Guide:

Lab 3 -- line 1 to line 20

Lab 4 -- line 21 to line 69

Assignment 1 -- line 70 to line 92

Assignment 2 -- line 26 to line 38 , line 92 to line 101

Import Libraries

In [1]:

```
import numpy as np
import pandas as pd
from pandas import DataFrame as df
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

In [2]:

```
canada = pd.read_excel('https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DV0101EN/labs/Data_Files/Canada.xlsx',
 sheet_name='Canada by Citizenship',
 skiprows=range(20),
 skipfooter=2)
```

In [3]:

```
canada.head()
#spare df
canada1 = canada
```

Rename columns

In [4]:

```
canada.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':'
Continent-Region'}, inplace=True)
canada.drop(['AREA','REG','DEV','Type','Coverage'], inplace=True, axis='column
s')
canada.isnull().sum().sum()
```

Out[4]:

0

In [5]:

```
canada.head(5)
```

Out[5]:

	Country	Continent	Continent-Region	DevName	1980	1981	1982	1983	1984	1985	...
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...

5 rows × 38 columns

In [6]:

```
canada.index.values
```

Out[6]:

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,
12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,
25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,
38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,
51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,
64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,
77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,
90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102,
103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180,
181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193,
194])
```

Reassign Index

In [7]:

```
canada.set_index('Country', inplace=True)
```

In [8]:

```
canada.head()
```

Out[8]:

Country	Continent	Continent-Region	DevName	1980	1981	1982	1983	1984	1985	1986
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2

5 rows × 37 columns

Convert columns to str

In [9]:

```
canada.dtypes
canada.columns = list(map(str,canada.columns));
```

Subsetting

In [10]:

```
canada[canada['Continent']=='Asia'].head(4)
```

Out[10]:

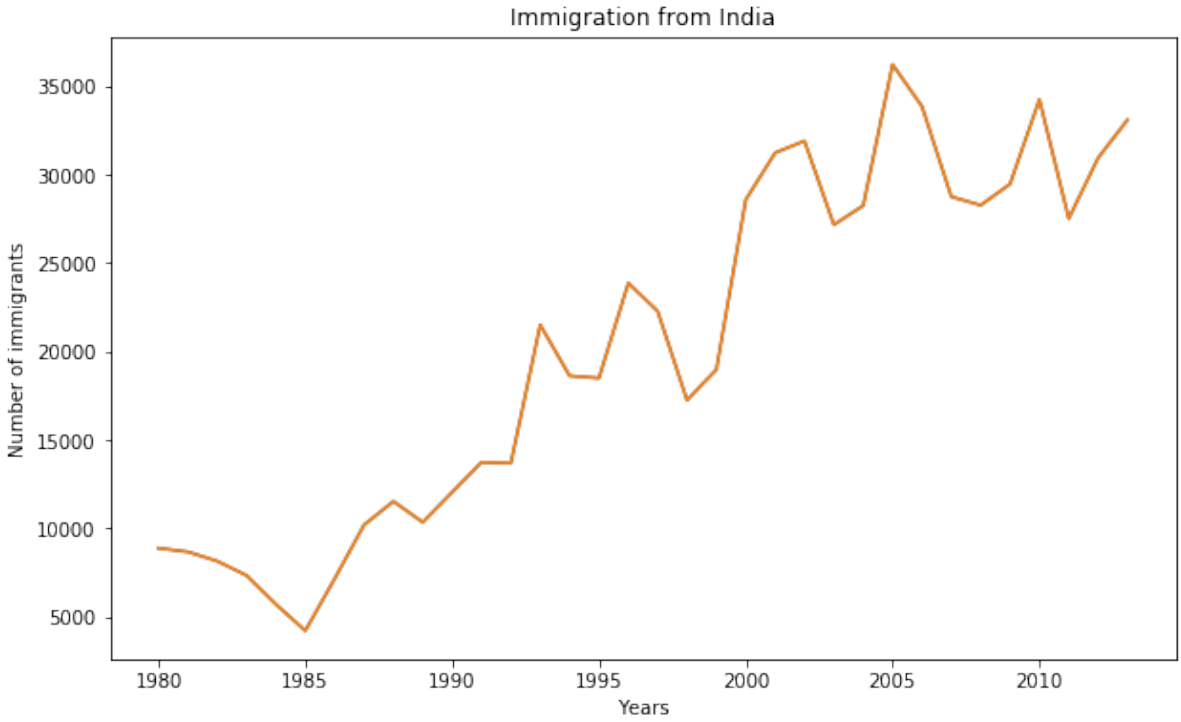
Country	Continent	Continent-Region	DevName	1980	1981	1982	1983	1984	1985	1986
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496
Armenia	Asia	Western Asia	Developing regions	0	0	0	0	0	0	0
Azerbaijan	Asia	Western Asia	Developing regions	0	0	0	0	0	0	0
Bahrain	Asia	Western Asia	Developing regions	0	2	1	1	1	3	0

4 rows × 37 columns

Viewing Line chart

In [11]:

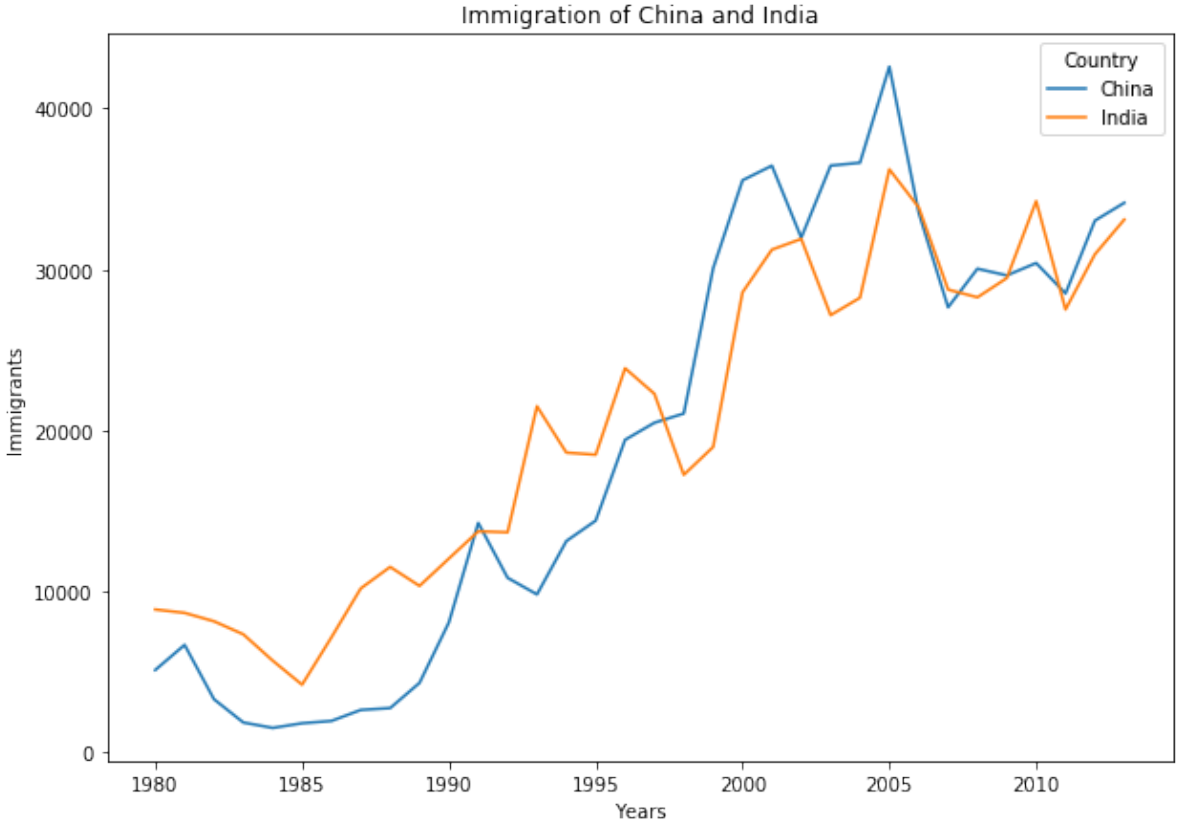
```
years = list(map(str,range(1980,2014)))
india_imgt = canada.loc['India', years]
india_imgt.plot(figsize=(10,6));
india_imgt.plot(kind='line');
plt.title('Immigration from India');
plt.ylabel('Number of immigrants');
plt.xlabel('Years');
```



Plotting two countries and comparing

In [12]:

```
canada.loc[['China','India'], years].transpose().plot(figsize =(10,7));
plt.title('Immigration of China and India');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```

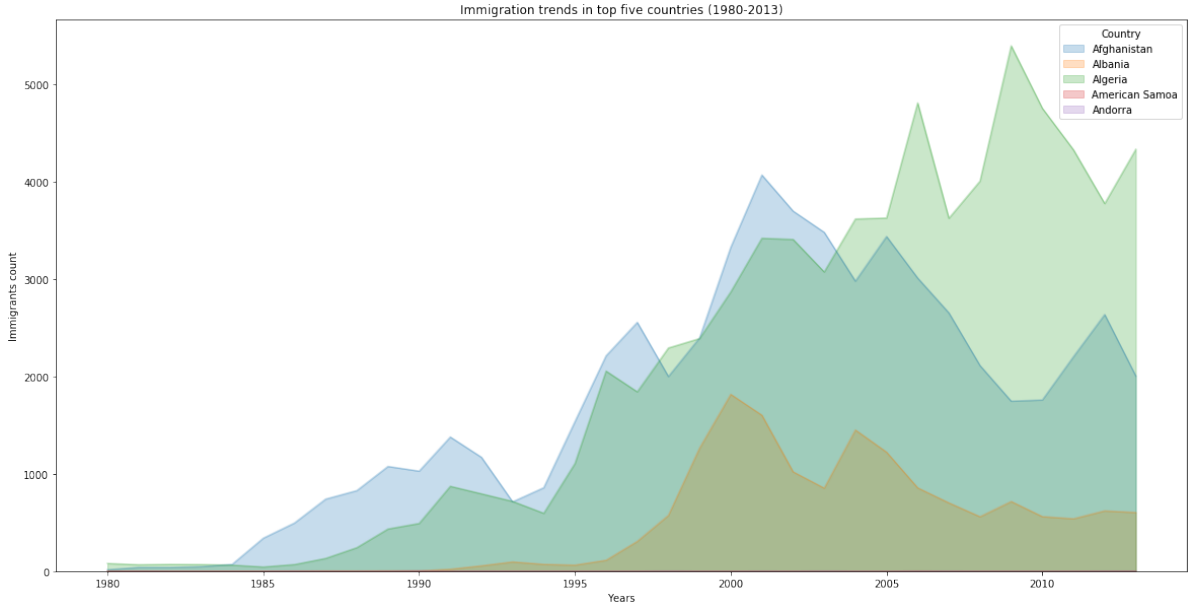


Area Plots

Unstacked

In [13]:

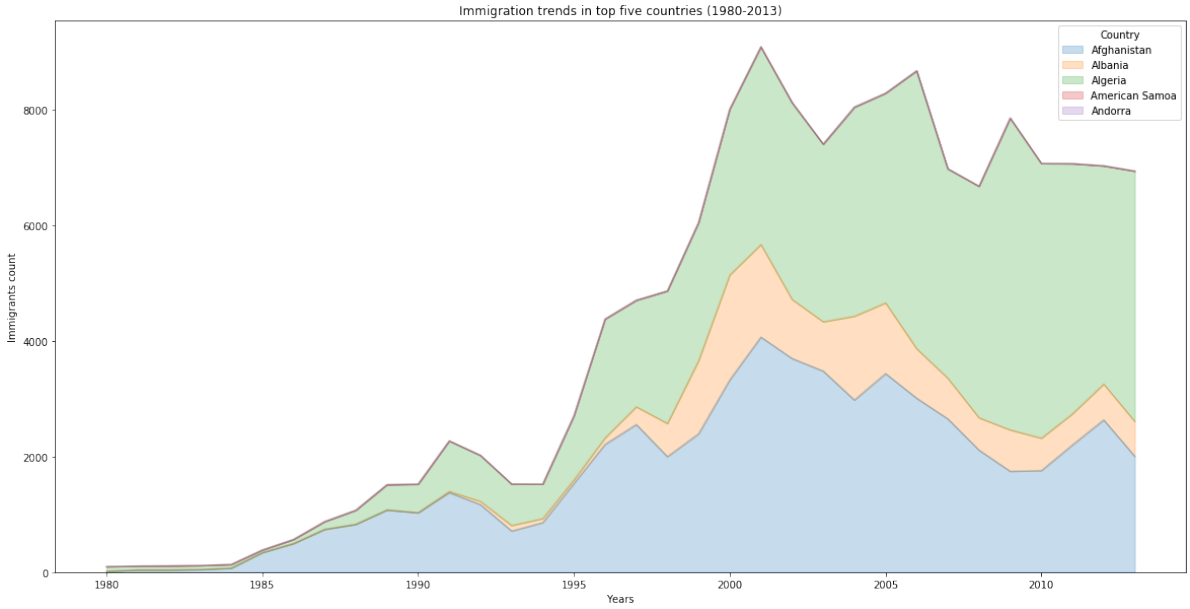
```
top5_clean = canada.head(5)[years].transpose()
top5_clean.index
top5_clean.index = top5_clean.index.map(int)
top5_clean.plot(kind="area", stacked=False, figsize=(20,10), alpha=0.25)
plt.title('Immigration trends in top five countries (1980-2013)')
plt.xlabel('Years')
plt.ylabel('Immigrants count')
plt.show()
```



Stacked

In [14]:

```
top5_clean.plot(kind="area", stacked=True, figsize=(20,10), alpha=0.25)
plt.title('Immigration trends in top five countries (1980-2013)')
plt.xlabel('Years')
plt.ylabel('Immigrants count')
plt.show()
```



Bar PLots and Histograms

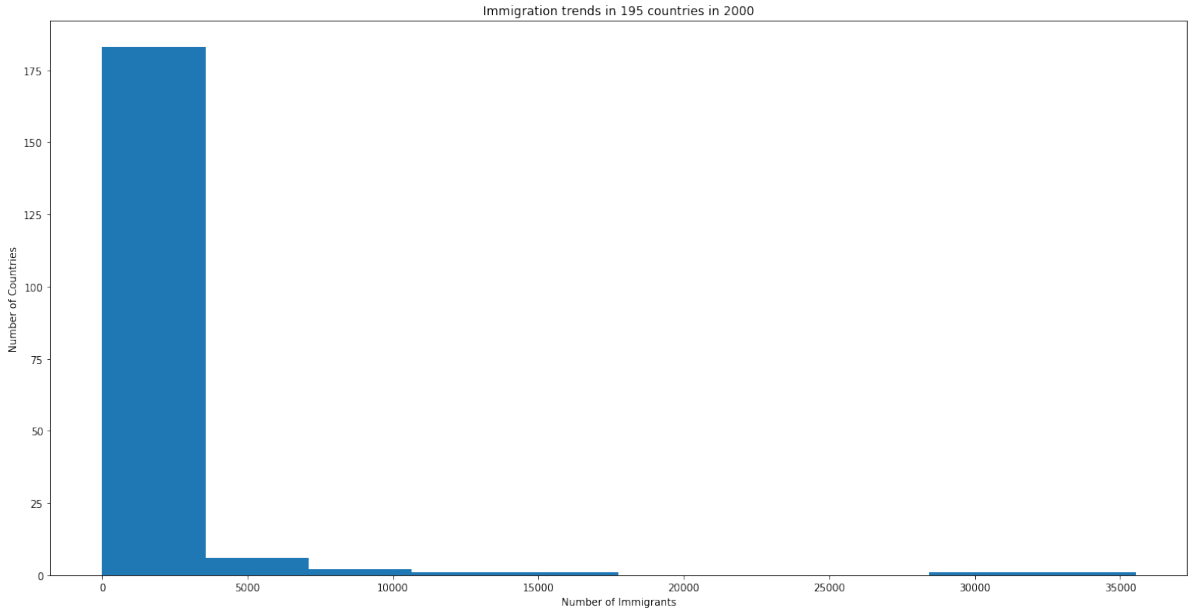
In [15]:

```
df_can = canada
```

Bar

In [16]:

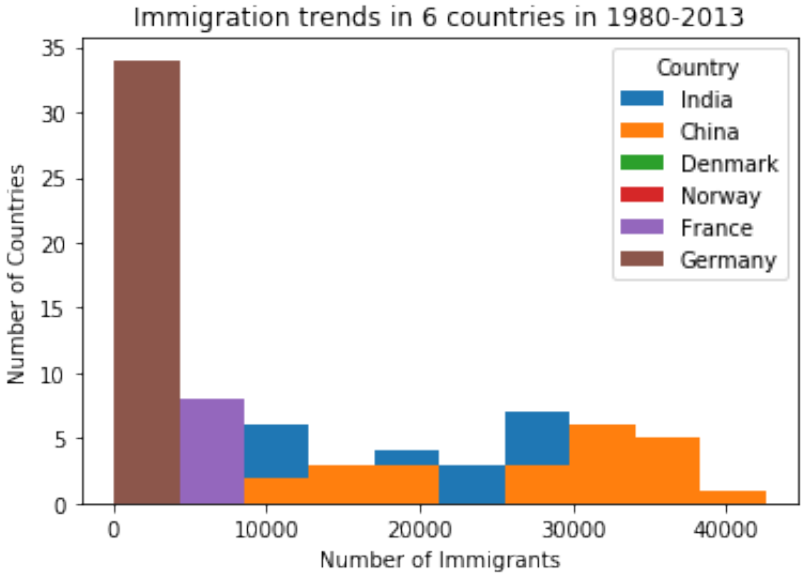
```
df_can['2000'].plot(kind='hist', figsize=(20,10))
plt.title('Immigration trends in 195 countries in 2000')
plt.xlabel('Number of Immigrants')
plt.ylabel('Number of Countries')
plt.show()
```



Histogram

In [17]:

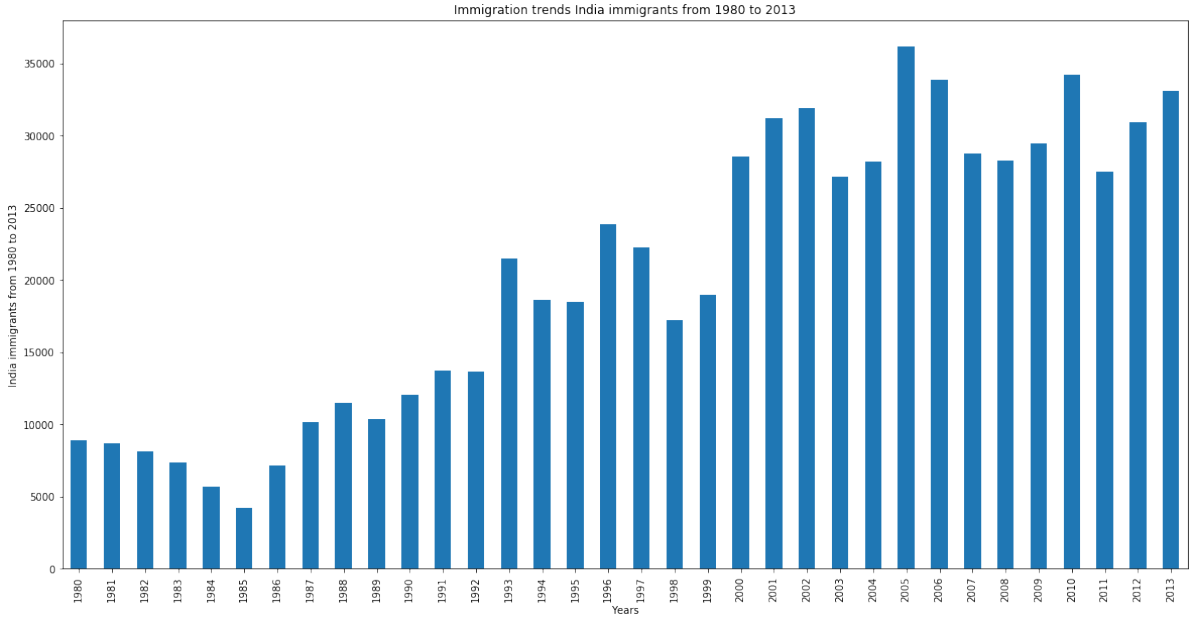
```
df_can.loc[['India','China','Denmark','Norway','France','Germany'], years].tra
nspose().plot.hist()

plt.title('Immigration trends in 6 countries in 1980-2013')
plt.xlabel('Number of Immigrants')
plt.ylabel('Number of Countries')

plt.show()
```



Vertical Bar Plots

In [18]:

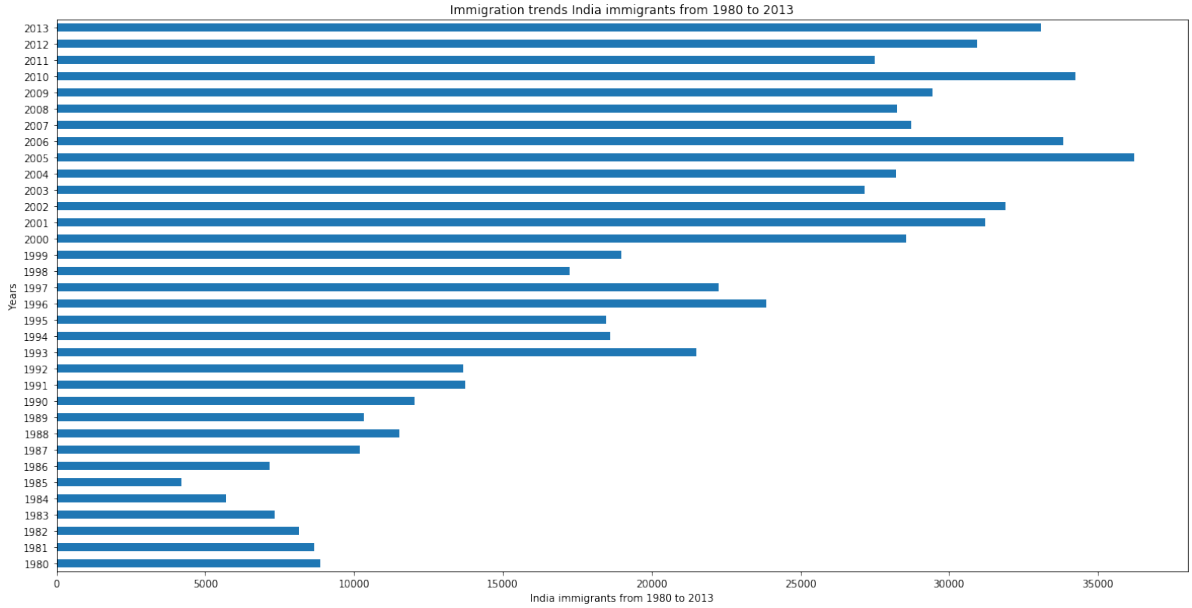
```
india = df_can.loc['India', years]

india.plot(kind='bar', figsize=(20,10))

plt.title('Immigration trends India immigrants from 1980 to 2013')
plt.xlabel('Years')
plt.ylabel('India immigrants from 1980 to 2013')

plt.show()
```



Horizontal Bar Plots

In [19]:

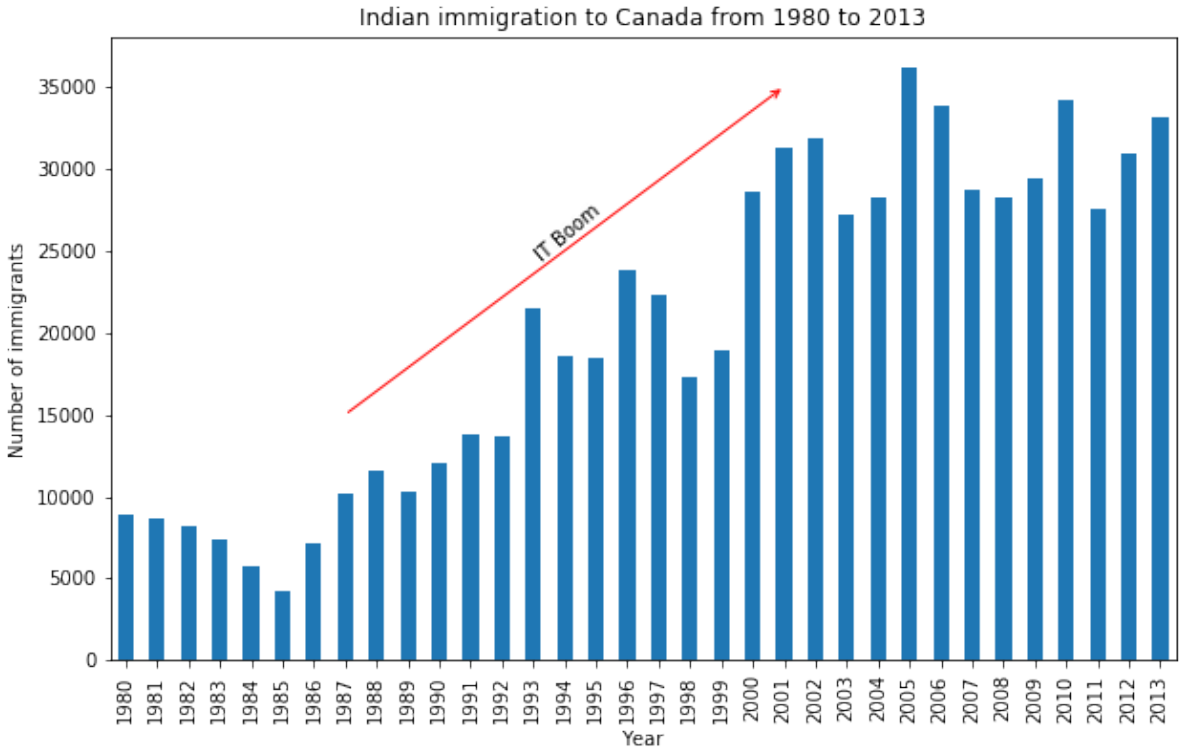
```
india = df_can.loc['India', years]

india.plot(kind='barh', figsize=(20,10))

plt.title('Immigration trends India immigrants from 1980 to 2013')
plt.xlabel('India immigrants from 1980 to 2013')
plt.ylabel('Years')
plt.show()
```



Using annotate

In [20]:

```
df_india = df_can.loc['India',years]
df_india.plot(kind='bar', figsize=(10,6))
plt.title("Indian immigration to Canada from 1980 to 2013")
plt.xlabel("Year")
plt.ylabel("Number of immigrants")
plt.annotate('',  #arrow title
             xy=(21,35000), #x,y of arrow head
             xytext=(7,15000), #x, y of arrow tail
             xycoords='data',  #keep unchanged
             arrowprops=dict(arrowstyle='->',color='red') #arrow style with co
lor
             )
plt.annotate('IT Boom', #add text to arrow
             xy=(13,28000),  #x, y of text position
             rotation=40, # counter clockwise rotate text by angle
             xycoords='data', #keep unchanged
             va='top', #position text
             ha='left') #position text
plt.show()
```



Lab 4 starts here ---

Scatter

In [21]:

```
indopak = canada.loc[['India','Pakistan'], years].transpose()
```

In [22]:

```
indopak.reset_index(inplace=True)
indopak.rename(columns={'Country':'index','index':'Year'}, inplace=True)
indopak.head()
```

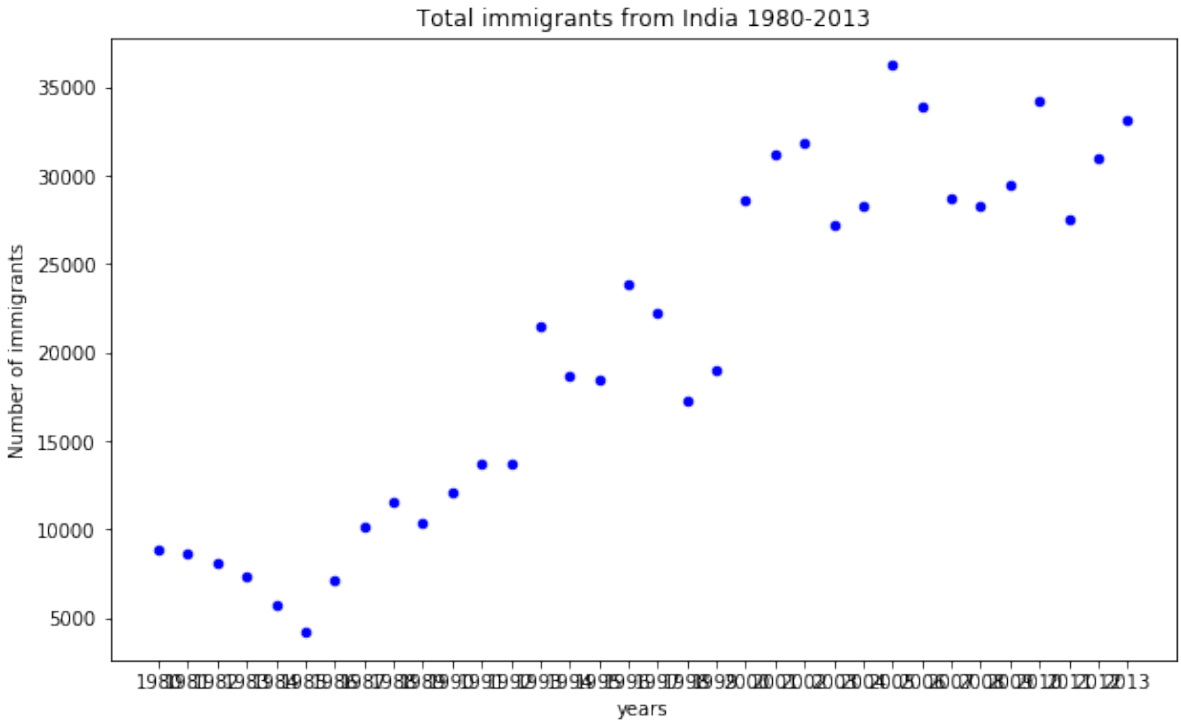
Out[22]:

Country	Year	India	Pakistan
0	1980	8880	978
1	1981	8670	972
2	1982	8147	1201
3	1983	7338	900
4	1984	5704	668

In [23]:

```
import numpy as np

x = indopak['Year'].astype('int64')
y = indopak['India']
fit = np.polyfit(x,y,deg=1)
print(fit)
indopak.plot(kind='scatter', x='Year', y='India', color='blue', figsize=(10,6)
)
plt.title('Total immigrants from India 1980-2013')
plt.xlabel("years")
plt.ylabel("Number of immigrants")

#plt.plot(x, fit[0]*x+fit[1], color='red')
plt.show();
```

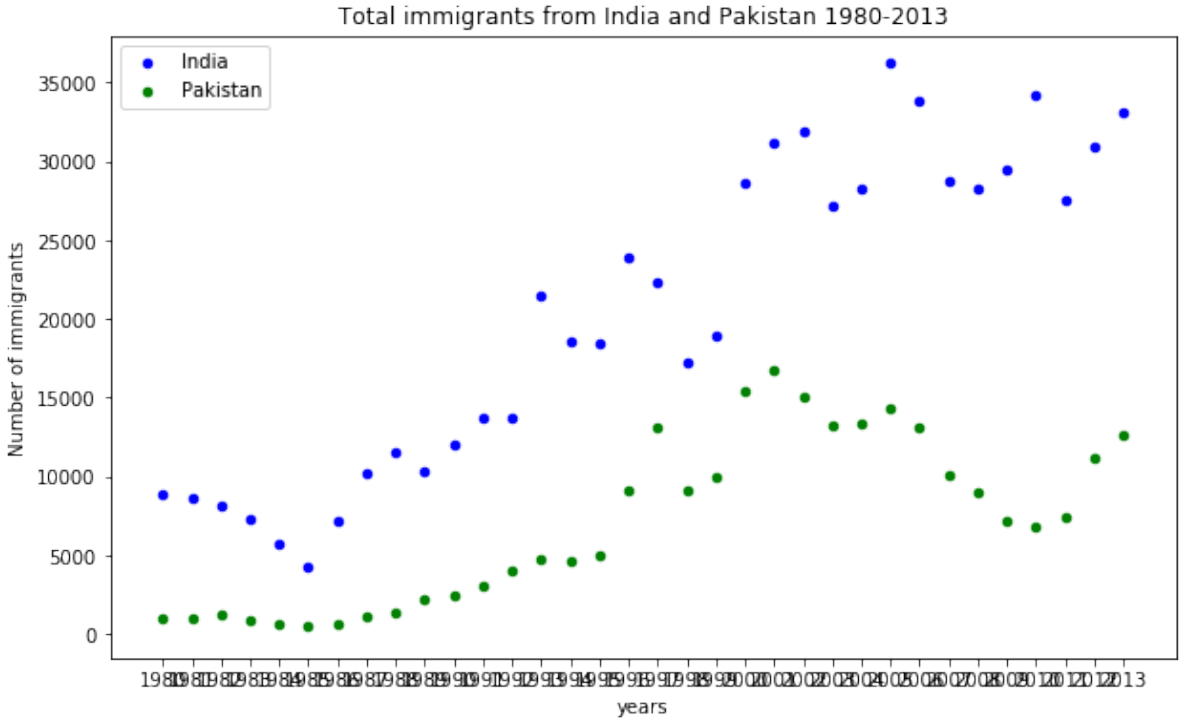
```
[ 9.34267991e+02 -1.84491593e+06]
```



Plotting 2 Scatter Plots in the Same Figure

In [24]:

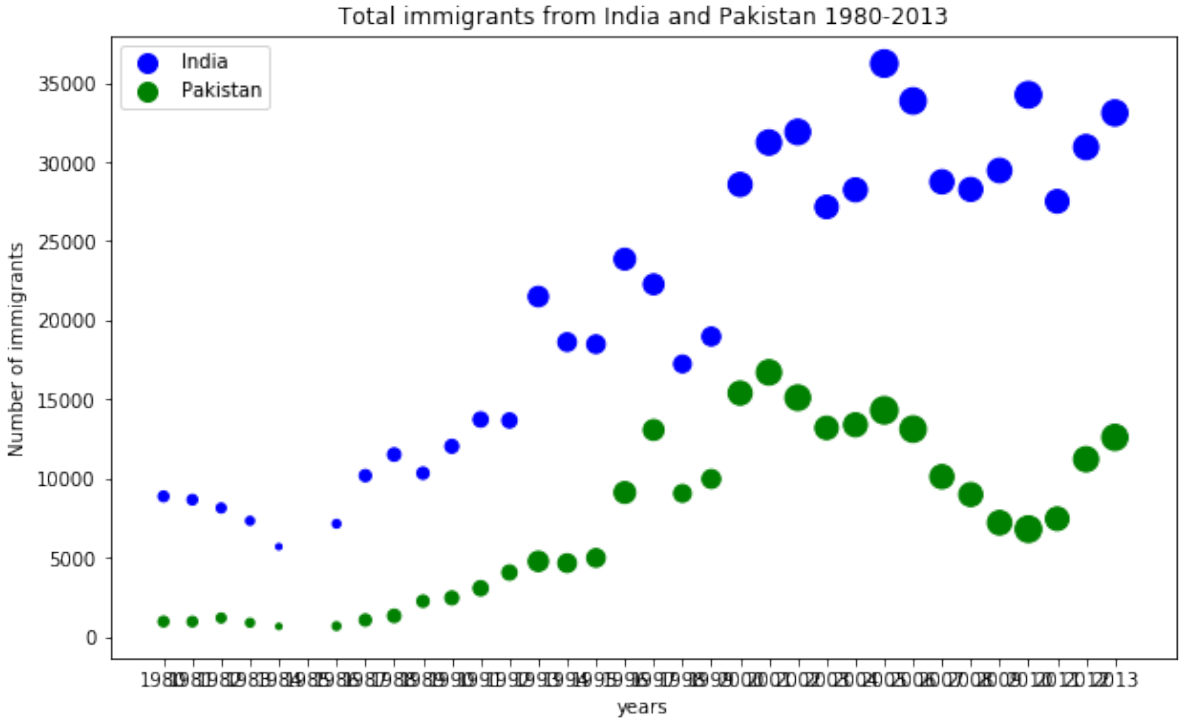
```
ax0 = indopak.plot(kind='scatter', x='Year', y='India', color='blue', figsize=(10,6))

indopak.plot(kind='scatter', x='Year', y='Pakistan', color='green', figsize=(10,6), ax=ax0)

plt.title('Total immigrants from India and Pakistan 1980-2013')
plt.xlabel("years")
plt.ylabel("Number of immigrants")
ax0.legend(['India','Pakistan'], loc='upper left')

plt.show();
```



Plotting 2 Scatter Plots in the Same Figure

In [25]:

```
norm_india = (indopak['India'] - indopak['India'].min()) / (indopak['India'].m
ax() - indopak['India'].min())
norm_pak = (indopak['Pakistan'] - indopak['Pakistan'].min()) / (indopak['Pakis
tan'].max() - indopak['Pakistan'].min())

ax0 = indopak.plot(kind='scatter', x='Year', y='India', color='blue', s=norm_i
ndia*200, figsize=(10,6))

indopak.plot(kind='scatter', x='Year', y='Pakistan', color='green', s=norm_ind
ia*200, figsize=(10,6), ax=ax0)

plt.title('Total immigrants from India and Pakistan 1980-2013')
plt.xlabel("years")
plt.ylabel("Number of immigrants")
ax0.legend(['India','Pakistan'], loc='upper left')

plt.show();
```



Waffle Charts

Exp 4 Assignment included

In [26]:

```
%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

In [27]:

```
df_can['Total'] =  df_can.sum (axis = 1)
```

In [28]:

```
# let's create a new dataframe for these three countries
df_dsn = df_can.loc[['Denmark', 'Norway', 'Sweden'], :]

# let's take a look at our dataframe
df_dsn
```

Out[28]:

Country	Continent	Continent-Region	DevName	1980	1981	1982	1983	1984	1985	1986	...
Denmark	Europe	Northern Europe	Developed regions	272	293	299	106	93	73	93	...
Norway	Europe	Northern Europe	Developed regions	116	77	106	51	31	54	56	...
Sweden	Europe	Northern Europe	Developed regions	281	308	222	176	128	158	187	...

3 rows × 38 columns

In [29]:

```
# compute the proportion of each category with respect to the total
total_values = sum(df_dsn['Total'])
category_proportions = [(float(value) / total_values) for value in df_dsn['Tot
al']]

# print out proportions
for i, proportion in enumerate(category_proportions):
    print (df_dsn.index.values[i] + ': ' + str(proportion))
```

```
Denmark: 0.32255663965602777
Norway: 0.1924094592359848
Sweden: 0.48503390110798744
```

In [30]:

```
width = 40 # width of chart
height = 10 # height of chart

total_num_tiles = width * height # total number of tiles

print ('Total number of tiles is ', total_num_tiles)
```

```
Total number of tiles is  400
```

In [31]:

```
# compute the number of tiles for each catagory
tiles_per_category = [round(proportion * total_num_tiles) for proportion in ca
tegory_proportions]

# print out number of tiles per category
for i, tiles in enumerate(tiles_per_category):
    print (df_dsn.index.values[i] + ': ' + str(tiles))
```

```
Denmark: 129
Norway: 77
Sweden: 194
```

In [32]:

```
# initialize the waffle chart as an empty matrix
waffle_chart = np.zeros((height, width))

# define indices to loop through waffle chart
category_index = 0
tile_index = 0

# populate the waffle chart
for col in range(width):
    for row in range(height):
        tile_index += 1

        # if the number of tiles populated for the current category is equal t
o its corresponding allocated tiles...
        if tile_index > sum(tiles_per_category[0:category_index]):
            # ...proceed to the next category
            category_index += 1

        # set the class value to an integer, which increases with class
        waffle_chart[row, col] = category_index

print ('Waffle chart populated!')
```

```
Waffle chart populated!
```

In [33]:

```
waffle_chart
```

Out[33]:

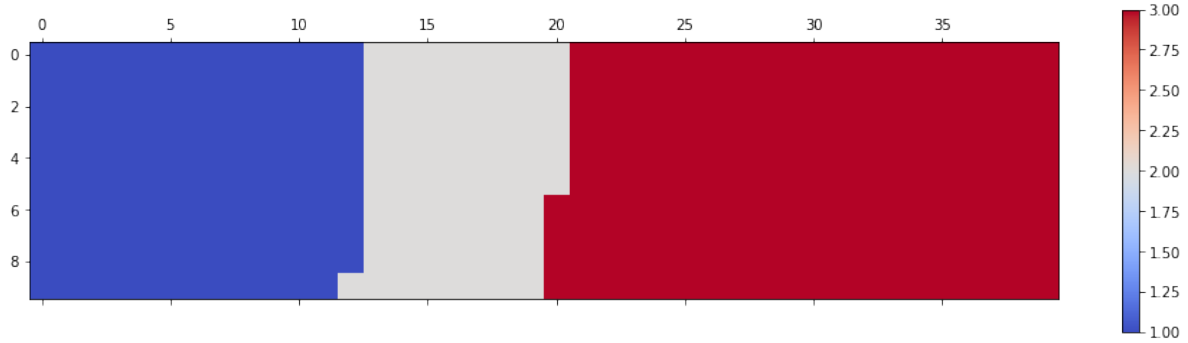
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2.
, 2.,
        2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2.
, 2.,
        2., 2., 2., 2., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.
, 3.,
        3., 3., 3., 3., 3., 3., 3., 3.]])
```

In [34]:

```
# instantiate a new figure object
fig = plt.figure()

# use matshow to display the waffle chart
colormap = plt.cm.coolwarm
plt.matshow(waffle_chart, cmap=colormap)
plt.colorbar()
```

Out[34]:

```
<matplotlib.colorbar.Colorbar at 0x7f8a918cf190>

<Figure size 432x288 with 0 Axes>
```

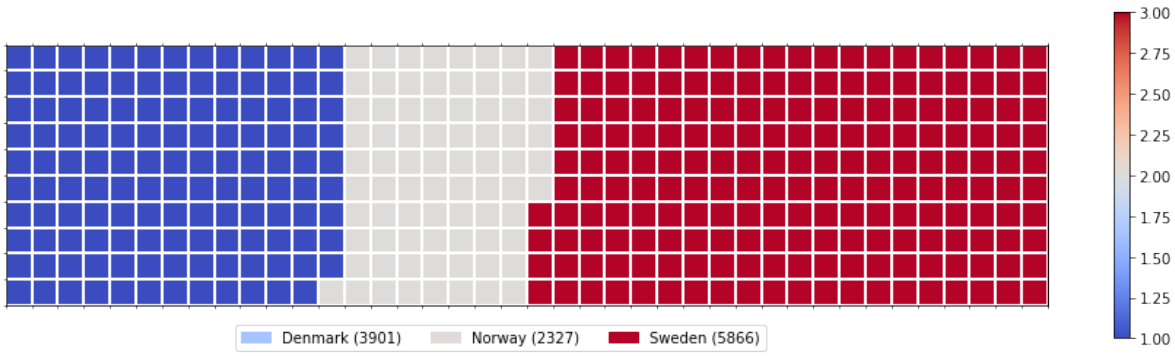


In [35]:

```
# instantiate a new figure object
fig = plt.figure()

# use matshow to display the waffle chart
colormap = plt.cm.coolwarm
plt.matshow(waffle_chart, cmap=colormap)
plt.colorbar()

# get the axis
ax = plt.gca()

# set minor ticks
ax.set_xticks(np.arange(-.5, (width), 1), minor=True)
ax.set_yticks(np.arange(-.5, (height), 1), minor=True)

# add gridlines based on minor ticks
ax.grid(which='minor', color='w', linestyle='-', linewidth=2)

plt.xticks([])
plt.yticks([])
# compute cumulative sum of individual categories to match color schemes betwe
en chart and legend
values_cumsum = np.cumsum(df_dsn['Total'])
total_values = values_cumsum[len(values_cumsum) - 1]

# create legend
legend_handles = []
for i, category in enumerate(df_dsn.index.values):
    label_str = category + ' (' + str(df_dsn['Total'][i]) + ')'
    color_val = colormap(float(values_cumsum[i])/total_values)
    legend_handles.append(mpatches.Patch(color=color_val, label=label_str))

# add legend to chart
plt.legend(handles=legend_handles,
           loc='lower center',
           ncol=len(df_dsn.index.values),
           bbox_to_anchor=(0., -0.2, 0.95, .1)
          )
```

Out[35]:

```
<matplotlib.legend.Legend at 0x7f8a90841090>

<Figure size 432x288 with 0 Axes>
```



Or use a waffle function

In [36]:

```
def create_waffle_chart(categories, values, height, width, colormap, value_sig
n=''):

    # compute the proportion of each category with respect to the total
    total_values = sum(values)
    category_proportions = [(float(value) / total_values) for value in values]

    # compute the total number of tiles
    total_num_tiles = width * height # total number of tiles
    print ('Total number of tiles is', total_num_tiles)

    # compute the number of tiles for each catagory
    tiles_per_category = [round(proportion * total_num_tiles) for proportion i
n category_proportions]

    # print out number of tiles per category
    for i, tiles in enumerate(tiles_per_category):
        print (df_dsn.index.values[i] + ': ' + str(tiles))

    # initialize the waffle chart as an empty matrix
    waffle_chart = np.zeros((height, width))

    # define indices to loop through waffle chart
    category_index = 0
    tile_index = 0

    # populate the waffle chart
    for col in range(width):
        for row in range(height):
            tile_index += 1
```

```
            # if the number of tiles populated for the current category
            # is equal to its corresponding allocated tiles...
            if tile_index > sum(tiles_per_category[0:category_index]):
                # ...proceed to the next category
                category_index += 1

            # set the class value to an integer, which increases with class
            waffle_chart[row, col] = category_index

    # instantiate a new figure object
    fig = plt.figure()

    # use matshow to display the waffle chart
    colormap = plt.cm.coolwarm
    plt.matshow(waffle_chart, cmap=colormap)
    plt.colorbar()

    # get the axis
    ax = plt.gca()

    # set minor ticks
    ax.set_xticks(np.arange(-.5, (width), 1), minor=True)
    ax.set_yticks(np.arange(-.5, (height), 1), minor=True)

    # add dridlines based on minor ticks
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)

    plt.xticks([])
    plt.yticks([])

    # compute cumulative sum of individual categories to match color schemes b
etween chart and legend
    values_cumsum = np.cumsum(values)
    total_values = values_cumsum[len(values_cumsum) - 1]

    # create legend
    legend_handles = []
    for i, category in enumerate(categories):
        if value_sign == '%':
            label_str = category + ' (' + str(values[i]) + value_sign + ')'
        else:
            label_str = category + ' (' + value_sign + str(values[i]) + ')'

        color_val = colormap(float(values_cumsum[i])/total_values)
        legend_handles.append(mpatches.Patch(color=color_val, label=label_str)
)

    # add legend to chart
    plt.legend(
        handles=legend_handles,
        loc='lower center',
        ncol=len(categories),
        bbox_to_anchor=(0., -0.2, 0.95, .1)
```

```
    )
```

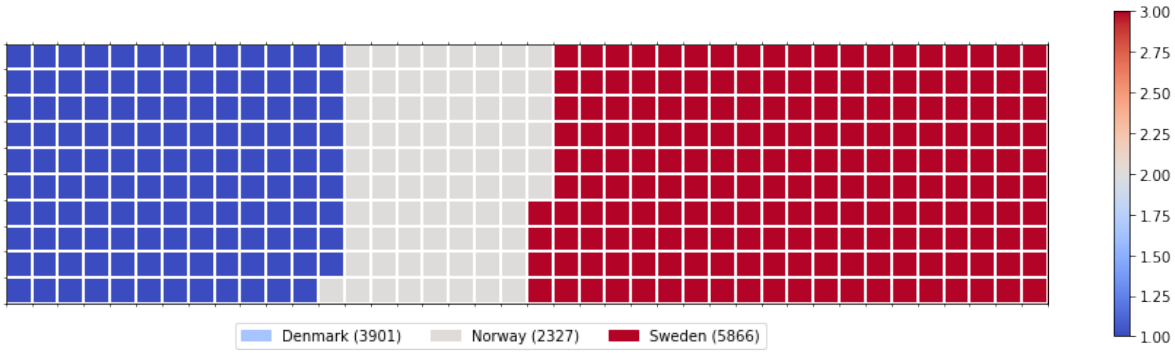
In [37]:

```
width = 40 # width of chart
height = 10 # height of chart

categories = df_dsn.index.values # categories
values = df_dsn['Total'] # correponding values of categories

colormap = plt.cm.coolwarm # color map class
```

In [38]:

```
create_waffle_chart(categories, values, height, width, colormap)
```

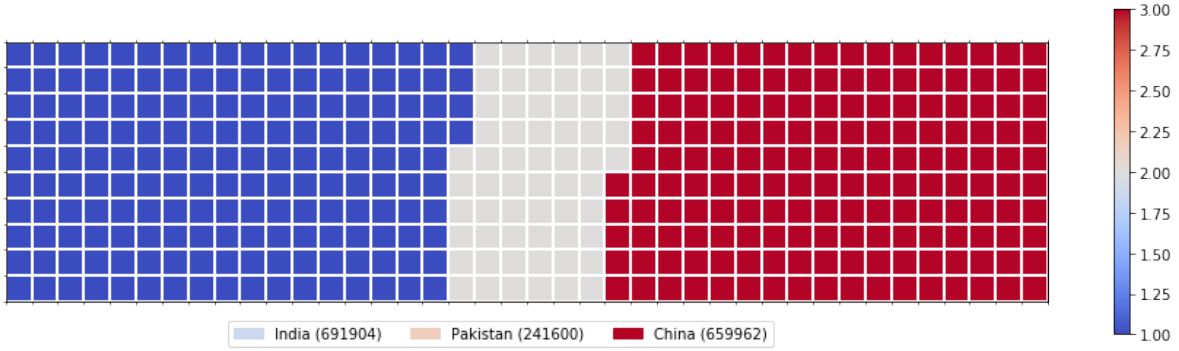
```
Total number of tiles is 400
Denmark: 129
Norway: 77
Sweden: 194

<Figure size 432x288 with 0 Axes>
```



In [39]:

```
indpakchi = canada.loc[['India', 'Pakistan', 'China'], :]
categories = list(indpakchi.index.values)
values = indpakchi['Total']
create_waffle_chart(categories, values, height, width, colormap)
```

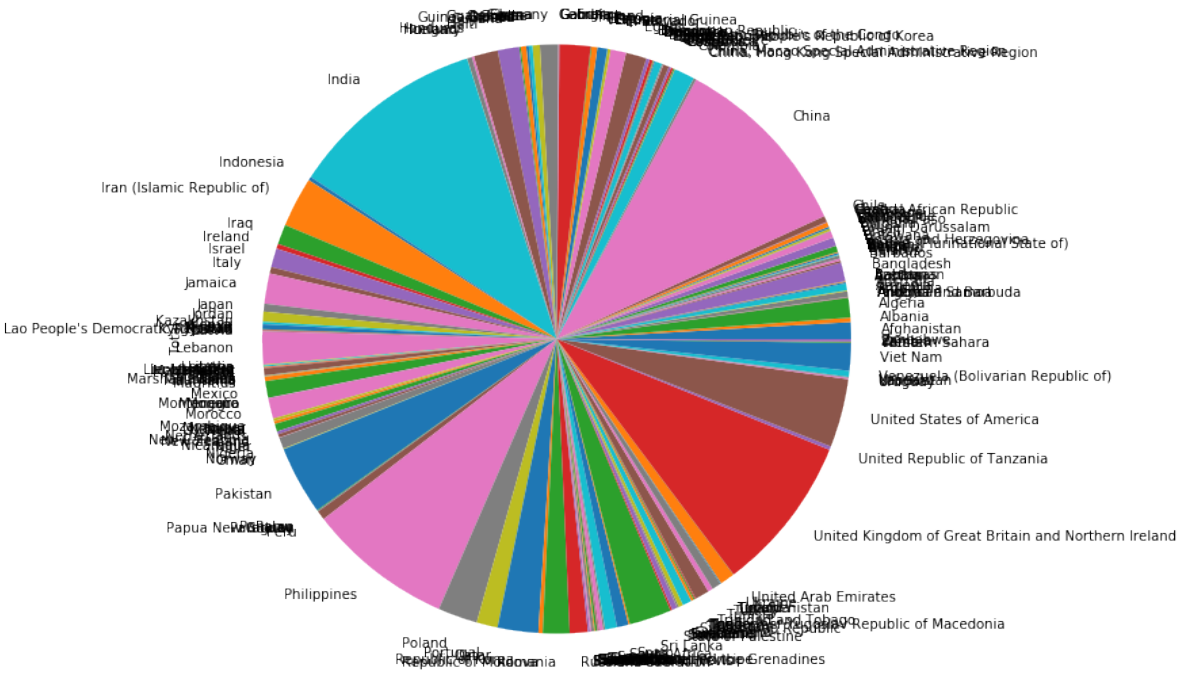
```
Total number of tiles is 400
Denmark: 174
Norway: 61
Sweden: 166

<Figure size 432x288 with 0 Axes>
```



Total Count of Immigration across all countries

In [40]:

```
canada['Total'].plot(kind='pie',figsize=(10,10));
```



Plotting the total immigration data by each continent

In [41]:

```
can_continent = canada.groupby('Continent', axis='index').sum()
can_continent.head()
```

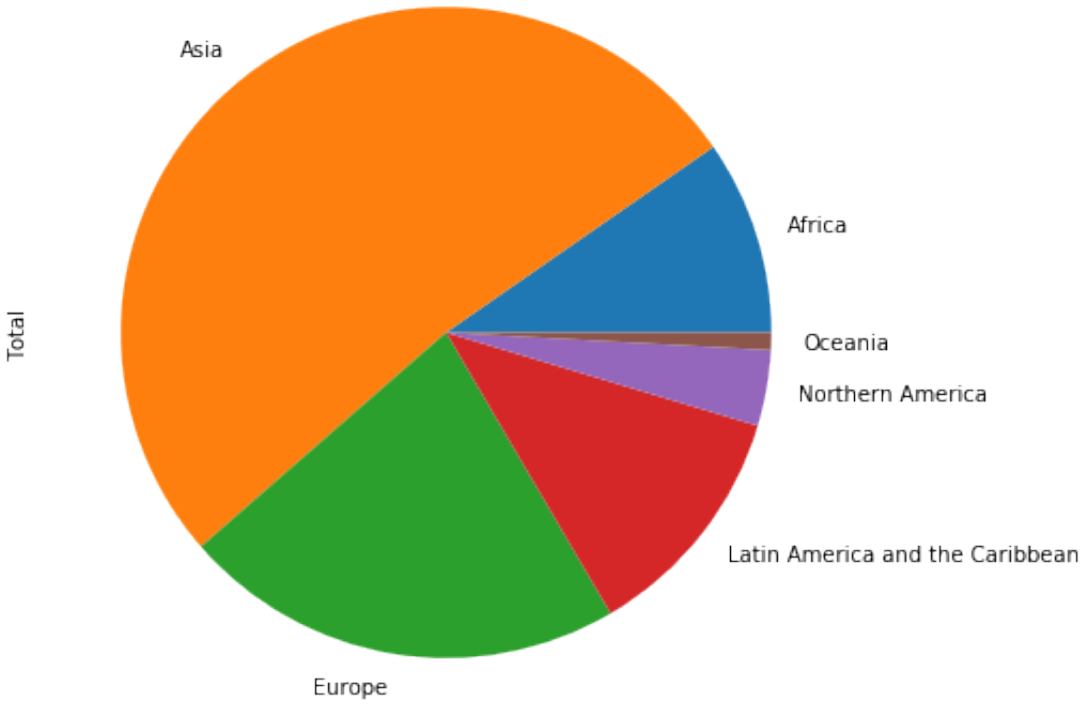
Out[41]:

Continent	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	
Africa	3951	4363	3819	2671	2639	2650	3782	7494	7552	9894	...	2
Asia	31025	34314	30214	24696	27274	23850	28739	43203	47454	60256	...	15
Europe	39760	44802	42720	24638	22287	20844	24370	46698	54726	60893	...	3
Latin America and the Caribbean	13081	15215	16769	15427	13678	15171	21179	28471	21924	25060	...	2
Northern America	9378	10030	9074	7100	6661	6543	7074	7705	6469	6790	...	

5 rows × 35 columns

In [42]:

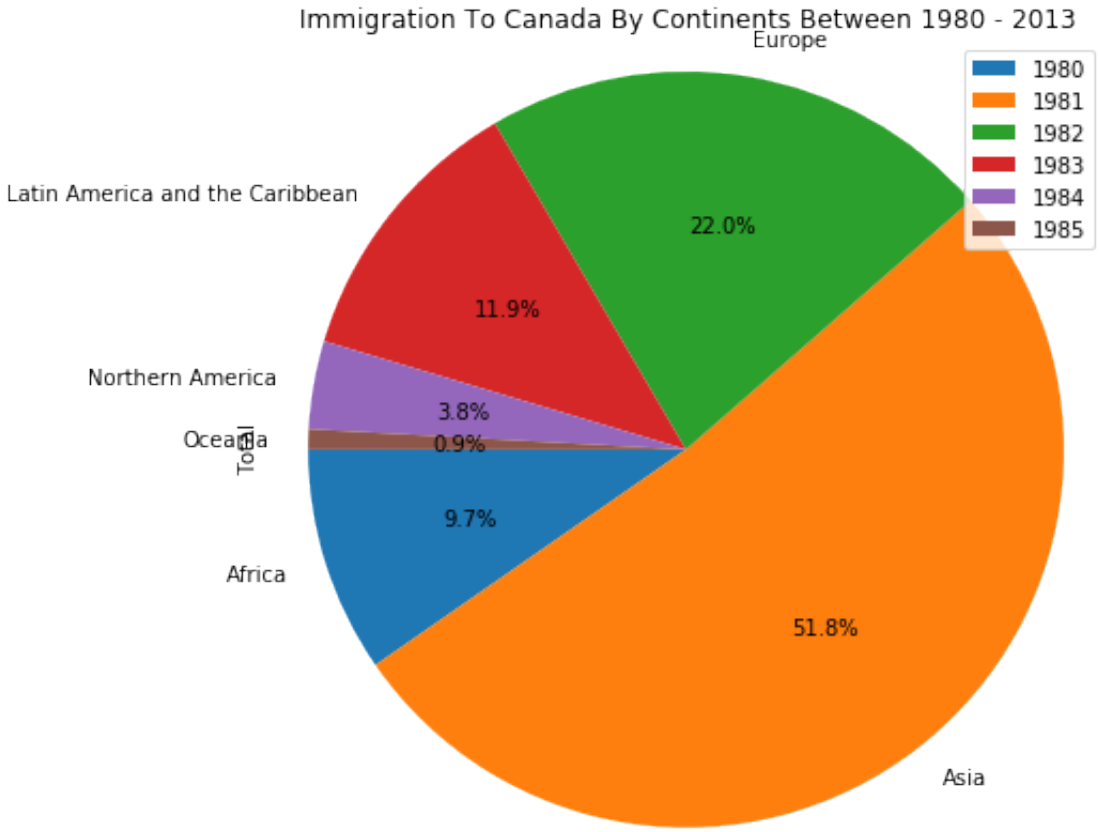
```
can_continent['Total'].plot(kind='pie', figsize=(7, 7));
```



Modifying the graph

In [43]:

```
can_continent['Total'].plot(kind='pie', figsize=(7, 7),
                                    startangle = 180, autopct = '%1.1f%%')
plt.title('Immigration To Canada By Continents Between 1980 - 2013')
plt.axis('equal')
plt.legend(labels = can_continent.columns, loc='upper right')
plt.show()
```



Plotting Immigration Data by Development Status

In [44]:

```
can_devstatus = canada.groupby('DevName', axis='index').sum()
can_devstatus.head()
```

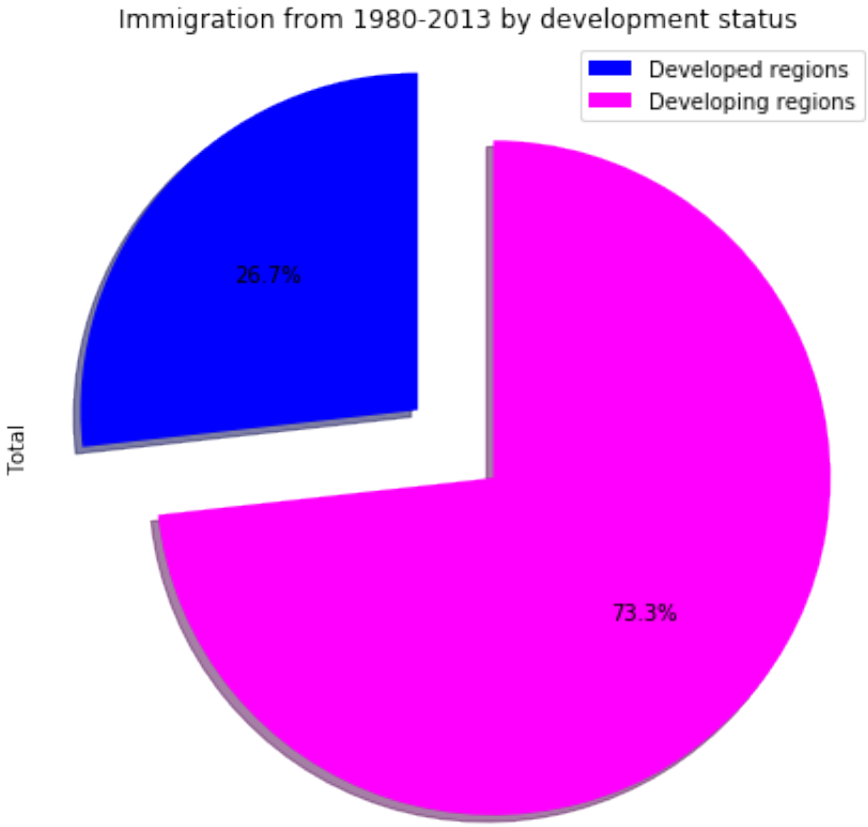
Out[44]:

DevName	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	
Developed regions	51143	56707	53240	32504	29675	28052	32224	55500	62129	68967	...	
Developing regions	47994	53856	51031	43046	43742	41926	53824	79271	77177	95465	...	2

2 rows × 35 columns

In [45]:

```
explodeList = [0.3,0]
colorList=['blue','magenta']
can_devstatus['Total'].plot(kind='pie', figsize=(7,7), startangle=90,autopct='
%1.1f%%',
                                    shadow=True, labels = None, colors = colorLis
t, explode = explodeList)
plt.title('Immigration from 1980-2013 by development status')
plt.axis('equal')
plt.legend(labels= can_devstatus.index,loc ='upper right')
plt.show()
```



Plotting a pie chart for Immigration with respect to Country-Region

In [46]:

```
can_contreg = canada.groupby('Continent-Region', axis='index').sum()
can_contreg.head()
```

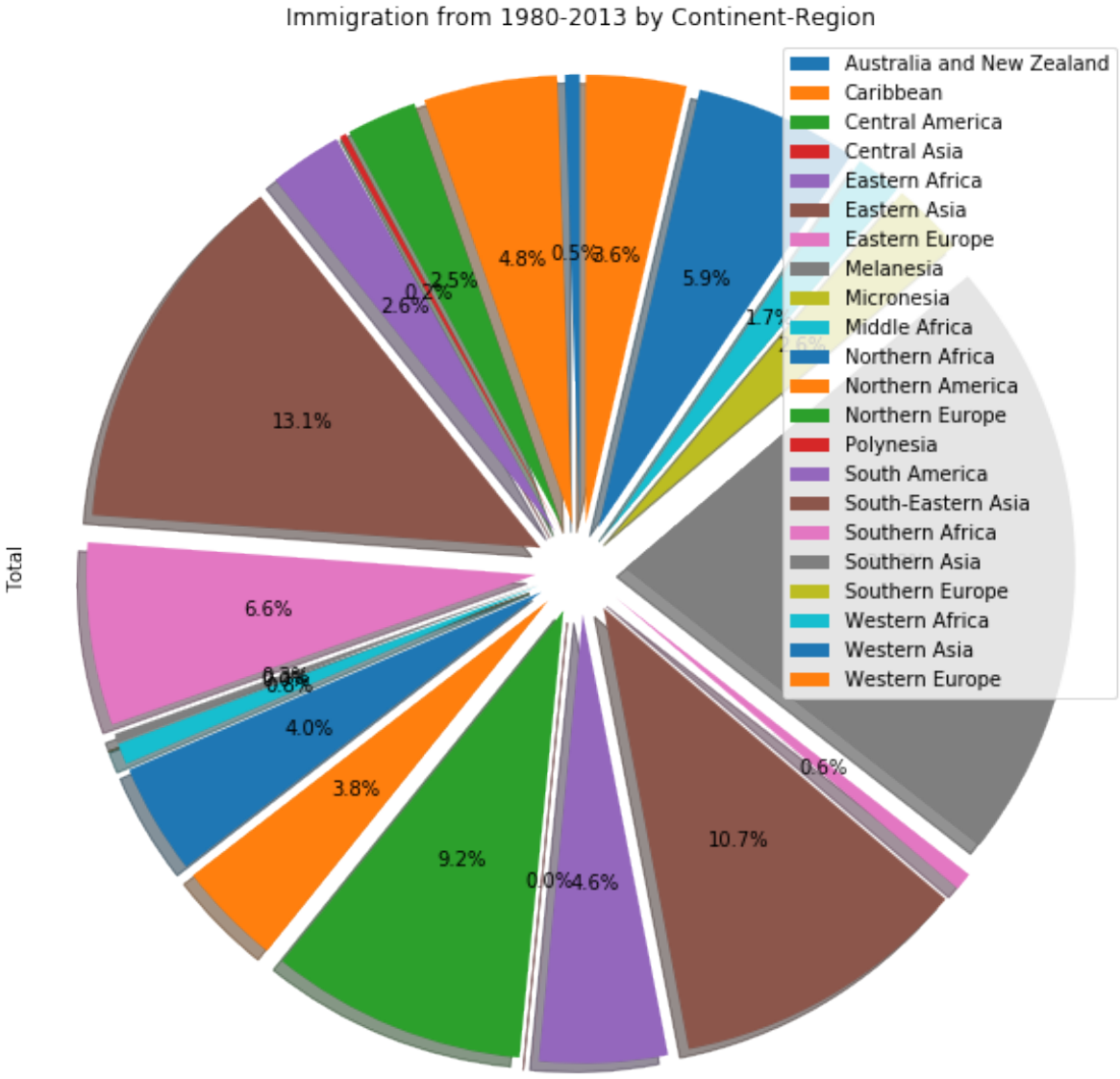
Out[46]:

Continent-Region	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2005	2006
Australia and New Zealand	1304	1119	848	457	481	467	532	675	610	790	...	1279	1193
Caribbean	7045	8310	8326	6998	5553	6048	8716	10932	9229	10786	...	6816	6652
Central America	734	921	1612	3648	4087	4862	5909	6804	5596	5821	...	3990	4140
Central Asia	0	0	0	0	0	0	0	0	0	0	...	1134	903
Eastern Africa	1471	1641	1426	1094	1187	1134	1454	2734	3237	4094	...	7083	6750

5 rows × 35 columns

In [47]:

```
explodeList = [0.1]*22
#colorList=['blue','magenta']
can_contreg['Total'].plot(kind='pie', figsize=(10,10), startangle=90,autopct='
%1.1f%%',
                                  shadow=True, labels = None, explode = explode
List)
plt.title('Immigration from 1980-2013 by Continent-Region')
plt.axis('equal')
plt.legend(labels= can_contreg.index,loc ='upper right')
plt.show()
```

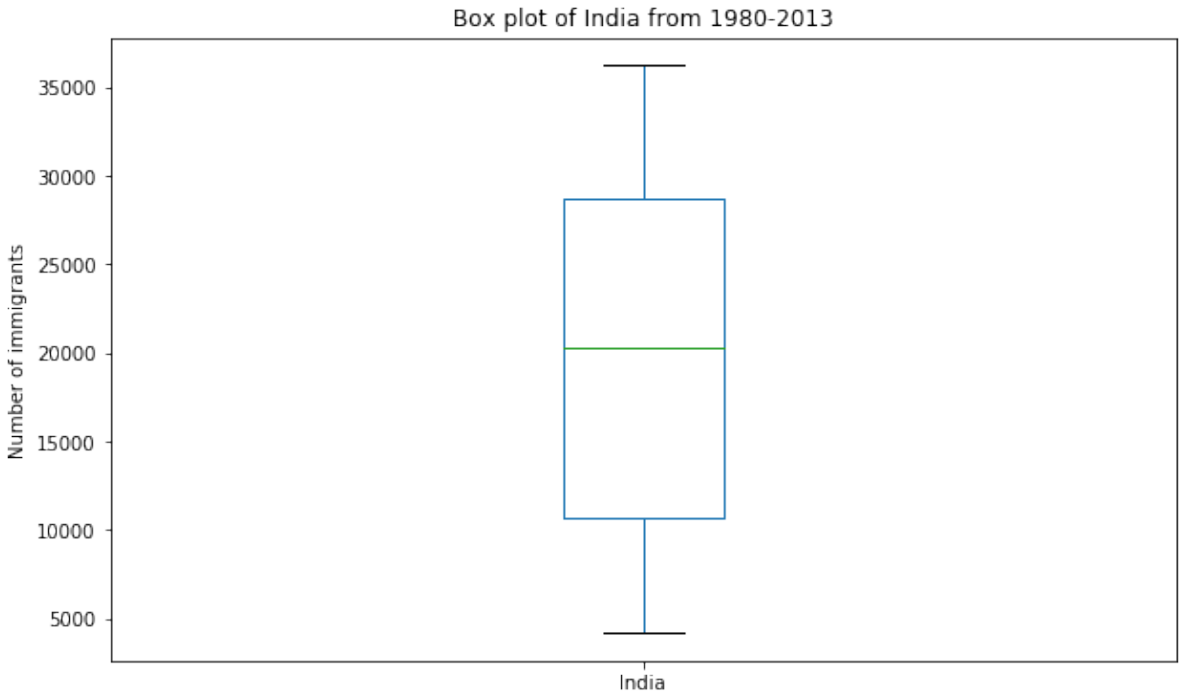


Box Plots

Plotting a boxplot for Immigration from India across the years

In [48]:

```
india = canada.loc['India', years]

india.plot(kind='box', figsize=(10,6))
plt.title('Box plot of India from 1980-2013')
plt.ylabel('Number of immigrants')
plt.show()
```



Comparative Boxplot for Immigration from India and Pakistan

In [49]:

```
indopak = canada.loc[['India','Pakistan'], years].transpose()

indopak.plot(kind='box', figsize=(10,6))
plt.title('Box plot of India and Pakistan from 1980-2013')
plt.ylabel('Number of immigrants')
plt.show()
```

Describing the Ind-Pak Immigration Data

In [50]:

```
indopak.describe()
```

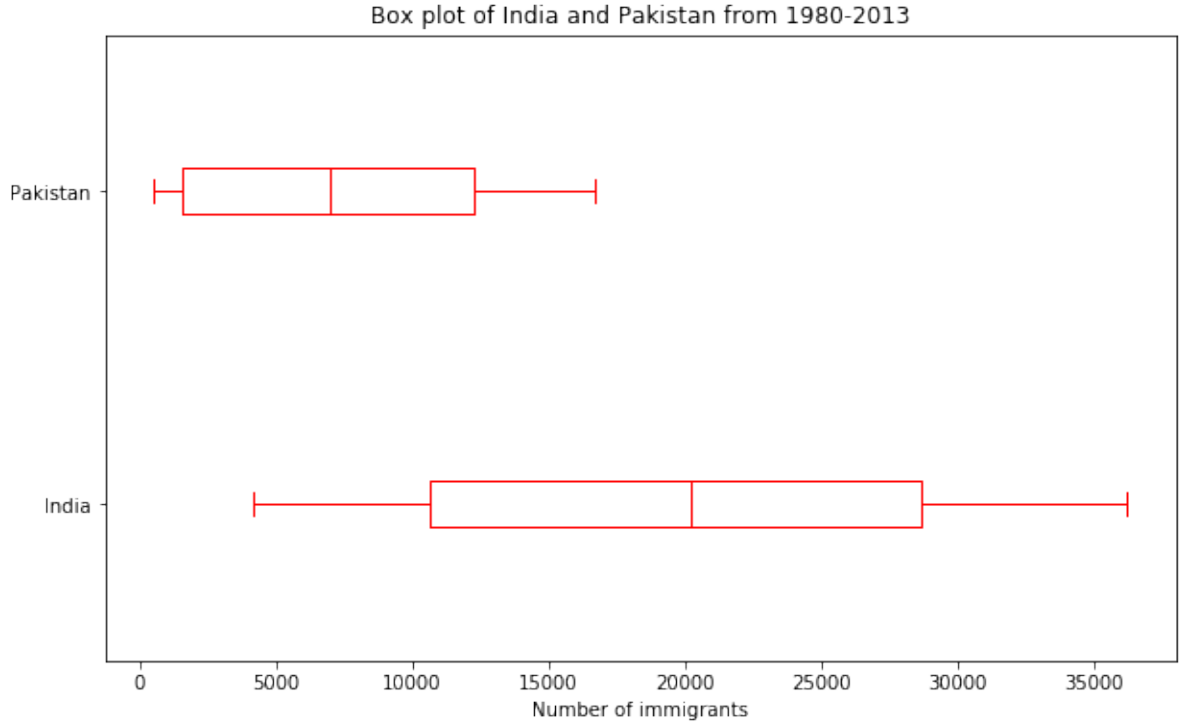
Out[50]:

Country	India	Pakistan
count	34.000000	34.000000
mean	20350.117647	7105.882353
std	10007.342579	5315.849587
min	4211.000000	514.000000
25%	10637.750000	1565.750000
50%	20235.000000	7014.000000
75%	28699.500000	12259.000000
max	36210.000000	16708.000000

Modifying the Boxplot for Ind-Pak Data

In [51]:

```
indopak.plot(kind='box', figsize=(10,6), color='red', vert=False)

plt.title('Box plot of India and Pakistan from 1980-2013')
plt.xlabel('Number of immigrants')
plt.show()
```



Creating 4 subplots of for boxplots of top 5 countries In Europe, Asia, Africa and Oceania

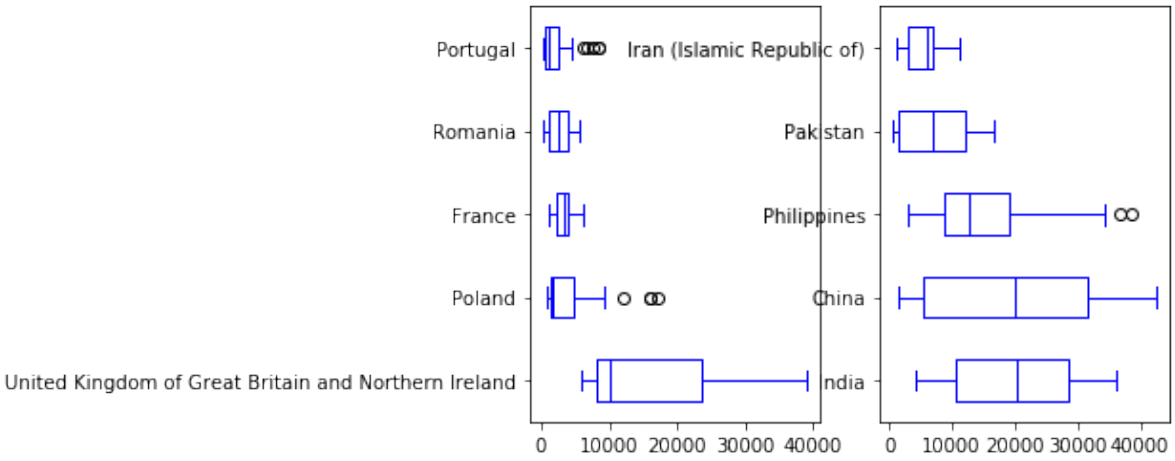
In [52]:

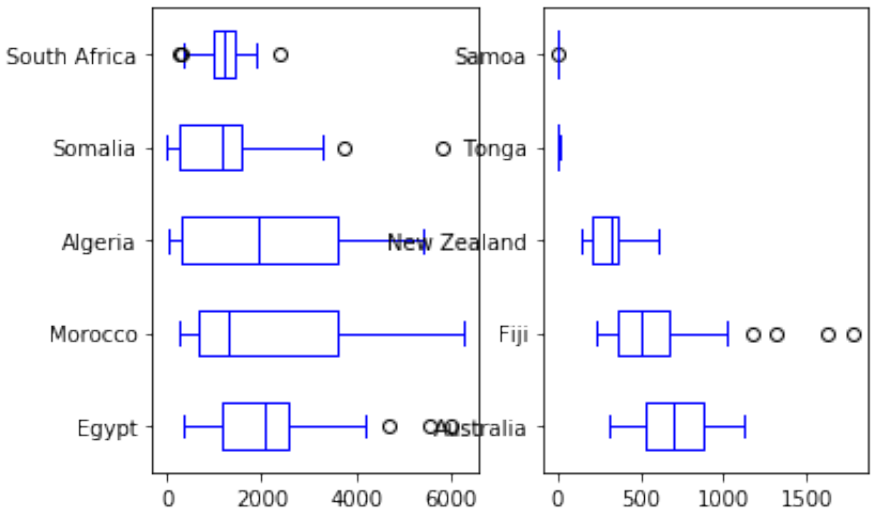
```
asia = canada[(canada['Continent']=='Asia')].sort_values('Total', ascending=Fa
lse).head()
eur = canada[(canada['Continent']=='Europe')].sort_values('Total', ascending=F
alse).head()
afr = canada[(canada['Continent']=='Africa')].sort_values('Total', ascending=F
alse).head()
ocn = canada[(canada['Continent']=='Oceania')].sort_values('Total', ascending=
False).head()

asia = asia[years].transpose()
eur = eur[years].transpose()
afr = afr[years].transpose()
ocn = ocn[years].transpose()
```

PLots

In [53]:

```
fig = plt.figure();
ax1 = fig.add_subplot(1,2,1);
eur.plot(kind='box', color='blue', vert=False, ax=ax1);
ax2 = fig.add_subplot(1,2,2);
asia.plot(kind='box', color='blue', vert=False, ax=ax2);

fig2 = plt.figure();
ax3 = fig2.add_subplot(1,2,1);
afr.plot(kind='box', color='blue', vert=False, ax=ax3);
ax4 = fig2.add_subplot(1,2,2);
ocn.plot(kind='box', color='blue', vert=False, ax=ax4);
```





Outliers

In [54]:

```
temp = asia.describe()
Q1 = temp.loc['25%', 'Philippines']
Q3 = temp.loc['75%', 'Philippines']
IQR = Q3-Q1

Outlier1 = Q3 + 1.5*IQR
Outlier2 = Q1 - 1.5*IQR
print(Outlier1)
print(Outlier2)

asia[asia['Philippines']>Outlier1]
```

```
35128.0
-7216.0
```

Out[54]:

Country	India	China	Philippines	Pakistan	Iran (Islamic Republic of)
2010	34235	30391	38617	6811	7477
2011	27509	28502	36765	7468	7479

Word Clouds

importing

In [55]:

```
from wordcloud import WordCloud, STOPWORDS
```

In [56]:

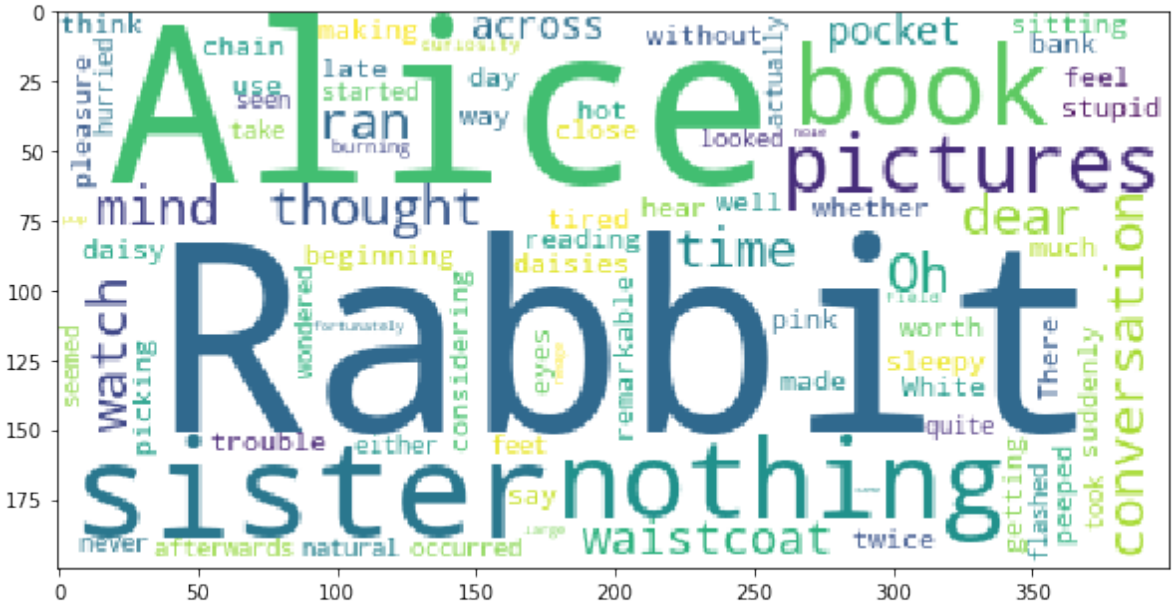
```
alice_novel = open('/Users/home/Downloads/1462444-088c12e7f74fcff9be6c4faa556f961cb7a675bd/alice.txt', 'r').read()
```

In [57]:

```
stopwordsremove = set(STOPWORDS)

alice_wc = WordCloud(background_color = 'white', max_words = 2000,
    stopwords = stopwordsremove)
```

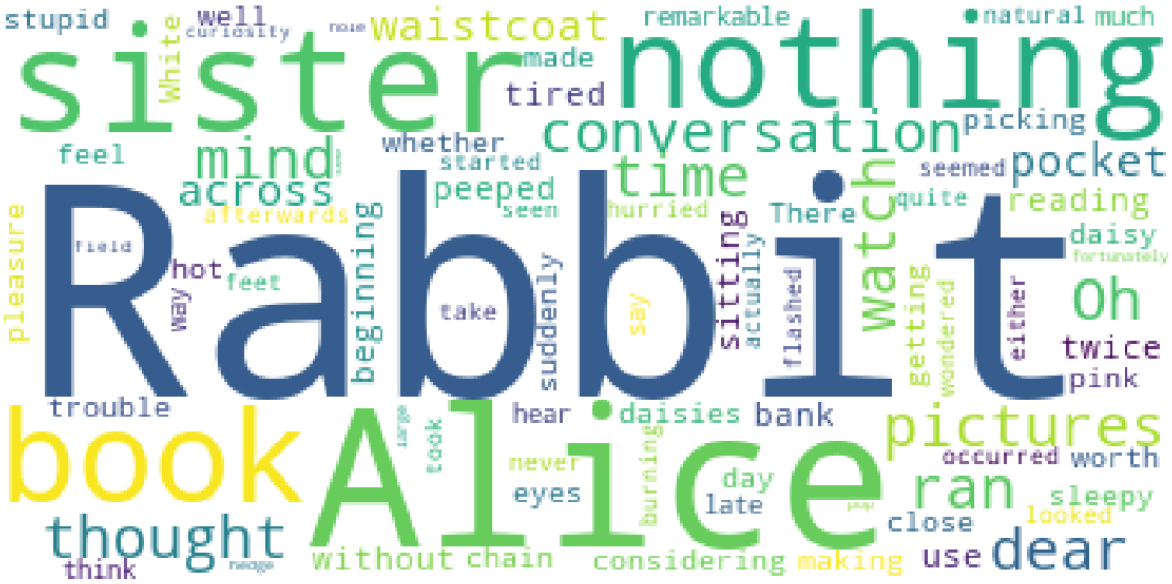
In [58]:

```
alice_wc.generate(alice_novel)

plt.figure(figsize = (10,15))
plt.imshow(alice_wc)
plt.show()
```



In [59]:

```
stopwordsremove = set(STOPWORDS)
stopwordsremove.add('said')

alice_wc = WordCloud(background_color = 'white', max_words = 3000,
    stopwords = stopwordsremove)

alice_wc.generate(alice_novel)

fig = plt.figure()
fig.set_figheight(20)
fig.set_figwidth(20)
plt.imshow(alice_wc)
plt.axis('off')
plt.show()
```



Maps

In [60]:

```
import folium
```

Map of Dahisar

In [61]:

```
worldmap = folium.Map(location=[19.25, 72.86], zoom_start=15)
worldmap
```

Out[61]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [62]:

```
worldmap = folium.Map(location=[19.25, 72.86], zoom_start=14, tiles='Stamen To
ner')
worldmap
```

Out[62]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [63]:

```
worldmap = folium.Map(location=[19.25, 72.86], zoom_start=14, tiles='Stamen Terrain')
worldmap
```

Out[63]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Loading different dataset

In [64]:

```
incidents = pd.read_csv('https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DV0101EN/labs/Data_Files/Police_Department_Incidents_-_Previous_Year__2016_.csv')
print('Dataset downloaded and read into a pandas dataframe!')
print(incidents.head())
print(incidents.shape)
```

```
Dataset downloaded and read into a pandas dataframe!
   IncidntNum      Category
Descript  \
0   120058272   WEAPON LAWS                         POSS OF PROHIBIT
ED WEAPON
1   120058272   WEAPON LAWS  FIREARM, LOADED, IN VEHICLE, POSSESSI
ON OR USE
2   141059263      WARRANTS                                WARRA
NT ARREST
3   160013662  NON-CRIMINAL                                 LOST
PROPERTY
4   160002740  NON-CRIMINAL                                 LOST
PROPERTY

  DayOfWeek                     Date   Time  PdDistrict      Resolu
tion  \
0    Friday  01/29/2016 12:00:00 AM  11:00    SOUTHERN  ARREST, BO
OKED
1    Friday  01/29/2016 12:00:00 AM  11:00    SOUTHERN  ARREST, BO
OKED
2    Monday  04/25/2016 12:00:00 AM  14:59     BAYVIEW  ARREST, BO
OKED
3   Tuesday  01/05/2016 12:00:00 AM  23:50  TENDERLOIN
NONE
4    Friday  01/01/2016 12:00:00 AM  00:30     MISSION
NONE

                  Address           X          Y  \
0  800 Block of BRYANT ST -122.403405  37.775421
1  800 Block of BRYANT ST -122.403405  37.775421
2   KEITH ST / SHAFTER AV -122.388856  37.729981
3  JONES ST / OFARRELL ST -122.412971  37.785788
4    16TH ST / MISSION ST -122.419672  37.765050

                             Location            PdId
0   (37.775420706711, -122.403404791479)  12005827212120
1   (37.775420706711, -122.403404791479)  12005827212168
2  (37.7299809672996, -122.388856204292)  14105926363010
3  (37.7857883766888, -122.412970537591)  16001366271000
4  (37.7650501214668, -122.419671780296)  16000274071000
(150500, 13)
```

In [65]:

```
limit = 100
inc = incidents.iloc[0:limit,:]
```

In [66]:

```
latitude = 37.77
longitude = -122.42
map1 = folium.Map(location = [latitude, longitude], zoom_start=12)
map1
```

Out[66]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [67]:

```
#given information about city coordinates
latitude= 37.77
longitude = -122.42

#create map using folium library and Map class
map1 = folium.Map(location = [latitude, longitude], zoom_start=12)

map1

#incident we will impose on the map
incident = folium.map.FeatureGroup()

#reduce size of dataset
df1 = incidents.iloc[0:limit,:]

#loop through the crimes by their lat Y and long X and create markers
for lat, long in zip(df1.Y, df1.X):
    incident.add_child(folium.CircleMarker([lat, long], radius = 5, color = 'y
ellow', fill=True,
            fill_color='blue').add_to(incident))

#add pop ups
lats = list(df1.Y)
longs = list(df1.X)
labels = list(df1.Category)

for lat, long, labs in zip(lats, longs, labels):
        folium.Marker([lat, long], popup=labs).add_to(map1)

#add incidents/crimes to the map
map1.add_child(incident)
```

Out[67]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [68]:

```
#given information about city coordinates
latitude= 37.77
longitude = -122.42

#create map using folium library and Map class
map1 = folium.Map(location = [latitude, longitude], zoom_start=12)
map1

#incident we will impose on the map
incident = folium.map.FeatureGroup()

#reduce size of dataset
df1 = incidents.iloc[0:limit,:]

#loop through the crimes by their lat Y and long X and create markers
lats = list(df1.Y)
longs = list(df1.X)
labels = list(df1.Category)

for lat, long, labs in zip(lats, longs, labels):
        folium.CircleMarker([lat, long], popup=labs, radius=5, color='yellow',
fill=True,
                            fill_color='blue').add_to(map1)

#add incidents/crimes to the map
map1
```

Out[68]:

Make this Notebook Trusted to load map: File -> Trust Notebook

In [69]:

```
#given information about city coordinates
latitude= 37.77
longitude = -122.42

#create map using folium library and Map class
map1 = folium.Map(location = [latitude, longitude], zoom_start=12)
map1

#incident we will impose on the map
from folium import plugins
incident = plugins.MarkerCluster().add_to(map1)

#reduce size of dataset
df1 = incidents.iloc[0:limit,:]

#loop through the crimes by their lat Y and long X and create markers
lats = list(df1.Y)
longs = list(df1.X)
labels = list(df1.Category)

for lat, long, labs in zip(lats, longs, labels):
        folium.CircleMarker([lat, long], popup=labs, radius=5, color='yellow',
fill=True,
            fill_color='blue').add_to(incident)

#add incidents/crimes to the map
map1
```

Out[69]:

Make this Notebook Trusted to load map: File -> Trust Notebook

Choropeth Markers

world_geo = pd.read_json(r'/Users/home/Downloads/world_countries.json')canada = pd.read_excel('https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DV0101EN/labs/Data_Files/Canada.xlsx', sheet_name='Canada by Citizenship', skiprows=range(20), skipfooter=2) canada.drop(['AREA','REG','DEV','Type','Coverage'], inplace=True, axis='columns') canada.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':'Continent-Region', 'DevName':'Development-Status'}, inplace=True) canada['TotalImmigration'] = canada.sum(axis='columns') canada.head()world_map = folium.Map(location=[0, 0], zoom_start=2) world_map.choropleth(geo_data=world_geo, data=canada, columns=['Country', 'TotalImmigration'], key_on='feature.properties.name', fill_color='YlOrRd', fill_opacity=0.7, line_opacity=0.2, legend_name='Immigration to Canada') # display map world_mapworld_geo = r'world_countries.json' # create a numpy array of length 6 and has linear spacing from the minium total immigration to the maximum total immigration threshold_scale = np.linspace(df_can['Total'].min(), df_can['Total'].max(), 6, dtype=int) threshold_scale = threshold_scale.tolist() # change the numpy array to a list threshold_scale[-1] = threshold_scale[-1] + 1 # make sure that the last value of the list is greater than the maximum immigration # let Folium determine the scale. world_map = folium.Map(location=[0, 0], zoom_start=2, tiles='Mapbox Bright') world_map.choropleth(geo_data=world_geo, data=canada, columns=['Country', 'Total'], key_on='feature.properties.name', threshold_scale=threshold_scale, fill_color='YlOrRd', fill_opacity=0.7, line_opacity=0.2, legend_name='Immigration to Canada', reset=True) world_map

Assignment 1

Which two countries have similar immigration trends over the years 1980-2013?

In [70]:

```
canada['total_immigration'] = canada.sum(axis='columns')
canada.sort_values(by='total_immigration', ascending=False, axis='index', inpl
ace=True)
canada.head(10)
```

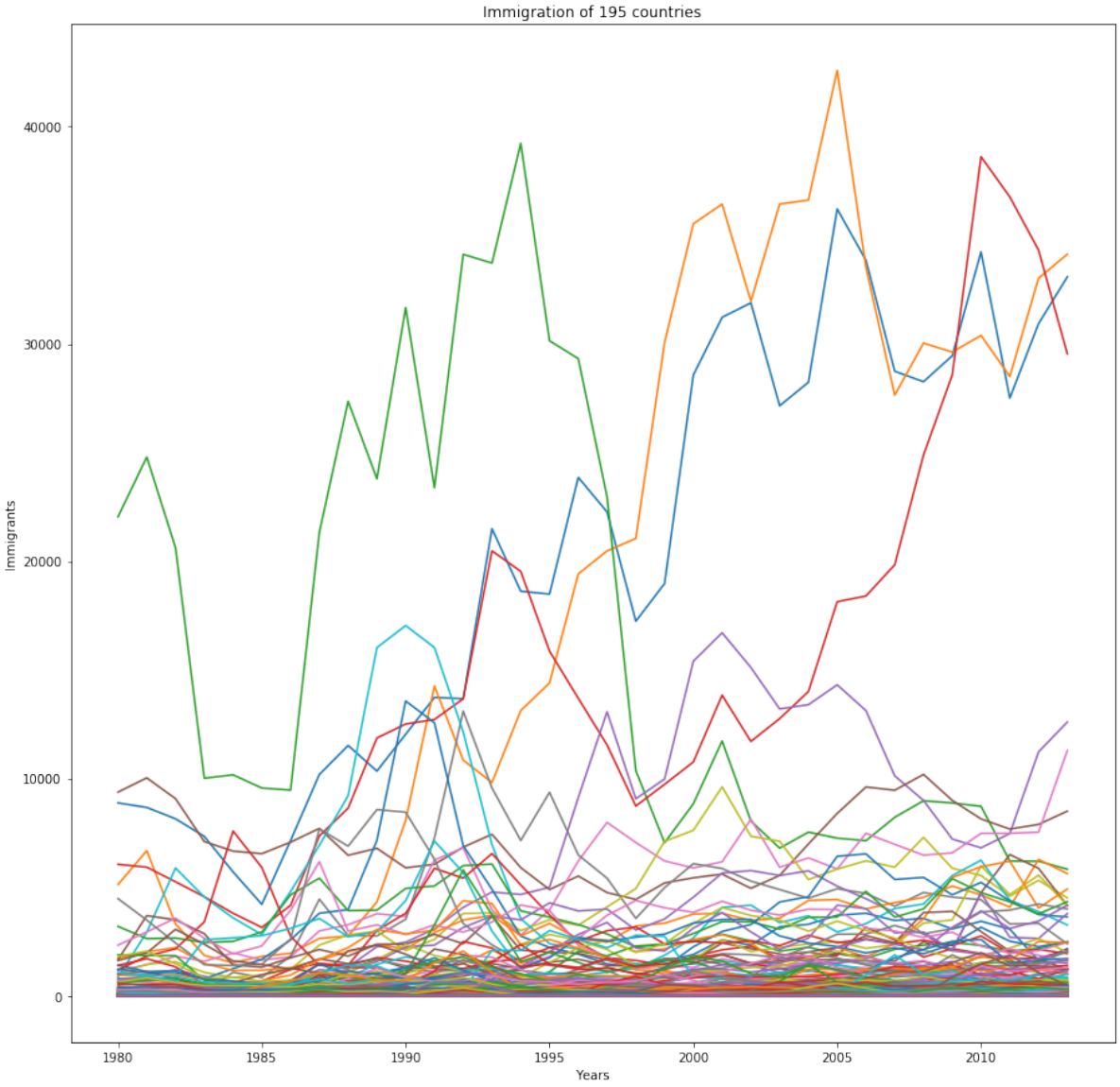
Out[70]:

Country	Continent	Continent-Region	DevName	1980	1981	1982	1983	1984	1985	1
India	Asia	Southern Asia	Developing regions	8880	8670	8147	7338	5704	4211	7
China	Asia	Eastern Asia	Developing regions	5123	6682	3308	1863	1527	1816	1
United Kingdom of Great Britain and Northern Ireland	Europe	Northern Europe	Developed regions	22045	24796	20620	10015	10170	9564	9
Philippines	Asia	South-Eastern Asia	Developing regions	6051	5921	5249	4562	3801	3150	4
Pakistan	Asia	Southern Asia	Developing regions	978	972	1201	900	668	514	
United States of America	Northern America	Northern America	Developed regions	9378	10030	9074	7100	6661	6543	7
Iran (Islamic Republic of)	Asia	Southern Asia	Developing regions	1172	1429	1822	1592	1977	1648	1
Sri Lanka	Asia	Southern Asia	Developing regions	185	371	290	197	1086	845	1
Republic of Korea	Asia	Eastern Asia	Developing regions	1011	1456	1572	1081	847	962	1
Poland	Europe	Eastern Europe	Developed regions	863	2930	5881	4546	3588	2819	4

10 rows × 39 columns

In [71]:

```
canada.loc[canada.index.tolist(), years].transpose().plot(figsize=(15,15),lege
nd=False);
plt.title('Immigration of 195 countries');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Plotting three similar countries after year 2009 in the top 10 list.

In [72]:

```
canada.loc[['India','China','Philippines'], years].transpose().plot(figsize=(15,15));
plt.title('Immigration of India, China and Philippines');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```

Since 2009, India China and Philippines have shown similar trends.

As it is impossible to compare 195 countries and their charts, Last 3 year's average is taken and compared in the following steps. The selection of n years is random.

Taking avg. of last 3 years and cleaning the Series

In [73]:

```
canada['avglast3'] = (canada['2011']+canada['2012']+canada['2013'])/3
l3 = canada['avglast3']
l3 = l3.sort_values(ascending = False )
l3 = l3.round()
l3 = l3[l3>2]
```

In [74]:

```
l3.head(10)
```

Out[74]:

```
Country
Philippines                                                 33541.0
China                                                       31885.0
India                                                       30510.0
Pakistan                                                    10433.0
Iran (Islamic Republic of)                                   8768.0
United States of America                                     8023.0
United Kingdom of Great Britain and Northern Ireland         6075.0
Haiti                                                        5508.0
France                                                       5328.0
Iraq                                                         5052.0
Name: avglast3, dtype: float64
```

Removing outliers

In [75]:

```
l5 = l3[l3<11000];
```

Calculating Standard Deviation for bins

In [76]:

```
np.std(l5)
```

Out[76]:

```
1629.5934270734047
```

Creating bins

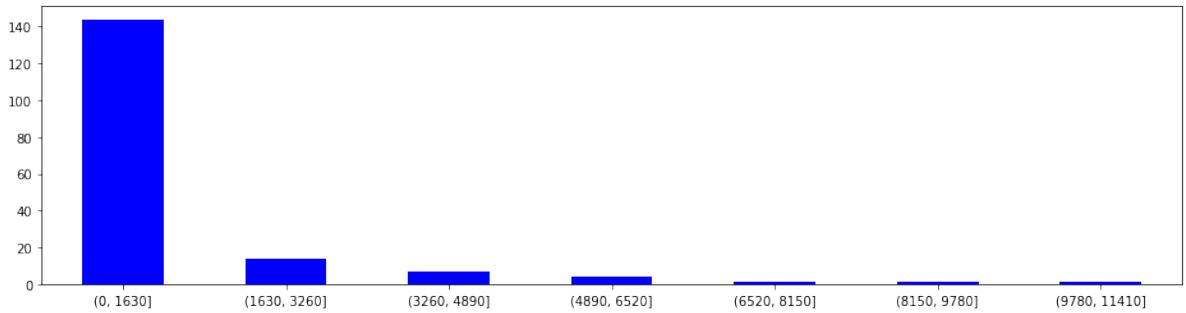
In [77]:

```
type(l5)
dl5 = df(l5)
dl5
ahg = pd.Series(range(0,12000,1630))
bins = ahg
dl5['binned'] = pd.cut(dl5['avglast3'], bins)
```

Plotting the bar graph

In [78]:

```
out = dl5['binned']
ax = out.value_counts(sort=False).plot.bar(rot=0, color="b", figsize=(16,4))
plt.show()
```



Bin values

In [79]:

```
dl5['binned'].value_counts(sort=True)
```

Out[79]:

```
(0, 1630]        144
(1630, 3260]      14
(3260, 4890]       7
(4890, 6520]       4
(9780, 11410]      1
(8150, 9780]       1
(6520, 8150]       1
Name: binned, dtype: int64
```

Countries showing similar trends (2011-2013), grouped

In [80]:

```
print('(9780, 11410]')
print(' {ho} {jo}'.format(ho = l5.index[0], jo = l5[0]))
print('')
print('')
print('(8150, 9780]')
print(' {ho} {jo}'.format(ho = l5.index[1], jo = l5[1]))
print('')
print('')
print('(6520, 8150]')
print(' {ho} {jo}'.format(ho = l5.index[2], jo = l5[2]))
print('')
print('')
print('(4890, 6520]')
print(' {jo}'.format(jo = l5[3:7]))
print('')
print('')
print('(3260, 4890]')
print(' {jo}'.format(jo = l5[7:14]))
print('')
print('')
print('(1630, 3260]')
print(' {jo}'.format(jo = l5[14:28]))
print('')
print('')
print('(0, 1630]')
print(' {jo}'.format(jo = l5[28:172]))
```

```
(9780, 11410]
 Pakistan 10433.0


(8150, 9780]
 Iran (Islamic Republic of) 8768.0


(6520, 8150]
 United States of America 8023.0


(4890, 6520]
 Country
United Kingdom of Great Britain and Northern Ireland    6075.0
Haiti                                                    5508.0
France                                                   5328.0
Iraq                                                     5052.0
Name: avglast3, dtype: float64


(3260, 4890]
 Country
Republic of Korea    4804.0
```

```
Egypt                 4794.0
Algeria               4143.0
Mexico                4057.0
Colombia              3913.0
Morocco               3846.0
Nigeria               3573.0
Name: avglast3, dtype: float64


(1630, 3260]
 Country
Bangladesh                          3041.0
Sri Lanka                           3014.0
Ukraine                             2422.0
Lebanon                             2286.0
Afghanistan                         2281.0
Jamaica                             2240.0
Cameroon                            2195.0
Russian Federation                  2169.0
Israel                              2016.0
Ethiopia                            1878.0
Viet Nam                            1855.0
Somalia                             1715.0
Democratic Republic of the Congo    1663.0
Germany                             1657.0
Name: avglast3, dtype: float64


(0, 1630]
 Country
Romania               1625.0
Brazil                1621.0
Tunisia               1524.0
Eritrea               1412.0
Jordan                1365.0
                       ...
Equatorial Guinea        6.0
Mozambique               6.0
Brunei Darussalam        5.0
Cabo Verde               4.0
Tonga                    3.0
Name: avglast3, Length: 144, dtype: float64
```

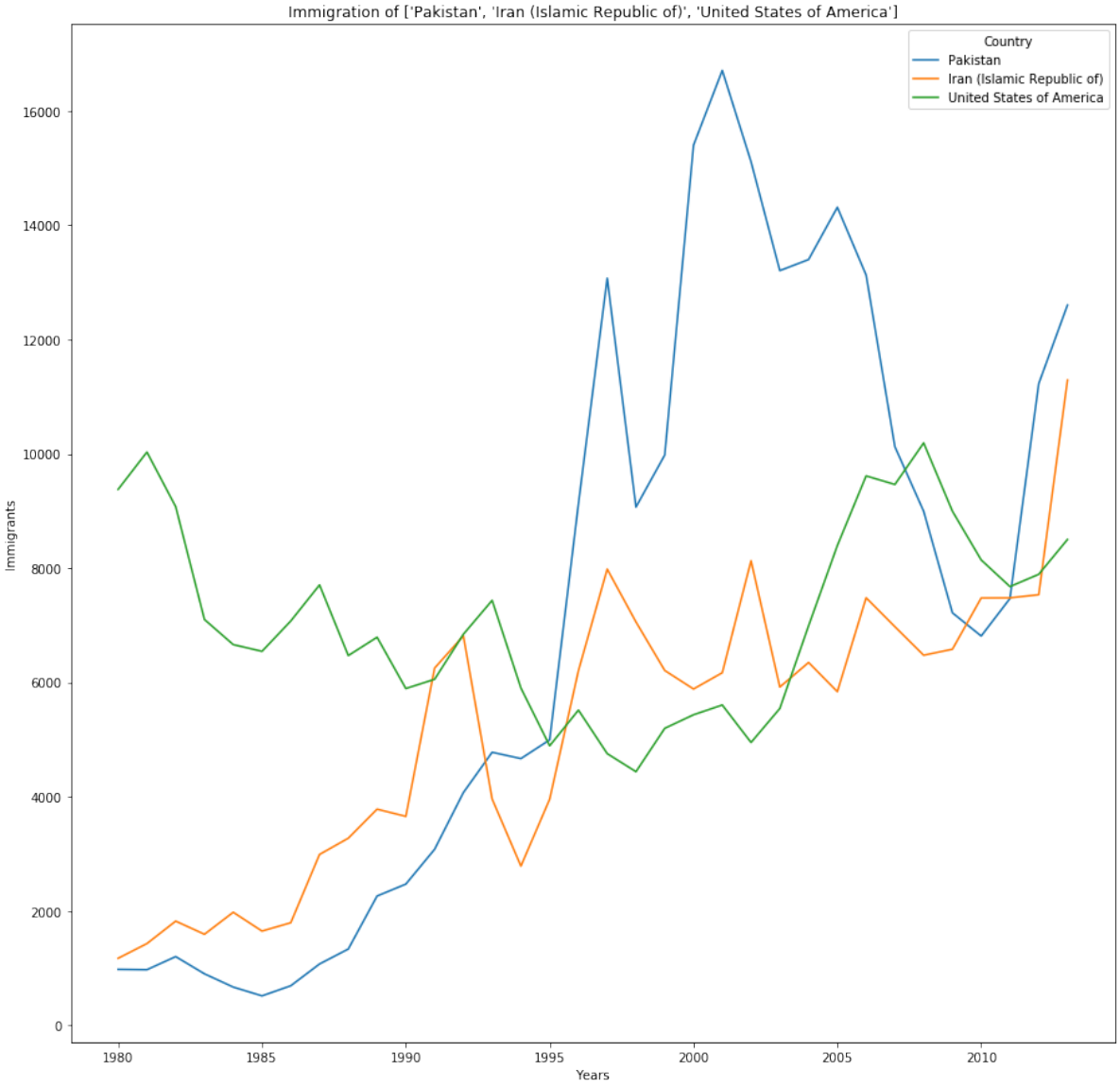
Plots

Comparison on basis of average of immigrants between years 2011 and 2013

Top 3 - excluding India, China and Philippines

In [81]:

```
a1 = l5[0:3].index.tolist()
canada.loc[a1, years].transpose().plot(figsize=(15,15));
plt.title('Immigration of {as1}'.format(as1 = a1));
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



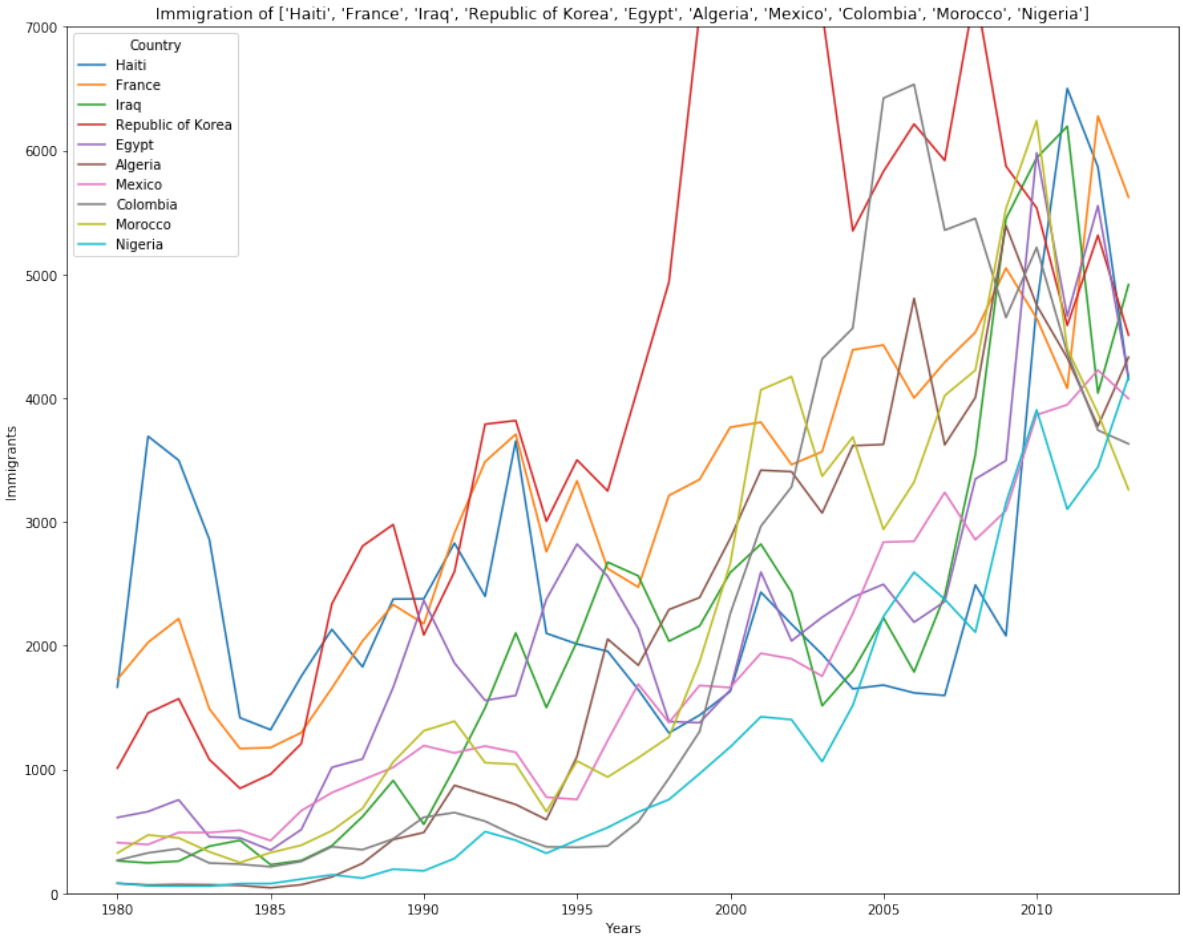
Inference

We can see that Pakistan and Iran show similar trend 2011 onwards, while on the other hand USA and Pakistan show similar trends between the years 2007 and 2011

Countries in Range - (3260, 6520]

In [82]:

```
years = list(map(str,range(1980,2014)))
a2 = l5[4:14].index.tolist()
canada.loc[a2, years].transpose().plot(figsize=(15,12));
plt.title('Immigration of {as2}'.format(as2 = a2));
plt.xlabel('Years');
plt.ylabel('Immigrants');
plt.ylim((0,7000));
```



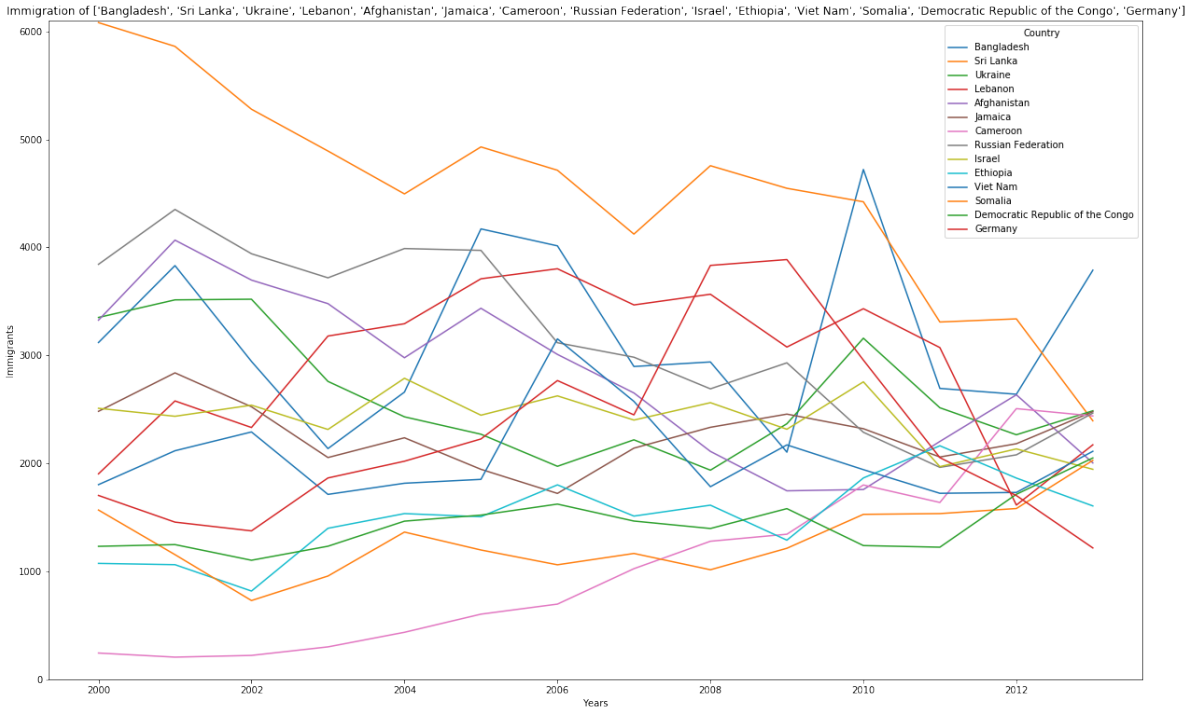
Inference

We can see that all countries except South Korea show almost similar trend throughout from 1980 to 2013.

Countries in Range - (1630, 3260]

In [83]:

```
years1 = list(map(str,range(2000,2014)))
a4 = l5[14:28].index.tolist()
canada.loc[a4, years1].transpose().plot(figsize=(21,13));
plt.title('Immigration of {as4}'.format(as4 = a4));
plt.xlabel('Years');
plt.ylabel('Immigrants');
plt.ylim((0,6100));
```



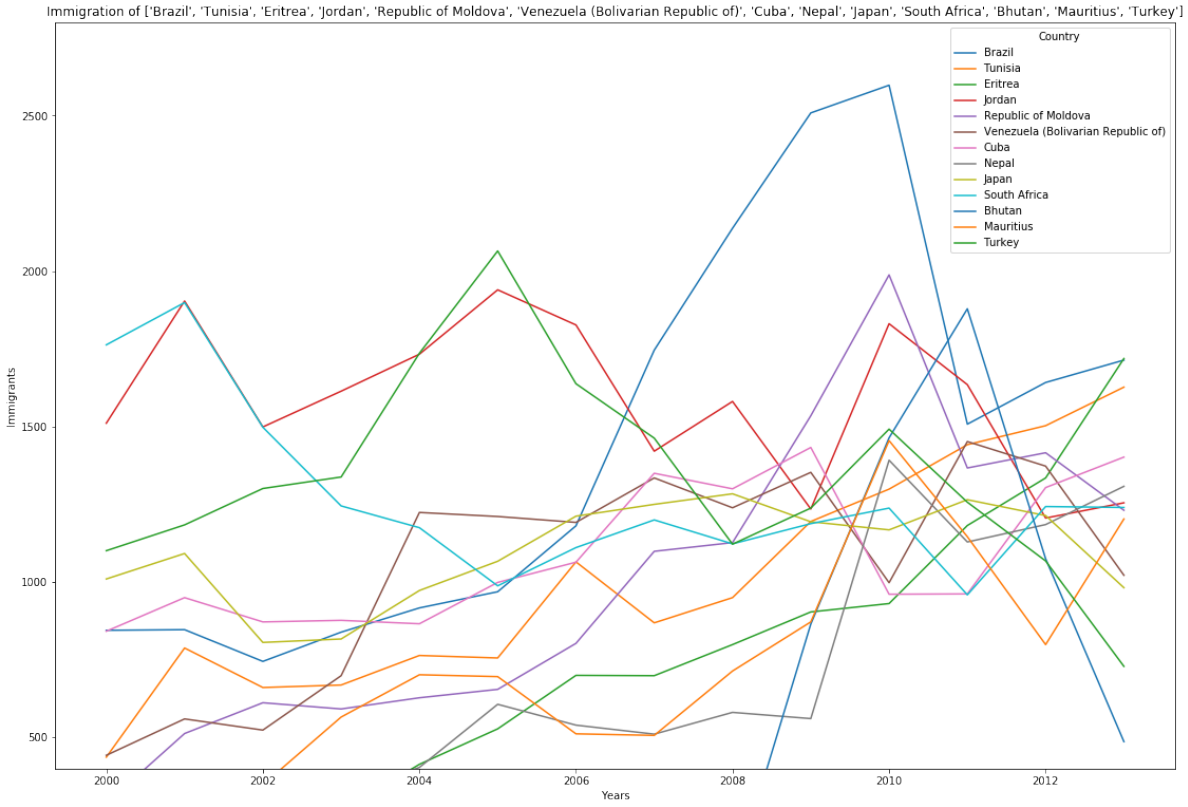
Inference

For ease of comparison, timeline has been reduced to 14 years. Other than Sri Lanka (highest value, 2000) and Cameroon (lowest value, 2000) show similar trends throughout from 2000 to 2013.

Countries in Range - (1000, 1630]

In [84]:

```
a4 = l5[29:42].index.tolist()
canada.loc[a4, years1].transpose().plot(figsize=(19,13));
plt.title('Immigration of {as4}'.format(as4 = a4));
plt.xlabel('Years');
plt.ylabel('Immigrants');
plt.ylim((400,2800));
```



Inference

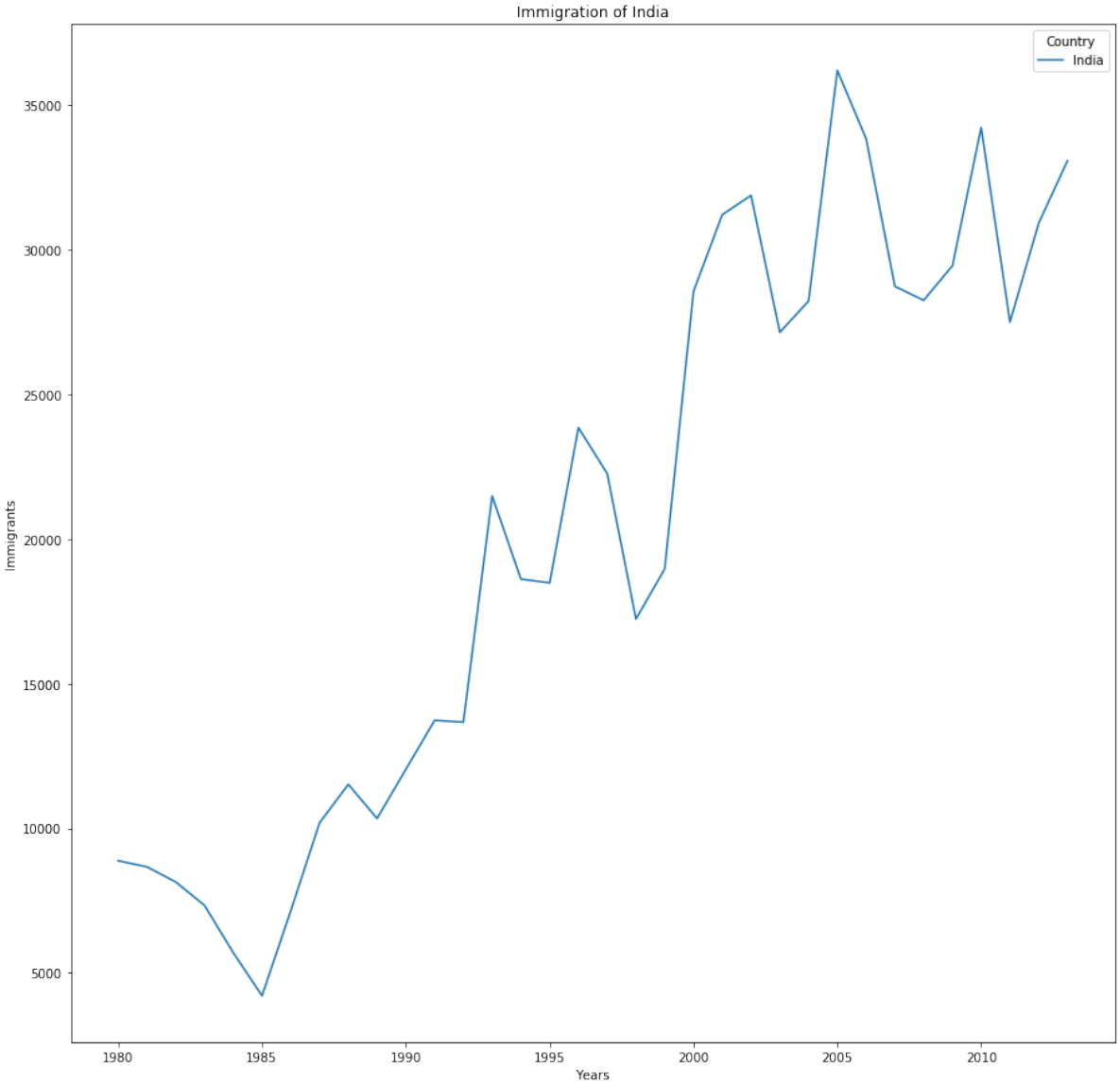
For ease of comparison, timeline has been reduced to 14 years. Except Brazil, which shows a sudden spike after 2006 and Bhutan, which is low throughout and values only spike in year 2011, all other countries are in the 500 - 2000 bracket with inconsistent trends.

Key comparisons and analysis

India

In [85]:

```
canada.loc[['India'], years].transpose().plot(figsize =(15,15));
plt.title('Immigration of India');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Inference

Fist spike is observed during FY 1985-1986. Next spike is observed in 1990-1991. This can be attributed to Liberalization. Following that in 1995-1996 india became the fifth largest economy in world with 3.9% share in world GDP. We can see that 1999-2000 there is a spike. This can be attributed to the Y2k bug solution.

India and China

In [86]:

```
canada.loc[['China','India'], years].transpose().plot(figsize =(15,15));
plt.title('Immigration of China and India');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```

Inference

Both countries show a steady growth rate, which starts to plateau after 2001.

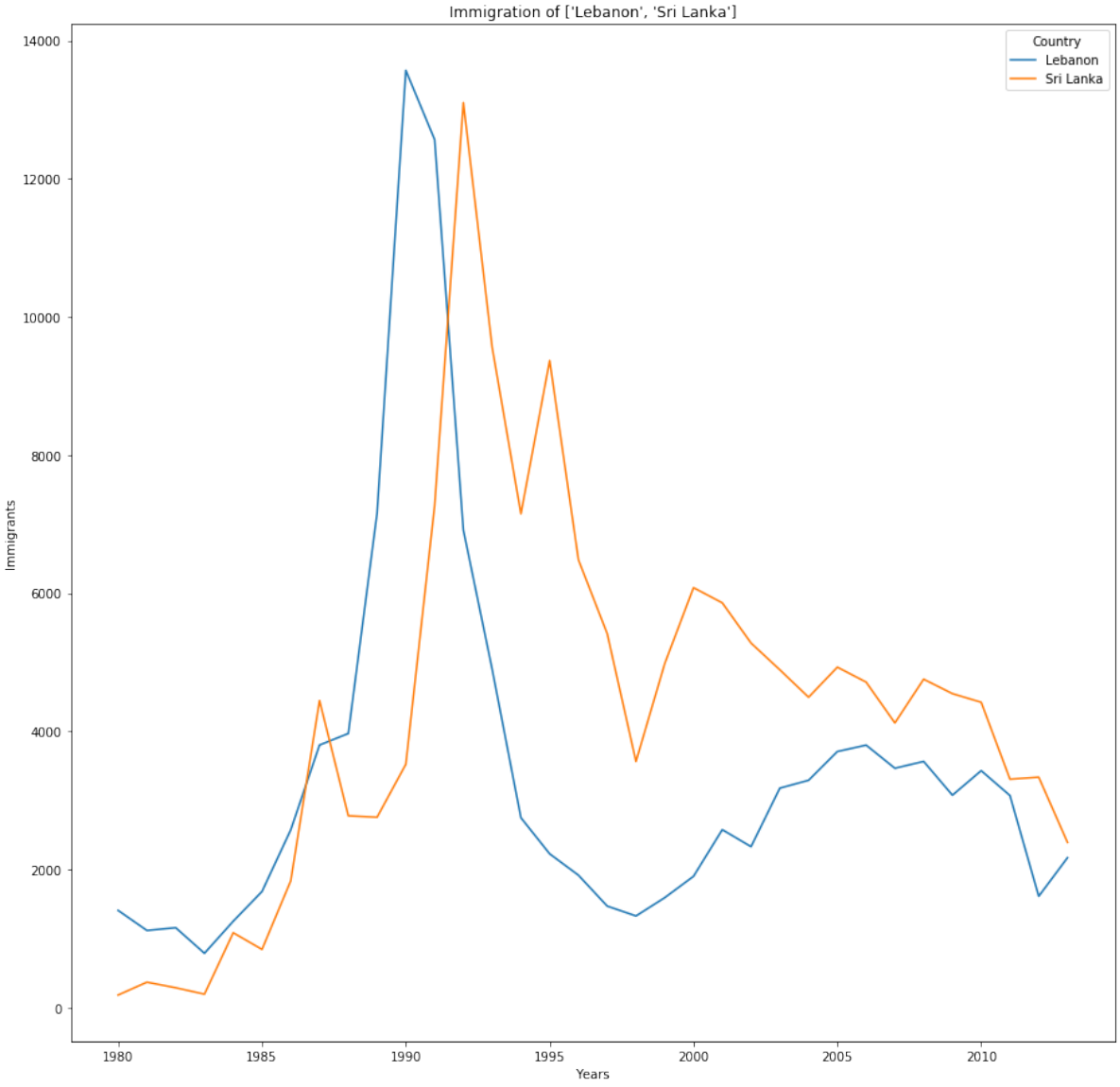
China

China has strengthened its export sector, which has attributed to a decline in the increasing immigration rate.

Lebanon and Sri Lanka

In [87]:

```
a9 = ['Lebanon', 'Sri Lanka']
canada.loc[['Lebanon', 'Sri Lanka'], years].transpose().plot(figsize=(15,15));
plt.title('Immigration of {as9}'.format(as9 = a9));
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Inference

Sri Lanka

Increased privatization, economic reform, and a stress on export-oriented growth helped improve the economic performance, increasing GDP growth to 7% in 1993. This attributes to the drop in immigrants from Sri Lanka.

Lebanon

The Lebanese Civil war which ended in 1990, severely damaged the Lebanese economy. This dropped continously for four years until 1994, when the Real GDP rose to 8%. This was possible because of the government's 20 billion dollar reconstruction program, which managed to create new jobs, making way for immigrants to return back

Pakistan and South Korea (Republic of Korea)

In [88]:

```
a9 = ['Pakistan',' Republic of Korea']
canada.loc[['Pakistan', 'Republic of Korea'], years].transpose().plot(figsize=
(15,15));
plt.title('Immigration of {as9}'.format(as9 = a9));
plt.xlabel('Years');
plt.ylabel('Immigrants');
```


Immigration of ['Pakistan', ' Republic of Korea']
Country
Pakistan
Republic of Korea
Immigrants
Years
0
2000
4000
6000
8000
10000
12000
14000
16000
1980
1985
1990
1995
2000
2005
2010

Inference

Both countries follow a similar trendline. The graph shows steady increase till about the year 1995. Both countries show a drop in the 2000-2002 period.

Pakistan:

In 1994 inflation rate in Pakistan went to a whopping 14.5%. The following year the GDP growth rate dropped from 4.5% to 1.7%, lowest in decades. Pakistan's GDP growth rate dropped from 7.68% to 1.7% in a period of 4 years.

In 1999 Pervez Musharraf lead a millitary coup and overthrew the Nawaz Sharif government. This resulted in Liberalization. During Musharraf's term (1999-2008), around 11,9 million new jobs were created. Therefore this can be attributed to the drop in immigrants. Also, immigration was affected due to various events that happened in USA during 2001.

South Korea:

Korea shows a spike in the year 1997, this indicates the Asian Financial Crisis. The Korean Won began to depreciate heavily. Within months, a third of Korea's merchant banks were closed. Korean economy shrinked at an average of -6.65% per year. Conglomerates became a casuality of this.

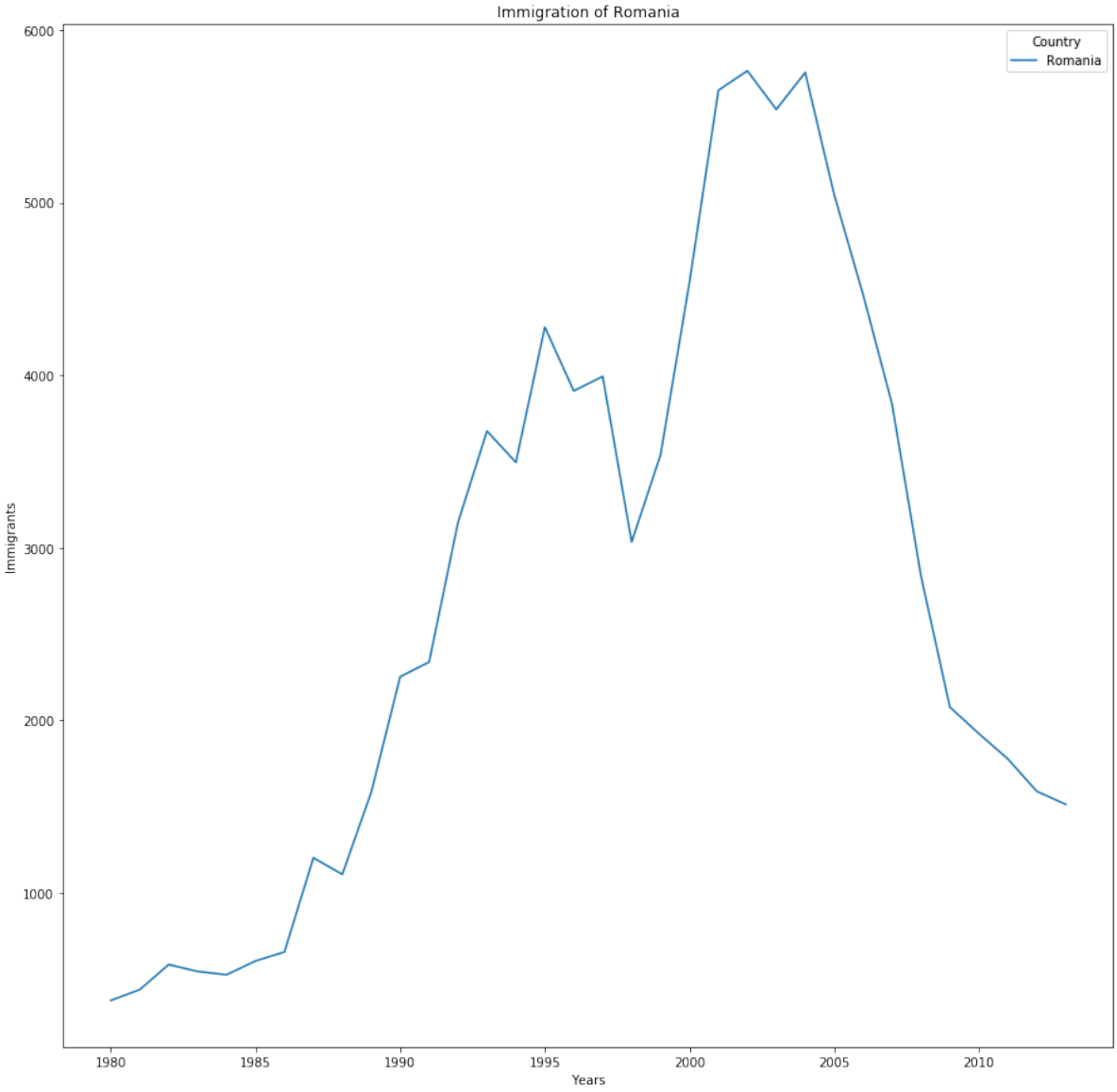
In 2000, the Korean economy shifted to a more market oriented economy. Growth rates touched closer to 11%, highest in decades. This resulted a drop in immigrants.

Individual trend analysis

Romania

In [89]:

```
years = list(map(str,range(1980,2014)))
canada.loc[['Romania'], years].transpose().plot(figsize=(15,15));
plt.title('Immigration of Romania');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Inference

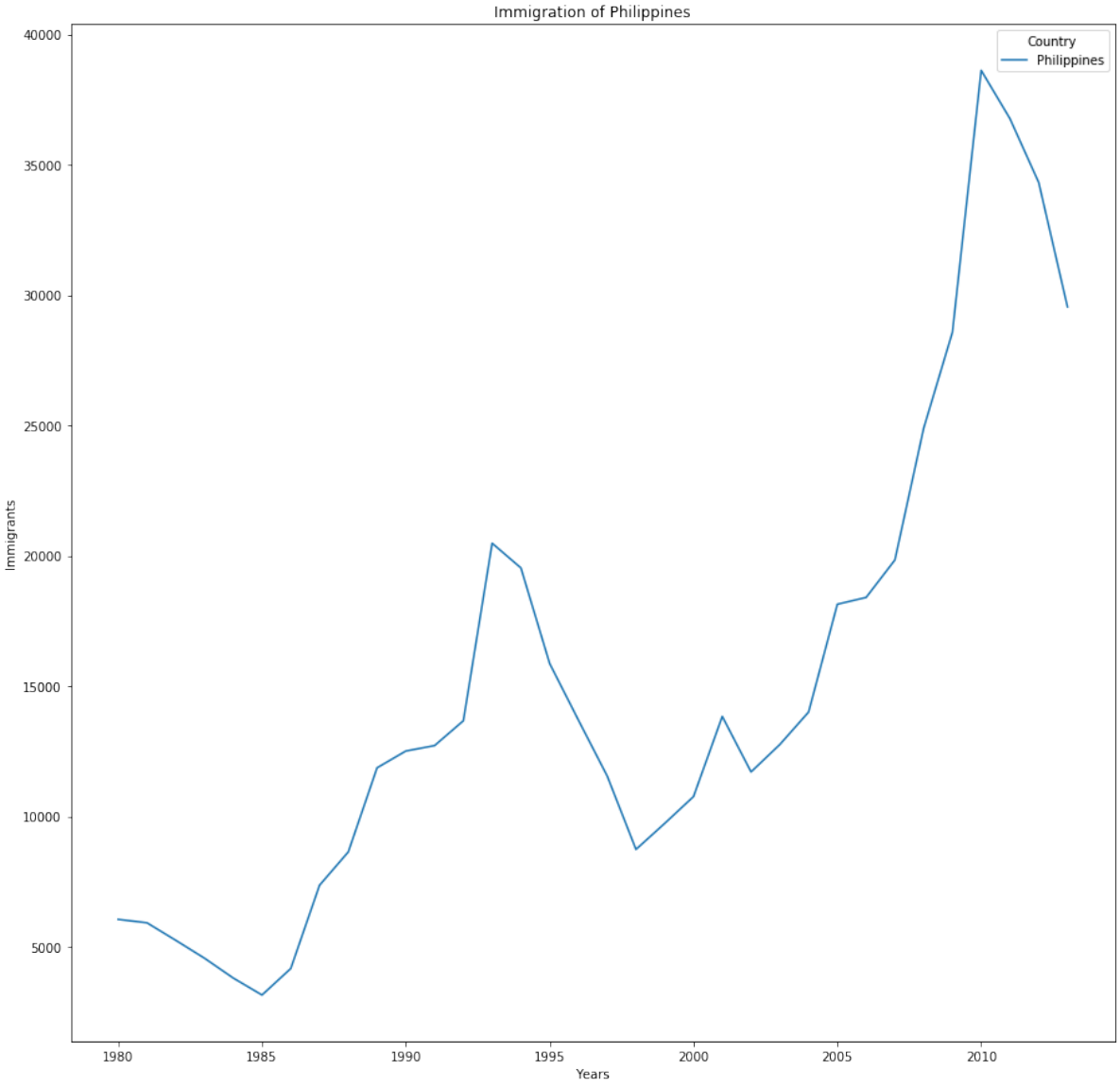
Point of Interest: 2004-2006 period.

This was the period of FDI in Romania. Demand of Romanian products increased in EU. Also, as Romania closed in towards joining EU, immigration to other countries substantially reduced,

Philippines

In [90]:

```
years = list(map(str,range(1980,2014)))
canada.loc[['Philippines'], years].transpose().plot(figsize=(15,15));
plt.title('Immigration of Philippines');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Inference

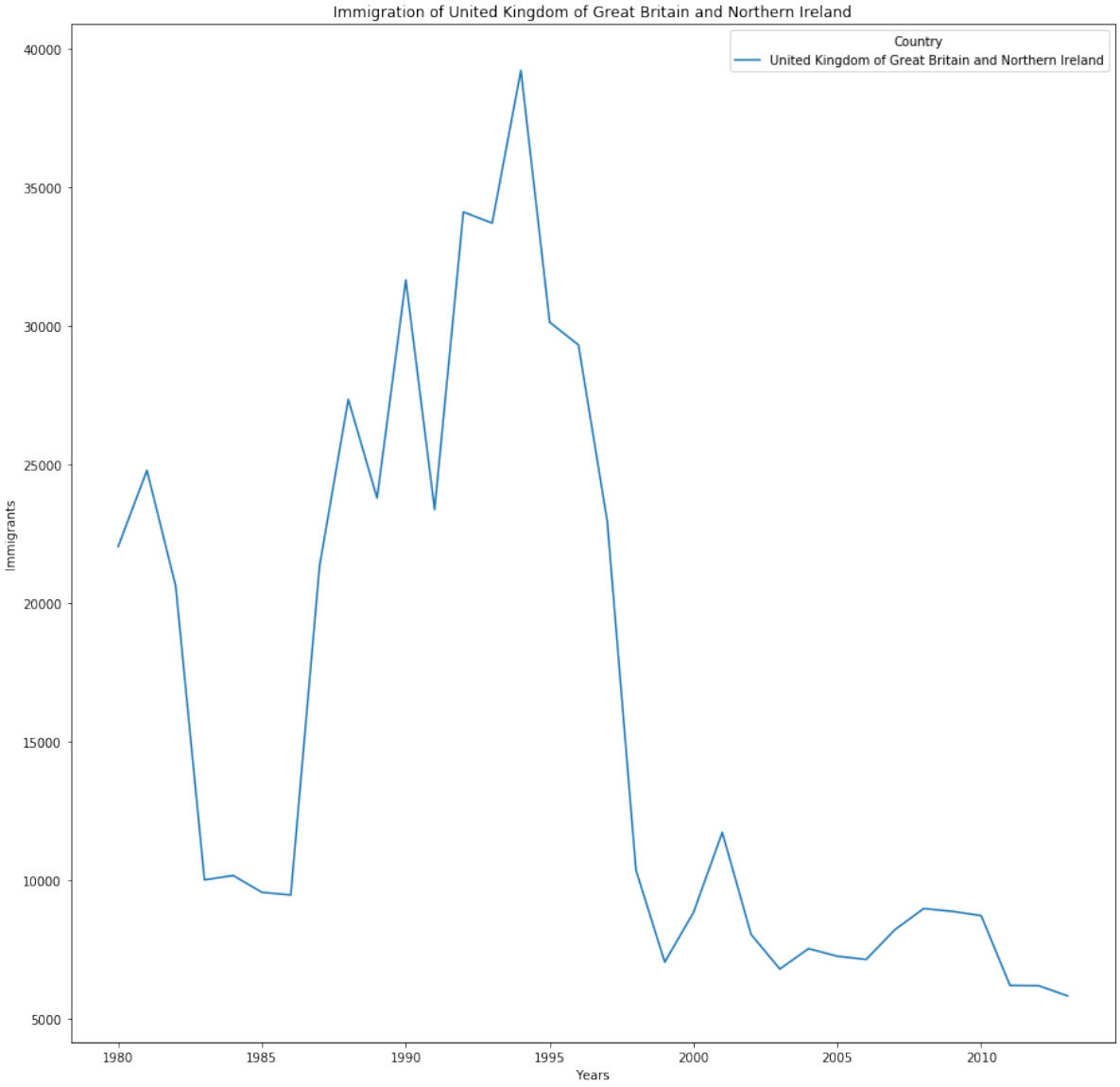
Points of Interests: 1985 and 1998

In the 1980s, annual GDP growth rate dropped to 2%, which was the lowest in decades. This resulted in people immigrating for jobs. Just like South Korea, the Asian Financial Crisis affected Philippines, which explains the spike after 1998,

United Kingdom

In [91]:

```
years = list(map(str,range(1980,2014)))
canada.loc[['United Kingdom of Great Britain and Northern Ireland'], years].tr
anspose().plot(figsize=(15,15));
plt.title('Immigration of United Kingdom of Great Britain and Northern Ireland
');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Inference

Points of interests: 1984-1986 and 1994

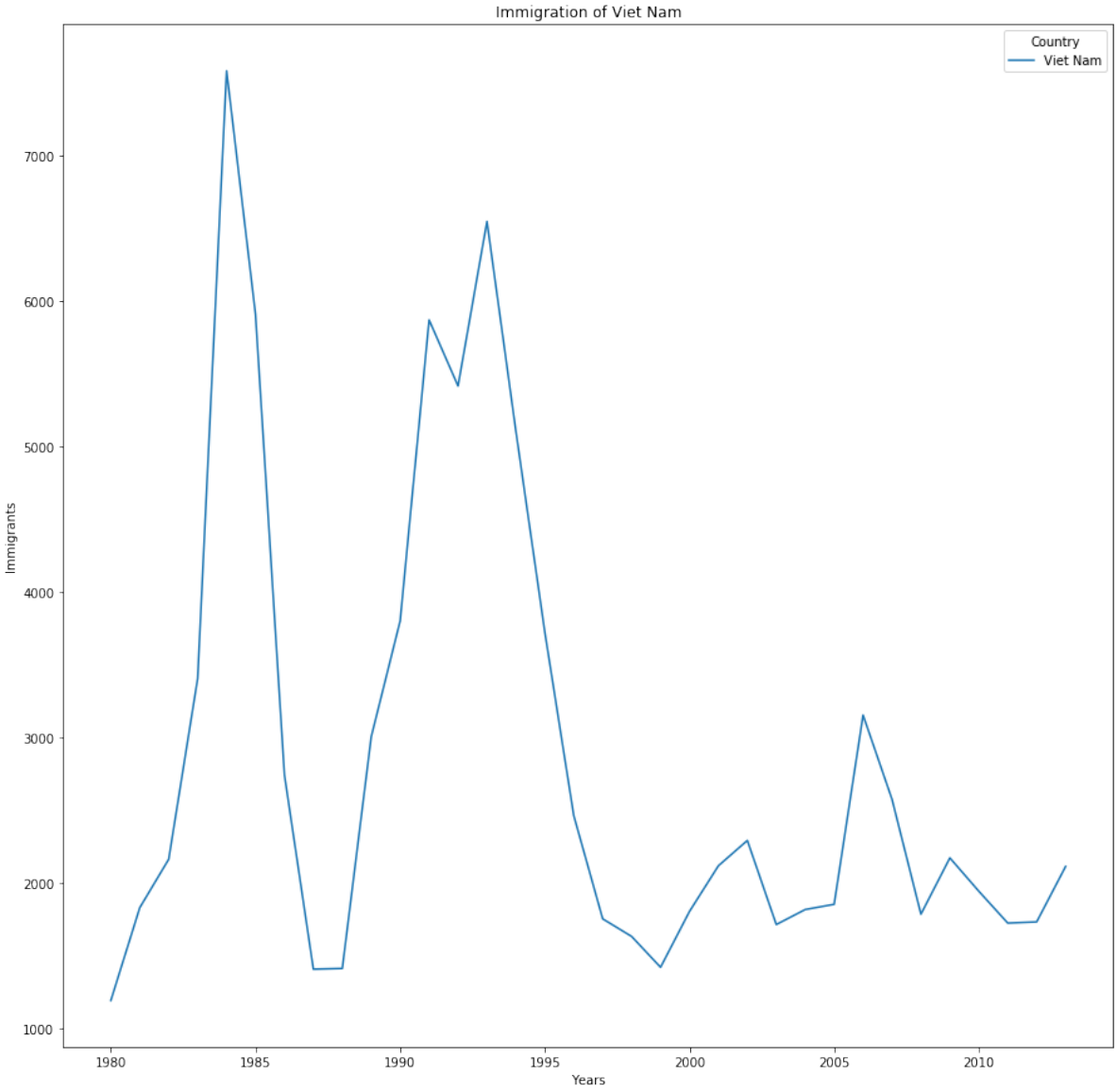
In 1984, an all time high unemployment rate of 11.9% was witnessed in the UK, forcing people to search for jobs in other countries. This attributes to the spike in 1984-1985.

The John Major government in 1990 fucussed to creating new jobs, which substancially rediced the unemployment rate. This resulted in tyhe immigrants returning back to the UK, which can be seen as the graph is seen dipping post 1994.

Viet Nam

In [92]:

```
years = list(map(str,range(1980,2014)))
canada.loc[['Viet Nam'], years].transpose().plot(figsize=(15,15));
plt.title('Immigration of Viet Nam');
plt.xlabel('Years');
plt.ylabel('Immigrants');
```



Inference

Points of interest: 1983-1998 and 1993,

Vietnam entered its Third Five-Year-Plan scheme in 1981. This resulted in a massive economic failure, slowing down the economic growth heavily. The government swiftly corrected its path. Increase in exports is the main reason for immigrants to drop post 1993. This was done to bring back the economy on track.

Assignment 2

Mumbai Crimes (line)

In [93]:

```
indcri = pd.read_csv('https://storage.googleapis.com/kagglesdsdata/datasets%2F1850%2F3879%2Fcrime%2Fcrime%2F01_District_wise_crimes_committed_IPC_2001_2012.csv?GoogleAccessId=gcp-kaggle-com@kaggle-161607.iam.gserviceaccount.com&Expires=1599078097&Signature=JDSu2lsgzbxZ4cxzIe4Ib4Tm6qYbCR4DpiywJPr5tbFQyTGGv0NO19slzn%2BWP0qDORJuKwhVhjvdo3J5%2FcPfjTWnRMKLD%2FHcnlksjbB%2F9m7LE0%2FGE%2FeQt4pBmaBaVVdT6%2F944FheXw0GXFE%2BomBRJOvme9hZOxQ2rmCxj4WsxhvLEgcKvgtUKytHES%2Fl%2FMwbvtF7gYuwzmK%2BmJL9yaO%2Fqyk3vwyUjNvYo7lf%2FMpWM8YAAnyC1osbta3hMPwEsf01K27GgVhCzMOykQAB45vZUvVfRoWU3o1%2BxvHKw8kdKVCL9ZSy7tefz8IB5nPydgJv8%2Bznfqun%2BudngkGe5C%2BPZA%3D%3D')
```

In [94]:

```
df = indcri
indcri.head()
```

Out[94]:

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE
0	ANDHRA PRADESH	ADILABAD	2001	101	60	17	50	0
1	ANDHRA PRADESH	ANANTAPUR	2001	151	125	1	23	0
2	ANDHRA PRADESH	CHITTOOR	2001	101	57	2	27	0
3	ANDHRA PRADESH	CUDDAPAH	2001	80	53	1	20	0
4	ANDHRA PRADESH	EAST GODAVARI	2001	82	67	1	23	0

5 rows × 33 columns

In [95]:

```
mumcri = df.loc[df['DISTRICT'] == 'MUMBAI']
mumcri.head(10)
```

Out[95]:

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTC
370	MAHARASHTRA	MUMBAI	2001	295	200	6	127	
1087	MAHARASHTRA	MUMBAI	2002	252	134	7	128	
1816	MAHARASHTRA	MUMBAI	2003	242	115	8	133	
2544	MAHARASHTRA	MUMBAI	2004	253	127	4	187	
3278	MAHARASHTRA	MUMBAI	2005	212	136	8	201	

5 rows × 33 columns

In [96]:

```
mumcri.columns
```

Out[96]:

```
Index(['STATE/UT', 'DISTRICT', 'YEAR', 'MURDER', 'ATTEMPT TO MURDE
R',
       'CULPABLE HOMICIDE NOT AMOUNTING TO MURDER', 'RAPE', 'CUSTO
DIAL RAPE',
       'OTHER RAPE', 'KIDNAPPING & ABDUCTION',
       'KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS',
       'KIDNAPPING AND ABDUCTION OF OTHERS', 'DACOITY',
       'PREPARATION AND ASSEMBLY FOR DACOITY', 'ROBBERY', 'BURGLAR
Y', 'THEFT',
       'AUTO THEFT', 'OTHER THEFT', 'RIOTS', 'CRIMINAL BREACH OF T
RUST',
       'CHEATING', 'COUNTERFIETING', 'ARSON', 'HURT/GREVIOUS HURT'
,
       'DOWRY DEATHS', 'ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HE
R MODESTY',
       'INSULT TO MODESTY OF WOMEN', 'CRUELTY BY HUSBAND OR HIS RE
LATIVES',
       'IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES',
       'CAUSING DEATH BY NEGLIGENCE', 'OTHER IPC CRIMES', 'TOTAL I
PC CRIMES'],
      dtype='object')
```

In [97]:

```
plt.plot('YEAR', 'TOTAL IPC CRIMES', data=mumcri);
```

PyWaffle

In [98]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from pywaffle import Waffle
%matplotlib inline
```

In [99]:

```
scan = canada.loc[['Sweden', 'Denmark', 'Norway'], :]
scan
```

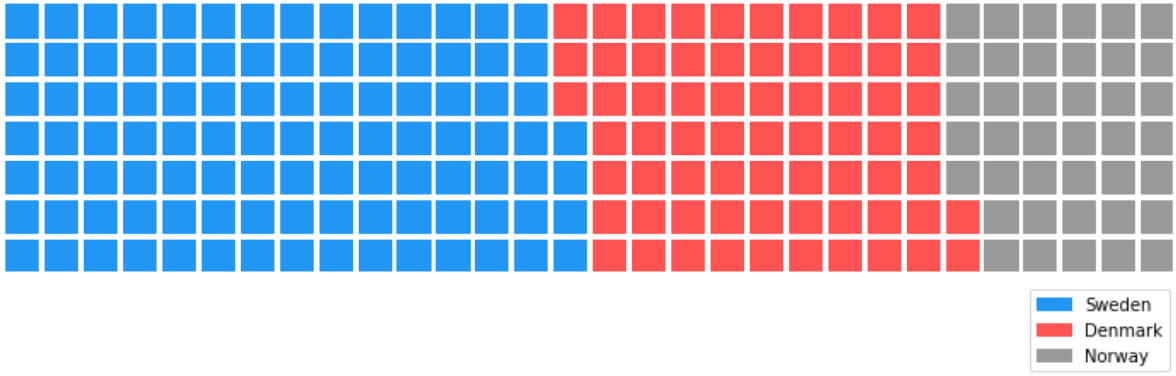
Out[99]:

Country	Continent	Continent-Region	DevName	1980	1981	1982	1983	1984	1985	1986	...
Sweden	Europe	Northern Europe	Developed regions	281	308	222	176	128	158	187	...
Denmark	Europe	Northern Europe	Developed regions	272	293	299	106	93	73	93	...
Norway	Europe	Northern Europe	Developed regions	116	77	106	51	31	54	56	...

3 rows × 40 columns

In [100]:

```
cat = list(scan.index.values)
vals = scan['Total']
fig = plt.figure(FigureClass = Waffle, rows = 7, columns = 30, values = vals,
labels = list(cat),
                 colors=["#2196f3", "#ff5252", "#999999"], figsize=(10,5),
                legend = {'loc': 'lower left', 'bbox_to_anchor': (0.87, -0.4)}
)
```



In [101]:

```
indpakchi = canada.loc[['India', 'Pakistan', 'China'], :]
cat = list(indpakchi.index.values)
vals = indpakchi['Total']
fig = plt.figure(FigureClass = Waffle, rows = 7, columns = 30, values = vals,
labels = list(cat),
                 colors=["#2196f3", "#ff5252", "#999999"], figsize=(10,5),
                legend = {'loc': 'lower left', 'bbox_to_anchor': (0.878, -0.4)
})
```