Australian Migration

Background

As a temporary migrant to Australia myself as a small child and for a longer period of 7 years from 2006 to 2013 with my own family, I was aware that Australia had been a destination for people from the UK for many decades. I was interested to investigate this topic further using current available data.

This notebook investigates country of origin for migration to Australia over the period 2005 to 2023 using the available statistics. The figures are derived from the Australian Bureau of Statistics and are reported for financial years ended 30 June for each year.

The purpose of the study is to examine migration trends and patterns and derive any interesting or useful insights regarding the movement of people to Australia. In short, has the pattern of migration to Australia changed over this period?

The inspiration for this notebook comes from Cognitive AI Data Visualization with Python Course.

Data Sources:

Australian Bureau of Statistics Migration Figures

Methodology Notes

World Bank GDP Figures

World Bank Population Figures

World Bank Country Lending Groups

Macrotrends

Worldometer

Statista

UN Data

1.0 Import the data and libraries

In [378…

```
! pip install pywaffle
```

```
Collecting pywaffle
  Downloading pywaffle-1.1.1-py2.py3-none-any.whl (30 kB)
Requirement already satisfied: matplotlib in c:\users\imoge\appdata\roaming\python\pytho
n37\site-packages (from pywaffle) (3.5.3)
Collecting fontawesomefree
  Downloading fontawesomefree-6.6.0-py3-none-any.whl (25.6 MB)
```

```
     ---------------------------------------- 25.6/25.6 MB 9.0 MB/s eta 0:00:00
Requirement already satisfied: cycler>=0.10 in c:\users\imoge\appdata\roaming\python\pyt
hon37\site-packages (from matplotlib->pywaffle) (0.11.0)
Requirement already satisfied: numpy>=1.17 in c:\users\imoge\appdata\roaming\python\pyth
on37\site-packages (from matplotlib->pywaffle) (1.21.6)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\imoge\appdata\roaming\pytho
n\python37\site-packages (from matplotlib->pywaffle) (1.4.5)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\imoge\appdata\roaming\pytho
n\python37\site-packages (from matplotlib->pywaffle) (4.38.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\imoge\appdata\roaming\python\py
thon37\site-packages (from matplotlib->pywaffle) (9.5.0)
Requirement already satisfied: packaging>=20.0 in c:\users\imoge\anaconda3\envs\machinel
earning\lib\site-packages (from matplotlib->pywaffle) (23.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\imoge\appdata\roaming\py
thon\python37\site-packages (from matplotlib->pywaffle) (2.9.0.post0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\imoge\appdata\roaming\python
\python37\site-packages (from matplotlib->pywaffle) (3.1.2)
Requirement already satisfied: typing-extensions in c:\users\imoge\appdata\roaming\pytho
n\python37\site-packages (from kiwisolver>=1.0.1->matplotlib->pywaffle) (4.7.1)
Requirement already satisfied: six>=1.5 in c:\users\imoge\appdata\roaming\python\python3
7\site-packages (from python-dateutil>=2.7->matplotlib->pywaffle) (1.16.0)
Installing collected packages: fontawesomefree, pywaffle
Successfully installed fontawesomefree-6.6.0 pywaffle-1.1.1
```

```
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
WARNING: Ignoring invalid distribution -illow (c:\users\imoge\anaconda3\envs\machinelear
ning\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\imoge\anaconda3\envs\machine
learning\lib\site-packages)
```

In [379…

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use(['ggplot'])
import plotly
```

```
import plotly.express as px
from pywaffle import Waffle
import warnings
warnings.filterwarnings("ignore")
```

In [269…

```
print(matplotlib.__version__)
```

```
3.5.3
```

In [270…

```
# Read in the data starting from the row where it starts in the spreadsheet
df = pd.read_excel(r'C:\Users\imoge\AllMLProjects\Data\AustralianMigration2324.xlsx', s
```

In [271…

```
# Get the table shape
df.shape
```

Out[271…

```
(257, 22)
```

In [272…

```
# Have a look at a summary of the datatypes and null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 257 entries, 0 to 256
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   SACC code(e)         251 non-null    object 
 1   Country of birth(e)  253 non-null    object 
 2   2004-05              253 non-null    float64
 3   2005-06              253 non-null    float64
 4   2006-07              253 non-null    float64
 5   2007-08              253 non-null    float64
 6   2008-09              253 non-null    float64
 7   2009-10              253 non-null    float64
 8   2010-11              253 non-null    float64
 9   2011-12              253 non-null    float64
 10  2012-13              253 non-null    float64
 11  2013-14              253 non-null    float64
 12  2014-15              253 non-null    float64
 13  2015-16              253 non-null    float64
 14  2016-17              253 non-null    float64
 15  2017-18              253 non-null    float64
 16  2018-19              253 non-null    float64
 17  2019-20              253 non-null    float64
 18  2020-21              253 non-null    float64
 19  2021-22              253 non-null    float64
 20  2022-23              253 non-null    float64
 21  2023-24(f)           253 non-null    float64
dtypes: float64(20), object(2)
memory usage: 44.3+ KB
```

In [273…

```
# Check the table head
df.head()
```

Out[273...

	SACC code(e)	Country of birth(e)	2004-05	2005-06	2006-07	2007-08	2008-09	2009-10	2010-11	2011-12	...	2014-15
0	1101	Australia	47290.0	50150.0	52920.0	53190.0	56470.0	54530.0	53150.0	50610.0	...	46350.0
1	1102	Norfolk Island(g)	30.0	30.0	40.0	40.0	30.0	30.0	30.0	30.0	...	20.0
2	1199	Aust E T, nec	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
3	1201	New Zealand	32030.0	32770.0	36300.0	42070.0	38090.0	31580.0	43430.0	48220.0	...	23300.0
4	1301	New Caledonia	200.0	150.0	160.0	190.0	190.0	220.0	170.0	130.0	...	150.0

5 rows × 22 columns

In [274...

```
# Check the tail
df.tail(10)
```

Out[274...

	SACC code(e)	Country of birth(e)	2004-05	2005-06	2006-07	2007-08	2008-09	2009-10	2010-11
247	9232	Zimbabwe	2510.0	2710.0	2950.0	3630.0	3230.0	1850.0	1560.0
248	9299	S & E Afr, nec	0.0	0.0	0.0	0.0	0.0	0.0	0.0
249	0000	Inadequately Described	0.0	0.0	0.0	0.0	0.0	0.0	0.0
250	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
251	NaN	Total Australian-born	47290.0	50150.0	52920.0	53190.0	56470.0	54530.0	53150.0
252	NaN	Total overseas-born	294120.0	326380.0	384520.0	448150.0	463310.0	383400.0	378630.0
253	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
254	NaN	Total	341400.0	376530.0	437440.0	501340.0	519790.0	437930.0	431780.0
255	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
256	© Commonwealth of Australia 2024	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

10 rows × 22 columns

We can see that we have quite a few rows at the bottom of the table that we do not want

2.0 Data Cleaning

Australia collects data for their financial year which is from 1 July to 30 June. We will rename the columns to the end of June + the year for ease of presentation.

In [275…

```
# Drop the 9 rows at the bottom we don't need
df.drop(df.tail(9).index,inplace = True)

# Check the column names
print(df.columns)

# Rename the Country column
df.rename(columns = {'Country of birth(e)':'Country'}, inplace = True)

# Drop the SACC code and set the Country of Birth to the index
df.drop(columns = ['SACC code(e)'], axis = 1, inplace = True)
df.set_index('Country', drop = True, inplace = True)

# Replace the dash with an empty string
df.columns = df.columns.str.replace("-", "", regex=True)

# Iterate over the column names and extract the first and last two characters and appen
cols = []

for i in list(df.columns):
    i = 'Jun ' + i[0:2] + i[-2:]
    cols.append(i)

# Set the list as the column names
df.columns = cols

# Rename the last column
df.rename(columns={"Jun 20f)": "Jun 2024"}, inplace = True)

# Rename UK
df = df.rename(index={'UK, CIs & IOM': 'UK'}) # Rename to UK

# Drop Australians returning home
df = df[df.index != 'Australia']

# Check for Nan values
print(df.isnull().sum())

df.head()
```

```
Index(['SACC code(e)', 'Country of birth(e)', '2004-05', '2005-06', '2006-07',
       '2007-08', '2008-09', '2009-10', '2010-11', '2011-12', '2012-13',
       '2013-14', '2014-15', '2015-16', '2016-17', '2017-18', '2018-19',
       '2019-20', '2020-21', '2021-22', '2022-23', '2023-24(f)'],
      dtype='object')
Jun 2005    0
```

```
Jun 2006    0
Jun 2007    0
Jun 2008    0
Jun 2009    0
Jun 2010    0
Jun 2011    0
Jun 2012    0
Jun 2013    0
Jun 2014    0
Jun 2015    0
Jun 2016    0
Jun 2017    0
Jun 2018    0
Jun 2019    0
Jun 2020    0
Jun 2021    0
Jun 2022    0
Jun 2023    0
Jun 2024    0
dtype: int64
```

Out[275…

Country	Jun 2005	Jun 2006	Jun 2007	Jun 2008	Jun 2009	Jun 2010	Jun 2011	Jun 2012	Jun 2013	Jun 2014	J 20
Norfolk Island(g)	30.0	30.0	40.0	40.0	30.0	30.0	30.0	30.0	40.0	30.0	2(
Aust E T, nec	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	(
New Zealand	32030.0	32770.0	36300.0	42070.0	38090.0	31580.0	43430.0	48220.0	41500.0	28440.0	2330
New Caledonia	200.0	150.0	160.0	190.0	190.0	220.0	170.0	130.0	130.0	110.0	15
PNG	1370.0	1620.0	1610.0	1820.0	1710.0	1550.0	1560.0	1640.0	1580.0	1270.0	124

In [276…

```
# Create a copy
df_cleaned = df.copy()
```

In [277…

```
# Create a total column
df_cleaned['Total'] = df_cleaned.sum(axis = 1)

# Create an average column
df_cleaned['Average'] = df_cleaned['Total']/len(df_cleaned.columns[0:-1])

# Set the values to integers
cols = df_cleaned.columns
df_cleaned[cols] = df_cleaned[cols].applymap(np.int64)

# Drop rows where the average is zero
df_cleaned = df_cleaned[df_cleaned['Average']>0]

# Get shape
df_cleaned.shape
```

Out[277… (215, 22)

In [278…
```
len(df_cleaned.index)
```

Out[278… 215

3.0 Data Exploration

In [279…
```
# Check the basic statistics
df_cleaned.describe()
```

Out[279…

	Jun 2005	Jun 2006	Jun 2007	Jun 2008	Jun 2009	Jun 2010	Jun 201
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000
mean	1368.000000	1518.139535	1789.069767	2085.023256	2155.162791	1783.441860	1761.72093(
std	4497.470338	5020.840239	6049.224332	7219.410290	7463.178950	5820.706212	5741.58411!
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
50%	120.000000	120.000000	120.000000	140.000000	150.000000	150.000000	130.000000
75%	645.000000	605.000000	700.000000	750.000000	755.000000	780.000000	755.000000
max	39870.000000	45530.000000	48650.000000	56810.000000	68350.000000	49020.000000	45750.000000

8 rows × 22 columns

The mean values across the years seem ok, with a dip in 2021 which is likely due to Covid restrictions but we will look at this further. There is some skew in the data with the mean values well above the median.

3.1 Average Migration from All Countries

In [280…
```
# Make a copy of the dataframe
df_migration = df_cleaned.copy()
```

In [281…
```
# Countries with 500 or less migrants per year on average
print('Total countries:',df_migration.shape[0])
print('Less than 500 per year: ',df_migration[df_migration['Average']<=500].shape[0])
print('%:',(df_migration[df_migration['Average']<=500].shape[0]/(df_migration.shape[0])
```

```
Total countries: 215
Less than 500 per year:  145
%: 67.44186046511628
```

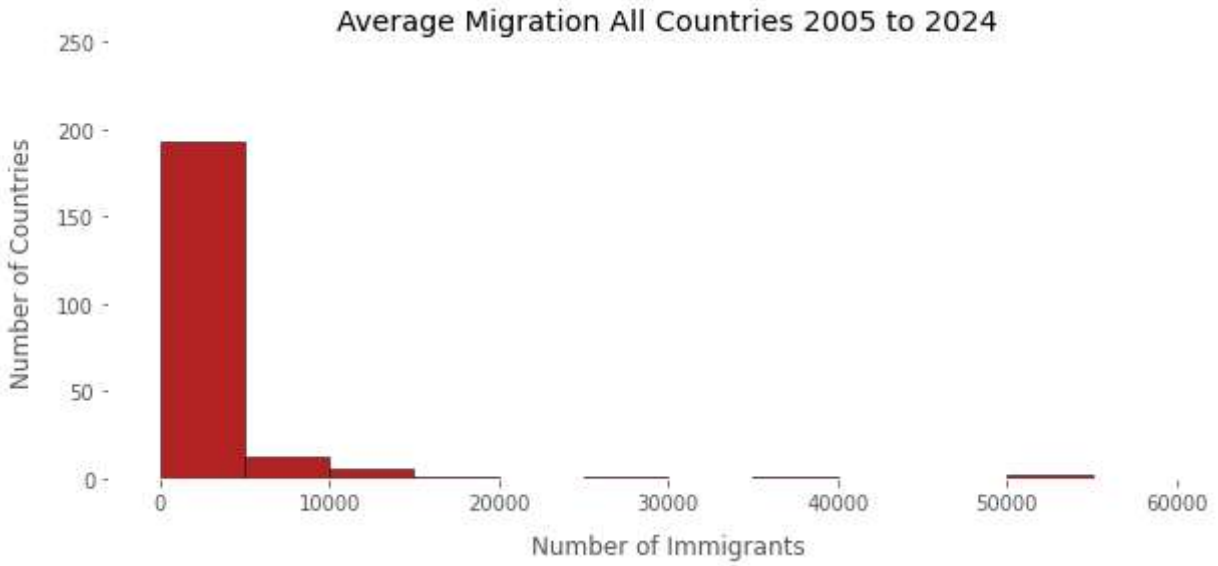
In [282…

```
# Countries with 5000 or less migrants per year on average
print('Total countries:',df_migration.shape[0])
print('Less than 5000 per year: ',df_migration[df_migration['Average']<=5000].shape[0])
print('%:',(df_migration[df_migration['Average']<=5000].shape[0]/(df_migration.shape[0]
```

```
Total countries: 215
Less than 5000 per year:  193
%: 89.76744186046511
```

In [283…

```
# Set the bins 5000 apart
bins = [0,5000,10000,15000,20000,25000,30000,35000,40000,45000,50000,55000,60000]

# Plot the histogram
ax = (df_migration['Average']).plot(kind='hist',
                                    figsize=(10,4),
                                    color = 'firebrick',
                                    bins = bins,
                                    ec = 'k',
                                    ylim=(0,250)
                                          )
ax.set_facecolor("white")
# Add labels
plt.title('Average Migration All Countries 2005 to 2024')
plt.ylabel('Number of Countries',labelpad = 10)
plt.xlabel("Number of Immigrants",labelpad = 10);
```



In [284…

```
df_migration[df_migration['Average']>=25000]
```

Out[284…

Country	Jun 2005	Jun 2006	Jun 2007	Jun 2008	Jun 2009	Jun 2010	Jun 2011	Jun 2012	Jun 2013	Jun 2014	...	Jun 2017	Jun 2018	J 20
New Zealand	32030	32770	36300	42070	38090	31580	43430	48220	41500	28440	...	22660	21270	209
UK	39870	45530	48650	53330	48910	42100	45750	50610	46250	37190	...	33080	29740	293
China	28870	30370	39540	49200	48540	49020	41210	40150	46560	53130	...	77760	82320	718

Country	Jun 2005	Jun 2006	Jun 2007	Jun 2008	Jun 2009	Jun 2010	Jun 2011	Jun 2012	Jun 2013	Jun 2014	...	Jun 2017	Jun 2018	J 20
India	21070	27770	43660	56810	68350	36610	23760	30390	35280	45520	...	60310	67440	863

4 rows × 22 columns

- Most countries (68%) contributed 500 or less migrants per year on average over the whole period
- There are a few countries that had more than 25000 migrants on average per year - China, India, UK, New Zealand

3.2 The Trend in Total Migration Across the Period

In [285…

```
# Calculate the average across the period in thousands
average_calc = df_migration.drop(columns = ['Total','Average'],axis = 1)
average_calc = average_calc.sum().sum()/19
print(average_calc)
```

```
450344.7368421053
```

In [286…

```
# Get total for all countries migration by year
all_countries = df_migration.iloc[:,:-2]
all_countries_total = all_countries.sum(axis = 0)
all_countries_total = all_countries_total/1000
```

In [287…

```
# Percentage growth
percent_growth = pd.DataFrame(all_countries_total).reset_index()
percent_growth.columns = ['Year','Migration(000s)']
percent_growth['%'] = percent_growth['Migration(000s)'].pct_change()*100
percent_growth
```

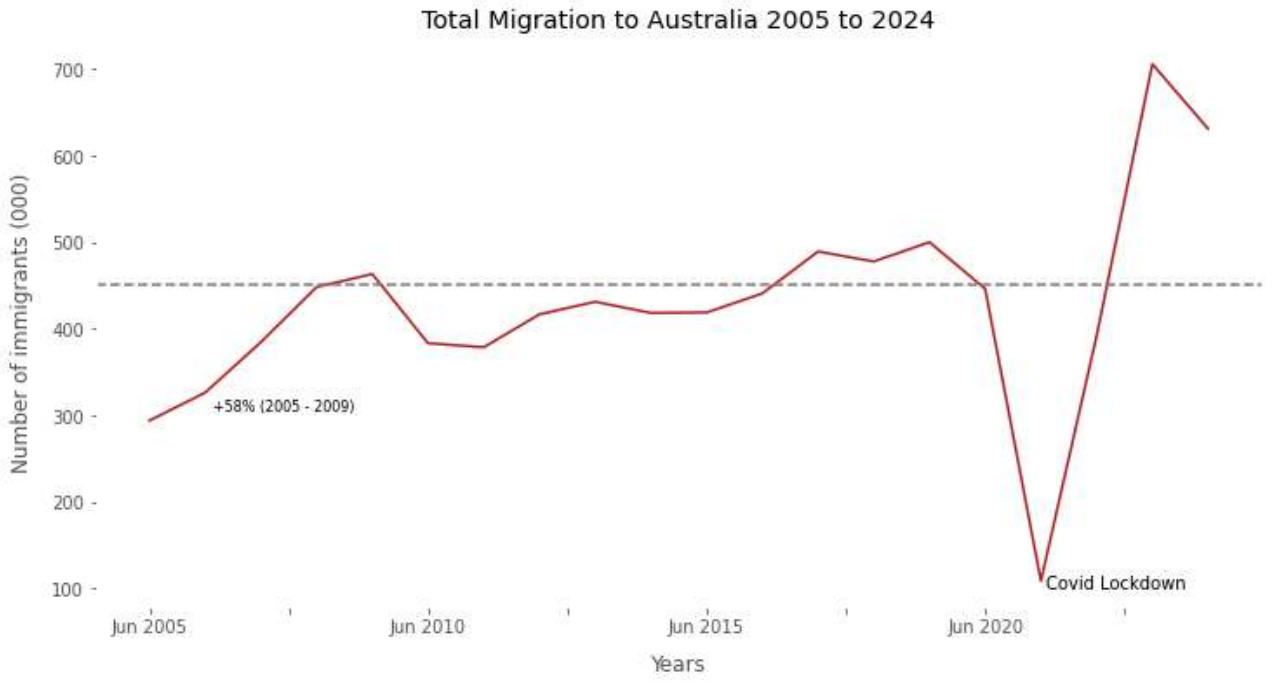
Out[287…

	Year	Migration(000s)	%
0	Jun 2005	294.12	NaN
1	Jun 2006	326.40	10.975112
2	Jun 2007	384.65	17.846201
3	Jun 2008	448.28	16.542311
4	Jun 2009	463.36	3.363969
5	Jun 2010	383.44	-17.247928
6	Jun 2011	378.77	-1.217922
7	Jun 2012	416.78	10.035114
8	Jun 2013	431.30	3.483852

	Year	Migration(000s)	%
9	Jun 2014	418.40	-2.990958
10	Jun 2015	418.94	0.129063
11	Jun 2016	440.87	5.234640
12	Jun 2017	489.31	10.987366
13	Jun 2018	477.80	-2.352292
14	Jun 2019	500.10	4.667225
15	Jun 2020	446.20	-10.777844
16	Jun 2021	109.03	-75.564769
17	Jun 2022	391.89	259.433184
18	Jun 2023	705.77	80.093904
19	Jun 2024	631.14	-10.574266

In [288…

```
# Plot the migration total for all years

fig, ax = plt.subplots(figsize = (12,6))
all_countries_total.plot(color = 'firebrick',
                  title = 'Total Migration to Australia 2005 to 2024')
ax.set_facecolor('white')
plt.ylabel('Number of immigrants (000)',labelpad = 10, fontsize = 12)
plt.xlabel('Years', labelpad = 10, fontsize = 12)
ax.text(16.1, 100, 'Covid Lockdown', fontsize = 10)
plt.axhline(y = average_calc/1000, color = 'gray', linestyle = '--');
plt.text(1.15,305,"+58% (2005 - 2009)",fontsize = 8);
```



Total Migration over the Period

- We can see from the chart that migration has been relatively stable around a mean of 417,000 until the lockdown, where migration from all countries into Australia fell to 109,000.
- Since then, there has been a recovery to 392,000 to June 2022 and jumping up to a peak of 704,000 to the year ending June 2023.
- However, if we average the last three years (2021,2022 and 2023), migration is 401,000 per year closer to the mean for the years 2005 to 2019 of 419,000.
- Overall, excluding the blips over the Covid years, migration has been relatively consistent at around 420,000 with the exception of the period of rapid growth between June 2005 and June 2009, where there was a significant increase in annual migration levels from 294,000 per year to 463,000 per year (58% growth rate).
- This was also the period where my own family emigrated to Australia (January 2006)

3.3 Total Migration by Country over the Period

In [289…

```
# Sort by the total column
sorted_total = df_cleaned.sort_values(by = 'Total',ascending = False)
top5 = sorted_total.head(5)
top5
```

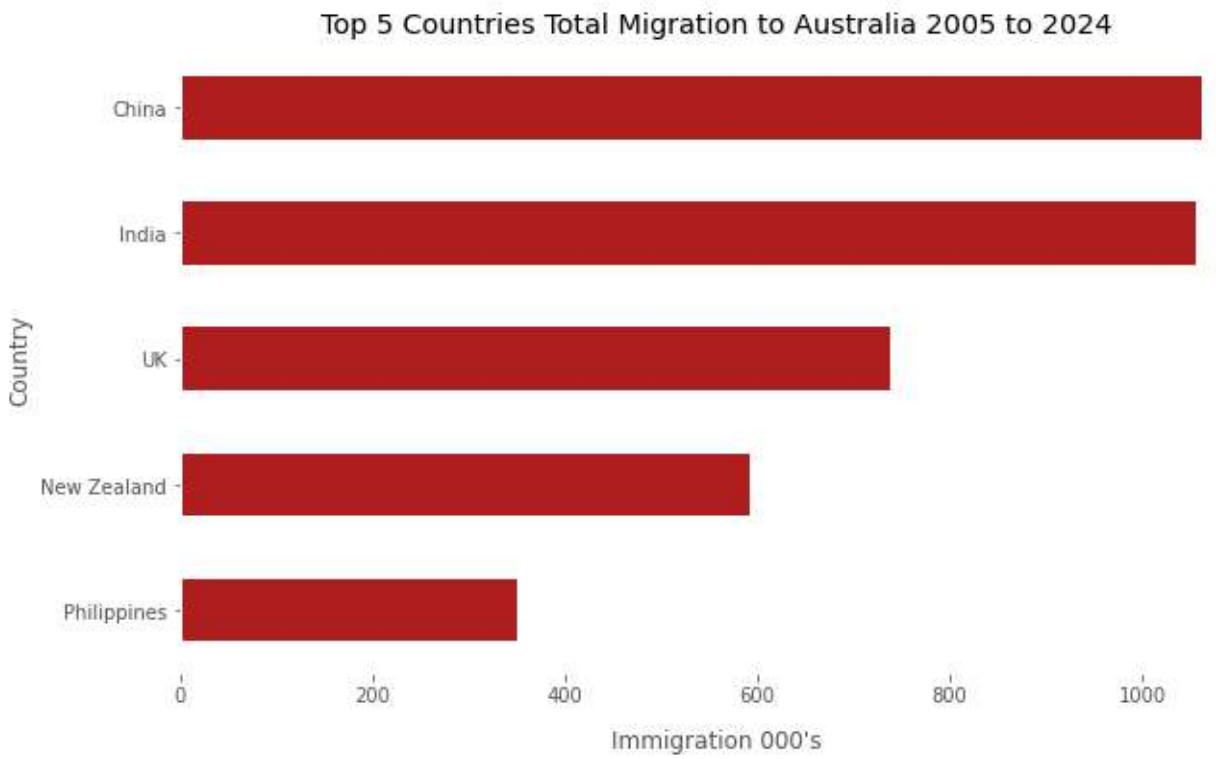
Out[289…

Country	Jun 2005	Jun 2006	Jun 2007	Jun 2008	Jun 2009	Jun 2010	Jun 2011	Jun 2012	Jun 2013	Jun 2014	...	Jun 2017	Jun 2018
China	28870	30370	39540	49200	48540	49020	41210	40150	46560	53130	...	77760	82320
India	21070	27770	43660	56810	68350	36610	23760	30390	35280	45520	...	60310	67440
UK	39870	45530	48650	53330	48910	42100	45750	50610	46250	37190	...	33080	29740
New Zealand	32030	32770	36300	42070	38090	31580	43430	48220	41500	28440	...	22660	21270
Philippines	7900	10140	13150	15120	15560	12520	13240	16910	17010	15890	...	17120	16370

5 rows × 22 columns

In [290…

```
# Chart the top five countries
ax = (top5['Total']/1000).sort_values().plot(kind = 'barh',
                                             figsize = (10,6),
                                             title = 'Top 5 Countries Total Migration to Aus
                                             color = 'firebrick')
ax.set_facecolor("white")
plt.xlabel("Immigration 000's", labelpad = 12);
```



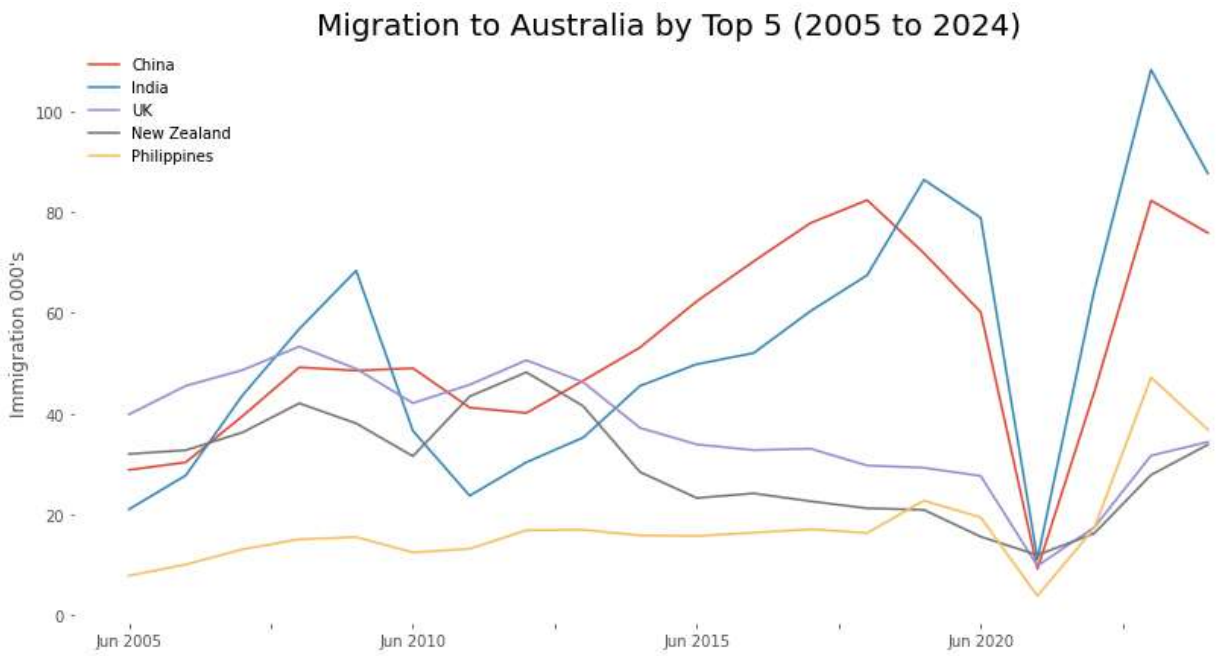
Total Migration by Country

- We can see that the highest total migration is from India and China, with the UK, New Zealand and the Phillipines making up the other three places

3.4 Top 5 Over the Period

In [291…

```
# Plot the movement in the top five over the period

ax = (top5.iloc[:,0:-2].T/1000).plot(kind = 'line', figsize = (14,7))
plt.title('Migration to Australia by Top 5 (2005 to 2024)', fontsize = 20)
plt.ylabel("Immigration 000's")
ax.set_facecolor('white')
plt.legend(facecolor = 'white',edgecolor = 'white')
props = dict(boxstyle='square', facecolor = 'lightgray', alpha=0.1)
```



Top 5 Countries by Total Migration over the Period

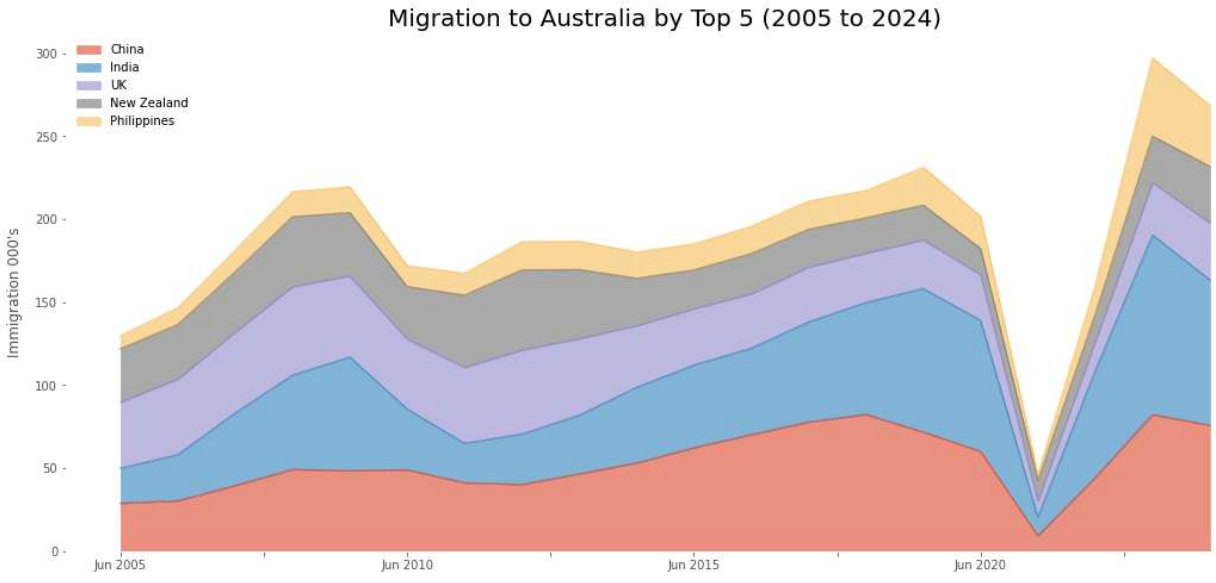
- If we look at the countries with the highest total migration over the time period we can see there is a clear trend upwards for China and India and a trend downwards for the UK and New Zealand. The Phillipines appears relatively constant over the period.
- All countries show a drop in 2021 over the lockdown period for Covid 19

In [292…

```
# Showing the data as an area chart
ax = (top5.iloc[:,0:-2].T/1000).plot(kind = 'area', alpha = 0.6,figsize = (18,8))
plt.title('Migration to Australia by Top 5 (2005 to 2024)', fontsize = 20)
plt.ylabel("Immigration 000's",labelpad = 10)
ax.set_facecolor('white')
plt.legend(facecolor = 'white', edgecolor = 'white')
;
```

Out[292…

''



- Looking at the area chart we can see the share of migration from China in India has been increasing significantly compared to the other countries.

3.5 Changes in the Top 5 Countries over the Period

In [293…

```
sorted_total.iloc[:]['Jun 2019']
```

Out[293…

```
Country
China              71810
India              86350
UK                 29300
New Zealand        20960
Philippines        22780
                   ...
Faroe Islands         10
Sint Maarten Dp        0
N Mariana Is           0
Virgin Is, Brit        0
St Kitts/Nevis         0
Name: Jun 2019, Length: 215, dtype: int64
```

In [294…

```
# Display the top five for each year
for i in range(len(sorted_total.columns[0:-2])):
    top = pd.DataFrame(sorted_total.iloc[:,i].sort_values(ascending = False).head(5))
    display(top)
```

	Jun 2005
Country	
UK	39870
New Zealand	32030
China	28870
India	21070
USA	10700

	Jun 2006
Country	
UK	45530
New Zealand	32770
China	30370
India	27770
Korea, South	13290

Jun 2007

Country	
UK	48650
India	43660
China	39540
New Zealand	36300
Korea, South	15500

Jun 2008

Country	
India	56810
UK	53330
China	49200
New Zealand	42070
Korea, South	17190

Jun 2009

Country	
India	68350
UK	48910
China	48540
New Zealand	38090
Korea, South	17030

Jun 2010

Country	
China	49020
UK	42100
India	36610
New Zealand	31580
Korea, South	14760

Jun 2011

Country	
UK	45750
New Zealand	43430
China	41210

Jun 2011

Country	
India	23760
Korea, South	13850

Jun 2012

Country	
UK	50610
New Zealand	48220
China	40150
India	30390
Philippines	16910

Jun 2013

Country	
China	46560
UK	46250
New Zealand	41500
India	35280
Philippines	17010

Jun 2014

Country	
China	53130
India	45520
UK	37190
New Zealand	28440
Philippines	15890

Jun 2015

Country	
China	62280
India	49810
UK	33920
New Zealand	23300
Philippines	15780

Jun 2016

Country	
China	70190
India	52020
UK	32790
New Zealand	24220
Malaysia	19260

Jun 2017

Country	
China	77760
India	60310
UK	33080
New Zealand	22660
Malaysia	20140

Jun 2018

Country	
China	82320
India	67440
UK	29740
Nepal	22070
New Zealand	21270

Jun 2019

Country	
India	86350
China	71810
UK	29300
Nepal	25470
Philippines	22780

Jun 2020

Country	
India	78830
China	60150
UK	27670

Jun 2020

Country	
Philippines	19460
Nepal	15910

Jun 2021

Country	
New Zealand	12000
India	11190
UK	9850
China	9260
USA	4650

Jun 2022

Country	
India	64380
China	44300
Nepal	23630
UK	17500
Philippines	17360

Jun 2023

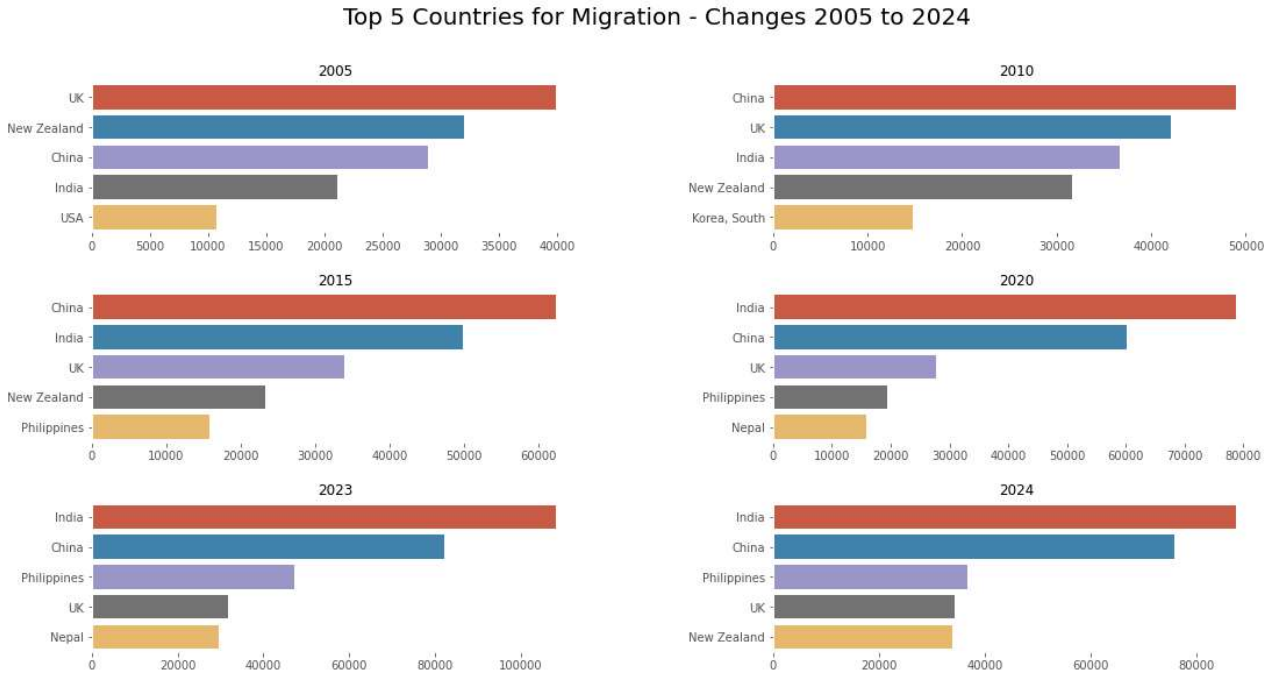
Country	
India	108140
China	82240
Philippines	47200
UK	31700
Nepal	29660

Jun 2024

Country	
India	87600
China	75830
Philippines	36830
UK	34410
New Zealand	33870

In [295…

```
# We can show some of these movements by showing bar charts of the top five in 2005, 201

# Split off the data we need into separate dataframes by years at 5 year periods
df2005 = pd.DataFrame(sorted_total.iloc[:,0].sort_values(ascending = False).head(5))
df2010 = pd.DataFrame(sorted_total.iloc[:,5].sort_values(ascending = False).head(5))
df2015 = pd.DataFrame(sorted_total.iloc[:,10].sort_values(ascending = False).head(5))
df2020 = pd.DataFrame(sorted_total.iloc[:,15].sort_values(ascending = False).head(5))
df2023 = pd.DataFrame(sorted_total.iloc[:,18].sort_values(ascending = False).head(5))
df2024 = pd.DataFrame(sorted_total.iloc[:,19].sort_values(ascending = False).head(5))


# Set the figure details
fig, axs = plt.subplots(nrows = 3,ncols = 2, figsize = (18,9))
fig.suptitle("Top 5 Countries for Migration - Changes 2005 to 2024", fontsize=20)
plt.subplots_adjust(wspace=0.4,hspace = 0.4)

# Plot barplots
a = sns.barplot(y=df2005.index, x="Jun 2005", data=df2005, ax = axs[0,0])
b = sns.barplot(y=df2010.index, x="Jun 2010", data=df2010, ax = axs[0,1])
c = sns.barplot(y=df2015.index, x="Jun 2015", data=df2015, ax = axs[1,0])
d = sns.barplot(y=df2020.index, x="Jun 2020", data=df2020, ax = axs[1,1])
e = sns.barplot(y=df2023.index, x="Jun 2023", data=df2023, ax = axs[2,0])
f = sns.barplot(y=df2024.index, x="Jun 2024", data=df2024, ax = axs[2,1])


plots = [a,b,c,d,e,f]

# Cycle over the plots and remove axis labels and set facecolour
for p in plots:
    p.set(xlabel = None)
    p.set(ylabel = None)
    p.set_facecolor('white')

# Add titles to the plots
axs[0,0].set_title("2005", fontsize = 12)
axs[0,1].set_title("2010", fontsize = 12)
axs[1,0].set_title("2015", fontsize = 12)
axs[1,1].set_title("2020", fontsize = 12)
axs[2,0].set_title("2023", fontsize = 12)
axs[2,1].set_title("2024", fontsize = 12);
```



3.6 Trend Analysis of the Top Migration Countries Over the Period

In [296…

```
# Split out China and India and the UK and New Zealand
china_india = top5[(top5.index == 'China') | (top5.index == 'India')].T.reset_index()
uk_newzeal = top5[(top5.index == 'New Zealand') | (top5.index == 'UK')].T.reset_index()

# Drop the index column
china_india.drop(columns = ['index'],axis = 1,inplace = True)
uk_newzeal.drop(columns = ['index'],axis = 1,inplace = True)
```

In [297…

```
# Create a new date index
dates = []
for i in range(2005,2025):
    dates.append(i)
dates = pd.DataFrame(dates)
```

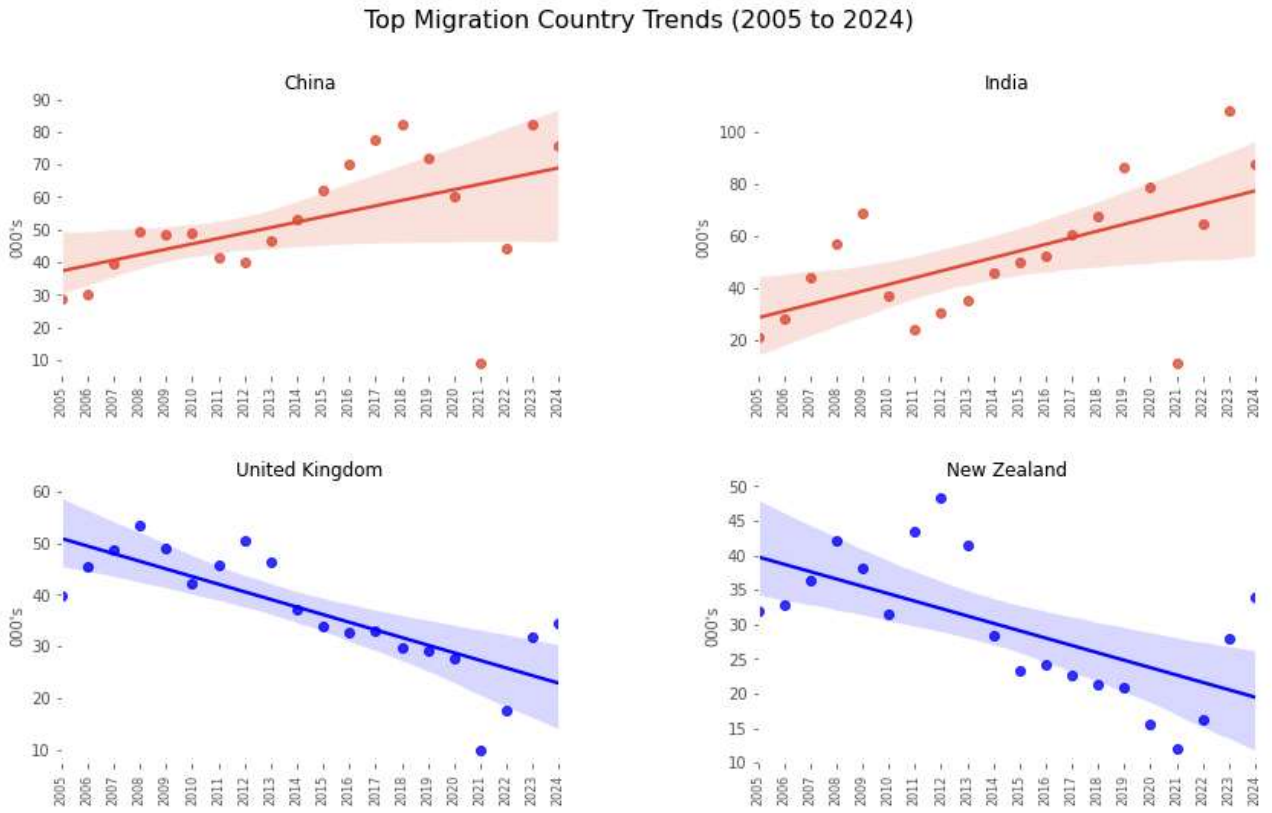
In [298…

```
# Concat the dates to the dataframes
new_china_india = pd.concat([dates,china_india],axis = 1).set_index(0)
new_uk_newzeal = pd.concat([dates,uk_newzeal],axis = 1).set_index(0)

# Remove index name
new_china_india.index.name = None
new_uk_newzeal.index.name = None

# Drop the last two row which are total and average
new_china_india2 = new_china_india.iloc[0:-2,:]
new_uk_newzeal2 = new_uk_newzeal.iloc[0:-2,:]
```

In [299…

```
# Set Xtick labels
labs = ['2005','2006','2007','2008','2009','2010','2011','2012','2013','2014','2015','2
        '2020','2021','2022','2023','2024']
```

```
# Set the figure details
fig, axs = plt.subplots(nrows = 2,ncols = 2, figsize = (14,8))
fig.suptitle("Top Migration Country Trends (2005 to 2024)", fontsize=16)

# Plot a regplot to show the trend and the separate datapoints
# India and China
a = sns.regplot(x=new_china_india2.index, y="China", data=new_china_india2/1000, ax = a
b = sns.regplot(x=new_china_india2.index, y="India", data=new_china_india2/1000, ax = a

# UK and New Zealand
c = sns.regplot(x=new_uk_newzeal2.index, y="UK", data=new_uk_newzeal2/1000, ax = axs[1,
d = sns.regplot(x=new_uk_newzeal2.index, y="New Zealand", data=new_uk_newzeal2/1000, ax

plots2 = [a,b,c,d]

# Cycle over the plots and set ylabels and facecolor
for p in plots2:
    p.set_ylabel("000's", fontsize = 10)
    p.set_facecolor('white')

# Set the titles
axs[0,0].set_title("China", fontsize = 12)
axs[0,1].set_title("India", fontsize = 12)
axs[1,0].set_title("United Kingdom", fontsize = 12)
axs[1,1].set_title("New Zealand", fontsize = 12)

# Set the Xticks
axs[0,0].set_xticks(new_china_india2.index,labs,rotation=90, fontsize = 8)
axs[0,1].set_xticks(new_china_india2.index,labs,rotation=90, fontsize = 8)
axs[1,0].set_xticks(new_uk_newzeal2.index,labs,rotation=90, fontsize = 8)
axs[1,1].set_xticks(new_uk_newzeal2.index,labs,rotation=90, fontsize = 8)

# Adjust spacing between plots
plt.subplots_adjust(wspace=0.4,hspace = 0.4)
```



In [300…

```
#calculate slope and intercept of regression equation
import scipy

for i in [a,b,c,d]:
    slope, intercept, r, p, sterr = scipy.stats.linregress(x=i.get_lines()[0].get_xdata
                                                           y=i.get_lines()[0].get_ydata())
    print(intercept,slope)
```

```
-3301.67245112775 1.6653308270676355
-5128.75669924802 2.572112781954838
3011.021390977246 -1.47635338345855
2183.1865864660253 -1.0690300751879005
```

We can see that the slope of the lines for China, India, UK and New Zealand respectively show a greater value for India than China which has an increase similar in rate to the decrease for the UK

Trends in the Top Countries for Immigration over the Period

- We can clearly see the trends here over the period with migration from the UK and New Zealand reducing while migration from China and India is upwards

3.7 Comparison Over Time for China, India, UK and New Zealand

In [301…

```
# Reset index and set the type of year to a string for both subsets

# China and India
new_china_india2_reset = new_china_india2.reset_index()
new_china_india2_reset.columns = ['Year','China','India']
new_china_india2_reset['Year'] = new_china_india2_reset['Year'].astype('int').astype('o
```

```
display(new_china_india2_reset.head())

# UK and New Zealand
new_uk_newzeal2_reset = new_uk_newzeal2.reset_index()
new_uk_newzeal2_reset.columns = ['Year','UK','New Zealand']
new_uk_newzeal2_reset['Year'] = new_uk_newzeal2_reset['Year'].astype('int').astype('obj
display(new_uk_newzeal2_reset.head())
```

	Year	China	India
0	2005	28870	21070
1	2006	30370	27770
2	2007	39540	43660
3	2008	49200	56810
4	2009	48540	68350

	Year	UK	New Zealand
0	2005	39870	32030
1	2006	45530	32770
2	2007	48650	36300
3	2008	53330	42070
4	2009	48910	38090

In [302…

```
new_china_india2.median()
```

Out[302…

```
China    49110.0
India    50915.0
dtype: float64
```

In [303…

```
# Melt the dataframes

melted_india_china = pd.melt(new_china_india2_reset, id_vars = ['Year'],
                             value_vars = ['China','India'],
                             var_name = 'Country',
                             value_name = 'Migration').sort_values(by = 'Year')

melted_uk_nz = pd.melt(new_uk_newzeal2_reset, id_vars = ['Year'],
                             value_vars = ['UK','New Zealand'],
                             var_name = 'Country',
                             value_name = 'Migration').sort_values(by = 'Year')
```

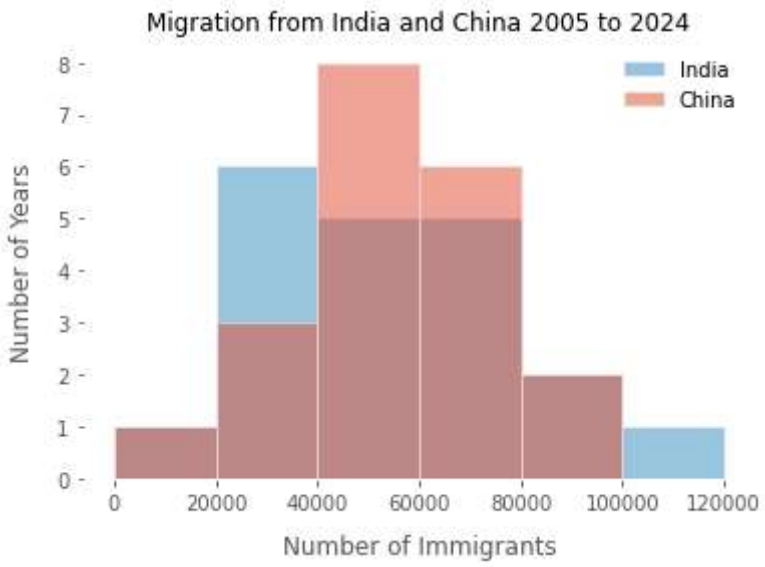
In [304…

```
# Plot a histogram with both countries on it
l = ['India','China']
bins = [0,20000,40000,60000,80000,100000,120000]
ax = sns.histplot(melted_india_china, x = 'Migration', hue = 'Country', bins = bins)
plt.title('Migration from India and China 2005 to 2024',fontsize = 12)
plt.ylabel("Number of Years",labelpad = 10)
plt.xlabel("Number of Immigrants",labelpad = 10)
ax.set_facecolor("white")
```

```
plt.legend(labels = l,loc='upper right', facecolor = 'white', edgecolor = 'white')
;
```

Out[304…

```
''
```

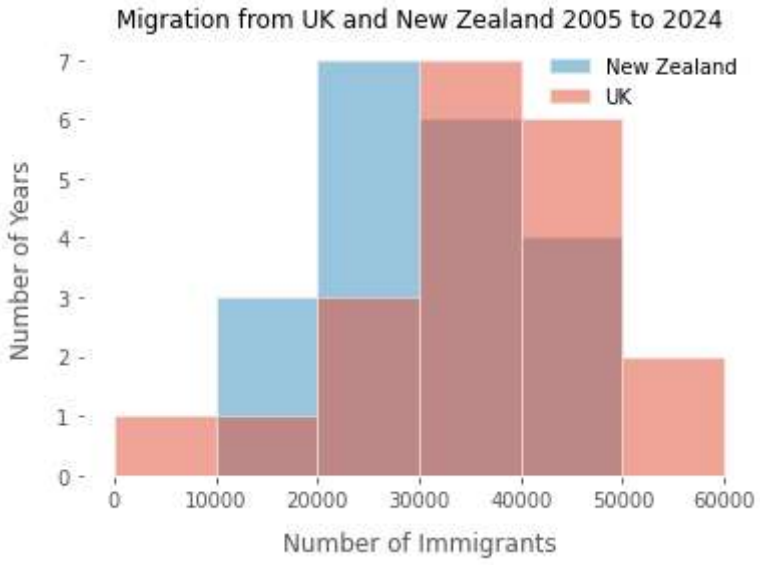


In [305…

```
# Plot a histogram with both countries on it
bins = [0,10000,20000,30000,40000,50000,60000]
l2 = ['New Zealand','UK']
ax = sns.histplot(melted_uk_nz, x = 'Migration', hue = 'Country', bins = bins)
plt.title('Migration from UK and New Zealand 2005 to 2024',fontsize = 12)
plt.ylabel("Number of Years",labelpad = 10)
plt.xlabel("Number of Immigrants",labelpad = 10)
ax.set_facecolor("white")
plt.legend(labels = l2,loc='upper right', facecolor = 'white', edgecolor = 'white')
;
```

Out[305…

```
''
```



In [306…

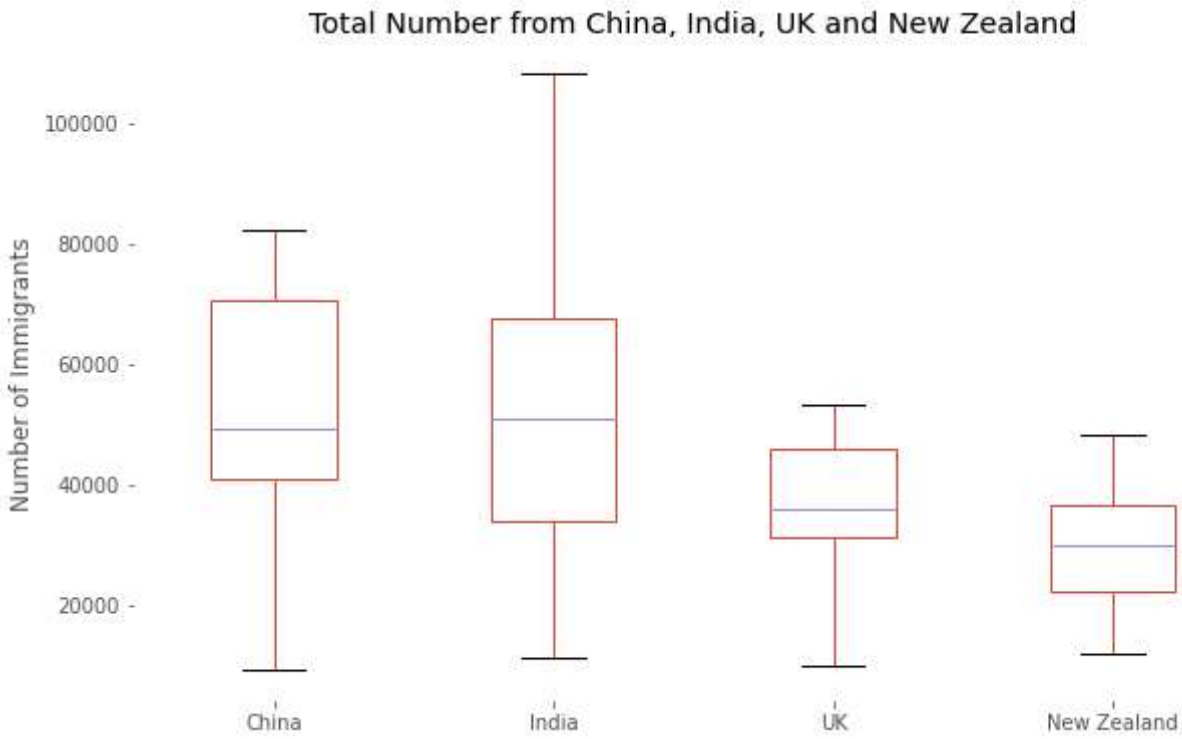
```
all_four = pd.concat([new_china_india2, new_uk_newzeal2],axis = 1)
all_four
```

Out[306…

	China	India	UK	New Zealand
2005.0	28870	21070	39870	32030
2006.0	30370	27770	45530	32770
2007.0	39540	43660	48650	36300
2008.0	49200	56810	53330	42070
2009.0	48540	68350	48910	38090
2010.0	49020	36610	42100	31580
2011.0	41210	23760	45750	43430
2012.0	40150	30390	50610	48220
2013.0	46560	35280	46250	41500
2014.0	53130	45520	37190	28440
2015.0	62280	49810	33920	23300
2016.0	70190	52020	32790	24220
2017.0	77760	60310	33080	22660
2018.0	82320	67440	29740	21270
2019.0	71810	86350	29300	20960
2020.0	60150	78830	27670	15640
2021.0	9260	11190	9850	12000
2022.0	44300	64380	17500	16280
2023.0	82240	108140	31700	27880
2024.0	75830	87600	34410	33870

In [307…

```
# Boxplots to compare the different countries migration
ax = all_four.plot(kind = 'box',
                   figsize = (10,6),
                   title = 'Total Number from China, India, UK and New Zealand',
                   ylabel = 'Number of Immigrants')
ax.set_facecolor('white');
```



3.8 Comparison by five year periods - all countries

In [308…

```
# Get figures by decade
y_2005 = pd.DataFrame(sorted_total.iloc[:,0])
y_2010 = pd.DataFrame(sorted_total.iloc[:,5])
y_2015 = pd.DataFrame(sorted_total.iloc[:,10])
y_2020 = pd.DataFrame(sorted_total.iloc[:,15])
y_2024 = pd.DataFrame(sorted_total.iloc[:,19])


all_years = pd.concat([y_2005,y_2010,y_2015, y_2020,y_2024],axis = 1)
```

In [309…

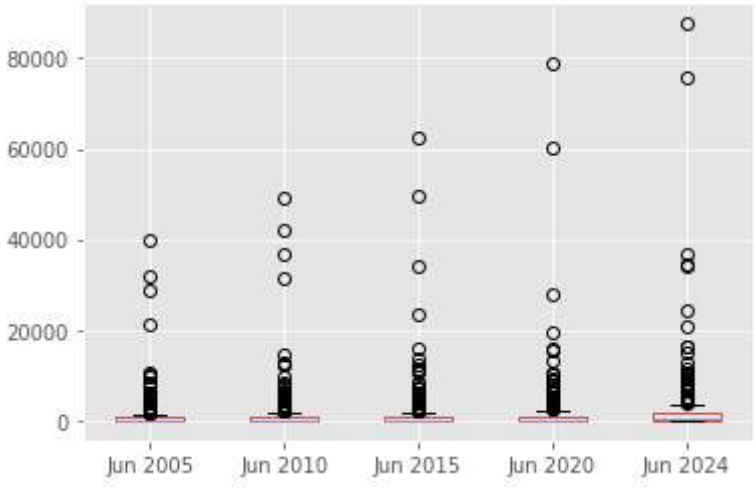
```
# Get basic statistics
all_years.describe()
```

Out[309…

	Jun 2005	Jun 2010	Jun 2015	Jun 2020	Jun 2024
count	215.000000	215.000000	215.000000	215.000000	215.000000
mean	1368.000000	1783.441860	1948.558140	2075.348837	2935.534884
std	4497.470338	5820.706212	6465.313724	7433.204158	9371.621837
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	20.000000	20.000000	30.000000	20.000000
50%	120.000000	150.000000	130.000000	140.000000	190.000000
75%	645.000000	780.000000	830.000000	1025.000000	1520.000000
max	39870.000000	49020.000000	62280.000000	78830.000000	87600.000000

In [310…

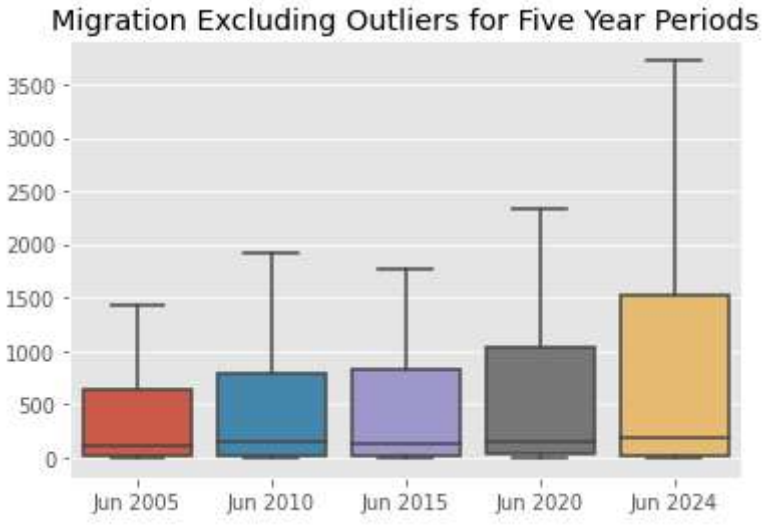
```
all_years.plot(kind = 'box');
```



Due to the outliers, it is difficult to see the spread of the other countries. We could do a log transform or there is a parameter in the seaborn function to set the whiskers parameter and showfliers = False.

In [311…

```
sns.boxplot(data = all_years,showfliers = False)
plt.title('Migration Excluding Outliers for Five Year Periods');
```



3.9 Immigration in relation to Population and GDP

For this part of the analysis, we are going to bring in two more tables of data, relating to population and gdp, also from the World Bank. We will need to do some further cleaning to get the data into a more useable format before being able to combine these three features to create some more charts.

In [312…

```
# Add a total migration column and an average column to our original cleaned dataframe
df_average = df_cleaned.copy()

# Drop those with zero migration
df_non_zero = df_average[df_average['Average']!=0]
```

```
df_average.head(2)
```

Out[312…

Country	Jun 2005	Jun 2006	Jun 2007	Jun 2008	Jun 2009	Jun 2010	Jun 2011	Jun 2012	Jun 2013	Jun 2014	...	Jun 2017	Jun 2018	2(
Norfolk Island(g)	30	30	40	40	30	30	30	30	40	30	...	80	10	
New Zealand	32030	32770	36300	42070	38090	31580	43430	48220	41500	28440	...	22660	21270	20

2 rows × 22 columns

In [313…

```
# Load data on population and GDP
pop = pd.read_csv(r'C:\Users\imoge\AllMLProjects\Data\WorldBankPopulationConverted.csv'
gdp = pd.read_csv(r'C:\Users\imoge\AllMLProjects\Data\WorldBankGDPConverted.csv',skipro
```

In [314…

```
# Show head and tail for each dataset
display(pop.head(2))
display(pop.tail(2))
display(gdp.head(2))
display(gdp.tail(2))

print(pop.shape, gdp.shape)
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54922.0	55578.0	56320.0	57002.0	
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130075728.0	133534923.0	137171659.0	140945536.0	1449

2 rows × 70 columns

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964
264	Zambia	ZMB	Population, total	SP.POP.TOTL	3153729.0	3254086.0	3358099.0	3465907.0	3577017.0
265	Zimbabwe	ZWE	Population, total	SP.POP.TOTL	3809389.0	3930401.0	4055959.0	4185877.0	4320006.0

2 rows × 70 columns

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	19
0	Aruba	ABW	GDP (current US$)	NY.GDP.MKTP.CD	NaN	NaN	NaN	Na
1	Africa Eastern and Southern	AFE	GDP (current US$)	NY.GDP.MKTP.CD	2.420993e+10	2.496326e+10	2.707802e+10	3.177483e+

2 rows × 70 columns

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	
264	Zambia	ZMB	GDP (current US$)	NY.GDP.MKTP.CD	6.987397e+08	6.823597e+08	6.792797e+08	7.04339
265	Zimbabwe	ZWE	GDP (current US$)	NY.GDP.MKTP.CD	1.052990e+09	1.096647e+09	1.117602e+09	1.15951

2 rows × 70 columns

```
(266, 70) (266, 70)
```

In [315…

```
# Get the columns from 2005 onwards
pop_filtered = pop.iloc[:,50:-1]
gdp_filtered = gdp.iloc[:,50:-1]

# Get the index from the previous dataframe
ind1 = pop['Country Name']
ind2 = gdp['Country Name']

# Set this as the index for the new dataframe
pop_filtered.index = ind1
gdp_filtered.index = ind2

# Display the filtered dataframes
display(pop_filtered.head(2))
display(gdp_filtered.head(2))
```

Country Name	2006	2007	2008	2009	2010	2011	2012
Aruba	99405.0	100150.0	100917.0	101604.0	101838.0	102591.0	104110.0
Africa Eastern and Southern	475606210.0	488580707.0	502070763.0	516003448.0	530308387.0	544737983.0	559609961.0

Country Name	2006	2007	2008	2009	2010	2011
Aruba	2.469783e+09	2.677641e+09	2.843025e+09	2.553793e+09	2.453597e+09	2.637859e+09
Africa Eastern and Southern	5.802408e+11	6.655987e+11	7.135021e+11	7.154853e+11	8.494096e+11	9.454390e+11

In [316…

```
# List of rows to exclude
to_exclude = ['Africa Eastern and Southern','Africa Western and Central','Arab World','
              'Central Europe and the Baltics','Caribbean small states','East Asia & Pa
              'Early-demographic dividend','East Asia & Pacific','Europe & Central Asia
              'Europe & Central Asia', 'Euro area','European Union','Fragile and confli
              'High income','Heavily indebted poor countries (HIPC)','IBRD only','IDA &
              'IDA only','Not classified','Latin America & Caribbean (excluding high in
              'Least developed countries: UN classification','Low income','Lower middle
              'Late-demographic dividend','Middle East, North Africa, Afghanistan & Pak
              'Middle East, North Africa, Afghanistan & Pakistan (excluding high income
              'Other small states','Pre-demographic dividend','Pacific island small sta
              'South Asia','Sub-Saharan Africa (excluding high income)','Sub-Saharan Af
              'East Asia & Pacific (IDA & IBRD countries)','Europe & Central Asia (IDA &
              'Latin America & the Caribbean (IDA & IBRD countries)','Middle East, Nort
              'South Asia (IDA & IBRD)','Sub-Saharan Africa (IDA & IBRD countries)','Up
```

In [317…

```
# Filter those out
pop_filtered = pop_filtered[~pop_filtered.index.isin(to_exclude)]
gdp_filtered = gdp_filtered[~gdp_filtered.index.isin(to_exclude)]
```

In [318…

```
# Check the shapes
print(pop_filtered.shape)
print(gdp_filtered.shape)
```

```
(216, 19)
(216, 19)
```

In [319…

```
# Check the datatypes
pop_filtered.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 216 entries, Aruba to Zimbabwe
Data columns (total 19 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   2006    216 non-null    float64
 1   2007    216 non-null    float64
 2   2008    216 non-null    float64
 3   2009    216 non-null    float64
 4   2010    216 non-null    float64
 5   2011    216 non-null    float64
 6   2012    216 non-null    float64
 7   2013    216 non-null    float64
 8   2014    216 non-null    float64
 9   2015    216 non-null    float64
 10  2016    216 non-null    float64
 11  2017    216 non-null    float64
 12  2018    216 non-null    float64
 13  2019    216 non-null    float64
 14  2020    216 non-null    float64
 15  2021    216 non-null    float64
 16  2022    216 non-null    float64
 17  2023    216 non-null    float64
 18  2024    216 non-null    float64
dtypes: float64(19)
memory usage: 33.8+ KB
```

In [320…

```
gdp_filtered.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 216 entries, Aruba to Zimbabwe
Data columns (total 19 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   2006    209 non-null    float64
 1   2007    209 non-null    float64
 2   2008    210 non-null    float64
 3   2009    212 non-null    float64
 4   2010    212 non-null    float64
 5   2011    213 non-null    float64
 6   2012    211 non-null    float64
 7   2013    211 non-null    float64
 8   2014    212 non-null    float64
 9   2015    210 non-null    float64
 10  2016    209 non-null    float64
 11  2017    209 non-null    float64
 12  2018    209 non-null    float64
 13  2019    209 non-null    float64
 14  2020    208 non-null    float64
 15  2021    208 non-null    float64
 16  2022    207 non-null    float64
 17  2023    199 non-null    float64
 18  2024    182 non-null    float64
dtypes: float64(19)
memory usage: 33.8+ KB
```

In [321…

```
# Check the missing values in the gdp dataframe for the last few years
gdp_filtered.isnull().sum().tail(20)
```

Out[321…

```
2006     7
2007     7
2008     6
2009     4
2010     4
2011     3
2012     5
2013     5
2014     4
2015     6
2016     7
2017     7
2018     7
2019     7
2020     8
2021     8
2022     9
2023    17
2024    34
dtype: int64
```

- Because we have 2023 and 2024 missing for both population figures, we are going to calculate an average population figure for each country over the period and an average gdp and compare this to an average migration figure in our chart.
- We also have some missing figures in the GDP which we will need to take account of

In [322…

```
# Add a total column to the population and gdp dataframes
pop_filtered['Total'] = pop_filtered.sum(axis=1)
gdp_filtered['Total'] = gdp_filtered.sum(axis = 1)

# Display results
display(pop_filtered.head(2))
display(gdp_filtered.head(2))
```

Country Name	2006	2007	2008	2009	2010	2011	2012	2
Aruba	99405.0	100150.0	100917.0	101604.0	101838.0	102591.0	104110.0	1056
Afghanistan	25424094.0	25909852.0	26482622.0	27466101.0	28284089.0	29347708.0	30560034.0	316227

Country Name	2006	2007	2008	2009	2010	2011	
Aruba	2.469783e+09	2.677641e+09	2.843025e+09	2.553793e+09	2.453597e+09	2.637859e+09	2.61
Afghanistan	6.971758e+09	9.747886e+09	1.010930e+10	1.241615e+10	1.585667e+10	1.780510e+10	1.99(

In [323…

```
# Now calculate an average population  and an average gdp column (excluding the total c
pop_filtered['Average_Pop'] = pop_filtered['Total']/len(pop_filtered.columns)-1
gdp_filtered['Average_GDP'] = gdp_filtered['Total']/len(pop_filtered.columns)-1

display(pop_filtered.head(2))
display(gdp_filtered.head(2))
```

Country Name	2006	2007	2008	2009	2010	2011	2012	2
Aruba	99405.0	100150.0	100917.0	101604.0	101838.0	102591.0	104110.0	1056
Afghanistan	25424094.0	25909852.0	26482622.0	27466101.0	28284089.0	29347708.0	30560034.0	316227

2 rows × 21 columns

Country Name	2006	2007	2008	2009	2010	2011
Aruba	2.469783e+09	2.677641e+09	2.843025e+09	2.553793e+09	2.453597e+09	2.637859e+09	2.61
Afghanistan	6.971758e+09	9.747886e+09	1.010930e+10	1.241615e+10	1.585667e+10	1.780510e+10	1.99(

2 rows × 21 columns

In [324…

```
# Reset index, select the average from migration dataframe, rename columns for consiste
migration_average = df_cleaned[['Average']].reset_index()
migration_average.columns = ['Country','Average_Mig']
migration_average = migration_average[migration_average['Average_Mig']>0]

# Reset index for other two dataframes
pop_average = pop_filtered[['Average_Pop']].reset_index()
gdp_average = gdp_filtered[['Average_GDP']].reset_index()

# Display all the dataframes
display(migration_average.head())
display(pop_average.head())
display(gdp_average.head())

# Check the shape of the dataframes
print(migration_average.shape)
print(pop_average.shape)
print(gdp_average.shape)
```

	Country	Average_Mig
0	Norfolk Island(g)	26

	Country	Average_Mig
1	New Zealand	29625
2	New Caledonia	146
3	PNG	1486
4	Solomon Islands	473

	Country Name	Average_Pop
0	Aruba	100256.80
1	Afghanistan	32023033.20
2	Angola	27015352.35
3	Albania	2728940.55
4	Andorra	74053.80

	Country Name	Average_GDP
0	Aruba	2.467609e+09
1	Afghanistan	1.391334e+10
2	Angola	7.716270e+10
3	Albania	1.333138e+10
4	Andorra	3.067741e+09

```
(215, 2)
(216, 2)
(216, 2)
```

- We will merge the tables on matches in country name.
- We are using a left join as we want to see where we are missing values

In [325…

```
# Clean up the text first to make sure we don't have any blanks
pop_average['Country Name'].str.strip()
gdp_average['Country Name'].str.strip()
migration_average['Country'].str.strip()

# Merge the population with the migration table to create one new table with the data we
merged_pop_migration_gdp = migration_average.merge(pop_average,
                                                   left_on='Country',
                                                   right_on='Country Name',
                                                   how = 'left').merge(gdp_average,
                                                                       left_on='Country
                                                                       right_on='Countr
                                                                       how = 'left')

merged_pop_migration_gdp.head()
```

Out[325…

	Country	Average_Mig	Country Name_x	Average_Pop	Country Name_y	Average_GDP
0	Norfolk Island(g)	26	NaN	NaN	NaN	NaN
1	New Zealand	29625	New Zealand	4449559.00	New Zealand	1.723478e+11

	Country	Average_Mig	Country Name_x	Average_Pop	Country Name_y	Average_GDP
2	New Caledonia	146	New Caledonia	261171.85	New Caledonia	7.566223e+09
3	PNG	1486	NaN	NaN	NaN	NaN
4	Solomon Islands	473	Solomon Islands	611105.25	Solomon Islands	1.147713e+09

In [326…

```
# Check the info
merged_pop_migration_gdp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 215 entries, 0 to 214
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Country         215 non-null    object 
 1   Average_Mig     215 non-null    int64  
 2   Country Name_x  162 non-null    object 
 3   Average_Pop     162 non-null    float64
 4   Country Name_y  162 non-null    object 
 5   Average_GDP     162 non-null    float64
dtypes: float64(2), int64(1), object(3)
memory usage: 11.8+ KB
```

In [327…

```
# How many are missing from the table?
merged_pop_migration_gdp[pd.isnull(merged_pop_migration_gdp).any(axis=1)].shape
```

Out[327…

```
(53, 6)
```

- We can see which ones are missing, sorted by average migration descending and then only look at those with at least 100 people as an average

In [328…

```
# Which ones are missing? Sorted by average migration and with levels greater than 100
missing = merged_pop_migration_gdp[pd.isnull(merged_pop_migration_gdp).any(axis=1)].sor
missing[missing['Average_Mig']>=100]
```

Out[328…

	Country	Average_Mig	Country Name_x	Average_Pop	Country Name_y	Average_GDP
23	UK	36907	NaN	NaN	NaN	NaN
108	Korea, South	11771	NaN	NaN	NaN	NaN
94	Vietnam	11046	NaN	NaN	NaN	NaN
127	USA	11005	NaN	NaN	NaN	NaN
105	Taiwan	7258	NaN	NaN	NaN	NaN
102	Hong Kong	5955	NaN	NaN	NaN	NaN
77	Iran	3754	NaN	NaN	NaN	NaN
86	Syria	1555	NaN	NaN	NaN	NaN
87	Türkiye	1519	NaN	NaN	NaN	NaN
3	PNG	1486	NaN	NaN	NaN	NaN

	Country	Average_Mig	Country Name_x	Average_Pop	Country Name_y	Average_GDP
69	Egypt	1430	NaN	NaN	NaN	NaN
65	Russia	1330	NaN	NaN	NaN	NaN
88	Unit Arab Emir	1069	NaN	NaN	NaN	NaN
178	Congo, Dem Rep	517	NaN	NaN	NaN	NaN
141	Venezuela	489	NaN	NaN	NaN	NaN
13	Cook Islands	391	NaN	NaN	NaN	NaN
92	Laos	342	NaN	NaN	NaN	NaN
46	Bosnia/Herzegov	337	NaN	NaN	NaN	NaN
66	Slovakia	299	NaN	NaN	NaN	NaN
95	Brunei	228	NaN	NaN	NaN	NaN
103	Macau	219	NaN	NaN	NaN	NaN

- Some of these will just be countries spelt or entered differently in the two dataframes.
- We will add in some population and gdp figures on some of these by referencing back to the spreadsheet of population.

In [329…

```
# Fix up the migration table matching the country names to our migration table for the p

list_names = [pop_average,gdp_average]

for i in list_names:
    i['Country Name'].replace({'United Kingdom': 'UK',
                               'Korea, Rep.': 'Korea, South',
                               'United States': 'USA',
                               'Viet Nam':'Vietnam',
                               'Hong Kong SAR, China':'Hong Kong',
                               'Iran, Islamic Rep.':'Iran',
                               'Syrian Arab Republic':'Syria',
                               'Turkiye':'Türkiye',
                               'Papua New Guinea': 'PNG',
                               'Egypt, Arab Rep.': 'Egypt',
                               'Russian Federation':'Russia',
                               'United Arab Emirates':'Unit Arab Emir',
                               'Venezuela, RB': 'Venezuela',
                               'Congo, Dem. Rep.': 'Congo, Dem Rep',
                               'Bosnia and Herzegovina':'Bosnia/Herzegov',
                               'Lao PDR':'Laos',
                               'Slovak Republic':'Slovakia',
                               'Brunei Darussalam': 'Brunei',
                               'Macao SAR, China': 'Macau',
                                }, inplace=True)
```

In [330…

```
# Merge the population with the migration table again
merged_pop_migration_gdp = migration_average.merge(pop_average, left_on='Country', righ
```

```
# How many are missing?
merged_pop_migration_gdp[pd.isnull(merged_pop_migration_gdp).any(axis=1)].shape
```

Out[330…

```
(34, 6)
```

In [331…

```
# Drop the Country Name column
merged_pop_migration_gdp.drop(['Country Name_x','Country Name_y'], axis = 1, inplace =
```

In [332…

```
# Look at the table
merged_pop_migration_gdp.head()
```

Out[332…

	Country	Average_Mig	Average_Pop	Average_GDP
0	Norfolk Island(g)	26	NaN	NaN
1	New Zealand	29625	4449559.00	1.723478e+11
2	New Caledonia	146	261171.85	7.566223e+09
3	PNG	1486	8279301.60	1.896404e+10
4	Solomon Islands	473	611105.25	1.147713e+09

In [333…

```
# Which ones are missing now?
merged_pop_migration_gdp[pd.isnull(merged_pop_migration_gdp).any(axis=1)].sort_values(b
```

Out[333…

	Country	Average_Mig	Average_Pop	Average_GDP
105	Taiwan	7258	NaN	NaN
13	Cook Islands	391	NaN	NaN
177	Congo, Rep	86	NaN	NaN
166	Trinidad/Tobago	69	NaN	NaN
107	Korea, North	67	NaN	NaN

Taiwan and the Cook Islands have average migration over 100 per year so we will try to find some population and GDP average figures to input into the missing fields.

Taiwan population - sourced from Macro Trends. (Average from 2005 to 2024)

Cook Islands population - sourced from Worldometer (Average of every five years from 2005)

Taiwan GDP - sourced from Statista

Cook Islands GDP - sourced from UN Data

In [334…

```
# Add in some figures for Taiwan and the Cook Islands (SOURCE: United Nations, latest p

merged_pop_migration_gdp.loc[merged_pop_migration_gdp['Country'] == 'Cook Islands', 'Av
merged_pop_migration_gdp.loc[merged_pop_migration_gdp['Country'] == 'Cook Islands', 'Av

merged_pop_migration_gdp.loc[merged_pop_migration_gdp['Country'] == 'Taiwan', 'Average_
```

```
merged_pop_migration_gdp.loc[merged_pop_migration_gdp['Country'] == 'Taiwan', 'Average_

# Drop nulls
merged_pop_migration_gdp.dropna(inplace = True)

# Check for nulls in the table now
merged_pop_migration_gdp.isnull().sum()
```

Out[334…

```
Country        0
Average_Mig    0
Average_Pop    0
Average_GDP    0
dtype: int64
```

In [335…

```
# Create a copy to work with
final = merged_pop_migration_gdp.copy()
final.columns = ['Country','Migration','Population','GDP']
```

In [336…

```
# Express GDP in millions
final['GDP'] = final['GDP']/1000000000

# Express population in millions
final['Population'] = final['Population']/1000000

# Change the column names to reflect these changes in numbers
final.rename(columns = {'GDP':'GDP(bn)','Population':'Pop(mn)'}, inplace = True)

# Add a GDP per capita column and a migration as % of population column
final['GDP per Capita'] = final['GDP(bn)']/final['Pop(mn)']*1000
final['% Pop'] = round((final['Migration']/(final['Pop(mn)']*1000000)*100),2)
```

In [337…

```
final.sort_values(by = 'Migration',ascending = False).head(10)
```

Out[337…

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop
101	China	53136	1304.326999	10081.880503	7729.565140	0.00
111	India	52764	1255.753205	2059.415837	1639.984536	0.00
23	UK	36907	61.668935	2611.185558	42341.991992	0.06
1	New Zealand	29625	4.449559	172.347771	38733.674722	0.67
98	Philippines	17534	98.929600	269.676628	2725.944783	0.02
97	Malaysia	12029	29.637164	281.156029	9486.603777	0.04
108	Korea, South	11771	48.101859	1217.251684	25305.709779	0.02
94	Vietnam	11046	87.928491	227.163787	2583.506028	0.01
127	USA	11005	304.408106	17418.358762	57220.417023	0.00
113	Nepal	10789	26.645374	23.314328	874.985951	0.04

3.10 Which countries have the highest migration as a percentage of population?

In [338…

```
# Sort by the % Pop descending
final.sort_values(by = '% Pop',ascending = False).head(10)
```

Out[338…

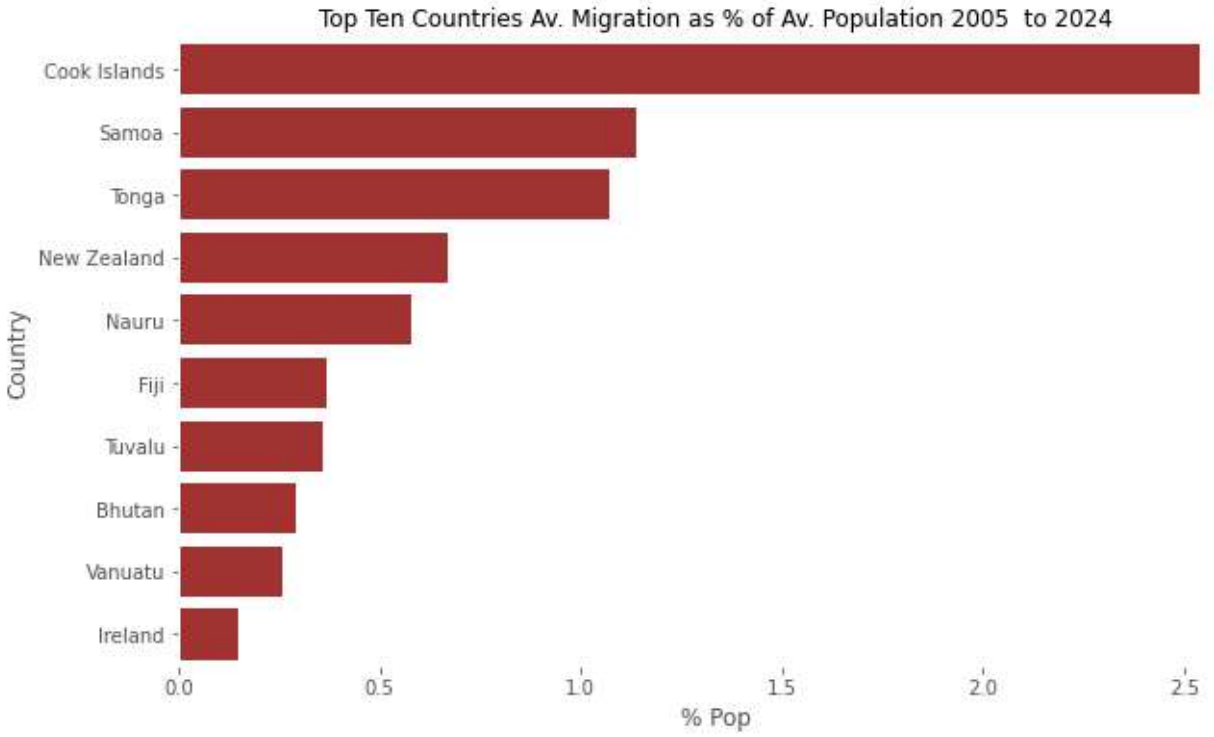
	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop
13	Cook Islands	391	0.015410	0.000017	1.124010	2.54
17	Samoa	2189	0.192216	0.715764	3723.743550	1.14
20	Tonga	1074	0.100826	0.368419	3654.006834	1.07
1	New Zealand	29625	4.449559	172.347771	38733.674722	0.67
10	Nauru	60	0.010360	0.088126	8506.688489	0.58
14	Fiji	3180	0.868184	3.988987	4594.631313	0.37
21	Tuvalu	36	0.009973	0.036175	3627.372291	0.36
110	Bhutan	2041	0.699407	1.816184	2596.750240	0.29
5	Vanuatu	665	0.255101	0.731674	2868.170006	0.26
24	Ireland	6863	4.546819	315.204421	69324.157527	0.15

In [339…

```
# Chart the countries with the highest average migration percentage of their population
top_percent = final.sort_values(by = '% Pop',ascending = False).head(10)

fig, ax = plt.subplots(figsize = (10,6))
ax = sns.barplot(data = top_percent, y = top_percent['Country'], x = top_percent['% Pop
plt.title("Top Ten Countries Av. Migration as % of Av. Population 2005  to 2024", fonts
ax.set_facecolor("white")
;
```

Out[339…

''



In [340…

```
# Look for high immigration source countries
high_immig = ['China','India','UK','New Zealand']

final[final['Country'].isin(high_immig)]
```

Out[340…

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop
1	New Zealand	29625	4.449559	172.347771	38733.674722	0.67
23	UK	36907	61.668935	2611.185558	42341.991992	0.06
101	China	53136	1304.326999	10081.880503	7729.565140	0.00
111	India	52764	1255.753205	2059.415837	1639.984536	0.00

Migration as % of Population

- We can see that the Cook Islands has the highest percentage for average yearly migration over population at 2.3%
- For some of the high immigration countries like China and India the percentage is quite small
- We cannot assume that the Cook Islands loses 2.3% of its population per year, so over the 19 year period as people will likely go backwards and forwards for work etc.
- Migration from the Cook Islands rather than New Zealand has coincided with declining economic conditions in New Zealand.

3.11 Average Migration by Country

In [341…

```
# Create interactive map of average migration
fig = px.choropleth(locationmode="country names",
```

```
                    locations = final.Country,
                    title = 'Average Annual Migration to Australia (2005 - 2024)',
                    color = final.Migration,
                    color_continuous_scale='viridis_r',
                    )
fig.update_layout(
    title_x=0.5,
    title_y=0.85)

fig.update_layout(coloraxis_colorbar_title_text = '')
fig.show()
```

3.12 Migration against GDP and Population

In [342...

```
# Have a look at the correlation coefficients
final[['Migration','Pop(mn)','GDP(bn)']].corr()
```

Out[342...

	Migration	Pop(mn)	GDP(bn)
Migration	1.000000	0.810031	0.469881
Pop(mn)	0.810031	1.000000	0.536986
GDP(bn)	0.469881	0.536986	1.000000

In [343…

```
# Calculate the correlation coefficient with a p-value
from scipy import stats
pearson_coef, p_value = stats.pearsonr(final['Migration'], final['Pop(mn)'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

```
The Pearson Correlation Coefficient is 0.8100312534446311  with a P-value of P = 8.01842
5017627745e-44
```

Since the p-value is <0.001, the correlation between migration and population is statistically significant, and the linear relationship is quite strong (~0.808, close to 1).

The P-value is the probability value that the correlation between these two variables is statistically significant. Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.

When the:

- p-value is 0.001: we say there is strong evidence that the correlation is significant.
- the p-value is 0.05: there is moderate evidence that the correlation is significant.
- the p-value is 0.1: there is weak evidence that the correlation is significant.
- the p-value is 0.1: there is no evidence that the correlation is significant.

In [344…

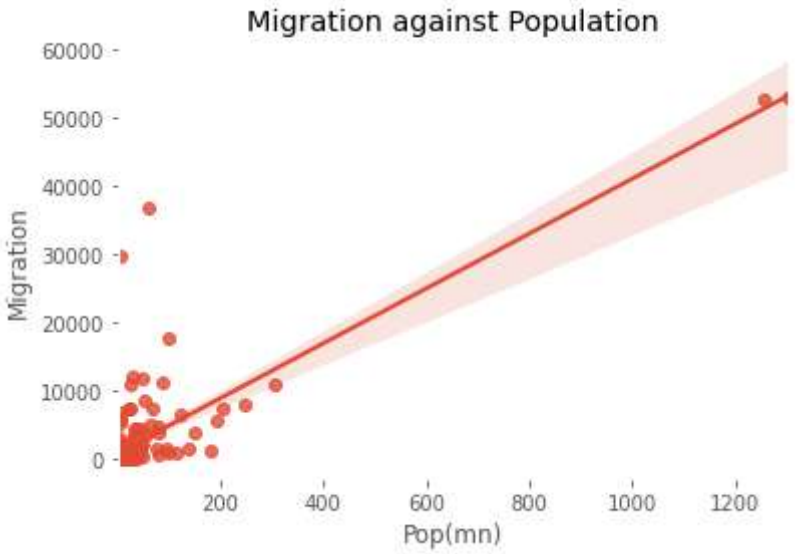
```
# Look at dataframe
final.head()
```

Out[344…

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop
1	New Zealand	29625	4.449559	172.347771	38733.674722	0.67
2	New Caledonia	146	0.261172	7.566223	28970.283835	0.06
3	PNG	1486	8.279302	18.964041	2290.536306	0.02
4	Solomon Islands	473	0.611105	1.147713	1878.093154	0.08
5	Vanuatu	665	0.255101	0.731674	2868.170006	0.26

In [345…

```
# Scatterplot of Migration against Population

ax = sns.regplot(data =final,
                 x = 'Pop(mn)',
                 y = 'Migration',
                 )
ax.set_facecolor("white")
plt.title('Migration against Population');
```



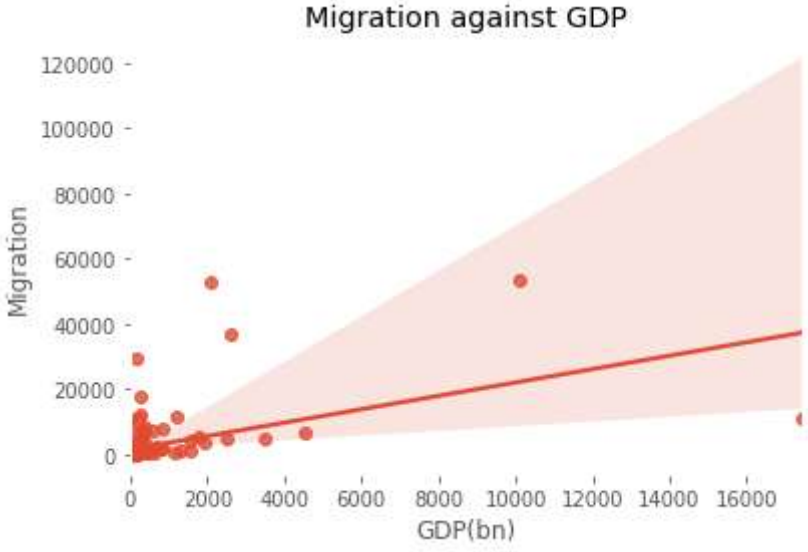
In [346…

```
# Get the slope of the line
slope, intercept, r, p, sterr = scipy.stats.linregress(x=ax.get_lines()[0].get_xdata(),
                                                       y=ax.get_lines()[0].get_ydata())
print(intercept,slope)
```

```
807.3014289479579 40.27131016850453
```

In [347…

```
# Scatterplot of Migration against GDP

ax = sns.regplot(data =final,
                 x = 'GDP(bn)',
                 y = 'Migration',
                 )
ax.set_facecolor("white")
plt.title('Migration against GDP');
```



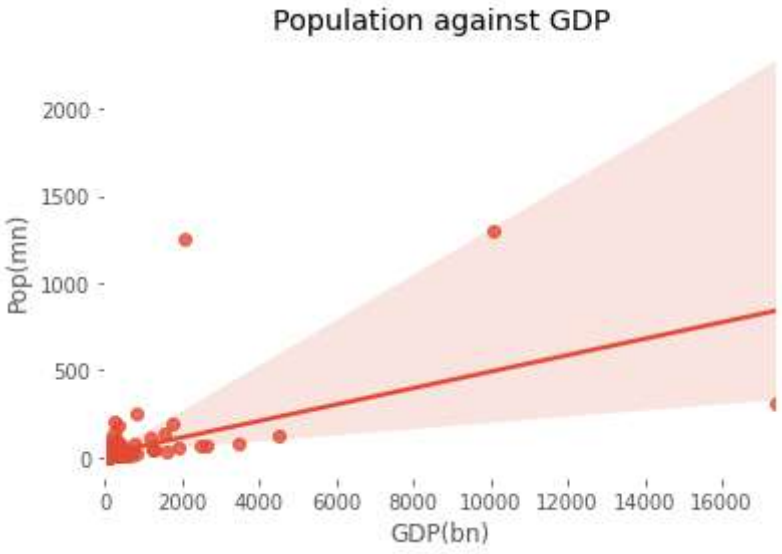
In [348…

```
# Get the slope of the line
slope, intercept, r, p, sterr = scipy.stats.linregress(x=ax.get_lines()[0].get_xdata(),
                                                       y=ax.get_lines()[0].get_ydata())
print(intercept,slope)
```

```
1539.008552815998 2.051814935916987
```

In [349…

```
# Scatterplot of Population against GDP

ax = sns.regplot(data =final,
                 x = 'GDP(bn)',
                 y = 'Pop(mn)',
                 )
ax.set_facecolor("white")
plt.title('Population against GDP');
```



In [350…

```
# Get the slope of the line
slope, intercept, r, p, sterr = scipy.stats.linregress(x=ax.get_lines()[0].get_xdata(),
                                                       y=ax.get_lines()[0].get_ydata())
print(intercept,slope)
```

```
19.635374724438407 0.04716499417607562
```

What happens if we exclude China and India from the analysis?

In [351…

```
# Exclude china and india and recalculate the correlation
test_df = final[(final['Country']!='China') & (final['Country']!='India')]
test_df[['Migration','Pop(mn)','GDP(bn)']].corr()
```

Out[351…

	Migration	Pop(mn)	GDP(bn)
Migration	1.000000	0.367377	0.310777
Pop(mn)	0.367377	1.000000	0.607187
GDP(bn)	0.310777	0.607187	1.000000

- Population and migration are highly positively correlated
- Migration is weakly correlated with GDP
- Population and GDP is also weakly correlated
- Removing China and India, affects the correlation between migration and population significantly

3.13 World Bank Income Categories

- We want to add another column which splits the countries into the World Bank Income Categories.
- The World Bank creates a yearly classification of countries by income, for all countries with population over 30,000.
- This classification stays the same throughout the fiscal year (from July 1 to June 30) even if the income data for a country changes.
- Low-income countries are those with a gross national income (GNI) per capita of US 1,135 or less in 2022
- Lower-middle-income countries are those with a GNI per capita between US 1,136 and US 4,465 in 2022
- Upper-middle-income countries are those with a GNI per capita between US 4,466 and US 13,845 in 2022
- High-income countries are those with a GNI per capita of US 13,846 or more in 2022

We will use this to split our countries into low, low-middle, upper-middle and high income

We will make a simplistic assumption that a country remains within category over the period which is not necessarily the case.

World Bank Income Categories

In [352…

```
# Define the bins and labels for categorization
bins = [0, 1135, 4465, 13845, float('inf')]
labels = ['Low', 'Low-middle', 'Upper-middle', 'High']

# Use pd.cut() to create categories based on 'GDP per Capita' column values
final['IncomeCat'] = pd.cut(final['GDP per Capita'], bins=bins, labels=labels, right=Fa

# Display the updated DataFrame with the new 'IncomeCat' column
final
```

Out[352…

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
1	New Zealand	29625	4.449559	172.347771	38733.674722	0.67	High
2	New Caledonia	146	0.261172	7.566223	28970.283835	0.06	High
3	PNG	1486	8.279302	18.964041	2290.536306	0.02	Low-middle
4	Solomon Islands	473	0.611105	1.147713	1878.093154	0.08	Low-middle
5	Vanuatu	665	0.255101	0.731674	2868.170006	0.26	Low-middle
…	…	…	…	…	…	…	…
210	Eswatini	19	1.094444	3.858769	3525.778952	0.00	Low-middle
211	Tanzania	230	50.353778	44.695951	887.638483	0.00	Low
212	Uganda	241	36.483971	28.169511	772.106500	0.00	Low

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
213	Zambia	355	15.695738	20.260215	1290.809959	0.00	Low-middle
214	Zimbabwe	2035	13.750970	20.211340	1469.811941	0.01	Low-middle

183 rows × 7 columns

In [353…

```
final.groupby('IncomeCat',as_index = False)['Migration'].mean()
```

Out[353…

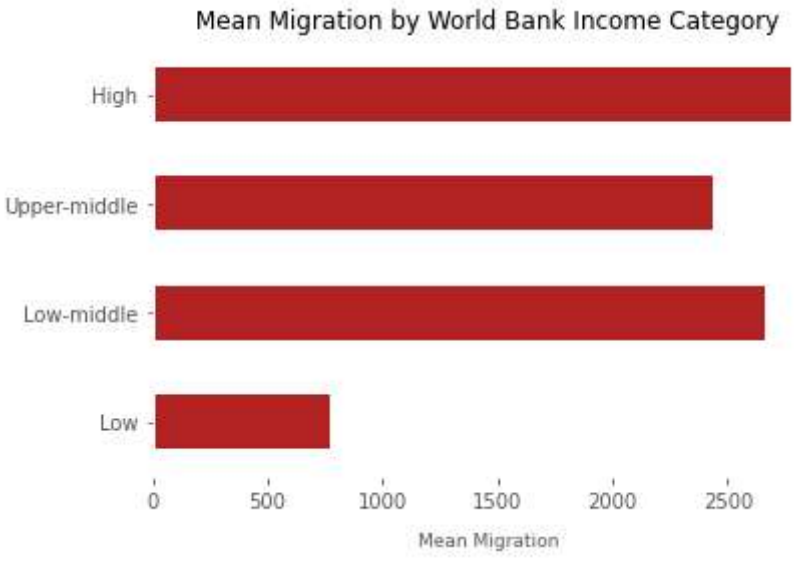
	IncomeCat	Migration
0	Low	770.428571
1	Low-middle	2658.714286
2	Upper-middle	2434.437500
3	High	2778.228070

In [354…

```
# Groupby category and apply to the migration averages to get a mean value for each grou
cat_income = final.groupby('IncomeCat',as_index = False)['Migration'].mean()

l = ['Low','Low-middle','Upper-middle','High']
ax = cat_income.plot(kind = 'barh',color = 'firebrick')
ax.set_yticklabels(cat_income.IncomeCat)
ax.set_title("Mean Migration by World Bank Income Category", fontsize = 12)
ax.get_legend().set_visible(False)
ax.set_xlabel("Mean Migration", fontsize = 9, labelpad = 10)
ax.set_facecolor("white")
;
```

Out[354…

```
''
```



In [355…

```
# Top migration countries that are high income
final[final['IncomeCat']=='High'].sort_values(by = 'Migration',ascending = False).head(
```

Out[355...

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
23	UK	36907	61.668935	2611.185558	42341.991992	0.06	High
1	New Zealand	29625	4.449559	172.347771	38733.674722	0.67	High
108	Korea, South	11771	48.101859	1217.251684	25305.709779	0.02	High
127	USA	11005	304.408106	17418.358762	57220.417023	0.00	High
105	Taiwan	7258	23.429643	553.651500	23630.385661	0.03	High

In [356...

```
# Top migration countries that are low-middle income
final[final['IncomeCat']=='Low-middle'].sort_values(by = 'Migration',ascending = False)
```

Out[356...

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
111	India	52764	1255.753205	2059.415837	1639.984536	0.00	Low-middle
98	Philippines	17534	98.929600	269.676628	2725.944783	0.02	Low-middle
94	Vietnam	11046	87.928491	227.163787	2583.506028	0.01	Low-middle
96	Indonesia	8007	247.444582	834.177376	3371.168482	0.00	Low-middle
114	Pakistan	7405	205.971490	252.744351	1227.084153	0.00	Low-middle

In [357...

```
# Top migration countries that are upper-middle income
final[final['IncomeCat']=='Upper-middle'].sort_values(by = 'Migration',ascending = Fals
```

Out[357...

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
101	China	53136	1304.326999	10081.880503	7729.565140	0.00	Upper-middle
97	Malaysia	12029	29.637164	281.156029	9486.603777	0.04	Upper-middle
209	South Africa	8566	53.712375	341.330774	6354.788288	0.02	Upper-middle
93	Thailand	7411	66.521448	374.348484	5627.485467	0.01	Upper-middle
130	Brazil	5454	190.844776	1762.263942	9234.017180	0.00	Upper-middle

In [358...

```
# Top migration countries that are low income
final[final['IncomeCat']=='Low'].sort_values(by = 'Migration',ascending = False).head()
```

Out[358...

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
113	Nepal	10789	26.645374	23.314328	874.985951	0.04	Low
116	Afghanistan	3585	32.023033	13.913336	434.479019	0.01	Low
90	Myanmar	2000	48.518715	51.382713	1059.028726	0.00	Low
72	Sudan	1143	38.827292	41.350715	1064.990959	0.00	Low
197	Ethiopia	777	99.612246	49.975318	501.698532	0.00	Low

- High income countries have the highest mean migration and the lowest is from low income countries

3.14 GDP per Capita

In [359…
```
# Check null values
final.isnull().sum()
```

Out[359…
```
Country           0
Migration         0
Pop(mn)           0
GDP(bn)           0
GDP per Capita    0
% Pop             0
IncomeCat         1
dtype: int64
```

In [360…
```
# Find null value
final[final['IncomeCat'].isnull()]
```

Out[360…

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
40	Gibraltar	10	0.032043	-1.000000e-09	-0.000031	0.03	NaN

We have missing values for Gibraltar across all years so we will just drop this row as it will not impact our chart significantly

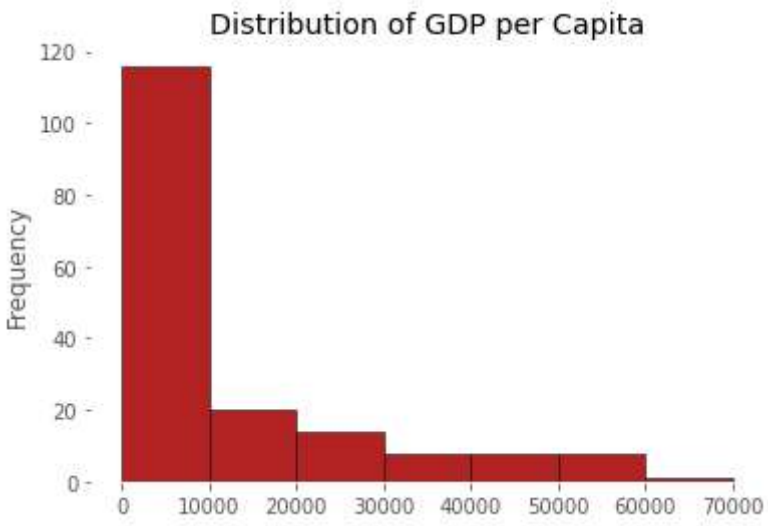
In [361…
```
# Drop Gibraltar
final = final[final['Country']!='Gibraltar']
```

In [362…
```
# Describe the GDP per Capital column
final['GDP per Capita'].describe()
```

Out[362…
```
count       182.000000
mean      15860.837143
std       23710.413907
min           1.124010
25%        1874.308497
50%        5647.248187
75%       19923.459253
max      169044.752554
Name: GDP per Capita, dtype: float64
```

- The GDP per Capita is very postively skewed as we might expect, there are a few outliers where the values are very high
- The median value is a long way from the mean value

In [363…
```
bins = [0,10000,20000,30000,40000,50000,60000,70000]
ax = final['GDP per Capita'].plot(kind = 'hist', ec = 'k', color = 'firebrick',bins = b
plt.title('Distribution of GDP per Capita')
ax.set_facecolor('white');
```



In [364...

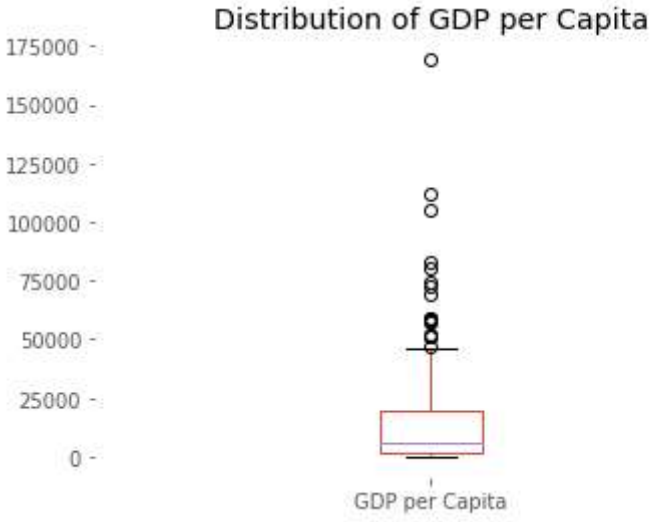
```
# Identify the outliers based on a z-score
from scipy import stats

z_scores = stats.zscore(final['GDP per Capita'])
outliers = final[(z_scores > 3) | (z_scores < -3)]
outliers.sort_values(by = 'GDP per Capita',ascending = False)
```

Out[364...

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
30	Monaco	11	0.034127	5.768973	169044.752554	0.03	High
29	Luxembourg	23	0.541930	60.480606	111602.186568	0.00	High
125	Bermuda	19	0.060622	6.367123	105030.079872	0.03	High

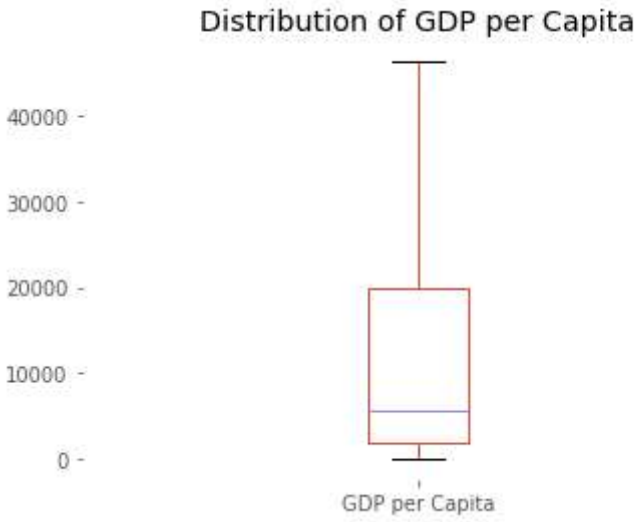
In [365...

```
ax = final['GDP per Capita'].plot(kind = 'box')
plt.title('Distribution of GDP per Capita')
ax.set_facecolor("white");
```



In [366...

```
ax = final['GDP per Capita'].plot(kind = 'box',showfliers = False)
plt.title('Distribution of GDP per Capita')
ax.set_facecolor("white");
```



- The problem with the z-score is that it assumes a normal distribution in the data.
- There are quite a few outliers bunched around the 50000 to 75000 level

3.15 Migration against Population and World Bank Income Groups

In [367…

```
# Copy the dataframe
scatter = final.copy()
```

In [368…

```
scatter.sort_values(by = 'Pop(mn)',ascending = False).head(10)
```

Out[368…

	Country	Migration	Pop(mn)	GDP(bn)	GDP per Capita	% Pop	IncomeCat
101	China	53136	1304.326999	10081.880503	7729.565140	0.00	Upper-middle
111	India	52764	1255.753205	2059.415837	1639.984536	0.00	Low-middle
127	USA	11005	304.408106	17418.358762	57220.417023	0.00	High
96	Indonesia	8007	247.444582	834.177376	3371.168482	0.00	Low-middle
114	Pakistan	7405	205.971490	252.744351	1227.084153	0.00	Low-middle
130	Brazil	5454	190.844776	1762.263942	9234.017180	0.00	Upper-middle
188	Nigeria	1082	181.009593	360.126203	1989.542085	0.00	Low-middle
109	Bangladesh	3862	151.464599	219.534019	1449.408112	0.00	Low-middle
65	Russia	1330	136.803054	1558.481054	11392.151070	0.00	Upper-middle
106	Japan	6563	120.520529	4527.850895	37569.125582	0.01	High

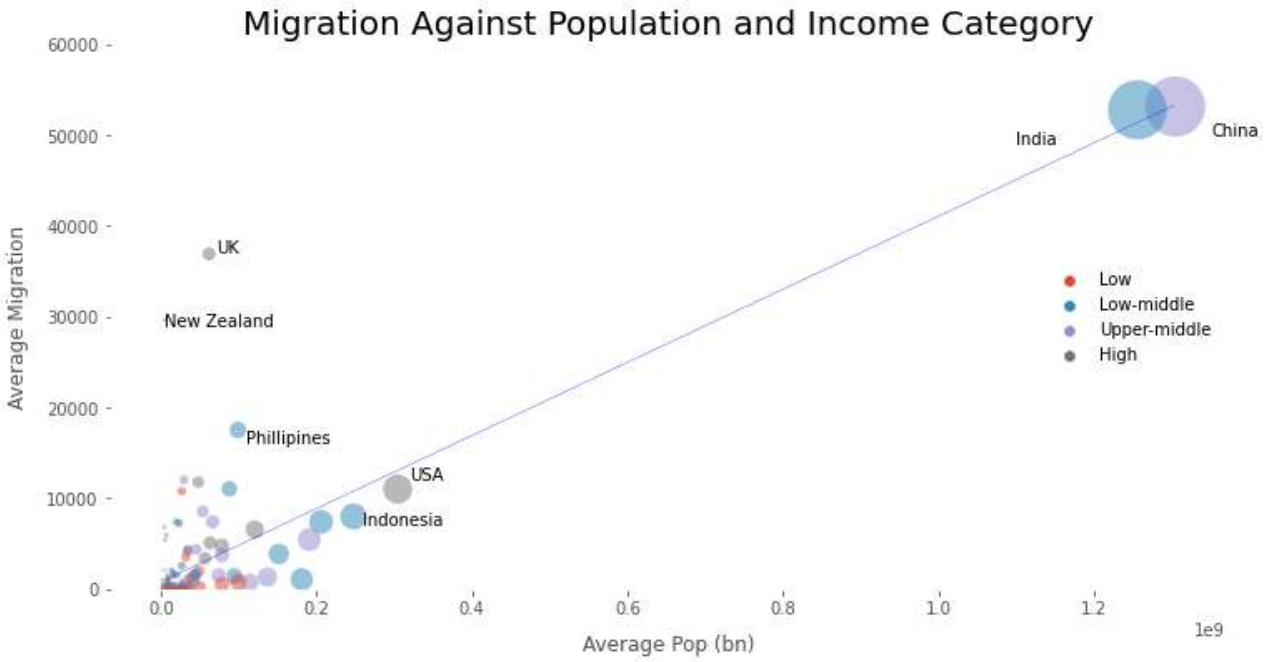
In [369…

```
# Plot a scatterplot
size = scatter["Pop(mn)"]
x = scatter['Pop(mn)'] * 1000000

z = np.polyfit(scatter['Pop(mn)'], scatter['Migration'], 1)
```

```
p = np.poly1d(z)

fig, ax = plt.subplots(figsize = (12,6))
sns.scatterplot(data =scatter,
                x = x,
                y = 'Migration',
                s = size,
                hue = 'IncomeCat',
                alpha = 0.5)
plt.title('Migration Against Population and Income Category', fontsize = 20)

plt.plot(scatter['Pop(mn)']*1000000,p(scatter['Pop(mn)']),"blue",linestyle = "--",linew
plt.xlabel('Average Pop (bn)', labelpad = 10)
plt.ylabel('Average Migration', labelpad = 10)
plt.legend(loc = 'center right',facecolor = "white",edgecolor = 'white')
ax.set_facecolor("white")
plt.ylim(0,60000)

plt.text(320000000,12000,"USA")
plt.text(260000000,7000,"Indonesia")
plt.text(4990000,29000,"New Zealand")
plt.text(70000000,37000,"UK")
plt.text(1100000000,49000,"India")
plt.text(1350000000,50000,"China")
plt.text(110000000,16000,"Phillipines")


;
```

Out[369…

```
''
```



In [370…

```
scatter_dropped = scatter[(scatter['Country']!='China') & (scatter['Country']!= 'India'
```

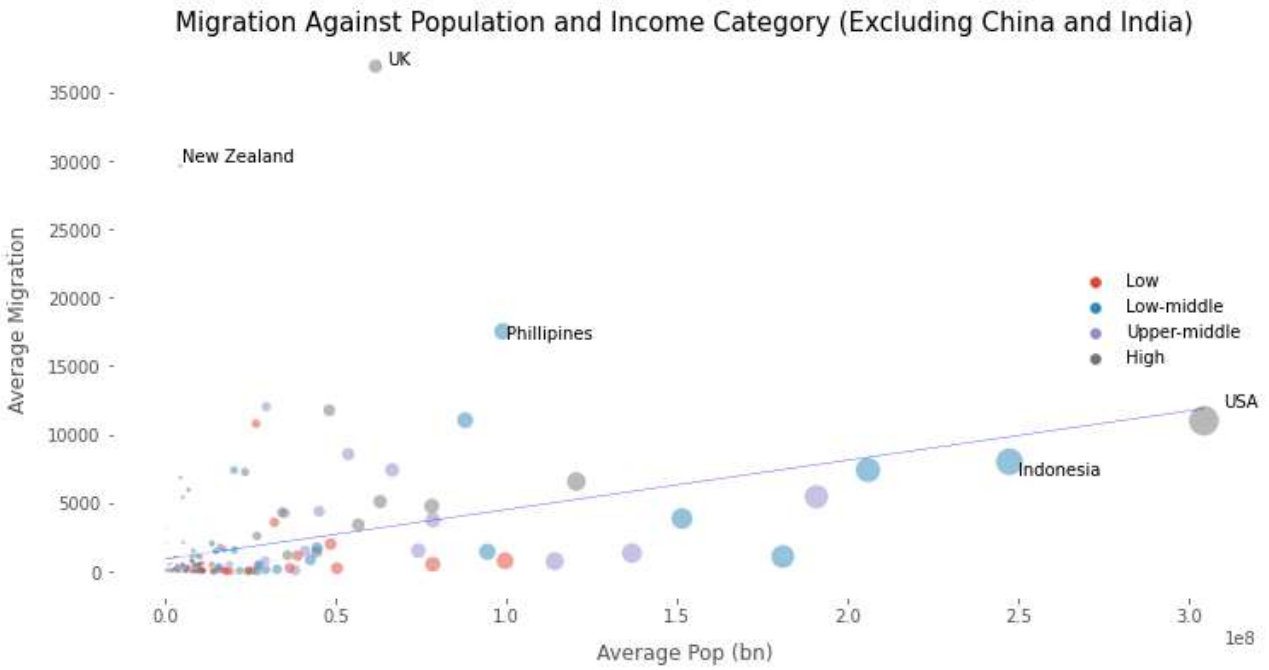
In [371…

```
# Plot a scatterplot
size = scatter_dropped["Pop(mn)"]
x = scatter_dropped['Pop(mn)'] * 1000000
```

```
z = np.polyfit(scatter_dropped['Pop(mn)'], scatter_dropped['Migration'], 1)
p = np.poly1d(z)

fig, ax = plt.subplots(figsize = (12,6))
sns.scatterplot(data =scatter_dropped,
                x = x,
                y = 'Migration',
                s = size,
                hue = 'IncomeCat',
                alpha = 0.5)
plt.title('Migration Against Population and Income Category (Excluding China and India)

plt.plot(scatter_dropped['Pop(mn)']*1000000,p(scatter_dropped['Pop(mn)']),"blue",linest
plt.xlabel('Average Pop (bn)', labelpad = 10)
plt.ylabel('Average Migration', labelpad = 10)
plt.legend(loc = 'center right',facecolor = "white",edgecolor = 'white')
ax.set_facecolor("white")
#plt.ylim(0,80000)

plt.text(310000000,12000,"USA")
plt.text(250000000,7000,"Indonesia")
plt.text(4900000,30000,"New Zealand")
plt.text(65000000,37000,"UK")
#plt.text(1100000000,95000,"India")
#plt.text(1292000000,90000,"China")
plt.text(100000000,17000,"Phillipines")


;
```

Out[371...

```
''
```



- We can see that with a few exceptions, the highest migration is from low-middle and upper-middle countries of which China and India are the highest
- Low income countries also have low migration on average

3.9.6 Income Categories as a Proportion of the Whole

If we filter the dataframe for each category and get the mean of country migration within each, we can compare the groups

In [372…

```
# Split out dataframe into separate ones according to the world bank income category
low = final[final['IncomeCat']=='Low']
low_mid = final[final['IncomeCat']=='Low-middle']
up_mid = final[final['IncomeCat']=='Upper-middle']
high = final[final['IncomeCat']=='High']
```

In [373…

```
# Get the totals and print
low_mig = low['Migration'].sum()
low_mid_mig = low_mid['Migration'].sum()
up_mid_mig = up_mid['Migration'].sum()
high_mig = high['Migration'].sum()

print(low_mig)
print(low_mid_mig)
print(up_mid_mig)
print(high_mig)
```

```
21572
130277
116853
158359
```

In [374…

```
# Lets express this in terms of 50 blocks
total = low_mig + low_mid_mig + up_mid_mig + high_mig
low_prop = low_mig/total*50
low_mid_prop = low_mid_mig/total*50
up_mid_prop = up_mid_mig/total*50
high_prop = high_mig/total*50
```

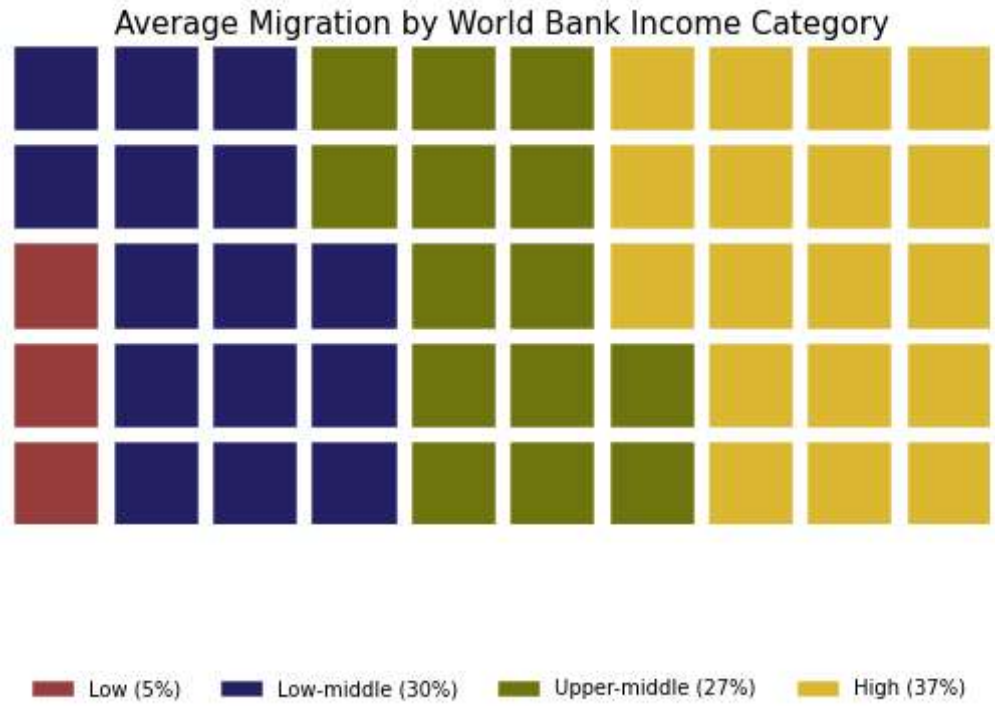
In [375…

```
data = {'Low': low_prop, 'Low-middle': low_mid_prop, 'Upper-middle': up_mid_prop,'High'
```

In [380…

```
# Plot waffle
fig = plt.figure(
    figsize = (15,5),
    FigureClass=Waffle,
    rows=5,
    columns=10,
    values=data,
    colors=["#983D3D","#232066","#6E750E", "#DCB732",],
    title={
        'label': 'Average Migration by World Bank Income Category',
        'loc': 'center',
        'fontdict': {
            'fontsize': 15
        }
    },
    labels=[f"{k} ({int(v / sum(data.values()) * 100)}%)" for k, v in data.items()],
    legend={
        # 'labels': [f"{k} ({v}%)" for k, v in data.items()],
```

```
        'loc': 'lower left',
        'bbox_to_anchor': (0, -0.4),
        'ncol': len(data),
        'framealpha': 0,
        'fontsize': 10
    }
)
```



- High income countries account for the highest mean migration and low income the lowest.
- Even though the other two groups include China and India, they are outliers amongst their groups so average annual migration from other low-middle and upper-middle income countries is relatively low in comparison.