In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
can=pd.read_excel('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsN
    sheet_name='Canada by Citizenship',
    skiprows=range(20),
    skipfooter=2)
```

In [3]:
```
can.head(2)
```

Out[3]:

| | Type | Coverage | OdName | AREA | AreaName | REG | RegName | DEV | DevName | 1980 | ... | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Immigrants | Foreigners | Afghanistan | 935 | Asia | 5501 | Southern Asia | 902 | Developing regions | 16 | ... | 2978 | 3436 | 3009 | 2652 | 2111 | 1746 | 1758 | 2203 | 2635 | 2( |
| **1** | Immigrants | Foreigners | Albania | 908 | Europe | 925 | Southern Europe | 901 | Developed regions | 1 | ... | 1450 | 1223 | 856 | 702 | 560 | 716 | 561 | 539 | 620 | ( |

2 rows × 43 columns

In [4]:
```
can.drop(['Type','Coverage','AREA','REG','DEV'],axis=1,inplace=True)
can.rename(columns={'OdName':'Country','AreaName':'Continent','RegName':'Region'}, inplace=True)
can['Total']=can.sum(axis=1)
```

```
C:\Users\ahmed\AppData\Local\Temp\ipykernel_8864\4146139127.py:3: FutureWarning: Dropping of nuisance columns in DataFrame redu
ctions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns befo
re calling the reduction.
  can['Total']=can.sum(axis=1)
```

In [6]:
```
can.set_index('Country', inplace=True)
```

In [7]:
```
can.columns=list(map(str, can.columns))
```

In [9]:
```
years = list(map(str, range(1980,2014)))
years
```

Out[9]:
```
['1980',
 '1981',
 '1982',
 '1983',
 '1984',
 '1985',
 '1986',
 '1987',
 '1988',
 '1989',
 '1990',
 '1991',
 '1992',
 '1993',
 '1994',
 '1995',
 '1996',
 '1997',
 '1998',
 '1999',
 '2000',
 '2001',
 '2002',
 '2003',
 '2004',
 '2005',
 '2006',
 '2007',
 '2008',
 '2009',
 '2010',
 '2011',
 '2012',
 '2013']
```

In [10]:

```
can.sort_values(['Total'], ascending=False, axis=0,inplace=True)
Top5=can.head()
Top5=Top5[years].transpose()
Top5.head()
```

Out[10]:

| Country | India | China | United Kingdom of Great Britain and Northern Ireland | Philippines | Pakistan |
|---|---|---|---|---|---|
| **1980** | 8880 | 5123 | 22045 | 6051 | 978 |
| **1981** | 8670 | 6682 | 24796 | 5921 | 972 |
| **1982** | 8147 | 3308 | 20620 | 5249 | 1201 |
| **1983** | 7338 | 1863 | 10015 | 4562 | 900 |
| **1984** | 5704 | 1527 | 10170 | 3801 | 668 |

In [12]:

```
mpl.style.use(['ggplot'])
```

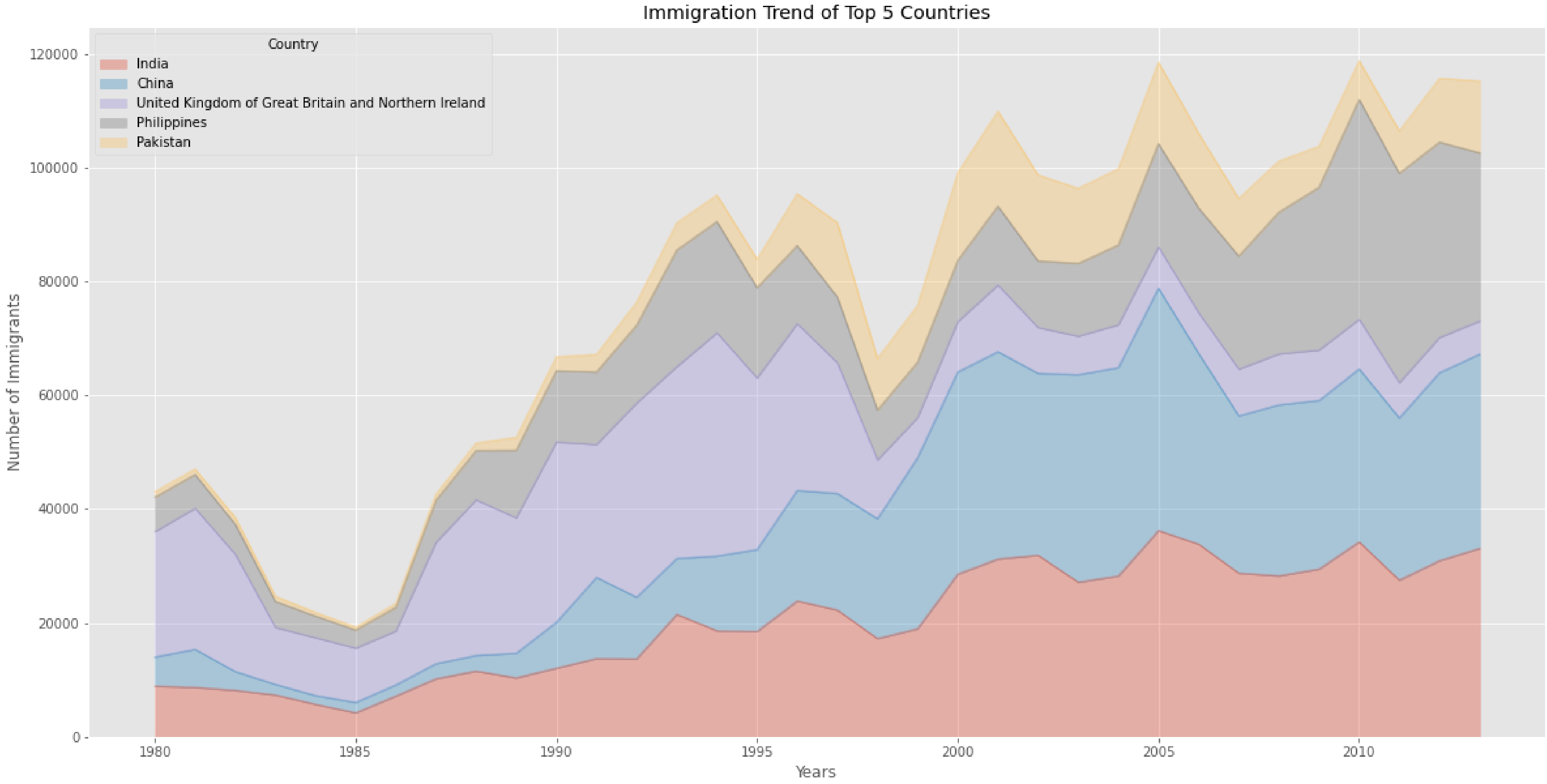
In [14]:

```
ax= Top5.plot(kind='area', alpha=0.35, figsize=(20,10))
```

```
ax.set_title('Immigration Trend of Top 5 Countries')
ax.set_ylabel('Number of Immigrants')
ax.set_xlabel('Years')
```

Out[14]:

```
Text(0.5, 0, 'Years')
```



In [19]:

```
df_continents = can.groupby('Continent', axis=0).sum()
df_continents.head()
```

Out[19]:

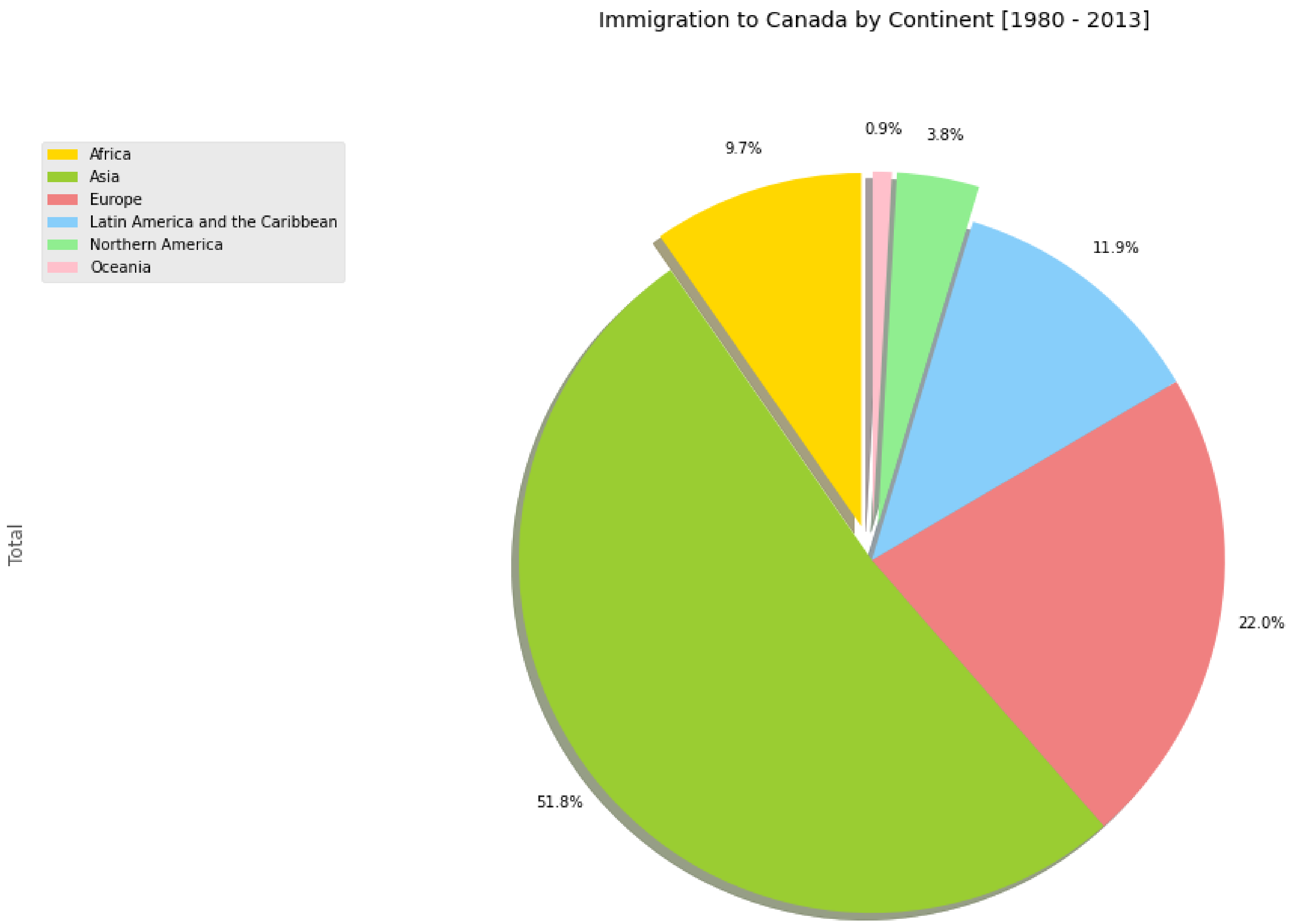
| | 1980 | 1981 | 1982 | 1983 | 1984 | 1985 | 1986 | 1987 | 1988 | 1989 | ... | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Continent** | | | | | | | | | | | | | | | | | | | | |
| **Africa** | 3951 | 4363 | 3819 | 2671 | 2639 | 2650 | 3782 | 7494 | 7552 | 9894 | ... | 27523 | 29188 | 28284 | 29890 | 34534 | 40892 | 35441 | 38083 | 38 |
| **Asia** | 31025 | 34314 | 30214 | 24696 | 27274 | 23850 | 28739 | 43203 | 47454 | 60256 | ... | 159253 | 149054 | 133459 | 139894 | 141434 | 163845 | 146894 | 152218 | 155 |
| **Europe** | 39760 | 44802 | 42720 | 24638 | 22287 | 20844 | 24370 | 46698 | 54726 | 60893 | ... | 35955 | 33053 | 33495 | 34692 | 35078 | 33425 | 26778 | 29177 | 28 |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Latin America and the Caribbean** | 13081 | 15215 | 16769 | 15427 | 13678 | 15171 | 21179 | 28471 | 21924 | 25060 | ... | 24747 | 24676 | 26011 | 26547 | 26867 | 28818 | 27856 | 27173 | 24 |
| **Northern America** | 9378 | 10030 | 9074 | 7100 | 6661 | 6543 | 7074 | 7705 | 6469 | 6790 | ... | 8394 | 9613 | 9463 | 10190 | 8995 | 8142 | 7677 | 7892 | 8 |

5 rows × 35 columns

In [23]:

```python
color_list = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'lightgreen', 'pink']
explode_list= [0.1,0,0,0,0.1,0.1]

df_continents['Total'].plot(kind='pie',
                  figsize=(20,10),
                  autopct='%1.1f%%',
                  startangle=90,
                  pctdistance=1.12,
                  labels=None,
                  shadow=True,
                  colors=color_list,
                  explode=explode_list)
plt.axis('equal')
plt.title('Immigration to Canada by Continent [1980 - 2013]', y=1.12)
plt.legend(labels=df_continents.index, loc='upper left')
plt.show()
```



In [24]:

```
df_top15=can.sort_values(['Total'],ascending=False,axis=0).head(15)
df_top15
```

Out[24]:

| | Continent | Region | DevName | 1980 | 1981 | 1982 | 1983 | 1984 | 1985 | 1986 | ... | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Country** | | | | | | | | | | | | | | | | | | | |
| **India** | Asia | Southern Asia | Developing regions | 8880 | 8670 | 8147 | 7338 | 5704 | 4211 | 7150 | ... | 36210 | 33848 | 28742 | 28261 | 29456 | 34235 | 27509 | 30933 |
| **China** | Asia | Eastern Asia | Developing regions | 5123 | 6682 | 3308 | 1863 | 1527 | 1816 | 1960 | ... | 42584 | 33518 | 27642 | 30037 | 29622 | 30391 | 28502 | 33024 |
| **United Kingdom** | Europe | Northern Europe | Developed regions | 22045 | 24796 | 20620 | 10015 | 10170 | 9564 | 9470 | ... | 7258 | 7140 | 8216 | 8979 | 8876 | 8724 | 6204 | 6195 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **of Great Britain and Northern Ireland** | | | | | | | | | | | | | | | | | | |
| **Philippines** | Asia | South-Eastern Asia | Developing regions | 6051 | 5921 | 5249 | 4562 | 3801 | 3150 | 4166 | ... | 18139 | 18400 | 19837 | 24887 | 28573 | 38617 | 36765 | 34315 |
| **Pakistan** | Asia | Southern Asia | Developing regions | 978 | 972 | 1201 | 900 | 668 | 514 | 691 | ... | 14314 | 13127 | 10124 | 8994 | 7217 | 6811 | 7468 | 11227 |
| **United States of America** | Northern America | Northern America | Developed regions | 9378 | 10030 | 9074 | 7100 | 6661 | 6543 | 7074 | ... | 8394 | 9613 | 9463 | 10190 | 8995 | 8142 | 7676 | 7891 |
| **Iran (Islamic Republic of)** | Asia | Southern Asia | Developing regions | 1172 | 1429 | 1822 | 1592 | 1977 | 1648 | 1794 | ... | 5837 | 7480 | 6974 | 6475 | 6580 | 7477 | 7479 | 7534 |
| **Sri Lanka** | Asia | Southern Asia | Developing regions | 185 | 371 | 290 | 197 | 1086 | 845 | 1838 | ... | 4930 | 4714 | 4123 | 4756 | 4547 | 4422 | 3309 | 3338 |
| **Republic of Korea** | Asia | Eastern Asia | Developing regions | 1011 | 1456 | 1572 | 1081 | 847 | 962 | 1208 | ... | 5832 | 6215 | 5920 | 7294 | 5874 | 5537 | 4588 | 5316 |
| **Poland** | Europe | Eastern Europe | Developed regions | 863 | 2930 | 5881 | 4546 | 3588 | 2819 | 4808 | ... | 1405 | 1263 | 1235 | 1267 | 1013 | 795 | 720 | 779 |
| **Lebanon** | Asia | Western Asia | Developing regions | 1409 | 1119 | 1159 | 789 | 1253 | 1683 | 2576 | ... | 3709 | 3802 | 3467 | 3566 | 3077 | 3432 | 3072 | 1614 |
| **France** | Europe | Western Europe | Developed regions | 1729 | 2027 | 2219 | 1490 | 1169 | 1177 | 1298 | ... | 4429 | 4002 | 4290 | 4532 | 5051 | 4646 | 4080 | 6280 |
| **Jamaica** | Latin America and the Caribbean | Caribbean | Developing regions | 3198 | 2634 | 2661 | 2455 | 2508 | 2938 | 4649 | ... | 1945 | 1722 | 2141 | 2334 | 2456 | 2321 | 2059 | 2182 |
| **Viet Nam** | Asia | South-Eastern Asia | Developing regions | 1191 | 1829 | 2162 | 3404 | 7583 | 5907 | 2741 | ... | 1852 | 3153 | 2574 | 1784 | 2171 | 1942 | 1723 | 1731 |
| **Romania** | Europe | Eastern Europe | Developed regions | 375 | 438 | 583 | 543 | 524 | 604 | 656 | ... | 5048 | 4468 | 3834 | 2837 | 2076 | 1922 | 1776 | 1588 |

15 rows × 38 columns

In [25]:

```
df_tot = pd.DataFrame(can[years].sum(axis=0))
df_tot.index = map(int, df_tot.index)
df_tot.reset_index(inplace = True)

df_tot.columns = ['year', 'total']
df_tot.head()
```

Out[25]:

|   | year | total |
|---|---|---|
| **0** | 1980 | 99137 |
| **1** | 1981 | 110563 |
| **2** | 1982 | 104271 |
| **3** | 1983 | 75550 |
| **4** | 1984 | 73417 |

In [26]:

```
x = df_tot['year']
y = df_tot['total']
fit= np.polyfit(x,y,deg=1)
fit
```

Out[26]:

```
array([ 5.56709228e+03, -1.09261952e+07])
```

In [29]:

```
'No. Immigrants = {0:.0f} * Year + {1:.0f}'.format(fit[0], fit[1])
```

Out[29]:

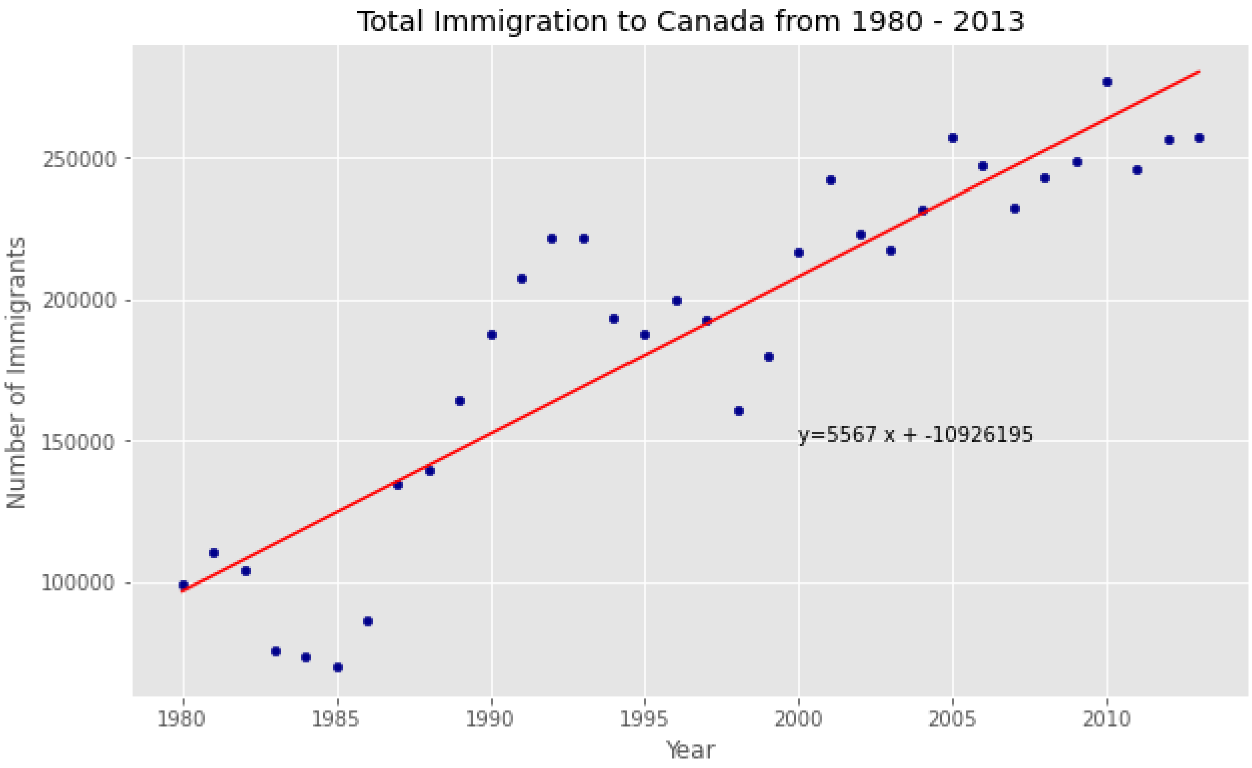
```
'No. Immigrants = 5567 * Year + -10926195'
```

In [30]:

```
df_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6), color='darkblue')

plt.title('Total Immigration to Canada from 1980 - 2013')
plt.xlabel('Year')
plt.ylabel('Number of Immigrants')

plt.plot(x, fit[0] * x + fit[1], color='red')
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit[0],fit[1]),xy=(2000,150000))
```

Out[30]:

```
Text(2000, 150000, 'y=5567 x + -10926195')
```



In [33]:
```
5567*2015-10926195
```

Out[33]:
```
291310
```

In [35]:
```
df_can_t = can[years].transpose()

# cast the Years (the index) to type int
df_can_t.index = map(int, df_can_t.index)

# let's label the index. This will automatically be the column name when we reset the index
df_can_t.index.name = 'Year'

# reset index to bring the Year in as a column
df_can_t.reset_index(inplace=True)

# view the changes
df_can_t.head()
```

Out[35]:

| Country | Year | India | China | United Kingdom of Great Britain and | Philippines | Pakistan | United States of America | Iran (Islamic Republic of) | Sri Lanka | Republic of Korea | ... | Kiribati | Vanuatu | Sao Tome and Principe | Tuvalu | American Samoa | San Marino | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

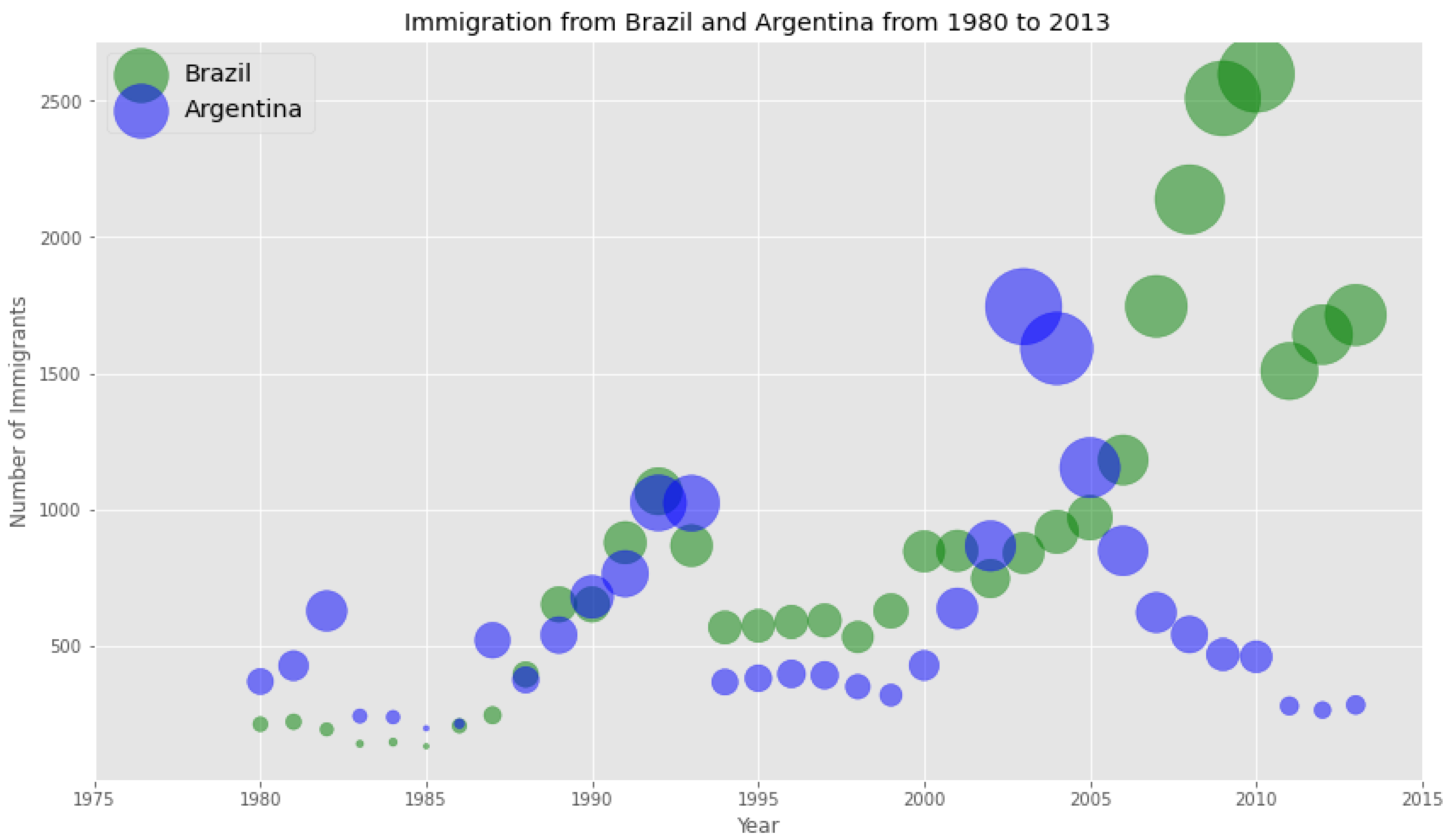
|  |  |  |  | Northern Ireland |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1980 | 8880 | 5123 | 22045 | 6051 | 978 | 9378 | 1172 | 185 | 1011 | ... | 0 | 0 | 0 | 0 | 0 | 1 |
| **1** | 1981 | 8670 | 6682 | 24796 | 5921 | 972 | 10030 | 1429 | 371 | 1456 | ... | 0 | 0 | 0 | 1 | 1 | 0 |
| **2** | 1982 | 8147 | 3308 | 20620 | 5249 | 1201 | 9074 | 1822 | 290 | 1572 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 1983 | 7338 | 1863 | 10015 | 4562 | 900 | 7100 | 1592 | 197 | 1081 | ... | 1 | 0 | 0 | 0 | 0 | 0 |
| **4** | 1984 | 5704 | 1527 | 10170 | 3801 | 668 | 6661 | 1977 | 1086 | 847 | ... | 0 | 0 | 0 | 1 | 0 | 0 |

5 rows × 196 columns

In [36]:
```
norm_brazil = (df_can_t['Brazil'] - df_can_t['Brazil'].min()) / (df_can_t['Brazil'].max() - df_can_t['Brazil'].min())

# normalize Argentina data
norm_argentina = (df_can_t['Argentina'] - df_can_t['Argentina'].min()) / (df_can_t['Argentina'].max() - df_can_t['Argentina'].m
```

In [37]:
```
# Brazil
ax0 = df_can_t.plot(kind='scatter',
                    x='Year',
                    y='Brazil',
                    figsize=(14, 8),
                    alpha=0.5,  # transparency
                    color='green',
                    s=norm_brazil * 2000 + 10,  # pass in weights
                    xlim=(1975, 2015)
                    )

# Argentina
ax1 = df_can_t.plot(kind='scatter',
                    x='Year',
                    y='Argentina',
                    alpha=0.5,
                    color="blue",
                    s=norm_argentina * 2000 + 10,
                    ax=ax0
                    )

ax0.set_ylabel('Number of Immigrants')
ax0.set_title('Immigration from Brazil and Argentina from 1980 to 2013')
ax0.legend(['Brazil', 'Argentina'], loc='upper left', fontsize='x-large')
```

Out[37]:
```
<matplotlib.legend.Legend at 0x206c0babf10>
```



In [38]:

```
from PIL import Image
```

In [40]:

```
df_dsn = can.loc[['Denmark', 'Norway', 'Sweden'], :]

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

Out[40]:

| | Continent | Region | DevName | 1980 | 1981 | 1982 | 1983 | 1984 | 1985 | 1986 | ... | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Country** | | | | | | | | | | | | | | | | | | | | | |
| **Denmark** | Europe | Northern Europe | Developed regions | 272 | 293 | 299 | 106 | 93 | 73 | 93 | ... | 62 | 101 | 97 | 108 | 81 | 92 | 93 | 94 | 81 | 3901 |
| **Norway** | Europe | Northern Europe | Developed regions | 116 | 77 | 106 | 51 | 31 | 54 | 56 | ... | 57 | 53 | 73 | 66 | 75 | 46 | 49 | 53 | 59 | 2327 |
| **Sweden** | Europe | Northern | Developed | 281 | 308 | 222 | 176 | 128 | 158 | 187 | ... | 205 | 139 | 193 | 165 | 167 | 159 | 134 | 140 | 140 | 5866 |

| | | |
|---|---|---|
| | Europe | regions |

3 rows × 38 columns

```
In [41]:
total_values = df_dsn['Total'].sum()
category_proportions = df_dsn['Total'] / total_values

# print out proportions
pd.DataFrame({"Category Proportion": category_proportions})
```

Out[41]:

| | Category Proportion |
|---|---|
| **Country** | |
| **Denmark** | 0.322557 |
| **Norway** | 0.192409 |
| **Sweden** | 0.485034 |

```
In [42]:
width = 40 # width of chart
height = 10 # height of chart

total_num_tiles = width * height # total number of tiles

print(f'Total number of tiles is {total_num_tiles}.')
```

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

```
In [43]:
tiles_per_category = (category_proportions * total_num_tiles).round().astype(int)

# print out number of tiles per category
pd.DataFrame({"Number of tiles": tiles_per_category})
```

Out[43]:

| | Number of tiles |
|---|---|
| **Country** | |
| **Denmark** | 129 |
| **Norway** | 77 |
| **Sweden** | 194 |

```
In [44]:
# initialize the waffle chart as an empty matrix
waffle_chart = np.zeros((height, width), dtype = np.uint)

# 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

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

```
Waffle chart populated!
```

In [45]:

```
waffle_chart
```

Out[45]:

```
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, 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]],
      dtype=uint32)
```

In [49]:

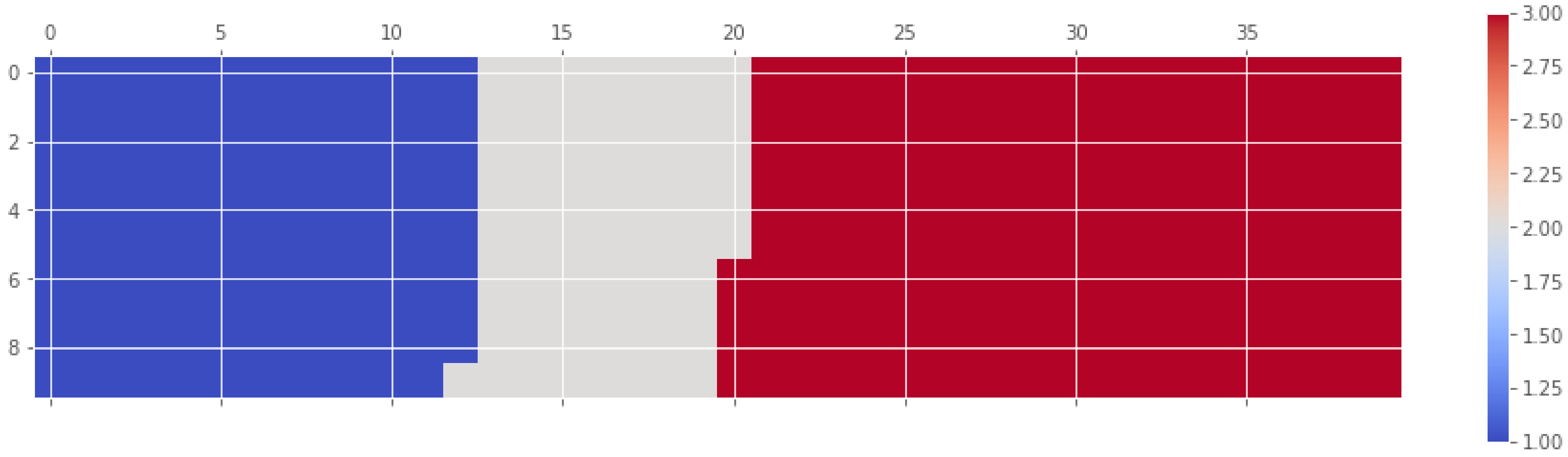
```
import matplotlib.patches as mpatches
```

In [46]:

```
# 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()
plt.show()
```

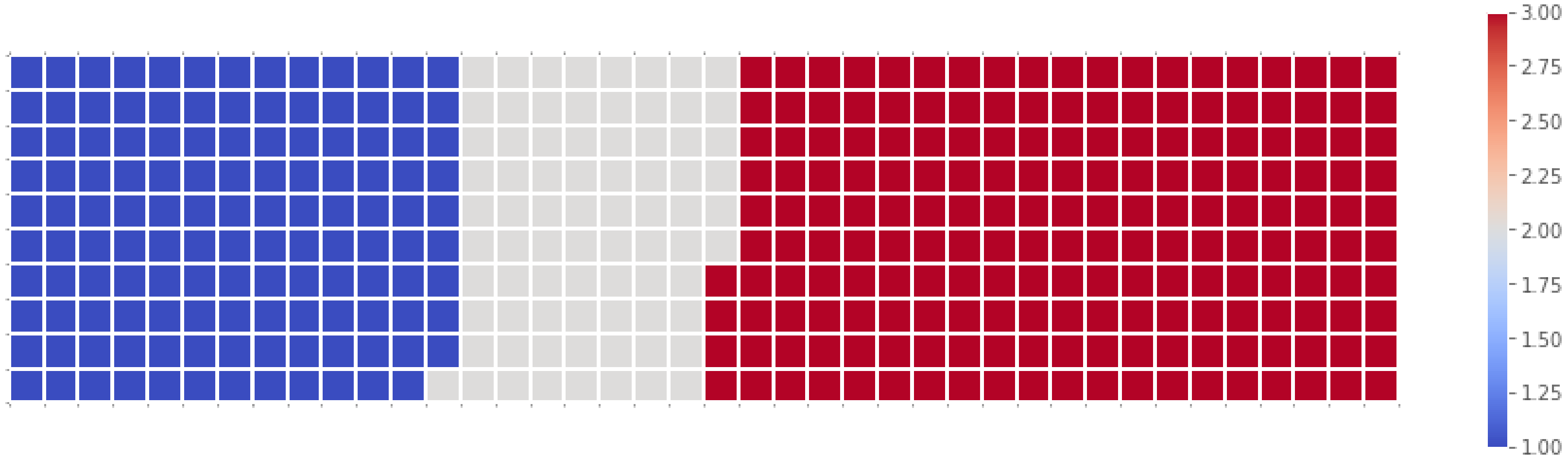
```
C:\Users\ahmed\AppData\Local\Temp\ipykernel_8864\103890981.py:7: MatplotlibDeprecationWarning: Auto-removal of grids by pcolor
() and pcolormesh() is deprecated since 3.5 and will be removed two minor releases later; please call grid(False) first.
  plt.colorbar()
<Figure size 432x288 with 0 Axes>
```



In [50]:

```
# 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([])
plt.show()
```

```
C:\Users\ahmed\AppData\Local\Temp\ipykernel_8864\1261046109.py:7: MatplotlibDeprecationWarning: Auto-removal of grids by pcolor
() and pcolormesh() is deprecated since 3.5 and will be removed two minor releases later; please call grid(False) first.
  plt.colorbar()
<Figure size 432x288 with 0 Axes>
```

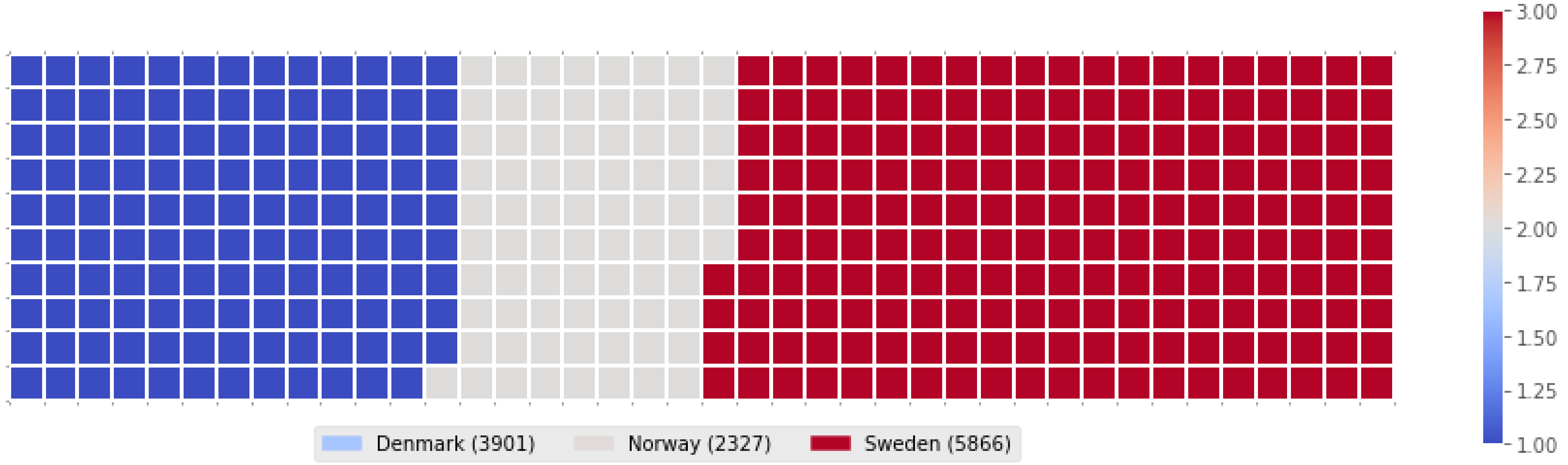


In [51]:

```
# 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 between 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)
          )
plt.show()
```

```
C:\Users\ahmed\AppData\Local\Temp\ipykernel_8864\2463873726.py:7: MatplotlibDeprecationWarning: Auto-removal of grids by pcolor
() and pcolormesh() is deprecated since 3.5 and will be removed two minor releases later; please call grid(False) first.
  plt.colorbar()
<Figure size 432x288 with 0 Axes>
```



In [53]:

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

    # 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 in 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 between 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)
    )
    plt.show()
```

In [54]:

```
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
```

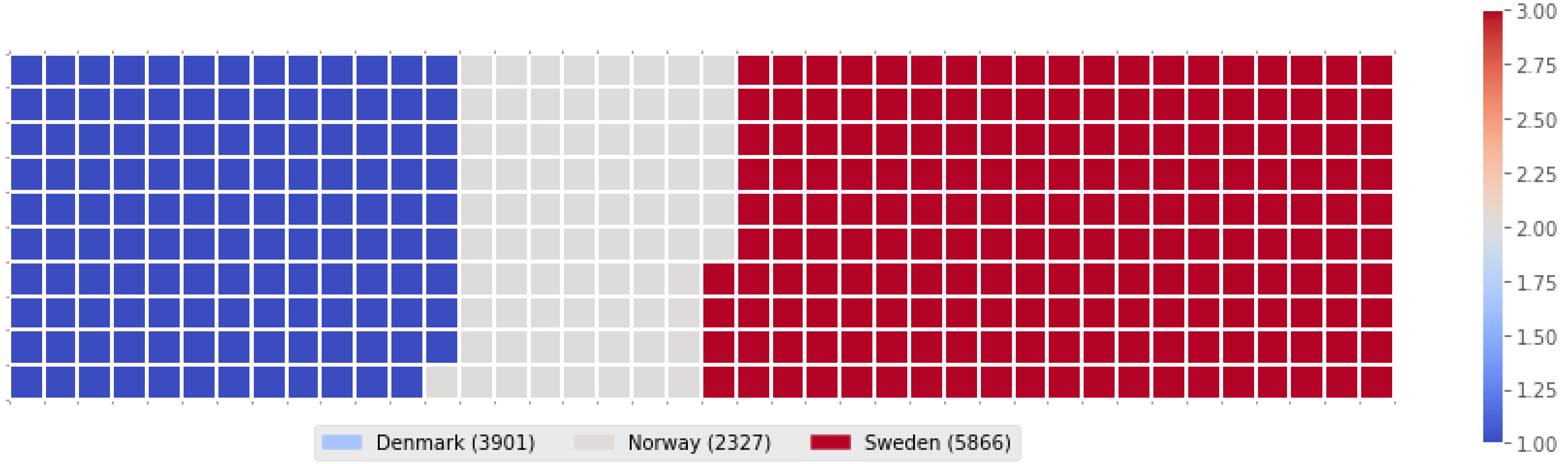
In [55]:

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

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

```
C:\Users\ahmed\AppData\Local\Temp\ipykernel_8864\3286913405.py:45: MatplotlibDeprecationWarning: Auto-removal of grids by pcolo
r() and pcolormesh() is deprecated since 3.5 and will be removed two minor releases later; please call grid(False) first.
  plt.colorbar()
```

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



In [58]:

```
import urllib

# open the file and read it into a variable alice_novel
alice_novel = urllib.request.urlopen('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetw
```

In [67]:

```
import seaborn as sns

print('Seaborn installed and imported!')
```

```
Seaborn installed and imported!
```

In [68]:

```
# we can use the sum() method to get the total population per year
df_tot = pd.DataFrame(can[years].sum(axis=0))

# change the years to type float (useful for regression later on)
df_tot.index = map(float, df_tot.index)

# reset the index to put in back in as a column in the df_tot dataframe
df_tot.reset_index(inplace=True)

# rename columns
df_tot.columns = ['year', 'total']

# view the final dataframe
df_tot.head()
```

Out[68]:

| | year | total |
|---|---|---|
| **0** | 1980.0 | 99137 |
| **1** | 1981.0 | 110563 |
| **2** | 1982.0 | 104271 |

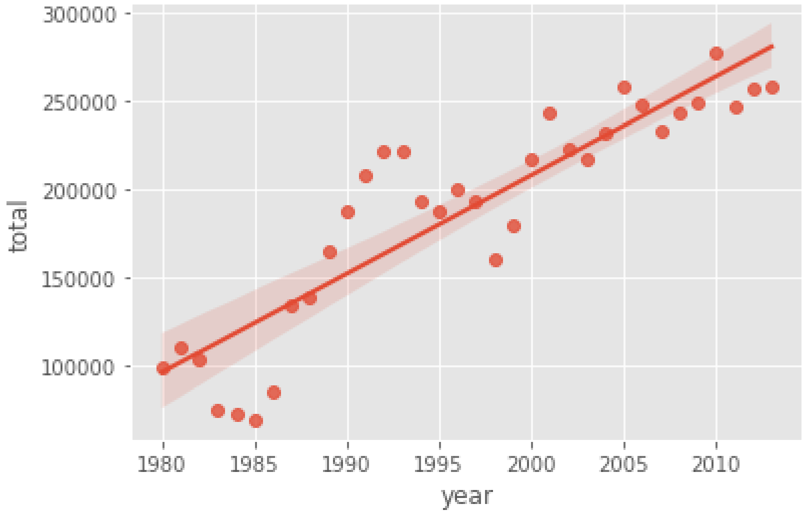
| | | |
|---|---|---|
| **3** | 1983.0 | 75550 |
| **4** | 1984.0 | 73417 |

In [69]:

```
sns.regplot(x='year', y='total', data=df_tot)
```

Out[69]:

```
<AxesSubplot:xlabel='year', ylabel='total'>
```



In [ ]: