In [ ]:
```
import numpy as np
import pandas as pd
from PIL import Image
```

In [ ]:
```
df = pd.read_excel('Canada.xlsx', sheet_name='Canada by Citizenship', skiprows = range(20), skipfooter = 2)
df.head()
```

Out[ ]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	2008
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3436	3009	2652	2111
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1223	856	702	560
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3626	4807	3623	4005
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	0	1	0	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	0	1	1	0

5 rows × 43 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(195, 43)
```

In [ ]:
```
df.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis = 1, inplace = True)
```

In [ ]:
```
df.rename(columns = {'OdName': 'Country', 'AreaName': 'Continent', 'RegName': 'Region'}, inplace = True)
```

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

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

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

In [ ]:
```
df['Total'] = df[years].sum(axis = 1)
```

In [ ]:
```
df.head()
```

Out[ ]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	1758
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856	702	560	716	561
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	4752
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1	0	0	0	0
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1	1	0	0	0

5 rows × 38 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(195, 38)
```

In [ ]:
```
import matplotlib
import matplotlib.pyplot as plt
```

```
import matplotlib.patches as mpatches
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'gra
yscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid',
'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk',
'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

In [ ]:
```
matplotlib.style.use('Solarize_Light2')
```

In [ ]:
```
df1 = df.loc[['Denmark', 'Norway', 'Sweden'], :]
df1.head()
```

Out[ ]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	201
Denmark	Europe	Northern Europe	Developed regions	272	293	299	106	93	73	93	...	62	101	97	108	81	92	9
Norway	Europe	Northern Europe	Developed regions	116	77	106	51	31	54	56	...	57	53	73	66	75	46	4
Sweden	Europe	Northern Europe	Developed regions	281	308	222	176	128	158	187	...	205	139	193	165	167	159	13

3 rows × 38 columns

In [ ]:
```
df2 = df.loc[['Denmark', 'Norway', 'Sweden']]
df2.head()
```

Out[ ]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	201
Denmark	Europe	Northern Europe	Developed regions	272	293	299	106	93	73	93	...	62	101	97	108	81	92	9
Norway	Europe	Northern Europe	Developed regions	116	77	106	51	31	54	56	...	57	53	73	66	75	46	4

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	201
Sweden	Europe	Northern Europe	Developed regions	281	308	222	176	128	158	187	...	205	139	193	165	167	159	13

3 rows × 38 columns

In [ ]:
```
totals = df1['Total'].sum()
proportions = df1['Total'] / totals
proportions = pd.DataFrame({'Proportion': proportions})
```

In [ ]:
```
proportions
```

Out[ ]:

Country	Proportion
Denmark	0.322557
Norway	0.192409
Sweden	0.485034

In [ ]:
```
w, l = 10, 40
area = w*l
print(area)
```

```
400
```

In [ ]:
```
proportions['NumTiles'] = (proportions['Proportion'] * area).round().astype(int)
proportions
```

Out[ ]:

Country	Proportion	NumTiles

	Proportion	NumTiles
Country		
Denmark	0.322557	129
Norway	0.192409	77
Sweden	0.485034	194

In [ ]:

```
waffle_chart = np.zeros((w, l), dtype = np.uint)
category_index = 0
tile_index = 0
```

In [ ]:

```
for col in range(l):
    for row in range(w):
        tile_index +=1
        if tile_index > proportions['NumTiles'].iloc[0:category_index].sum():
            category_index +=1
        waffle_chart[row, col] = category_index
print("Done")
```

```
Done
```

In [ ]:

```
waffle_chart
```

Out[ ]:

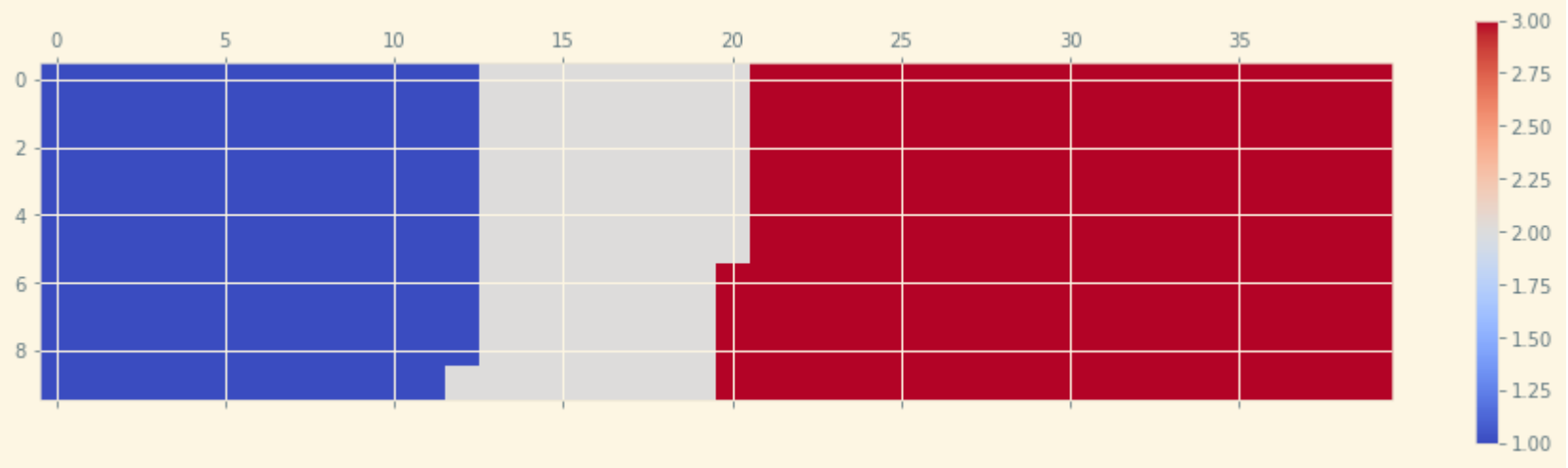
```
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
       [1, 1, 1, 1, 1, 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, 3, 3,
        3, 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, 3, 3, 3]],
      dtype=uint32)
```

In [ ]:

