Downloading and Prepping Data

Sanabila Khoirunnisa - 1301204097

Import primary modules.

In [20]:
```
import numpy as np  # useful for many scientific computing in Python
import pandas as pd # primary data structure library
```

Download the dataset and read it into a *pandas* dataframe.

In [21]:
```
df_can = pd.read_excel('https://s3-api.us-geo.objectstorage.softlayer.net/cf-course
                        sheet_name='Canada by Citizenship',
                        skiprows=range(20),
                        skipfooter=2
                       )

print('Data downloaded and read into a dataframe!')
```

```
Data downloaded and read into a dataframe!
```

Clean up data. We will make some modifications to the original dataset to make it easier to create our visualizations. Refer to *Introduction to Matplotlib and Line Plots* and *Area Plots, Histograms, and Bar Plots* for a detailed description of this preprocessing.

In [22]:
```
# clean up the dataset to remove unnecessary columns (eg. REG)
df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis=1, inplace=True)

# let's rename the columns so that they make sense
df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent','RegName':'Regio

# for sake of consistency, let's also make all column labels of type string
df_can.columns = list(map(str, df_can.columns))

# set the country name as index - useful for quickly looking up countries using .l

df_can.set_index('Country', inplace=True)

# add total column
df_can['Total'] = df_can.sum(axis=1)

# years that we will be using in this lesson - useful for plotting later on
years = list(map(str, range(1980, 2014)))
print('data dimensions:', df_can.shape)
```

```
data dimensions: (195, 38)
C:\Users\sanabila\AppData\Local\Temp\ipykernel_16076\3015018611.py:14: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only val
id columns before calling the reduction.
  df_can['Total'] = df_can.sum(axis=1)
```

In [23]:
```
df_can
```

Out[23]:

	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2
Country												
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
Viet Nam	Asia	South-Eastern Asia	Developing regions	1191	1829	2162	3404	7583	5907	2741	...	1
Western Sahara	Africa	Northern Africa	Developing regions	0	0	0	0	0	0	0	...	
Yemen	Asia	Western Asia	Developing regions	1	2	1	6	0	18	7	...	
Zambia	Africa	Eastern Africa	Developing regions	11	17	11	7	16	9	15	...	
Zimbabwe	Africa	Eastern Africa	Developing regions	72	114	102	44	32	29	43	...	

195 rows × 38 columns

Import Matplotlib

In [24]:
```python
%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('ggplot') # optional: for ggplot-like style

# check for latest version of Matplotlib
print('Matplotlib version: ', mpl.__version__) # >= 2.0.0
```

```
Matplotlib version:  3.5.2
```

Question 1: Using a pie chart, explore the proportion (percentage) of new immigrants grouped by continents in the year 2013.

Note: You might need to play with the explore values in order to fix any overlapping slice values.

In [25]:
```python
### type your answer here

# group countries by continents and apply sum() function
df_continents = df_can.groupby('Continent', axis=0).sum()
```

```
# note: the output of the groupby method is a `groupby' object.
# we can not use it further until we apply a function (eg .sum())
print(type(df_can.groupby('Continent', axis=0)))

df_continents.head()
```

```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

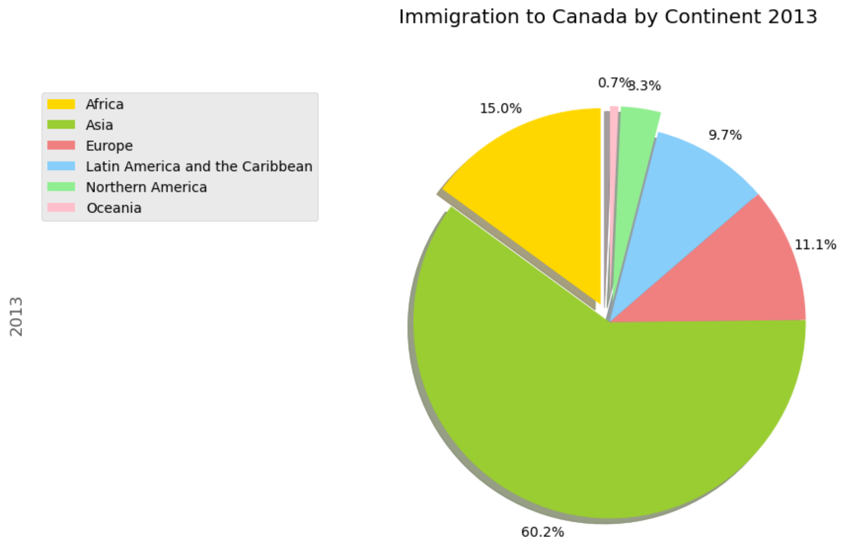
Out[25]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2005
Continent												
Africa	3951	4363	3819	2671	2639	2650	3782	7494	7552	9894	...	27523
Asia	31025	34314	30214	24696	27274	23850	28739	43203	47454	60256	...	159253
Europe	39760	44802	42720	24638	22287	20844	24370	46698	54726	60893	...	35955
Latin America and the Caribbean	13081	15215	16769	15427	13678	15171	21179	28471	21924	25060	...	24747
Northern America	9378	10030	9074	7100	6661	6543	7074	7705	6469	6790	...	8394

5 rows × 35 columns

In [26]:

```
colors_list = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'lightgreen',
explode_list = [0.1, 0, 0, 0, 0.1, 0.1] # ratio for each continent with which to o

df_continents['2013'].plot(kind='pie',
                            figsize=(15, 6),
                            autopct='%1.1f%%',
                            startangle=90,
                            shadow=True,
                            labels=None,         # turn off labels on pie chart
                            pctdistance=1.12,    # the ratio between the center of
                            colors=colors_list,  # add custom colors
                            explode=explode_list # 'explode' lowest 3 continents
                            )

# scale the title up by 12% to match pctdistance
plt.title('Immigration to Canada by Continent 2013', y=1.12)

plt.axis('equal')

# add legend
plt.legend(labels=df_continents.index, loc='upper left')

plt.show()
```



Question 2: Compare the distribution of the number of immigrants from Pakistan and Afghanistan from 1980 to 2013.

Step 1: Get the dataset for Pakistan and Afghanistan and call the dataframe **df_PA**.

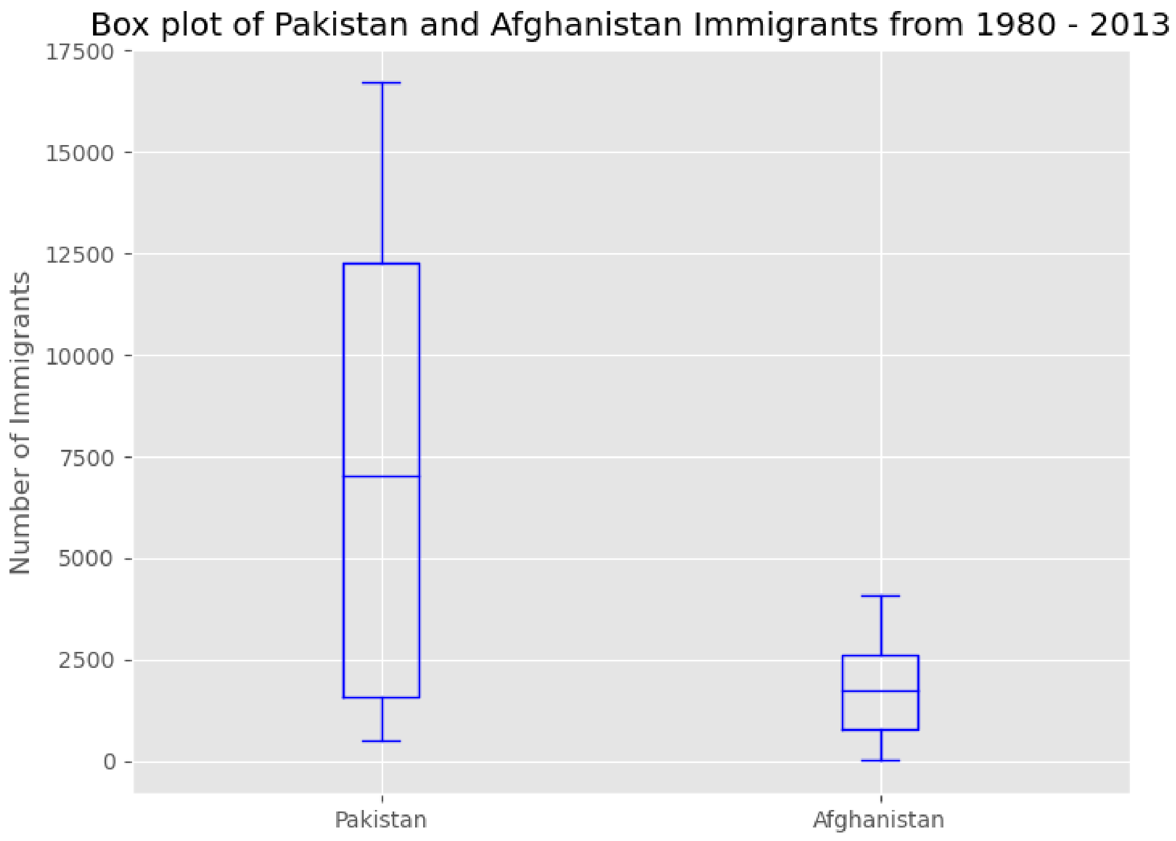
In [27]:
```
### type your answer here

df_PA = df_can.loc[['Pakistan', 'Afghanistan'], years].transpose()
df_PA
```

Out[27]:

Country	Pakistan	Afghanistan
1980	978	16
1981	972	39
1982	1201	39
1983	900	47
1984	668	71
1985	514	340
1986	691	496
1987	1072	741
1988	1334	828
1989	2261	1076
1990	2470	1028
1991	3079	1378
1992	4071	1170
1993	4777	713
1994	4666	858
1995	4994	1537
1996	9125	2212
1997	13073	2555
1998	9068	1999
1999	9979	2395
2000	15400	3326
2001	16708	4067
2002	15110	3697
2003	13205	3479
2004	13399	2978
2005	14314	3436
2006	13127	3009
2007	10124	2652
2008	8994	2111
2009	7217	1746
2010	6811	1758
2011	7468	2203
2012	11227	2635
2013	12603	2004

Step 2: Plot data.

In [28]:
```
### type your answer here

df_PA.plot(kind='box', figsize=(8, 6), color = 'blue')

plt.title('Box plot of Pakistan and Afghanistan Immigrants from 1980 - 2013')
plt.ylabel('Number of Immigrants')

plt.show()
```



Question 3: Create a scatter plot of the total immigration from Denmark, Norway, and Sweden to Canada from 1980 to 2013?

Step 1: Get the data:

1. Create a dataframe the consists of the numbers associated with Denmark, Norway, and Sweden only. Name it **df_countries**.
2. Sum the immigration numbers across all three countries for each year and turn the result into a dataframe. Name this new dataframe **df_total**.
3. Reset the index in place.
4. Rename the columns to **year** and **total**.
5. Display the resulting dataframe.

In [40]:
```
### type your answer here

df_countries = df_can.loc[['Denmark', 'Norway', 'Sweden'], years].transpose()

df_total = pd.DataFrame(df_countries.sum(axis=1))

df_total.reset_index(inplace=True)

df_total.columns = ['year', 'total']
```

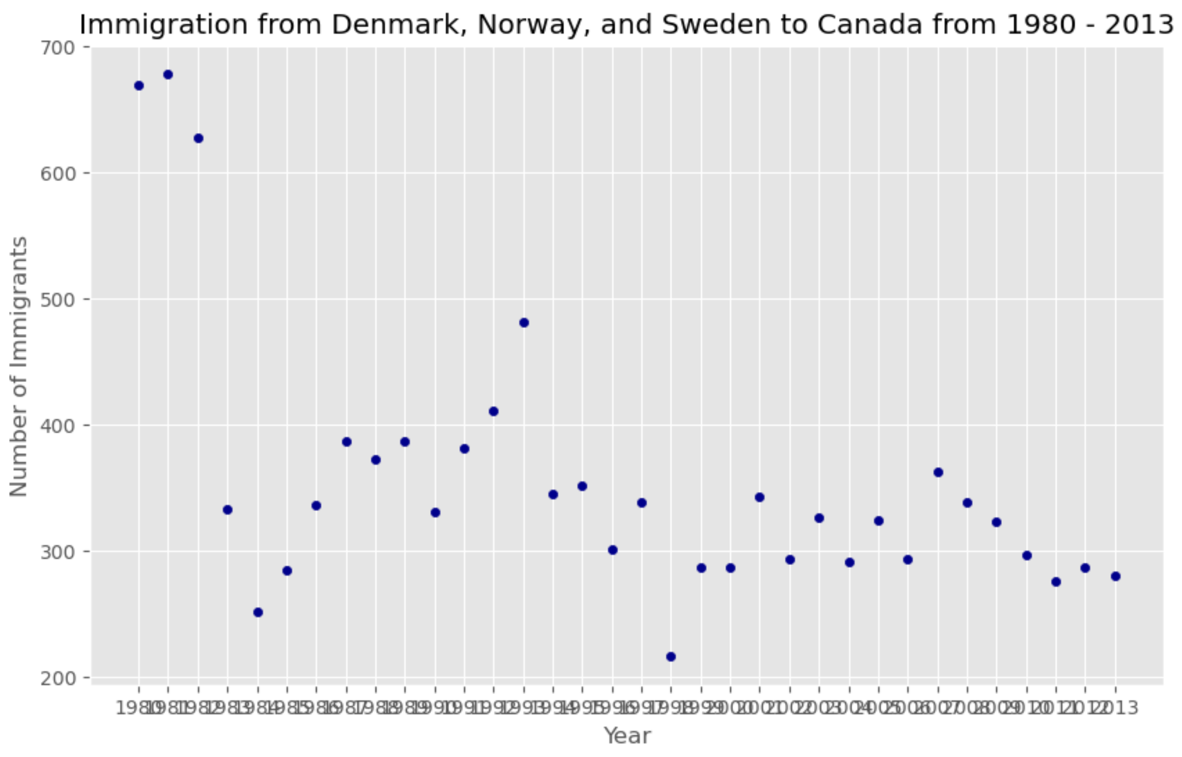
```
df_total
```

Out[40]:

	year	total
0	1980	669
1	1981	678
2	1982	627
3	1983	333
4	1984	252
5	1985	285
6	1986	336
7	1987	387
8	1988	373
9	1989	387
10	1990	331
11	1991	381
12	1992	411
13	1993	481
14	1994	345
15	1995	352
16	1996	301
17	1997	338
18	1998	217
19	1999	287
20	2000	287
21	2001	343
22	2002	293
23	2003	327
24	2004	291
25	2005	324
26	2006	293
27	2007	363
28	2008	339
29	2009	323
30	2010	297
31	2011	276
32	2012	287
33	2013	280

Step 2: Generate the scatter plot by plotting the total versus year in **df_total**.

In [41]:
```
### type your answer here

df_total.plot(kind='scatter', x='year', y='total', figsize=(10, 6), color='darkblu

plt.title('Immigration from Denmark, Norway, and Sweden to Canada from 1980 - 2013
plt.xlabel('Year')
plt.ylabel('Number of Immigrants')

plt.show()
```



Question 4: Previously in this lab, we created box plots to compare immigration from China and India to Canada. Create bubble plots of immigration from China and India to visualize any differences with time from 1980 to 2013. You can use **df_can_t** that we defined and used in the previous example.

Step 1: Normalize the data pertaining to China and India.

In [42]:
```
### type your answer here

df_can_t = df_can[years].transpose()
df_can_t.index = map(int, df_can_t.index)
df_can_t.index.name = 'Year'
df_can_t.reset_index(inplace=True)
df_can_t.head()
```

Out[42]:

Country	Year	Afghanistan	Albania	Algeria	American Samoa	Andorra	Angola	Antigua and Barbuda	Argentina
0	1980	16	1	80	0	0	1	0	368
1	1981	39	0	67	1	0	3	0	426
2	1982	39	0	71	0	0	6	0	626
3	1983	47	0	69	0	0	6	0	241
4	1984	71	0	63	0	0	4	42	237

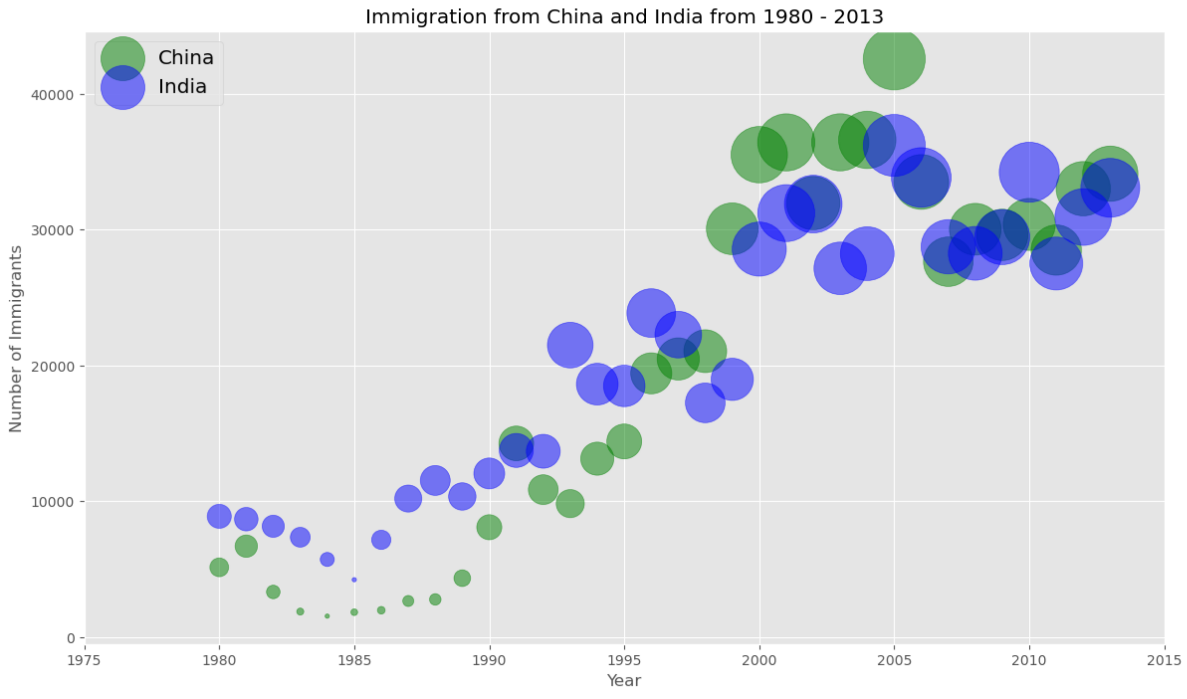
5 rows × 196 columns

In [43]:
```
#normalize China Data
norm_china = (df_can_t['China'] - df_can_t['China'].min()) / (df_can_t['China'].ma

#normalize India data
norm_india = (df_can_t['India'] - df_can_t['India'].min()) / (df_can_t['India'].ma
```

Step 2: Generate the bubble plots.

In [44]:
```
### type your answer here

ax0 = df_can_t.plot(kind='scatter',
                    x='Year',
                    y='China',
                    figsize=(14, 8),
                    alpha=0.5,
                    color='green',
                    s=norm_china * 2000 + 10,
                    xlim=(1975, 2015)
                   )

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

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

Out[44]:
```
<matplotlib.legend.Legend at 0x11b421c4400>
```



Thank you for completing this lab!
