Name: Shuang Liang
Student number: 1004190173
INF1340 Final Project Write up

Data Visualization on UN dataset

Introduction

In the world of Big Data, data visualization is essential to analyze massive amounts of information and making data-driven decisions. Data visualization is the process of transforming raw data into visual context, such as charts, graphs and so on. The dataset we would like to analyze for our final project is "Trends in International Migrant Stock: The 2015 Version" by United Nations. The dataset is in excel format and it contains 6 tables with a contents sheet, an annex sheet, and a notes sheet. I will mainly focus on interpreting the first five datasets by identifying the trends, patterns and outliers which is the main goal of data visualization. Besides, I will be using three different libraries, Matplotlib, Pandas Visualization and Plotly in the data visualization process. Furthermore, I will follow Tufte's principles of data visualization, which are Graphical Integrity, Data-Ink, Chartjunk, Data Density and Small Multiplies during the whole process.

Methods and Results

Data Visualization for Table1:

1. First, I import some libraries we use for data visualization, and we get the table1 ready from the data cleaning process.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
```

```
table1
```

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

4725 rows × 7 columns

2. I use pandas.DataFrame.pivot function to reshape the DataFrame by column values for moving the object under column "Gender" to column headers.

In [2958]:
```
table1.pivot(index = ['Sort order', 'Major area, region, country or area of destination', 'Country code', 'Type of d
new1 = table1.pivot(index = ['Sort order', 'Major area, region, country or area of destination', 'Country code', 'Ty
new1
```

Out[2958]:

Gender	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Both sexes	Female	Male
0	1	WORLD	900	NaN	1990	152563212.0	74815702.0	77747510.0
1	1	WORLD	900	NaN	1995	160801752.0	79064275.0	81737477.0
2	1	WORLD	900	NaN	2000	172703309.0	84818470.0	87884839.0
3	1	WORLD	900	NaN	2005	191269100.0	93402426.0	97866674.0
4	1	WORLD	900	NaN	2010	221714243.0	107100529.0	114613714.0
...	...	...	...	...	...	...	...	...
1570	265	Wallis and Futuna Islands	876	B	1995	1680.0	821.0	859.0
1571	265	Wallis and Futuna Islands	876	B	2000	2015.0	997.0	1018.0
1572	265	Wallis and Futuna Islands	876	B	2005	2365.0	1171.0	1194.0
1573	265	Wallis and Futuna Islands	876	B	2010	2776.0	1375.0	1401.0
1574	265	Wallis and Futuna Islands	876	B	2015	2849.0	1411.0	1438.0

1575 rows × 8 columns

3. Since the DataFrame index name is showing "Gender" after doing the pivot, I use DataFrame.index.name function to change it to an empty string.
Also, I use pandas.DataFrame.drop to drop the specific column "Both sexes" since I only need "Female" and "Male" for my data visualization.

In [2790]:
```
new1.columns.names = ['']
```

In [2791]:
```
new1 = new1.drop(labels = 'Both sexes', axis=1)
new1
```

Out[2791]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female	Male
0	1	WORLD	900	NaN	1990	74815702.0	77747510.0
1	1	WORLD	900	NaN	1995	79064275.0	81737477.0
2	1	WORLD	900	NaN	2000	84818470.0	87884839.0
3	1	WORLD	900	NaN	2005	93402426.0	97866674.0
4	1	WORLD	900	NaN	2010	107100529.0	114613714.0
...	...	...	...	...	...	...	...
1570	265	Wallis and Futuna Islands	876	B	1995	821.0	859.0
1571	265	Wallis and Futuna Islands	876	B	2000	997.0	1018.0
1572	265	Wallis and Futuna Islands	876	B	2005	1171.0	1194.0
1573	265	Wallis and Futuna Islands	876	B	2010	1375.0	1401.0
1574	265	Wallis and Futuna Islands	876	B	2015	1411.0	1438.0

1575 rows × 7 columns

4. I use pandas.DataFrame.loc to select the data of only "2015" under column "Year".

In [2961]:
```
new1 = new1.loc[new1['Year'] == 2015]
new1
```

Out[2961]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female	Male
5	1	WORLD	900	NaN	2015	117584801.0	126115435.0
11	2	Developed regions	901	NaN	2015	72863336.0	67618619.0
17	3	Developing regions	902	NaN	2015	44721465.0	58496816.0
23	4	Least developed countries	941	NaN	2015	5493028.0	6463217.0
29	5	Less developed regions excluding least develop...	934	NaN	2015	39228437.0	52033599.0
...	...	...	...	...	...	...	...
1550	261	Samoa	882	B	2015	2460.0	2469.0
1556	262	Tokelau	772	B	2015	254.0	233.0
1562	263	Tonga	776	B	2015	2604.0	3127.0
1568	264	Tuvalu	798	C	2015	63.0	78.0
1574	265	Wallis and Futuna Islands	876	B	2015	1411.0	1438.0

265 rows × 7 columns

And I would like to select the major regions by using pandas.loc and pandas.isin().

In [2962]:
```
# selecting the major region
new1 = new1.loc[new1['Sort order'].isin([7, 71, 127, 180, 232, 238])]
new1
```

Out[2962]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female	Male
41	7	Africa	903	NaN	2015	9526134.0	11123423.0
421	71	Asia	935	NaN	2015	31530709.0	43550416.0
757	127	Europe	908	NaN	2015	39873338.0	36272616.0
1071	180	Latin America and the Caribbean	904	NaN	2015	4650938.0	4583051.0
1376	232	Northern America	905	NaN	2015	27902348.0	26586377.0
1412	238	Oceania	909	NaN	2015	4101334.0	3999552.0

5. I use pandas.DataFrame.sort_values for sorting the rows and columns in ascending order, and we assign a new name "top_regions" to our DataFrame after sorting.

In [2963]:
```
#sorting rows and columns
new1.sort_values('Female', ascending = True)
```

Out[2963]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female	Male
1412	238	Oceania	909	NaN	2015	4101334.0	3999552.0
1071	180	Latin America and the Caribbean	904	NaN	2015	4650938.0	4583051.0
41	7	Africa	903	NaN	2015	9526134.0	11123423.0
1376	232	Northern America	905	NaN	2015	27902348.0	26586377.0
421	71	Asia	935	NaN	2015	31530709.0	43550416.0
757	127	Europe	908	NaN	2015	39873338.0	36272616.0

In [3182]:
```
#sorting rows and columns
top_regions = new1.sort_values('Female', ascending = True)
top_regions
```

Out[3182]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female	Male
1412	238	Oceania	909	NaN	2015	4101334.0	3999552.0
1071	180	Latin America and the Caribbean	904	NaN	2015	4650938.0	4583051.0
41	7	Africa	903	NaN	2015	9526134.0	11123423.0
1376	232	Northern America	905	NaN	2015	27902348.0	26586377.0
421	71	Asia	935	NaN	2015	31530709.0	43550416.0
757	127	Europe	908	NaN	2015	39873338.0	36272616.0

6. We need to reset the index to make it orderly by using

pandas.DataFrame.reset_index and the drop parameter is to avoid the old index being added as a column, so I set it to drop = True.

```
In [3183]: top_regions = top_regions.reset_index(drop = True)
           top_regions
```

Out[3183]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female	Male
0	238	Oceania	909	NaN	2015	4101334.0	3999552.0
1	180	Latin America and the Caribbean	904	NaN	2015	4650938.0	4583051.0
2	7	Africa	903	NaN	2015	9526134.0	11123423.0
3	232	Northern America	905	NaN	2015	27902348.0	26586377.0
4	71	Asia	935	NaN	2015	31530709.0	43550416.0
5	127	Europe	908	NaN	2015	39873338.0	36272616.0

7. I use pandas.DayaFrame.drop function to drop the columns we don't need for data visualization. Finally, we get our table ready for the data visualization process.

```
In [3184]: top_regions = top_regions.drop(['Sort order', 'Country code', 'Type of data (a)'], axis = 1)
           top_regions
```

Out[3184]:

	Major area, region, country or area of destination	Year	Female	Male
0	Oceania	2015	4101334.0	3999552.0
1	Latin America and the Caribbean	2015	4650938.0	4583051.0
2	Africa	2015	9526134.0	11123423.0
3	Northern America	2015	27902348.0	26586377.0
4	Asia	2015	31530709.0	43550416.0
5	Europe	2015	39873338.0	36272616.0

8. Before we dive into the data visualization process, we should start with a basic look at the details of DataFrame.
By using pandas.DataFrame.describe function, we will see how the data is distributed, size, minimum, maximum, quantiles and mean. We get the descriptive statistics about the international stock at mid-year respectively of female and male. And using pandas.DataFrame.info, we will see what kind of data each column includes.

```
In [3185]: top_regions.describe()
```

Out[3185]:

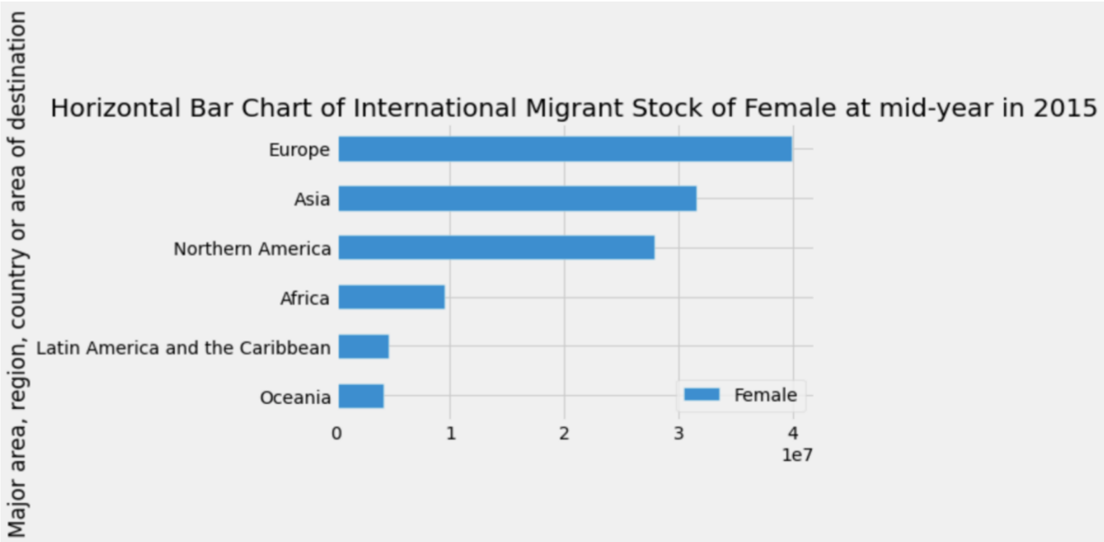
	Year	Female	Male
count	6.0	6.000000e+00	6.000000e+00
mean	2015.0	1.959747e+07	2.101924e+07
std	0.0	1.541068e+07	1.690596e+07
min	2015.0	4.101334e+06	3.999552e+06
25%	2015.0	5.869737e+06	6.218144e+06
50%	2015.0	1.871424e+07	1.885490e+07
75%	2015.0	3.062362e+07	3.385106e+07
max	2015.0	3.987334e+07	4.355042e+07

```
In [3186]: top_regions.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column                                              Non-Null Count  Dtype
---  ------                                              --------------  -----
 0   Major area, region, country or area of destination  6 non-null      object
 1   Year                                                6 non-null      int64
 2   Female                                              6 non-null      float64
 3   Male                                                6 non-null      float64
dtypes: float64(2), int64(1), object(1)
memory usage: 320.0+ bytes
```

9. The first plot we have is the horizontal bar plot using the pandas visualization,

pandas.DataFrame.plot.barh function with the parameter x "Major area, region, country or area of destination" and the parameter y "Female".

<u>Plot #1: Horizontal bar plot --- Horizontal Bar Chart of International Migrant Stock of Female at mid-year in 2015</u>

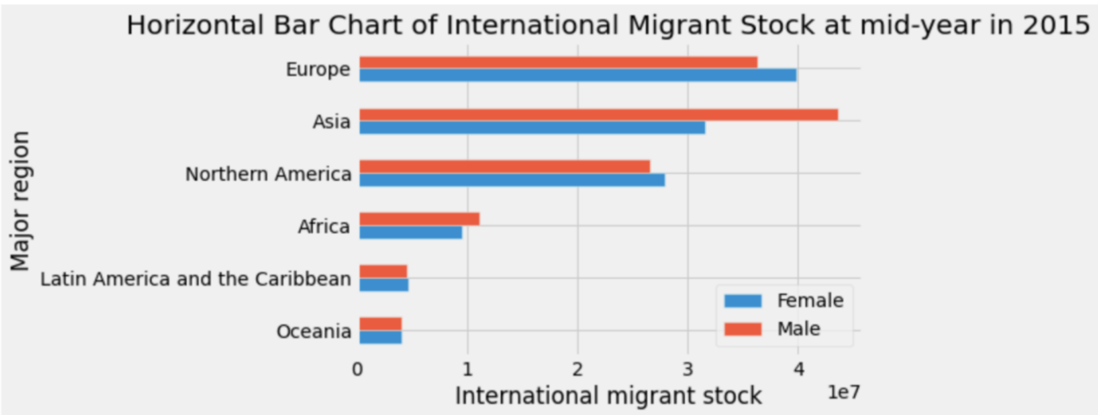


The horizontal bar plot represents the quantitative data horizontally with rectangular bars with lengths proportional to the values that they present, and it shows comparisons among discrete categories which are the major regions. The vertical axis shows the specific categories of major regions which are Europe, Asia, North America, Africa, Latin America and the Caribbean, Oceania, and the horizontal axis represents the measured values of the international migrant stock. The scale shows the value of 1 unit on the horizontal axis, it represents 1 million of international migrant stock. As we can see above, Europe is the region which has the highest international migrant stock since it has the longest bar, and it is almost reached 4 million. Oceania is the region which has the lowest international migrant stock since it has the shortest bar, and it is around 0.4 million.

During this process of graph making, I kept the graph correct and accurate to make sure that the graph told all the truth and avoid lie factors to keep the Graphic Integrity principle. All the graphs I make in this file all keep the same standard and method to make sure that the visualization obeys the Graphic Integrity. I keep the graph simple with a single color to make sure that nothing else would mislead the audience so that the visualization would keep a high Data-Ink ratio. In the rest of my graphs in this file, I would keep doing the same to obey the Data-Ink principle, there exist some expectations in Plot #2,3,4,5,6 that I used multi-colors, that was because I need to do the comparison and all the colors are labelled and they would not misguide the audience and make the graph too fancy to read, they all keep the highest Data-Ink ratio I could make. I used simple two-dimensional axes with clear labels on them, simple grids to show the graph clearly, and simple names to show the purpose of the graph. All of these are to make sure that I follow the Chartjunk principle to avoid useless information on the graph interfering with the audience. All my rest graphs follow this principle as well. The graph is made with high density which means that the data are shown clearly, and the analysis could be made easily by the graph, the concepts were in the graph and can be read directly. All my rest graphs follow this rule to make sure that I follow the Data Density principle.

10. Furthermore, we could plot a horizontal bar chart grouped by gender so that we could compare the international migrant stock by gender. We use the same function as the first plot, pandas.DataFrame.plot.barh function

Plot #2: Horizontal grouped bar plot --- Horizontal Bar Chart of International Migrant Stock at mid-year in 2015



The horizontal grouped bar chart is similar to the horizontal bar chart, it plots numeric values for levels of two categorical variables (Female and Male) instead of one. The bars are grouped by position for levels of one categorical variable, with color indicating the secondary category level within each region. The vertical axis shows the specific categories of major regions are essentially groups of female and male, and the categories are Europe, Asia, North America, Africa, Latin America and the Caribbean, Oceania. The horizontal axis represents the measured values of the international migrant stock. Also, the legend indicates the gender by different colors. For the region of Europe, female is higher in international migrant stock than male. For the region of Asia, male has way much larger proportion than male. Besides, there are approximately similar proportions by gender in the regions of Latin America and the Caribbean, and Oceania.

Data Visualization for Table2:

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

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
```

```
In [1404]: new2 = pd.read_excel('/Users/liangshuang/Desktop/UN_MigrantStockTotal_2015.xlsx', sheet_name = 2)
new2
```

Out[1404]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unnamed: 13	Unnamed: 14
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	United Nations	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	Population Division	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
275	261	Samoa	NaN	882	162.865	170.158	174.614	179.928	186.029	193.228	...	90.932	93.185	95.95
276	262	Tokelau	NaN	772	1.609	1.520	1.552	1.210	1.135	1.250	...	..	..	..
277	263	Tonga	NaN	776	95.152	95.889	97.898	100.858	103.947	106.170	...	49.788	50.574	52.055
278	264	Tuvalu	NaN	798	9.004	9.227	9.419	9.694	9.827	9.916	...	..	..	..
279	265	Wallis and Futuna Islands	NaN	876	13.88	14.143	14.497	14.246	13.565	13.151	...	..	..	..

280 rows × 22 columns

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

```
In [1405]: new2 = new2.drop(labels = range(0, 15), axis = 0)
new2
```

Out[1405]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unna
15	1	WORLD	NaN	900	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082
16	2	Developed regions	(b)	901	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962
17	3	Developing regions	(c)	902	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	269712
18	4	Least developed countries	(d)	941	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	...	331482.475	375757
19	5	Less developed regions excluding least develop...	NaN	934	3655147.008	3980172.519	4273424.303	4557911.390	4849094.485	5143963.209	...	2175044.969	2321362
...	...	...	...	...	...	...	...	...	...	...	...	...	
275	261	Samoa	NaN	882	162.865	170.158	174.614	179.928	186.029	193.228	...	90.932	93
276	262	Tokelau	NaN	772	1.609	1.520	1.552	1.210	1.135	1.250	...	..	
277	263	Tonga	NaN	776	95.152	95.889	97.898	100.858	103.947	106.170	...	49.788	50
278	264	Tuvalu	NaN	798	9.004	9.227	9.419	9.694	9.827	9.916	...	..	
279	265	Wallis and Futuna Islands	NaN	876	13.88	14.143	14.497	14.246	13.565	13.151	...	..	

265 rows × 22 columns

3. Since the index starts from 15 and it is wrong, we want to change it starts from 0. I use pandas.DataFrame.reset_index function, the drop parameter is to avoid the old index being added as a column, so I set it to drop = True. And I take a look at the first 20 rows since I only want to extract the row which contains Africa in the

following step.

In [1406]:
```
new2 = new2.reset_index(drop = True)
new2.head(20)
```

Out[1406]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 12	Unna
0	1	WORLD	NaN	900	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	328508
1	2	Developed regions	(b)	901	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	58796
2	3	Developing regions	(c)	902	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	269712
3	4	Least developed countries	(d)	941	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	...	331482.475	37575
4	5	Less developed regions excluding least develop...	NaN	934	3655147.008	3980172.519	4273424.303	4557911.390	4849094.485	5143963.209	...	2175044.969	232136
5	6	Sub-Saharan Africa	(e)	947	491497.691	562978.224	642172.298	733321.659	840390.129	962286.754	...	319998.713	36572
6	7	Africa	NaN	903	631614.304	720416.386	814063.149	920238.945	1044106.862	1186178.282	...	406405.564	45959
7	8	Eastern Africa	NaN	910	198231.687	225309.503	259372.541	297636.467	342742.625	394477.339	...	128612.117	14772
8	9	Burundi	NaN	108	5613.141	6239.030	6767.073	7934.213	9461.117	11178.921	...	3336.457	392
9	10	Comoros	NaN	174	415.144	479.574	547.696	618.632	698.695	788.474	...	275.584	31
10	11	Djibouti	NaN	262	588.356	661.076	722.562	778.406	830.802	887.861	...	363.204	39
11	12	Eritrea	NaN	232	3139.083	3164.095	3535.156	4191.273	4689.664	5227.791	...	1758.329	209
12	13	Ethiopia	NaN	231	48057.094	57237.226	66443.603	76608.431	87561.814	99390.750	...	33129.426	3821
13	14	Kenya	NaN	404	23446.229	27373.035	31065.820	35349.040	40328.313	46050.302	...	15487.89	176
14	15	Madagascar	NaN	450	11545.782	13452.526	15744.811	18290.394	21079.532	24235.390	...	7839.327	910
15	16	Malawi	NaN	454	9408.998	9822.812	11193.230	12747.846	14769.824	17215.232	...	5550.543	634
16	17	Mauritius	(1)	480	1055.865	1128.676	1185.143	1222.006	1247.951	1273.212	...	587.689	60
17	18	Mayotte	NaN	175	94.78	123.182	150.329	178.103	208.723	240.015	...	76.058	8
18	19	Mozambique	NaN	508	13371.971	15913.101	18264.536	21126.676	24321.457	27977.863	...	8766.267	1021
19	20	Réunion	NaN	638	610.582	673.542	736.711	791.602	830.516	861.154	...	359.525	38

20 rows × 22 columns

4. Then we need to define the unnamed columns before we select specific rows or columns.

In [1407]:
```
new2.columns = ["Sort order", "Major area, region, country or area of destination", "Notes", "Country code", "1990B"
new2
```

Out[1407]:

	Sort order	Major area, region, country or area of destination	Notes	Country code	1990B	1995B	2000B	2005B	2010B	2015B	...	2000M	2005M	2
0	1	WORLD	NaN	900	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	...	3084537.662	3285082.249	349395
1	2	Developed regions	(b)	901	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	...	578010.218	587962.213	59995
2	3	Developing regions	(c)	902	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	...	2506527.444	2697120.036	289400
3	4	Least developed countries	(d)	941	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	...	331482.475	375757.715	42239
4	5	Less developed regions excluding least develop...	NaN	934	3655147.008	3980172.519	4273424.303	4557911.390	4849094.485	5143963.209	...	2175044.969	2321362.321	247160
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
260	261	Samoa	NaN	882	162.865	170.158	174.614	179.928	186.029	193.228	...	90.932	93.185	
261	262	Tokelau	NaN	772	1.609	1.520	1.552	1.210	1.135	1.250	...	..	..	
262	263	Tonga	NaN	776	95.152	95.889	97.898	100.858	103.947	106.170	...	49.788	50.574	5
263	264	Tuvalu	NaN	798	9.004	9.227	9.419	9.694	9.827	9.916	...	..	..	
264	265	Wallis and Futuna Islands	NaN	876	13.88	14.143	14.497	14.246	13.565	13.151	...	..	..	

265 rows × 22 columns

5. Now, we can select the rows and columns we want for data visualization. So, I use pandas.DataFrame.loc to access a group of rows and columns by labels. Then we redefine the name of rows and columns to make it more clear. Finally, we get our

table ready for the data visualization process.

In [1408]:
```
rows = [1, 2]
columns = ['2005B', '2010B', '2015B']
new2 = new2.loc[rows, columns]
new2
```

Out[1408]:

	2005B	2010B	2015B
1	1208919.509	1233375.711	1251351.086
2	5310716.341	5696349.332	6098121.013

In [1409]:
```
new2.columns = ['Population_2005', 'Population_2010', 'Population_2015']
new2.index = ['Developed regions', 'Developing regions']
new2
```

Out[1409]:

	Population_2005	Population_2010	Population_2015
Developed regions	1208919.509	1233375.711	1251351.086
Developing regions	5310716.341	5696349.332	6098121.013

6. Similarly to table1, we should start with a basic look at the details of DataFrame.

In [1410]:
```
new2.describe()
```

Out[1410]:

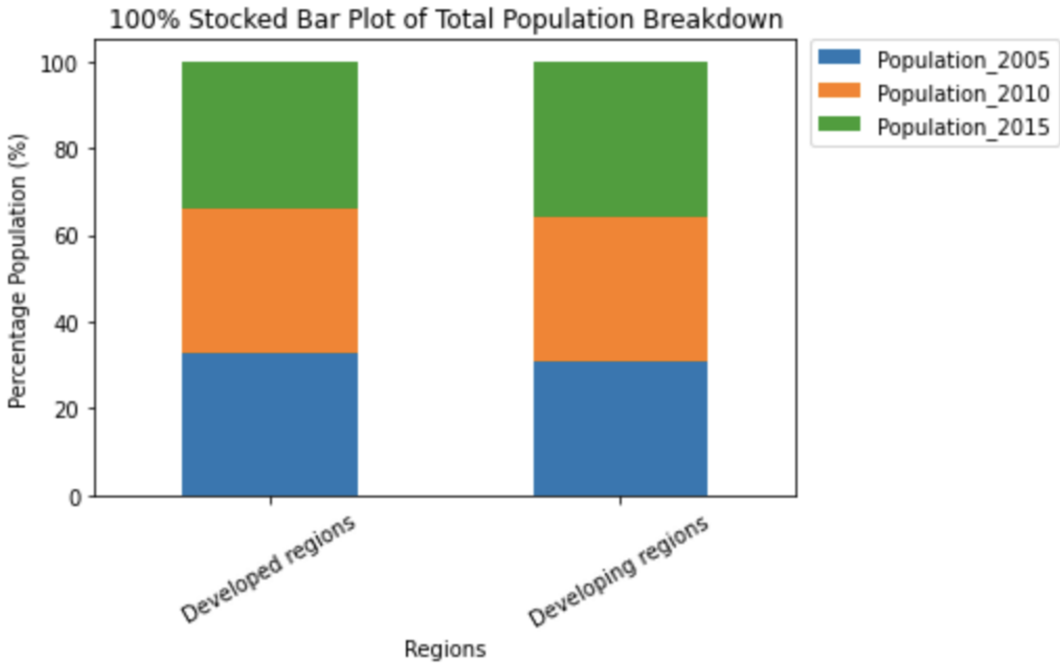
	Population_2005	Population_2010	Population_2015
count	2.000000e+00	2.000000e+00	2.000000e+00
mean	3.259818e+06	3.464863e+06	3.674736e+06
std	2.900408e+06	3.155799e+06	3.427184e+06
min	1.208920e+06	1.233376e+06	1.251351e+06
25%	2.234369e+06	2.349119e+06	2.463044e+06
50%	3.259818e+06	3.464863e+06	3.674736e+06
75%	4.285267e+06	4.580606e+06	4.886429e+06
max	5.310716e+06	5.696349e+06	6.098121e+06

In [1411]:
```
new2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2 entries, Developed regions to Developing regions
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Population_2005  2 non-null      float64
 1   Population_2010  2 non-null      float64
 2   Population_2015  2 non-null      float64
dtypes: float64(3)
memory usage: 64.0+ bytes
```

7. Now, we could plot our third plot which is a 100% stacked bar plot by using the matplotlib.pyplot.plot. And I applied a lambda function by using pandas.DataFrame.apply to change the values on the y-axis to percentage, because the y-axis of 100% stacked bar plot is in percentage.

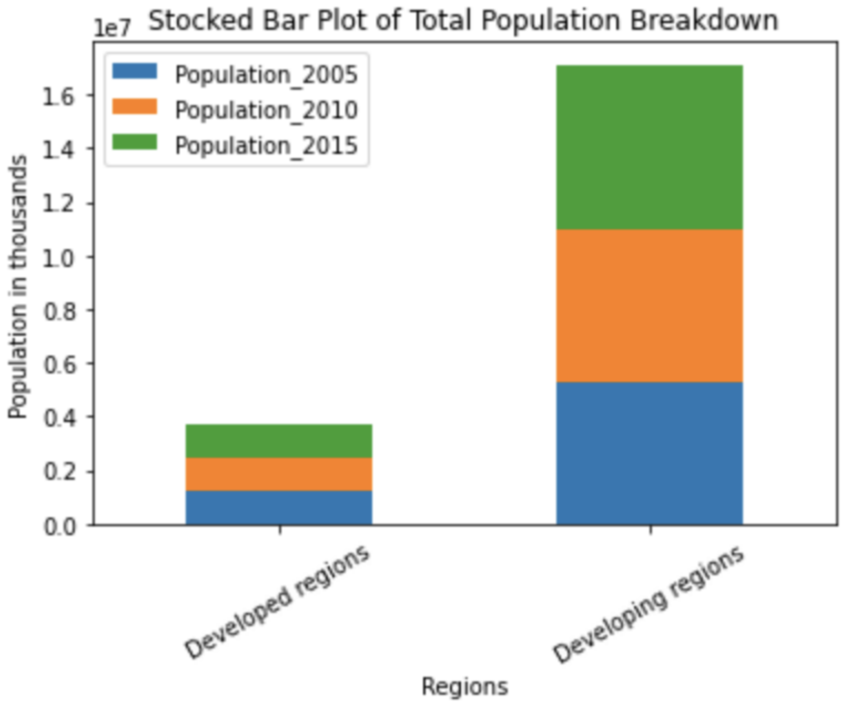
Plot #3: 100% stocked bar plot --- 100% Stocked Bar Plot of Total Population Breakdown



In a 100% stacked bar chart, the bars are split into colored bar segments placed on top of each other. Each bar height is 100%, and the colored bar segments represent the components' relative contributions to the total bar. As the plot above, the proportion of each bar is approximately evenly distributed, and we can barely see the difference in total population by year between developed regions and developing regions since the specific numbers of total populations are close as you can see in the DataFrame "new2".

8. I also create a stacked bar plot that shows two categorical variables by using the same matplotlib function above, matplotlib.pyplot.plot.

Plot #4: Stocked bar plot --- Stocked Bar Plot of Total Population Breakdown



The primary variable is shown along the entire length of the bar which is region: we can see that developing regions have much higher total populations than developed regions. Each bar is subdivided based on levels of the second categorical variable, year (2005, 2010 and 2015). We can see that for both developed regions and developing regions, the total population proportion by year in each bar is approximately evenly distributed.

Data Visualization for Table3:

1. First, I import some libraries we use for data visualization, and we get the table3 ready from the data cleaning process.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
```

```
In [3570]: table3
```

Out[3570]:

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

4343 rows × 7 columns

2. Then, I use pandas.DataFrame.drop to drop the columns that we don't need for data visualization, and I use pandas.DataFrame.loc to get only "Both sexes" under column "Gender".

```
In [3571]: new3 = table3.drop(['Sort order', 'Country code', 'Type of data (a)'], axis = 1)
           new3 = new3.loc[new3['Gender'] == 'Both sexes']
           new3
```

Out[3571]:

	Major area, region, country or area of destination	Year	Gender	International migrant stock as a percentage of the total population
0	WORLD	1990	Both sexes	2.9
1	Developed regions	1990	Both sexes	7.2
2	Developing regions	1990	Both sexes	1.7
3	Least developed countries	1990	Both sexes	2.2
4	Less developed regions excluding least develop...	1990	Both sexes	1.6
...	...	...	...	...
1566	Samoa	2015	Both sexes	2.6
1567	Tokelau	2015	Both sexes	39.0
1568	Tonga	2015	Both sexes	5.4
1569	Tuvalu	2015	Both sexes	1.4
1570	Wallis and Futuna Islands	2015	Both sexes	21.7

1571 rows × 4 columns

3. And I use pandas.DataFrame.loc again to access all the five major regions (Europe, Asia, North America, Africa, Latin America and the Caribbean, Oceania) under the column "Major area, region, country or area of destination".

	Major area, region, country or area of destination	Year	Gender	International migrant stock as a percentage of the total population
6	Africa	1990	Both sexes	2.5
69	Asia	1990	Both sexes	1.5
125	Europe	1990	Both sexes	6.8
177	Latin America and the Caribbean	1990	Both sexes	1.6
232	Oceania	1990	Both sexes	17.5
266	Africa	1995	Both sexes	2.3
329	Asia	1995	Both sexes	1.3
385	Europe	1995	Both sexes	7.3
437	Latin America and the Caribbean	1995	Both sexes	1.4
492	Oceania	1995	Both sexes	17.3
526	Africa	2000	Both sexes	1.8
589	Asia	2000	Both sexes	1.3
645	Europe	2000	Both sexes	7.7
697	Latin America and the Caribbean	2000	Both sexes	1.2
752	Oceania	2000	Both sexes	17.3
786	Africa	2005	Both sexes	1.7
849	Asia	2005	Both sexes	1.4
905	Europe	2005	Both sexes	8.8
957	Latin America and the Caribbean	2005	Both sexes	1.3
1013	Oceania	2005	Both sexes	18.1
1047	Africa	2010	Both sexes	1.6
1111	Asia	2010	Both sexes	1.6
1167	Europe	2010	Both sexes	9.8
1220	Latin America and the Caribbean	2010	Both sexes	1.4
1278	Oceania	2010	Both sexes	19.6
1312	Africa	2015	Both sexes	1.7
1376	Asia	2015	Both sexes	1.7
1432	Europe	2015	Both sexes	10.3
1485	Latin America and the Caribbean	2015	Both sexes	1.5
1543	Oceania	2015	Both sexes	20.6

4. I use pandas.DataFrame.pivot function to reshape the DataFrame by column values for moving the object under column "Major area, region, country or area of destination" to column headers.

```
In [3573]: new3.pivot(index = ['Year'], columns = 'Major area, region, country or area of destination')
           new3 = new3.pivot(index = ['Year'], columns = 'Major area, region, country or area of destination')['International m
           new3
```

Out[3573]:

Major area, region, country or area of destination	Year	Africa	Asia	Europe	Latin America and the Caribbean	Oceania
0	1990	2.5	1.5	6.8	1.6	17.5
1	1995	2.3	1.3	7.3	1.4	17.3
2	2000	1.8	1.3	7.7	1.2	17.3
3	2005	1.7	1.4	8.8	1.3	18.1
4	2010	1.6	1.6	9.8	1.4	19.6
5	2015	1.7	1.7	10.3	1.5	20.6

5. Since the DataFrame index name is showing "Major area, region, country or area of destination" after doing the pivot, I use DataFrame.index.name function to change it to an empty string. Finally, we get our table ready for the data visualization process.

```
In [3574]: new3.columns.names = ['']
           new3
```

Out[3574]:

	Year	Africa	Asia	Europe	Latin America and the Caribbean	Oceania
0	1990	2.5	1.5	6.8	1.6	17.5
1	1995	2.3	1.3	7.3	1.4	17.3
2	2000	1.8	1.3	7.7	1.2	17.3
3	2005	1.7	1.4	8.8	1.3	18.1
4	2010	1.6	1.6	9.8	1.4	19.6
5	2015	1.7	1.7	10.3	1.5	20.6

6. Similarly to table1, we should start with a basic look at the details of DataFrame.

```
In [3575]: new3.describe()
```

Out[3575]:

	Year	Africa	Asia	Europe	Latin America and the Caribbean	Oceania
count	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000
mean	2002.500000	1.933333	1.466667	8.450000	1.400000	18.400000
std	9.354143	0.372380	0.163299	1.412445	0.141421	1.385641
min	1990.000000	1.600000	1.300000	6.800000	1.200000	17.300000
25%	1996.250000	1.700000	1.325000	7.400000	1.325000	17.350000
50%	2002.500000	1.750000	1.450000	8.250000	1.400000	17.800000
75%	2008.750000	2.175000	1.575000	9.550000	1.475000	19.225000
max	2015.000000	2.500000	1.700000	10.300000	1.600000	20.600000

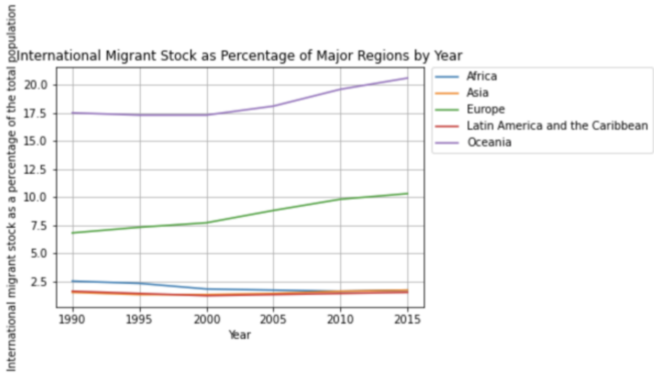
```
In [3576]: new3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Year                             6 non-null      int64
 1   Africa                           6 non-null      float64
 2   Asia                             6 non-null      float64
 3   Europe                           6 non-null      float64
 4   Latin America and the Caribbean  6 non-null      float64
 5   Oceania                          6 non-null      float64
dtypes: float64(5), int64(1)
memory usage: 416.0 bytes
```

7. Now, we could plot our fifth plot which is a line plot by using the matplotlib.pyplot.plot. I plot the line for each major region respectively for comparing the international migrant stock. The line plot uses points connected by line segments from left to right to demonstrate changes in the international migrant stock as a percentage of the total population. The horizontal axis indicates the continuous progression which is the year, and the vertical axis represents the values of international migrant stock as a percentage of the total population across those years, from 1990 to 2015.

<u>Plot #5: Line plot (Matplotlib) --- International Migrant Stock as Percentage of Major Regions by Year</u>

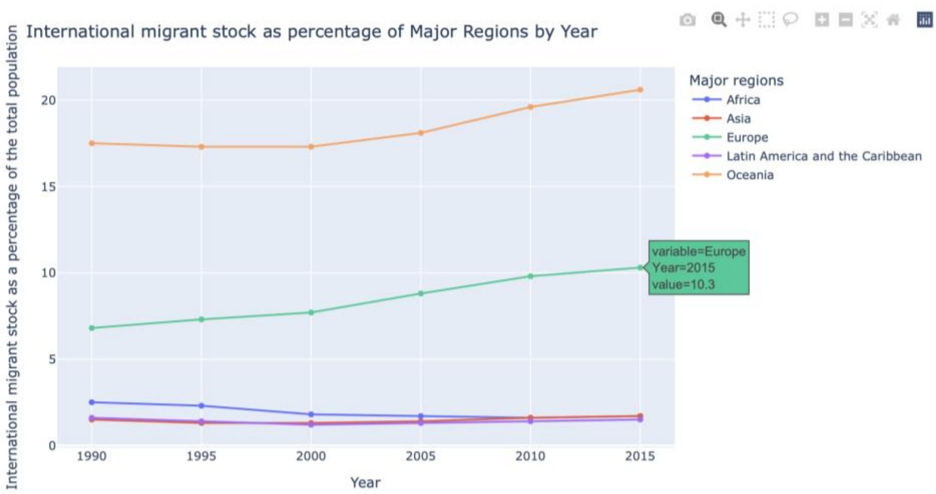
```
In [3617]: plt.plot(new3['Year'], new3['Africa'], label = 'Africa')
           plt.plot(new3['Year'], new3['Asia'], label = 'Asia')
           plt.plot(new3['Year'], new3['Europe'], label = 'Europe')
           plt.plot(new3['Year'], new3['Latin America and the Caribbean'], label = 'Latin America and the Caribbean')
           plt.plot(new3['Year'], new3['Oceania'], label = 'Oceania')

           plt.xlabel('Year')
           plt.ylabel('International migrant stock as a percentage of the total population')
           plt.title('International Migrant Stock as Percentage of Major Regions by Year')
           plt.grid(True)
           plt.legend(bbox_to_anchor = (1.02, 1), loc = 'upper left', borderaxespad = 0)

Out[3617]: <matplotlib.legend.Legend at 0x7fe3c85f2c70>
```



8. Moreover, I would like to use the library for the following data visualization since Plotly is insanely powerful at explaining and exploring data. It has a lot of advantages such as interactivity, customization, flexibility and so on. The horizontal axis also indicates the continuous progression which is the year, and the vertical axis also represents the values of international migrant stock as a percentage of the total population across those years, from 1990 to 2015.

Plot #6: Line plot (Plotly) --- International migrant stock as a percentage of the total population of Magjor Regions by Year

```
In [3578]: import plotly.io as pio
           pio.renderers.default = 'notebook'

           fig = px.line(new3, x = 'Year', y = new3.columns[1:6], markers = True)
           fig.update_layout(yaxis_title = 'International migrant stock as a percentage of the total population', title = 'Int
           fig.show()
```



As we could see, the graph plotting by Plotly can interact and it indicates the specific values of our data. We could find out that Oceania had the most international migrant stock as a percentage of total population from 1990 to 2015, and the second one in Europe. Oceania has an origin percentage of 17.5 from 1990 and kept the same until 2000, then it increased to 20.6 in 2015. Europe started at 6.8% and kept increasing to 10.3% in 2015. The rest three region, Africa, Asia and Latin America and the Caribbean, had lower percentages compared to the other two. Africa had 2.5% in 1990 while Asia and Latin America and the Caribbean had 1.5% and 1.6%, all of the three regions ended at 1.6% and 1.7% in 2015, they merged around the year 2007.

Data Visualization for Table4:

1. First, I import some libraries we use for data visualization, and we get the table4 ready from the data cleaning process.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
```

```
In [2019]: table4
```

Out[2019]:

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

1575 rows × 6 columns

2. Then I use pandas.DataFrame.drop function to drop the rows for getting all the countries in table4.

```
In [2020]: # selecting all the countries in table4 by deleting other rows
new4 = table4.drop([0,1,2,3,4,5,6,7,28,38,46,52,70,71,77,85,97,107,124,127,138,152,169,179,180,207,216,231,237,238,2
new4
```

Out[2020]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female migrants as a percentage of the international migrant stock
8	9	Burundi	108	B R	1990	51.0
9	10	Comoros	174	B	1990	52.3
10	11	Djibouti	262	B R	1990	47.4
11	12	Eritrea	232	I	1990	47.4
12	13	Ethiopia	231	B R	1990	47.4
...	...	...	...	...	...	...
1570	261	Samoa	882	B	2015	49.9
1571	262	Tokelau	772	B	2015	52.2
1572	263	Tonga	776	B	2015	45.4
1573	264	Tuvalu	798	C	2015	44.7
1574	265	Wallis and Futuna Islands	876	B	2015	49.5

1542 rows × 6 columns

3. We need to reset the index to make it orderly by using pandas.DataFrame.reset_index, and the drop parameter is to avoid the old index being added as a column, so I set it to drop = True.

In [2021]:
```
new4 = new4.reset_index(drop = True)
new4.head(10)
```

Out[2021]:

	Sort order	Major area, region, country or area of destination	Country code	Type of data (a)	Year	Female migrants as a percentage of the international migrant stock
0	9	Burundi	108	B R	1990	51.0
1	10	Comoros	174	B	1990	52.3
2	11	Djibouti	262	B R	1990	47.4
3	12	Eritrea	232	I	1990	47.4
4	13	Ethiopia	231	B R	1990	47.4
5	14	Kenya	404	B R	1990	45.9
6	15	Madagascar	450	C	1990	44.2
7	16	Malawi	454	B R	1990	51.5
8	17	Mauritius	480	C	1990	51.2
9	18	Mayotte	175	B	1990	42.3

4. Also, I use pandas.DataFrame.drop to drop the columns that we don't need for my data visualization. Finally, we get our table ready for the data visualization process.

In [2022]:
```
new4 = new4.drop(['Sort order', 'Country code', 'Type of data (a)'], axis = 1)
new4
```

Out[2022]:

	Major area, region, country or area of destination	Year	Female migrants as a percentage of the international migrant stock
0	Burundi	1990	51.0
1	Comoros	1990	52.3
2	Djibouti	1990	47.4
3	Eritrea	1990	47.4
4	Ethiopia	1990	47.4
...	...	...	...
1537	Samoa	2015	49.9
1538	Tokelau	2015	52.2
1539	Tonga	2015	45.4
1540	Tuvalu	2015	44.7
1541	Wallis and Futuna Islands	2015	49.5

1542 rows × 3 columns

5. Similarly to table1, we should start with a basic look at the details of DataFrame.

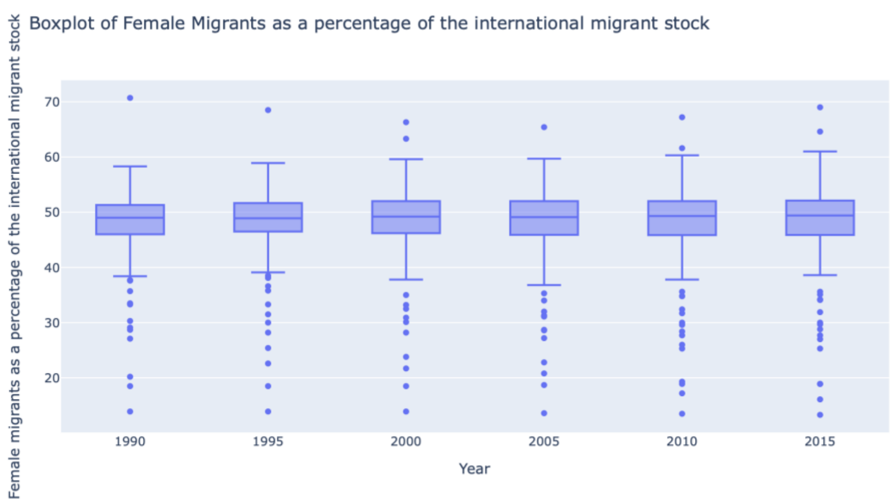
```
In [2023]: new4.describe()
Out[2023]:
```

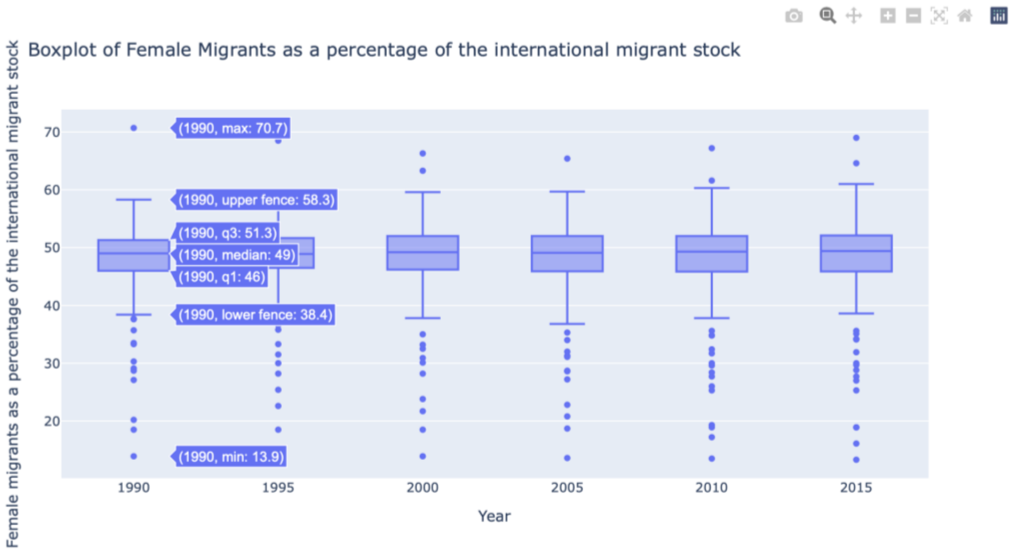
	Year	Female migrants as a percentage of the international migrant stock
count	1542.000000	1542.000000
mean	2002.821012	48.311349
std	8.438371	6.570394
min	1990.000000	13.300000
25%	1995.000000	46.000000
50%	2005.000000	49.100000
75%	2010.000000	51.900000
max	2015.000000	70.700000

```
In [2024]: new4.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1542 entries, 0 to 1541
Data columns (total 3 columns):
 #   Column                                                              Non-Null Count  Dtype
---  ------                                                              --------------  -----
 0   Major area, region, country or area of destination                  1542 non-null   object
 1   Year                                                                1542 non-null   int64
 2   Female migrants as a percentage of the international migrant stock  1542 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 36.3+ KB
```

9. Now, we could plot our sixth plot which is a box plot by using Plotly library, px.box function. I plot the box for female migrants as a percentage of the international migrant stock for all major area, region, country, or area of destination in 1990, 1995, 2000, 2005, 2010 and 2015. The highest point shows the highest percentage of the female migrants of the international migrant stock in a certain year, the lowest shows the lowest percentage. The upper line is the upper fence, and the lower line is the lower fence. Inside the box lies the middle 50% of the data, the horizontal border lines of the box are the separate lines between quartile groups and the line in the box is the median of the data. Six boxes mean the distributions of six years.
<u>Plot #7: Boxplot --- Boxplot of Female Migrants as a percentage of the international migrant stock</u>

```
In [2025]: import plotly.io as pio
pio.renderers.default = 'notebook'

fig = px.box(new4, x = 'Year', y = 'Female migrants as a percentage of the international migrant stock', title = 'Bo
fig.show()
```





From the graph, we could know several details. In 1990, the highest value of female migrants is 70.7% of the international migrant stock, and the lowest is 13.9%. The top quartile(Q4) of areas has at least 51.3% and the bottom quartile(Q1) has at most 46%. The upper fence is 58.3% and the lower fence is 38.4%. The median percentage is 49 in 1990.

In 1995, the highest value of female migrants is 68.5% of the international migrant stock, and the lowest is 13.9%. Q4 of data has at least 51.65% and Q1 has at most 46.5%. The upper fence is 58.9% and the lower fence is 39.1%. The median percentage is 48.9 in 1995.

In 2000, the highest value of female migrants is 66.3% of the international migrant stock, and the lowest is 13.9%. Q4 of data has at least 52% and Q1 has at most 46.2%. The upper fence is 59.6% and the lower fence is 37.8%. The median percentage is 49.2 in 2000.

In 2005, the highest value of female migrants is 65.4% of the international migrant stock, and the lowest is 13.6%. Q4 of data has at least 52% and Q1 has at most 45.9%. The upper fence is 59.7% and the lower fence is 36.8%. The median percentage is 49.1 in 2005.

In 2010, the highest value of female migrants is 67.2% of the international migrant stock, and the lowest is 13.5%. Q4 of data has at least 52% and Q1 has at most 45.85%. The upper fence is 60.3% and the lower fence is 37.8%. The median percentage is 49.3 in 2010.

In 2015, the highest value of female migrants is 69% of the international migrant stock, and the lowest is 13.3%. Q4 of data has at least 52.1% and Q1 has at most 45.875%. The upper fence is 61% and the lower fence is 38.6%. The median percentage is 49.4 in 2015.

To conclude, from this box plot, we can find out that for all areas, the highest value of female migrants is around 69% of the international migrant stock, and the lowest is around 13.7%. The lower border of Q4 of data is around 52% and the upper border of Q1 is around 46%. The upper fence is around 59% and the lower fence is 38%. The median percentage is around 49 in these 25 years.

Data Visualization for Table5:

1. First, I import some libraries we use for data visualization, and we get the table5 ready from the data cleaning process.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
```

```
In [1117]: table5
Out[1117]:
```

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

3930 rows × 7 columns

2. I use pandas.DayaFrame.drop function to drop the columns we don't need for data visualization, and I use pandas.DataFrame.loc to get only "Both sexes" under column "Gender".

```
In [1118]: new5 = table5.drop(['Sort order', 'Country code', 'Type of data (a)'], axis = 1)
           new5 = new5.loc[new5['Gender'] == 'Both sexes']
           new5
```

Out[1118]:

	Major area, region, country or area of destination	Year	Gender	Annual rate of change of the migrant stock
0	WORLD	1990-1995	Both sexes	1.05
1	Developed regions	1990-1995	Both sexes	2.28
2	Developing regions	1990-1995	Both sexes	-0.49
3	Least developed countries	1990-1995	Both sexes	1.12
4	Less developed regions excluding least develop...	1990-1995	Both sexes	-0.80
...	...	...	...	...
1305	Samoa	2010-2015	Both sexes	-0.77
1306	Tokelau	2010-2015	Both sexes	2.54
1307	Tonga	2010-2015	Both sexes	2.64
1308	Tuvalu	2010-2015	Both sexes	-1.76
1309	Wallis and Futuna Islands	2010-2015	Both sexes	0.52

1310 rows × 4 columns

3. Then, I would like to create a new DataFrame "hist_vertical" for our first plot of table 5. I use pandas.DataFrame.loc to get only "WORLD" under column "Major area, region, country or area of destination".

```
In [1119]: # plot#1, vertical histogram
           hist_vertical = new5.loc[new5['Major area, region, country or area of destination'] == 'WORLD']
           hist_vertical
```

Out[1119]:

	Major area, region, country or area of destination	Year	Gender	Annual rate of change of the migrant stock
0	WORLD	1990-1995	Both sexes	1.05
261	WORLD	1995-2000	Both sexes	1.43
522	WORLD	2000-2005	Both sexes	2.04
783	WORLD	2005-2010	Both sexes	2.95
1045	WORLD	2010-2015	Both sexes	1.89

4. We need to reset the index to make it orderly by using pandas.DataFrame.reset_index, and the drop parameter is to avoid the old index being added as a column, so I set it to drop = True.

```
In [1120]: hist_vertical = hist_vertical.reset_index(drop=True)
           hist_vertical
```

Out[1120]:

	Major area, region, country or area of destination	Year	Gender	Annual rate of change of the migrant stock
0	WORLD	1990-1995	Both sexes	1.05
1	WORLD	1995-2000	Both sexes	1.43
2	WORLD	2000-2005	Both sexes	2.04
3	WORLD	2005-2010	Both sexes	2.95
4	WORLD	2010-2015	Both sexes	1.89

5. Similarly to table1, we should start with a basic look at the details of DataFrame.

In [1121]:
```
hist_vertical.describe()
```
Out[1121]:

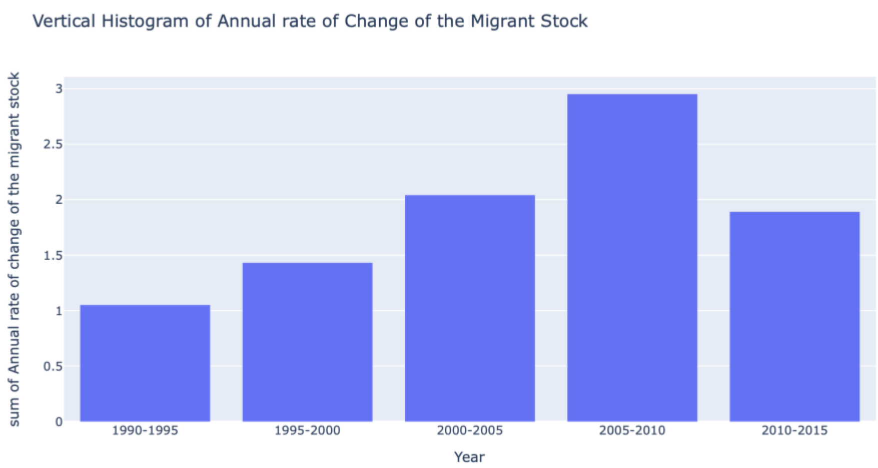
	Annual rate of change of the migrant stock
count	5.000000
mean	1.872000
std	0.717928
min	1.050000
25%	1.430000
50%	1.890000
75%	2.040000
max	2.950000

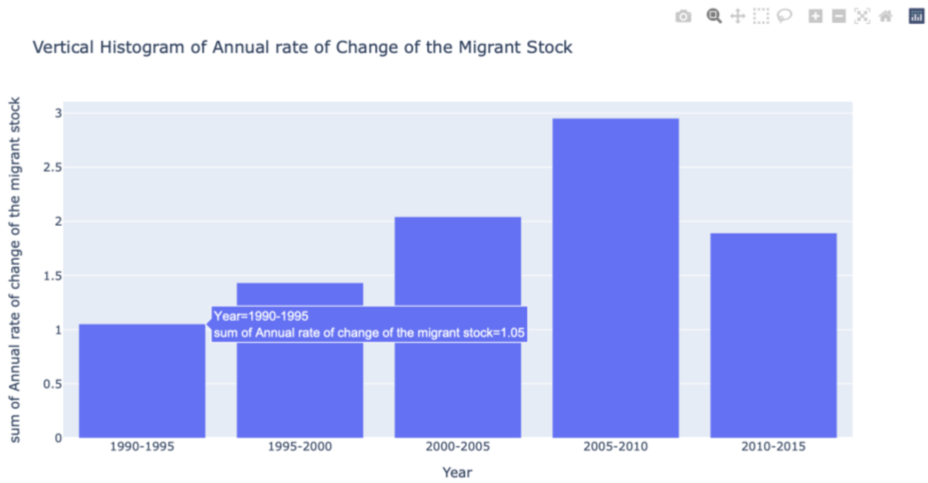
In [1122]:
```
hist_vertical.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column                                               Non-Null Count  Dtype
---  ------                                               --------------  -----
 0   Major area, region, country or area of destination  5 non-null      object
 1   Year                                                 5 non-null      object
 2   Gender                                               5 non-null      object
 3   Annual rate of change of the migrant stock          5 non-null      float64
dtypes: float64(1), object(3)
memory usage: 288.0+ bytes
```

6. Now, we could plot our first plot of the table5 which is a histogram by using the Plotly library, px.histogram function. I plot the histogram for the annual rate of change of the migrant stock in five time periods, 1990 to 1995, 1995 to 2000, 2000 to 2005, 2005 to 2010 and 2010 to 2015. Each histogram represents the certain annual rate of change of migrant stock in that period of time that is shown on x-axis.

<u>Plot #8: Vertical histogram --- Vertical Histogram of Annual rate of Change of the Migrant Stock</u>

In [1123]:
```
import plotly.io as pio
pio.renderers.default = 'notebook'

fig = px.histogram(hist_vertical,
                   x = 'Year',
                   y = 'Annual rate of change of the migrant stock',
                   range_x = ['1990-1995','2010-2015'],
                   title = 'Vertical Histogram of Annual rate of Change of the Migrant Stock',
                  )
fig.show()
```





From the plots above, we could find out that the annual rate of change of the migrant stock is 1.05 from 1990 to 1995, which is the minimum in the five time periods. From 1995 to 2000, it increased to 1.43. From 2000 to 2005, it increased to 2.04 and it reached the highest point of 2.95 from 2005 to 2010, at last, it decreased to 1.89 from 2010 to 2015.

7. Furthermore, I would like to create another new DataFrame "hist_horizontal" for our second plot of table5, which is also the last plot for our data visualization. I use pandas.DataFrame.loc to get only "Africa" under column "Major area, region, country or area of destination".

```
In [1124]: # plot#2, horizontal histogram
hist_horizontal = new5.loc[new5['Major area, region, country or area of destination'] == 'Africa']
hist_horizontal
```

Out[1124]:

	Major area, region, country or area of destination	Year	Gender	Annual rate of change of the migrant stock
6	Africa	1990-1995	Both sexes	0.83
267	Africa	1995-2000	Both sexes	-2.00
528	Africa	2000-2005	Both sexes	0.52
789	Africa	2005-2010	Both sexes	2.06
1051	Africa	2010-2015	Both sexes	4.08

8. We repeated steps #4 and #5 to get ready for plotting.

In [1125]:
```
hist_horizontal = hist_horizontal.reset_index(drop = True)
hist_horizontal
```

Out[1125]:

	Major area, region, country or area of destination	Year	Gender	Annual rate of change of the migrant stock
0	Africa	1990-1995	Both sexes	0.83
1	Africa	1995-2000	Both sexes	-2.00
2	Africa	2000-2005	Both sexes	0.52
3	Africa	2005-2010	Both sexes	2.06
4	Africa	2010-2015	Both sexes	4.08

In [1126]:
```
hist_horizontal.describe()
```

Out[1126]:

	Annual rate of change of the migrant stock
count	5.000000
mean	1.098000
std	2.226055
min	-2.000000
25%	0.520000
50%	0.830000
75%	2.060000
max	4.080000

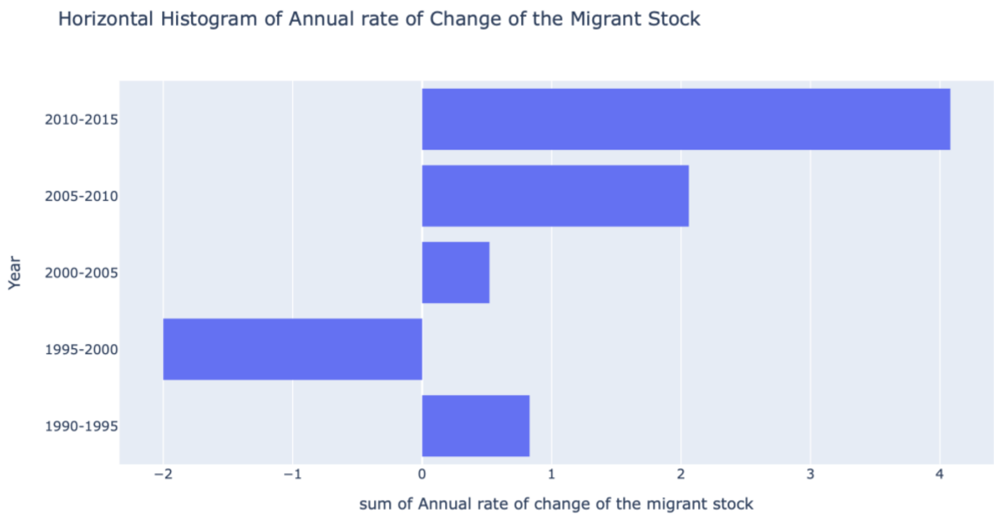
In [1127]:
```
hist_horizontal.info()
```

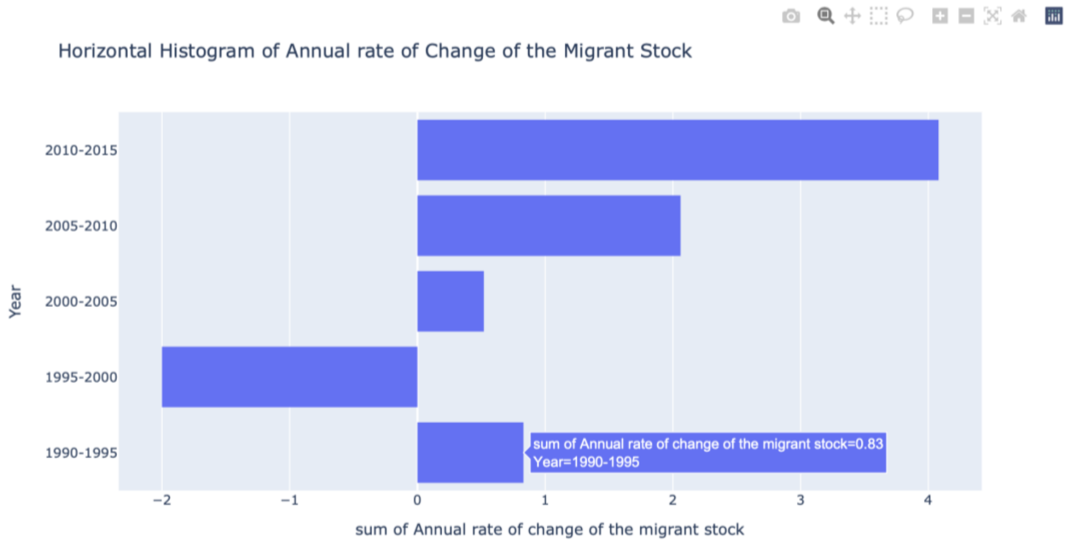
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column                                              Non-Null Count  Dtype  
---  ------                                              --------------  -----  
 0   Major area, region, country or area of destination  5 non-null      object 
 1   Year                                                5 non-null      object 
 2   Gender                                              5 non-null      object 
 3   Annual rate of change of the migrant stock          5 non-null      float64
dtypes: float64(1), object(3)
memory usage: 288.0+ bytes
```

9. Now, we could plot our second plot of table5 which is a histogram by using the Plotly library. The reason I would like to plot the horizontal histogram is that we might have negative values in our dataset, according to the values of table5, it is about the annual rate of change as a percentage of the migrant stock. Therefore, it is easier and clearer that use the horizontal histogram to visualize the annual rate of change.

Plot #9: Horizontal histogram --- Horizontal Histogram of Annual rate of Change of the Migrant Stock

```
In [1128]: import plotly.io as pio
pio.renderers.default = 'notebook'

fig = px.histogram(hist_horizontal,
                   x = 'Annual rate of change of the migrant stock',
                   y = 'Year',
                   range_y = ['1990-1995','2010-2015'],
                   title = 'Horizontal Histogram of Annual rate of Change of the Migrant Stock',
                  )
fig.show()
```





From the plots above, we could find out that the annual rate of change of the migrant stock is 0.83 from 1990 to 1995. From 1995 to 2000, it was -2, which is the only negative value. From 2000 to 2005, it was 0.52 which is the minimum positive value. It increased to 2.06 from 2005 to 2010, at last, it increased to its highest value of 4.08 from 2010 to 2015.

Conclusion

In the world of Big Data, data visualization is essential to analyze massive amounts of information and making data-driven decisions. I made five different types of graphs, horizontal bar plot, stocked bar plot, line plot, box plot and histogram plot as examples of data visualization. During plotting graphs of five tables, to make sure that my graphs are clear and correct, I followed Tufte's principles of data visualization, which are Graphical Integrity, Data-Ink, Chartjunk, Data Density and Small Multiplies during the whole process. So that the graph should be easily read and analyzed. Several comparisons are made so that the trend could be shown easily. Data visualization is the most direct and easy way to let the audience escape the tons of messy data and get what the data means, and what the data analysis wants to tell.