```
fig = plt.figure(figsize = (13, 5))
colormap = plt.cm.coolwarm
plt.matshow(waffle_chart, cmap = colormap)
plt.colorbar()
plt.show()
```

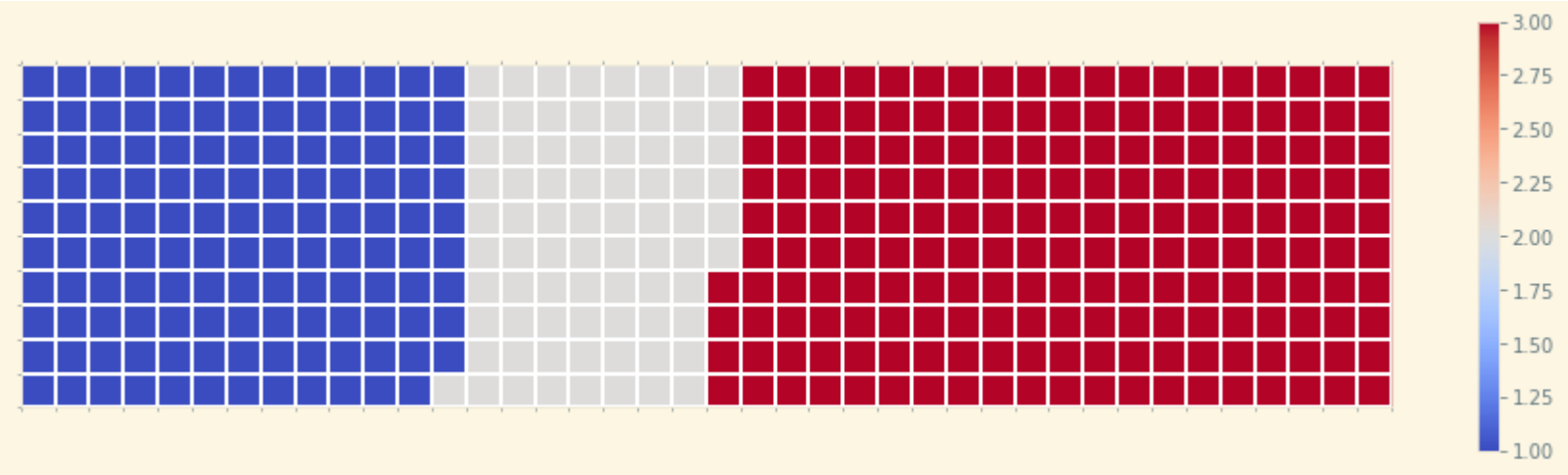
```
<Figure size 936x360 with 0 Axes>
```



In [ ]:

```
fig = plt.figure(figsize = (13, 5))
colormap = plt.cm.coolwarm
plt.matshow(waffle_chart, cmap = colormap)
plt.colorbar()
ax = plt.gca()
ax.set_xticks(np.arange(-.5, (l), 1), minor = True)
ax.set_yticks(np.arange(-.5, (w), 1), minor = True)
ax.grid(which = 'minor', color = 'w', linestyle = '-', linewidth = 2)
plt.xticks([])
plt.yticks([])
plt.show()
```

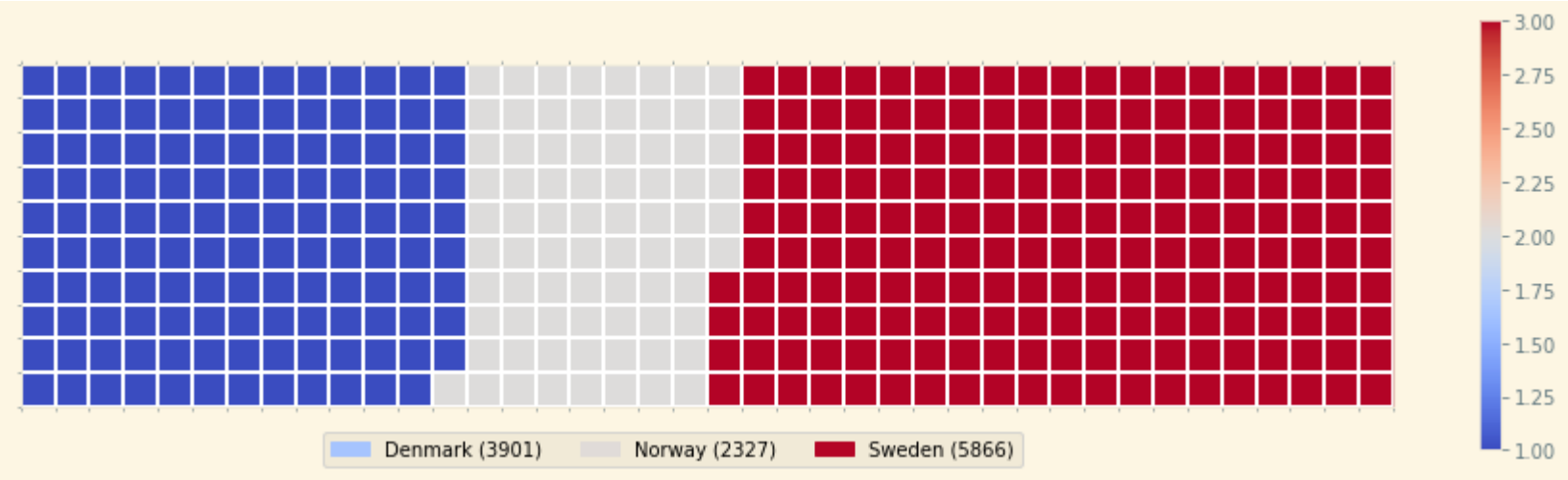
```
<Figure size 936x360 with 0 Axes>
```



In [ ]:

```
fig = plt.figure(figsize = (13, 5))
colormap = plt.cm.coolwarm
plt.matshow(waffle_chart, cmap = colormap)
plt.colorbar()
ax = plt.gca()
ax.set_xticks(np.arange(-.5, (l), 1), minor = True)
ax.set_yticks(np.arange(-.5, (w), 1), minor = True)
ax.grid(which = 'minor', color = 'w', linestyle = '-', linewidth = 2)
plt.xticks([])
plt.yticks([])

values_cumsum = np.cumsum(df1['Total'])
total_values = values_cumsum[len(values_cumsum) - 1]

#create legend
legend_handles = []
for i, category in enumerate(df1.index.values):
    label_str = category + ' (' + str(df1['Total'][i]) + ')'
    color_val = colormap(float(values_cumsum[i])/total_values)
    legend_handles.append(mpatches.Patch(color = color_val, label = label_str))

plt.legend(handles = legend_handles,
    loc = 'lower center', ncol = len(df1.index.values), bbox_to_anchor = (0, -0.2, 0.95, .1))


plt.show()
```

```
<Figure size 936x360 with 0 Axes>
```



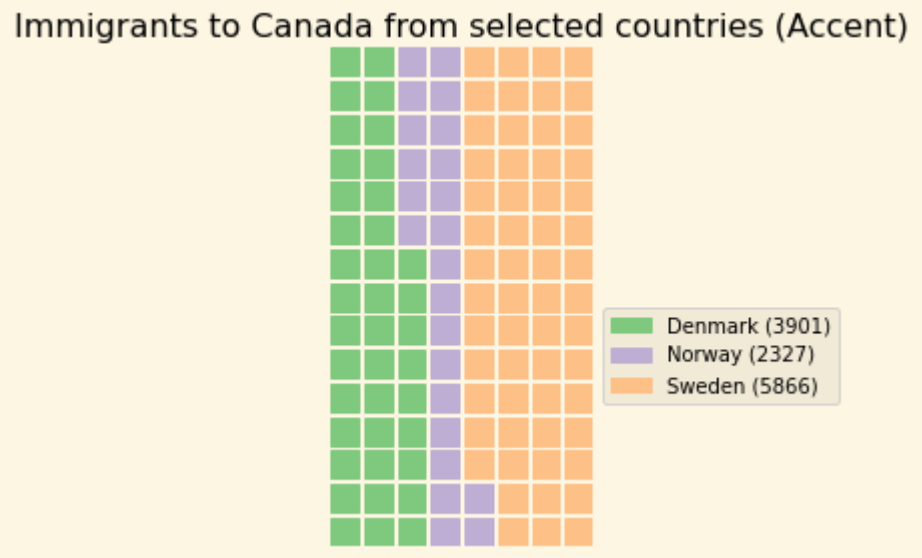
In [ ]:
```
!pip install pywaffle
```

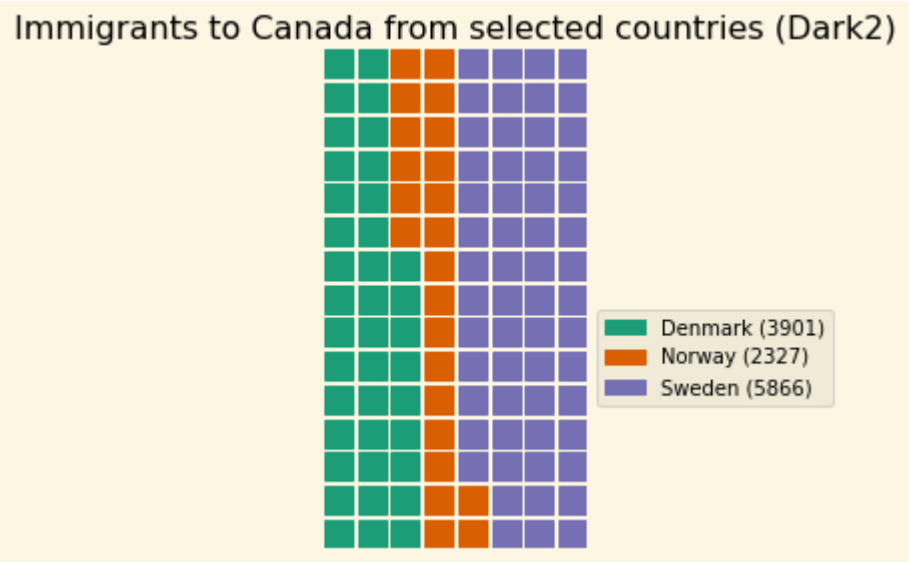
```
Requirement already satisfied: pywaffle in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (0.6.
3)
Requirement already satisfied: matplotlib in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (fro
m pywaffle) (3.4.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\gabri\appdata\roaming\python\python39\site-packages (from
matplotlib->pywaffle) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packag
es (from matplotlib->pywaffle) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-package
s (from matplotlib->pywaffle) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib->pywaffle) (8.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (f
rom matplotlib->pywaffle) (0.10.0)
Requirement already satisfied: numpy>=1.16 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (fr
om matplotlib->pywaffle) (1.21.1)
Requirement already satisfied: six in c:\users\gabri\appdata\roaming\python\python39\site-packages (from cycler>=0.10->ma
tplotlib->pywaffle) (1.16.0)
```

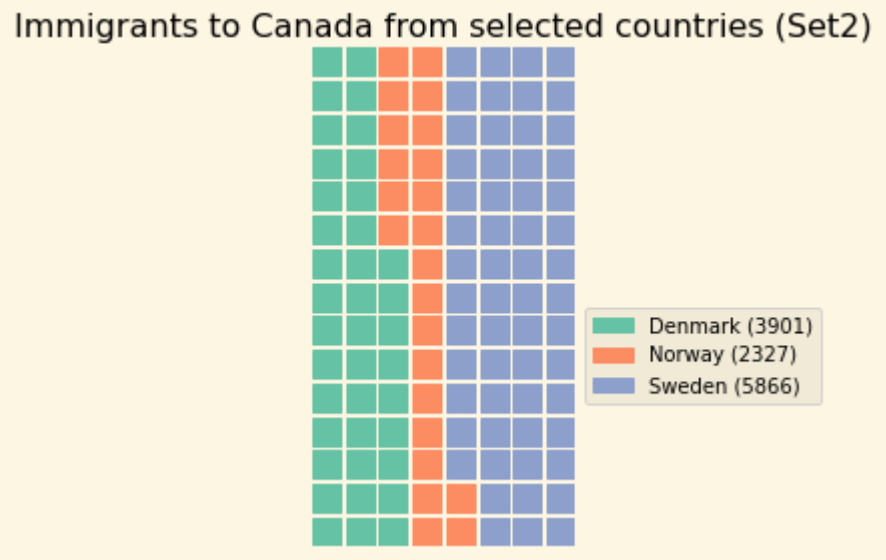
In [ ]:
```
from pywaffle import Waffle
```

In [ ]:
```
#Create waffle chart using pywaffle instead of manually
dict1 = {}
for i, category in enumerate(df1.index.values):
```

```
    dict1[category] = df1['Total'][i]
    i+=1

list = []
for key in dict1.keys():
    list.append(key)

#Accent, Dark2, Set2 are nice colormaps

colorlist = ['Accent', 'Dark2', 'Set2']
#for color in plt.colormaps():
for color in colorlist:
    try:
        fig = plt.figure(FigureClass= Waffle, rows = 15, columns = 8, values = dict1,
            labels = [f"{k} ({int(v)})" for k, v in dict1.items()],
            legend = {'loc': 'upper left', 'bbox_to_anchor': (1, .5)}, title = {'label': f'Immigrants to Canada from sele
            cmap_name = color
            )
        plt.show()
    except:
        pass
```







In [ ]:
```
df1.head()
```

Out[ ]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	201
Denmark	Europe	Northern Europe	Developed regions	272	293	299	106	93	73	93	...	62	101	97	108	81	92	9

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	201
Norway	Europe	Northern Europe	Developed regions	116	77	106	51	31	54	56	...	57	53	73	66	75	46	4
Sweden	Europe	Northern Europe	Developed regions	281	308	222	176	128	158	187	...	205	139	193	165	167	159	13

3 rows × 38 columns

In [ ]:
```
dict1 = {}
for i, category in enumerate(df1.index.values):
    dict1[category] = df1['Total'][i]
    i+=1
dict1
```

Out[ ]:
```
{'Denmark': 3901, 'Norway': 2327, 'Sweden': 5866}
```

In [ ]:
```
#Custom waffle chart function:

def create_waffle_chart(categories, values, height, width, colormap, dfname, 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 (dfname.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 [ ]:

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

categories = df1.index.values # categories
values = df1['Total'] # correponding values of categories

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

In [ ]:

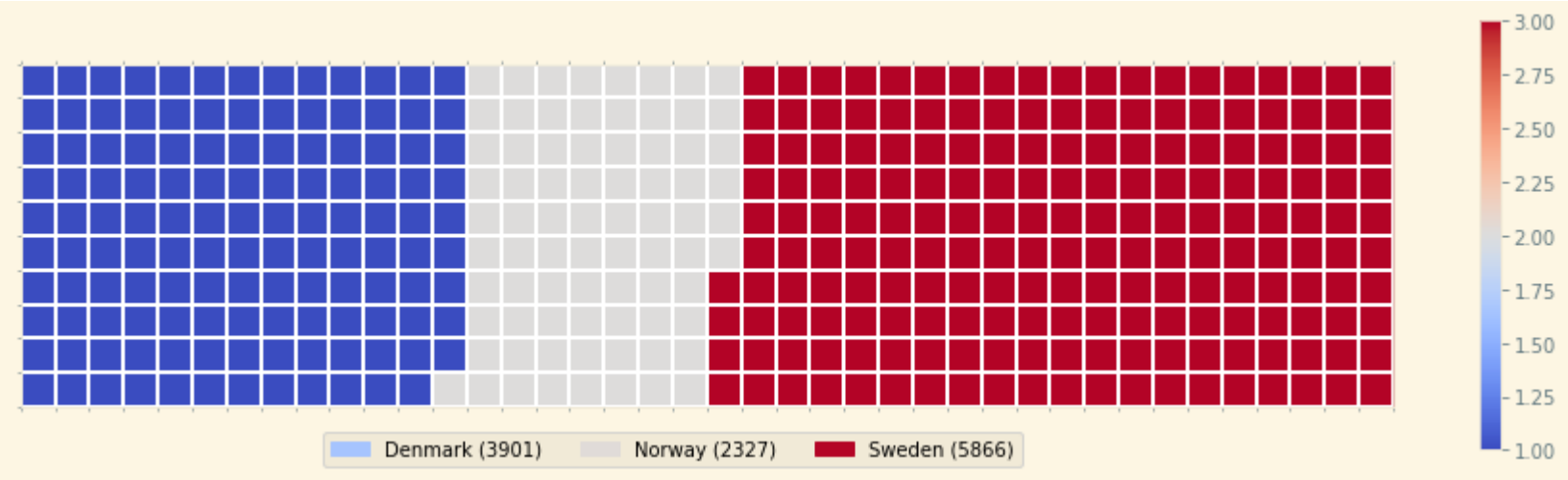
```
print(values)
```

```
Country
Denmark    3901
Norway     2327
Sweden     5866
Name: Total, dtype: int64
```

In [ ]:

```
create_waffle_chart(categories, values, height, width, colormap, df1)
```

```
Total number of tiles is 400
Denmark: 129
Norway: 77
Sweden: 194
<Figure size 432x288 with 0 Axes>
```



In [ ]:
```
import seaborn as sns
import pandas as pd
```

In [ ]:
```
df_tot = pd.DataFrame(df[years].sum(axis = 0))
df_tot.index = map(float, df_tot.index)
df_tot.reset_index(inplace = True)
df_tot.columns = ['year', 'total']
```

In [ ]:
```
df_tot.head()
```

Out[ ]:

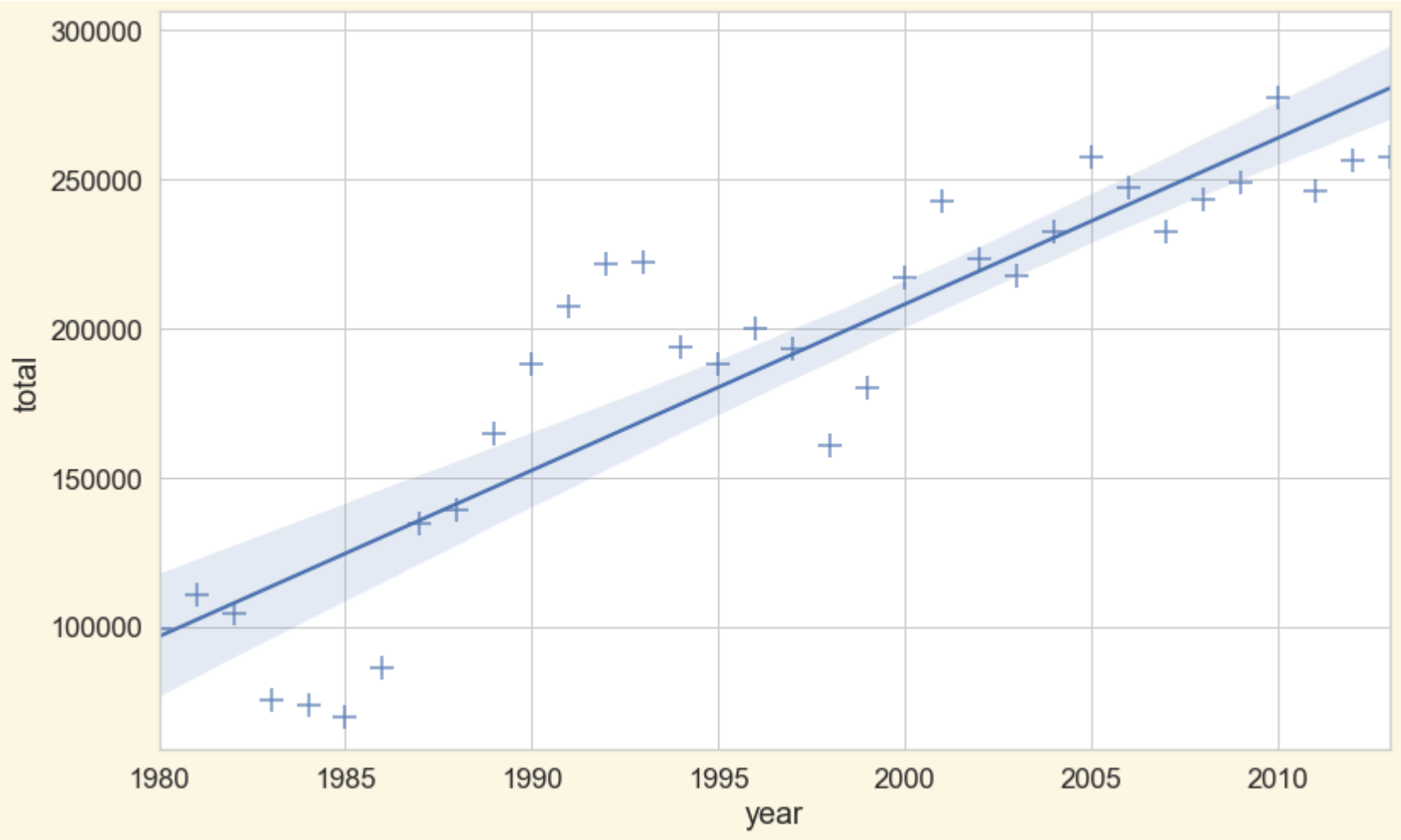
	year	total
0	1980.0	99137
1	1981.0	110563
2	1982.0	104271
3	1983.0	75550
4	1984.0	73417

In [ ]:
```
plt.figure(figsize = (13,8))
sns.set (font_scale = 1.5)
```

```
sns.set_style('whitegrid')
sns.regplot(x = 'year', y = 'total', data = df_tot, marker = "+", scatter_kws={'s': 200})
```

Out[ ]:

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



In [ ]:

```
df.head()
```

Out[ ]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	1758
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856	702	560	716	561
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	4752
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1	0	0	0	0
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1	1	0	0	0

5 rows × 38 columns

In [ ]:
```
df_tot2 = pd.DataFrame(df.loc[['Denmark', 'Sweden', 'Norway'],years].sum(axis = 0))
df_tot2.index = map(float, df_tot2.index)
df_tot2.reset_index(inplace = True)
df_tot2.columns = ['year', 'total']
df_tot2['year'] = df_tot2['year'].astype(int)
```

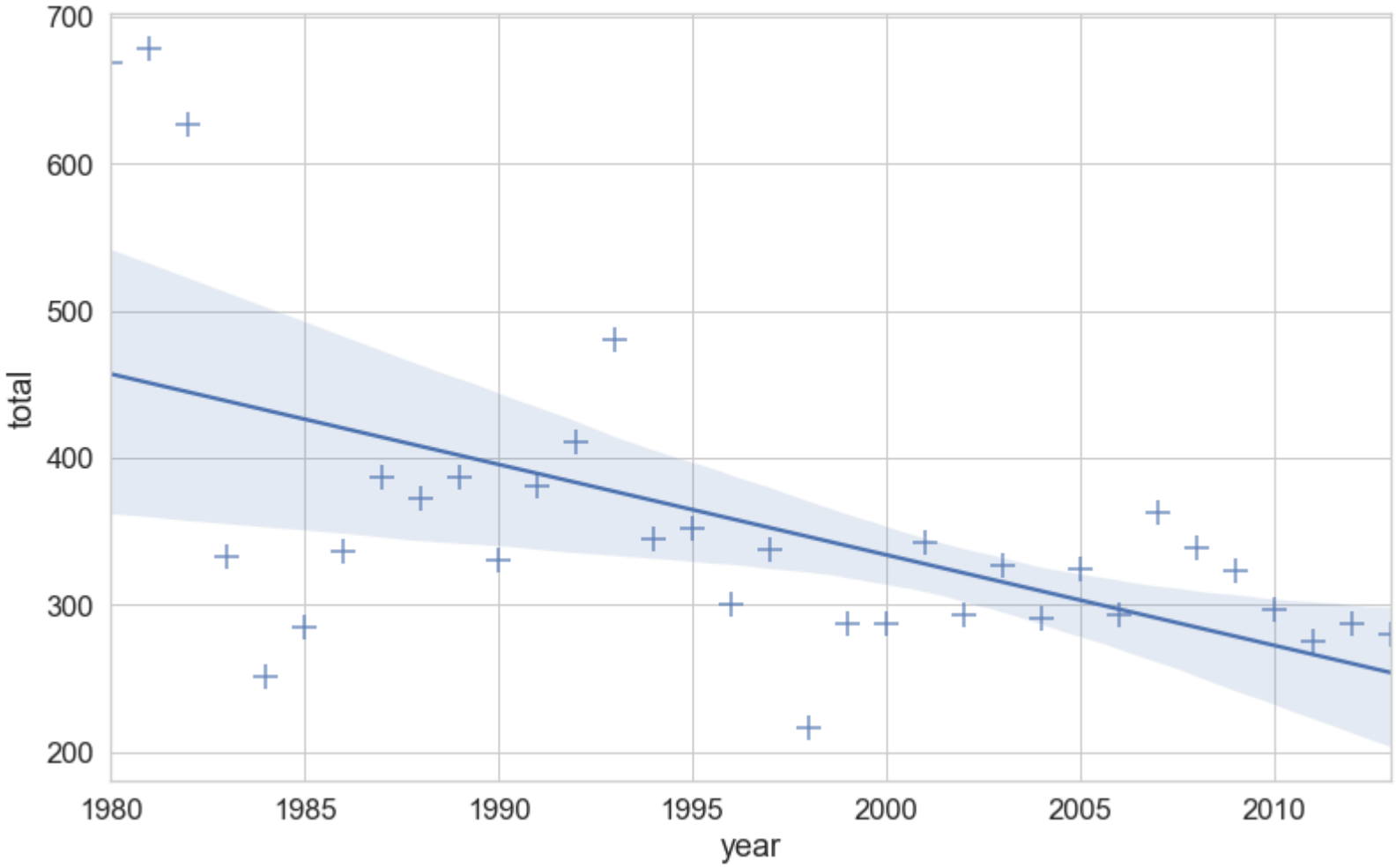
In [ ]:
```
df_tot2.head()
```

Out[ ]:

	year	total
0	1980	669
1	1981	678
2	1982	627
3	1983	333
4	1984	252

In [ ]:
```
plt.figure(figsize = (13,8))
sns.set (font_scale = 1.5)
sns.set_style('whitegrid')
sns.regplot(x = 'year', y = 'total', data = df_tot2, marker = "+", scatter_kws={'s': 200})
```

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



In [ ]:
```
!pip3 install wordcloud
```

```
Collecting wordcloud
  Using cached wordcloud-1.8.1.tar.gz (220 kB)
Requirement already satisfied: numpy>=1.6.1 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (f
rom wordcloud) (1.21.1)
```

```
Requirement already satisfied: pillow in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (from wo
rdcloud) (8.3.1)
Requirement already satisfied: matplotlib in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (fro
m wordcloud) (3.4.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-package
s (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packag
es (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\gabri\appdata\roaming\python\python39\site-packages (from
matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\gabri\appdata\local\programs\python\python39\lib\site-packages (f
rom matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: six in c:\users\gabri\appdata\roaming\python\python39\site-packages (from cycler>=0.10->ma
tplotlib->wordcloud) (1.16.0)
Using legacy 'setup.py install' for wordcloud, since package 'wheel' is not installed.
Installing collected packages: wordcloud
    Running setup.py install for wordcloud: started
    Running setup.py install for wordcloud: finished with status 'done'
Successfully installed wordcloud-1.8.1
```

In [ ]:

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

In [ ]:

```
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/IBMDeveloperSkil
```

In [ ]:

```
stopwords = set(STOPWORDS)
```

In [ ]:

```
print(stopwords)
```

```
{"where's", 'both', 'whom', 'in', "it's", 'him', "wasn't", 'at', 'few', 'since', 'r', 'below', "they'd", 'yourselves', "l
et's", 'against', 'had', "you'd", 'however', 'has', 'me', 'under', 'above', 'are', "he'll", 'get', 'nor', 'is', 'so', 'af
ter', "he'd", 'theirs', "they'll", 'otherwise', "who's", 'to', 'about', "can't", 'would', 'you', 'not', 'an', 'our', 'of
f', 'between', "weren't", 'this', 'when', 'shall', 'before', 'own', 'but', 'be', 'am', 'her', "that's", 'he', "why's", "t
hey're", 'do', 'by', 'than', 'with', 'why', 'for', "what's", 'on', 'ours', "he's", "here's", 'as', "i'm", "shouldn't", "y
ou'll", 'over', 'no', 'www', 'once', "didn't", "how's", 'into', 'what', 'more', 'else', 'from', 'itself', 'could', "we'r
e", 'did', 'same', "she'll", 'its', 'was', 'up', 'k', 'only', 'of', 'some', "hadn't", 'all', "i'll", "mustn't", "she's",
'most', 'further', 'those', 'also', "won't", 'other', 'were', 'during', 'having', 'any', 'there', 'have', 'while', 'becau
se', "we'll", "hasn't", 'very', 'where', "aren't", "they've", 'such', 'here', "you're", 'who', 'i', 'she', 'which', 'ough
t', 'does', 'that', 'their', 'it', 'out', "don't", 'we', 'these', "we'd", 'themselves', 'therefore', 'until', 'being', 'h
```

```
ence', 'how', 'too', "i'd", 'my', 'or', "when's", 'should', 'com', 'herself', "there's", "isn't", 'yourself', 'can', 'eve
r', 'his', 'like', 'myself', 'ourselves', 'cannot', "couldn't", "wouldn't", "haven't", 'just', 'the', 'hers', 'himself',
'and', "we've", 'if', 'been', 'doing', "she'd", 'through', 'down', 'them', 'your', "doesn't", 'a', "you've", 'again',
"i've", 'yours', 'http', "shan't", 'each', 'they', 'then'}
```

In [ ]:
```
alice_wc = WordCloud(background_color='white', max_words = 2000, stopwords = stopwords)
alice_wc.generate(alice_novel)
```

Out[ ]:
```
<wordcloud.wordcloud.WordCloud at 0x2687558ca30>
```

In [ ]:
```
plt.imshow(alice_wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [ ]:
```
stopwords.add('said')
alice_wc.generate(alice_novel)
plt.imshow(alice_wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [ ]:
```
# save mask to alice_mask
alice_mask = np.array(Image.open(urllib.request.urlopen('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clo
```

In [ ]:
```
fig = plt.figure(figsize=(14, 18))

plt.imshow(alice_mask, cmap=plt.cm.gray, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [ ]:

```
alice_wc = WordCloud(background_color='white', max_words = 2000, stopwords = stopwords, mask = alice_mask)
alice_wc.generate(alice_novel)
fig = plt.figure(figsize = (15, 15))
plt.imshow(alice_wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```


Alice
little
head
one
went
well
Queen
time
thing
know
now
see
began
Hatter
Mock Turtle
round
two
go
quite
Cat
Duchess
must
much
never
say
got
first
tell
back
looked
come
came
great
last
thought
voice
long
Dormouse
Mouse
Gryphon
large
day
eye
tone
make
door
replied
March Hare
looking
nothing
hand
found
Oh
word
heard
minute
put
made
Caterpillar
White Rabbit
something
might
two
hastily
asked
soon
felt
face
court
garden
seemed
talking
getting
idea
please
jury
really
next
turned
course
perhaps
upon
half
sat
sort
another
dear
till
tree
three
baby
people
everything
mouth
find
may
tea
even
sure
right
speak
question
leave
saying
going
old
enough
curious
live
added
change
took
hear
thought Alice

In [ ]:

```
df.head()
```

Out[ ]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	1758
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856	702	560	716	561
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	4752
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1	0	0	0	0
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1	1	0	0	0

5 rows × 38 columns

In [ ]:

```
max_words = 100
total_immigration = df['Total'].sum()
word_string = ''
for country in df.index.values:
    if country.count(" ") ==0:
        repeat_num_times = int(df.loc[country, 'Total']/ total_immigration * max_words)
        word_string = word_string + ((country + ' ')*repeat_num_times)
word_string
```

Out[ ]:

```
'Algeria Bangladesh China China China China China China China China China China Colombia Egypt France Guyana Haiti India
India India India India India India India India India India Iraq Israel Jamaica Lebanon Morocco Pakistan Pakistan Pakistan Phil
ippines Philippines Philippines Philippines Philippines Philippines Philippines Poland Poland Portugal Romania '
```

In [ ]:

```
wordcloud2 = WordCloud(background_color='white').generate(word_string)
plt.figure(figsize = (11, 6))
plt.imshow(wordcloud2, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```


Iraq
Poland
Pakistan
Algeria
Bangladesh
Israel
China China
Philippines
France
Morocco
Egypt
Haiti
Guyana
Romania
India India
Portugal
Colombia
Lebanon
Jamaica

In [ ]: