futuna Islands	265	Oceania	909	238	Polynesia	957	256	No	No	No
228	732	Western Sahara	46	Africa	903	7	Northern Africa	912	39	No	No	No
229	887	Yemen	126	Asia	935	71	Western Asia	922	108	No	Yes	No
230	894	Zambia	27	Africa	903	7	Eastern Africa	910	8	No	Yes	Yes
231	716	Zimbabwe	28	Africa	903	7	Eastern Africa	910	8	No	No	Yes

232 rows × 12 columns

Null value in the data frame has been dropped so that the accuracy of the data can be increased.

Step 7: Replacing unwanted value with zero using regex function

```
[ ] df73 = df72.replace('\..','0',regex=True)
    df73.head(5)
```

	Country_code	Country or area	Sort_order	Major area	Code	Sort_order_1	Region	Code_1	Sort_order_2	Developed region	Least developed country	Sub-Saharan Africa
0	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	Yes	No
1	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	No	No
2	12	Algeria	40	Africa	903	7	Northern Africa	912	39	No	No	No
3	16	American Samoa	257	Oceania	909	238	Polynesia	957	256	No	No	No
4	20	Andorra	155	Europe	908	127	Southern Europe	925	153	Yes	No	No

The "..” value has been replaced with 0 using the regex function to get accurate data in the dataframe.

Step 8: To get top level evaluation

```
[ ] print(df73.eval("Country_code + Code_1"))

    0      5505
    1       933
    2       924
    3       973
    4       945
           ...
    227    1833
    228    1644
    229    1809
    230    1804
    231    1626
    Length: 232, dtype: int64
```

Top level evaluation is done in the system using .eval() function

Step 9: Data manipulation

```
[ ] df74 = pd.melt(df73, id_vars=['Sort_order_1'], value_vars=list(df73.columns)[1:],
                   var_name='Sort_order_2', value_name='Code_1')
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: This dataframe has a column name that matches the 'value_name' column name of the resulting Dataframe.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
[ ] df74.head(8)
```

	Sort_order_1	Sort_order_2	Code_1
0	71	Country or area	Afghanistan
1	127	Country or area	Albania
2	7	Country or area	Algeria
3	238	Country or area	American Samoa
4	127	Country or area	Andorra
5	7	Country or area	Angola
6	180	Country or area	Anguilla
7	180	Country or area	Antigua and Barbuda

Data manipulation is done using the melt function.

Step 10: Map function to apply a function to each item to get iterable value of the data

```
[ ] df73['Least developed country'] = df73['Least developed country'].map({'Yes': 1, 'No': 0})
    df73
```

	Country_code	Country or area	Sort_order	Major area	Code	Sort_order_1	Region	Code_1	Sort_order_2	Developed region	Least developed country	Sub-Saharan Africa
0	4	Afghanistan	99	Asia	935	71	Southern Asia	5501	98	No	1	No
1	8	Albania	154	Europe	908	127	Southern Europe	925	153	Yes	0	No
2	12	Algeria	40	Africa	903	7	Northern Africa	912	39	No	0	No
3	16	American Samoa	257	Oceania	909	238	Polynesia	957	256	No	0	No
4	20	Andorra	155	Europe	908	127	Southern Europe	925	153	Yes	0	No
...	...	...	...	...	...	...	...	...	...	...	...	...
227	876	Wallis and Futuna Islands	265	Oceania	909	238	Polynesia	957	256	No	0	No
228	732	Western Sahara	46	Africa	903	7	Northern Africa	912	39	No	0	No
229	887	Yemen	126	Asia	935	71	Western Asia	922	108	No	1	No
230	894	Zambia	27	Africa	903	7	Eastern Africa	910	8	No	1	Yes
231	716	Zimbabwe	28	Africa	903	7	Eastern Africa	910	8	No	0	Yes

232 rows × 12 columns

Map function is used to input a list and to get iterable value of the data

Step 11: Getting unique value based on hash table and print it

```
[ ] df76 = df73['Major area'].unique()
```

```
[ ] print(df76)
```

```
['Asia' 'Europe' 'Africa' 'Oceania' 'Latin America and the Caribbean'
 'Northern America']
```

This step consists of getting the union value based on the hash table and getting the value of it.