NAME : Nidhi Kaushik

DATASET USED : Canada Immigration from year 1980-2013

we can get this data following this link: https://www.kaggle.com/datasets/ammaraahmad/immigration-to-canada

I have already downloaded the data and uploaded it here in colab notebook, so I am directly using it's path to read.

**Description: Analysis of trends and hidden patterns of Immigrants from different countries to Canada. This dataset consists of immigrants record from 150+ countries to Canada between 1980 to 2013.

I have used the following python libraries: pandas, numpy, matplotlib for preprocessing and for doing the immigration trend analysis using scatter plot, line plot and pie chart.**

importing libraries and dataset

```
import numpy as np
import pandas as pd
```

```
df_Cnd_Img = pd.read_excel('/content/Canada.xlsx',
                       sheet_name='Canada by Citizenship',
                       skiprows=range(20),
                       skipfooter=2)
```

```
df_Cnd_Img
```

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
190	Immigrants	Foreigners	Viet Nam	935	Asia	920	South-Eastern Asia	902	Developing regions	1191	...	1816	1
191	Immigrants	Foreigners	Western Sahara	903	Africa	912	Northern Africa	902	Developing regions	0	...	0	

```
type(df_Cnd_Img)
```

```
pandas.core.frame.DataFrame
```

```
#The info() method prints information about the DataFrame.
#The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in
#Note: the info() method actually prints the info.
```

```
df_Cnd_Img.info()
```

```
df_Cnd_Img.isnull().any()       #checking if any null value present
```

```
#checking the dtype for column variables
```

```
df_Cnd_Img.columns.values
```

```
array(['Type', 'Coverage', 'OdName', 'AREA', 'AreaName', 'REG', 'RegName',
       'DEV', 'DevName', 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], dtype=object)
```

PREPROCESSING

```
#Let's clean the data set to remove a few unnecessary columns. We can use pandas drop() method as follows:
# in pandas axis=0 represents rows (default) and axis=1 represents columns.
df_Cnd_Img.drop(['AREA','REG','DEV','Type','Coverage'], axis=1, inplace=True)
#view the top 2 rows of the dataset using the head() function.
df_Cnd_Img.head(2)
```

	OdName	AreaName	RegName	DevName	1980	1981	1982	1983	1984	1985	...	2004	2005	2006
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2978	3436	3009
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1450	1223	856

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

```
print(df_Cnd_Img.columns.values)
```

```
['Country' 'Continent' 'Region' 'DevName' 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]
```

```
#Column names that are integers (such as the years) might introduce some confusion
#For example, when we are referencing the year 2013, one might confuse that when the 2013th positional index.
#To avoid this ambuigity, let's convert the column names into strings: '1980' to '2013'.
```

```
df_Cnd_Img.columns = list(map(str, df_Cnd_Img.columns))
```

```
print(df_Cnd_Img.columns.values)
```

```
['Country' 'Continent' 'Region' 'DevName' '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']
```

```
df_Cnd_Img.describe()        #The .describe() method returns description of the data in the DataFrame.
```

	1980	1981	1982	1983	1984	1985	1986	
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	1
mean	508.394872	566.989744	534.723077	387.435897	376.497436	358.861538	441.271795	6
std	1949.588546	2152.643752	1866.997511	1204.333597	1198.246371	1079.309600	1225.576630	21
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.500000	
50%	13.000000	10.000000	11.000000	12.000000	13.000000	17.000000	18.000000	
75%	251.500000	295.500000	275.000000	173.000000	181.000000	197.000000	254.000000	4
max	22045.000000	24796.000000	20620.000000	10015.000000	10170.000000	9564.000000	9470.000000	213

8 rows × 34 columns

```
df_Cnd_Img
```

	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2004	2005	20
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2978	3436	30
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1450	1223	8
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3616	3626	48
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	0	

Before we proceed, notice that the defaut index of the dataset is a numeric range from 0 to 194. This makes it very difficult to do a query by a specific country. For example to search for data on Japan, we need to know the corressponding index value.

This can be fixed very easily by setting the 'Country' column as the index using `set_index()` method.

```
df_Cnd_Img = df_Cnd_Img.set_index('Country')

# tip : The opposite of set is reset. So to reset the index, we can use df_can.reset_index()
```

QUESTION :

Let's view the number of immigrants from Japan (row 87) for the following scenarios:

1. The full row data (all columns)
2. For year 2013
3. For years 1980 to 1985

```
# 1. the number of immigrants from Japan the full row data (all columns)
print(df_Cnd_Img.loc['Japan'])

# alternate methods
# print(df_can.iloc[87])
# print(df_can[df_can.index == 'Japan'].T.squeeze())
```

```
Continent                Asia
Region            Eastern Asia
DevName      Developed regions
1980                       701
1981                       756
1982                       598
1983                       309
1984                       246
1985                       198
1986                       248
1987                       422
1988                       324
1989                       494
1990                       379
1991                       506
1992                       605
1993                       907
1994                       956
1995                       826
1996                       994
1997                       924
1998                       897
1999                      1083
2000                      1010
2001                      1092
2002                       806
2003                       817
2004                       973
2005                      1067
2006                      1212
2007                      1250
2008                      1284
2009                      1194
2010                      1168
2011                      1265
2012                      1214
2013                       982
Name: Japan, dtype: object
```

```
#2. for year 2013
total_immigrants = df_Cnd_Img.iloc[87, 36]
```

```
print(f'No. of immigrants from japan in 2013 is : {total_immigrants}')

#alternative method : using .loc
#print(df_Cnd_Img.loc['Japan', '2013'])
```

```
     No. of immigrants from japan in 2013 is : 982
```

```
# 3. for years 1980 to 1985
print(df_Cnd_Img.iloc[87, 3:9])
```

```
     1980    701
     1981    756
     1982    598
     1983    309
     1984    246
     1985    198
     Name: Japan, dtype: object
```

QUESTION :

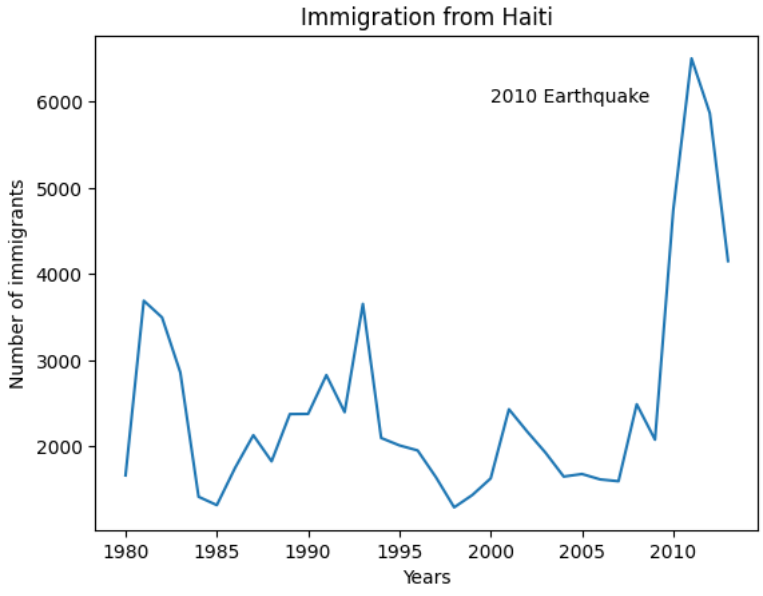
OBJECTIVE : In 2010, Haiti suffered a catastrophic magnitude 7.0 earthquake. The quake caused widespread devastation and loss of life and aout three million people were affected by this natural disaster. As part of Canada's humanitarian effort, the Government of Canada stepped up its effort in accepting refugees from Haiti. We can quickly visualize this effort using a `Line` plot:

Plot a line graph of immigration from Haiti using `df.plot()`.

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

```
haiti = df_Cnd_Img.loc['Haiti', '1980':'2013']
```

```
haiti.index = haiti.index.map(int) # let's change the index values of Haiti to type integer for plotting
haiti.plot(kind='line')

plt.title('Immigration from Haiti')
plt.ylabel('Number of immigrants')
plt.xlabel('Years')

# annotate the 2010 Earthquake.
# syntax: plt.text(x, y, label)
plt.text(2000, 6000, '2010 Earthquake') # see note below

plt.show() # need this line to show the updates made to the figure
```



QUESTION :

Let's compare the number of immigrants from India and China from 1980 to 2013.

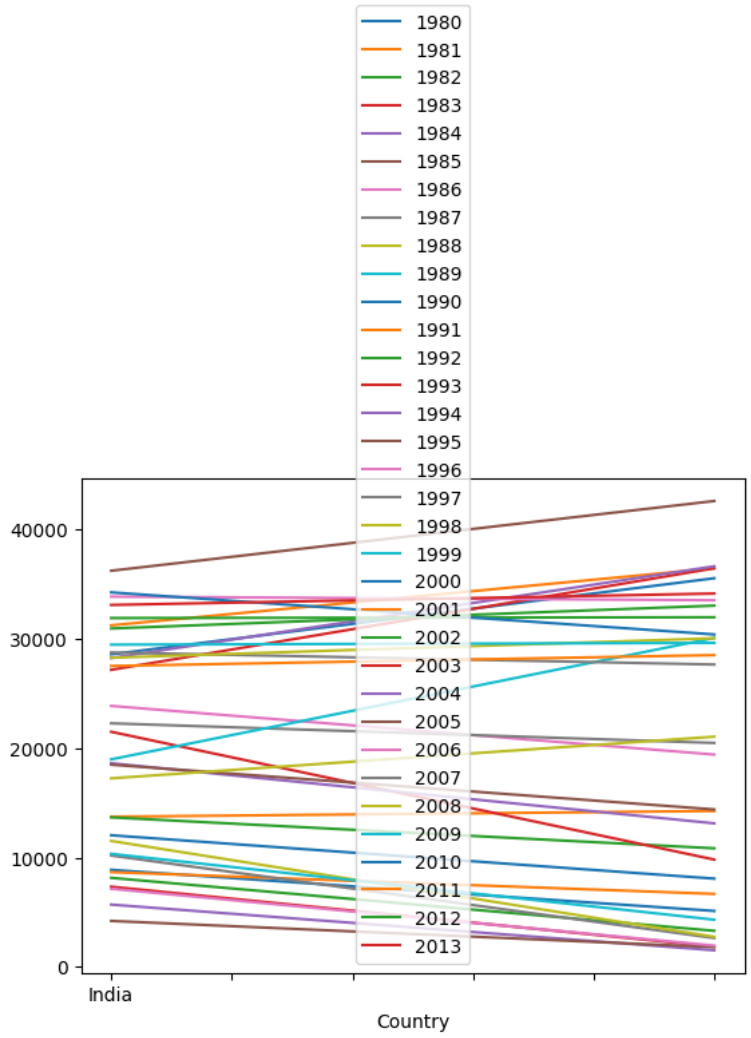
```
df_IC = df_Cnd_Img.loc[['India', 'China'], '1980':'2013']
df_IC.head()
```

Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	...	28235	36210	33848	28742
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	...	36619	42584	33518	27642

2 rows × 34 columns

```
df_IC.plot(kind='line')
```

```
<Axes: xlabel='Country'>
```



That doesn't look right...

Recall that *pandas* plots the indices on the x-axis and the columns as individual lines on the y-axis. Since `df_CI` is a dataframe with the `country` as the index and `years` as the columns, we must first transpose the dataframe using `transpose()` method to swap the row and columns.

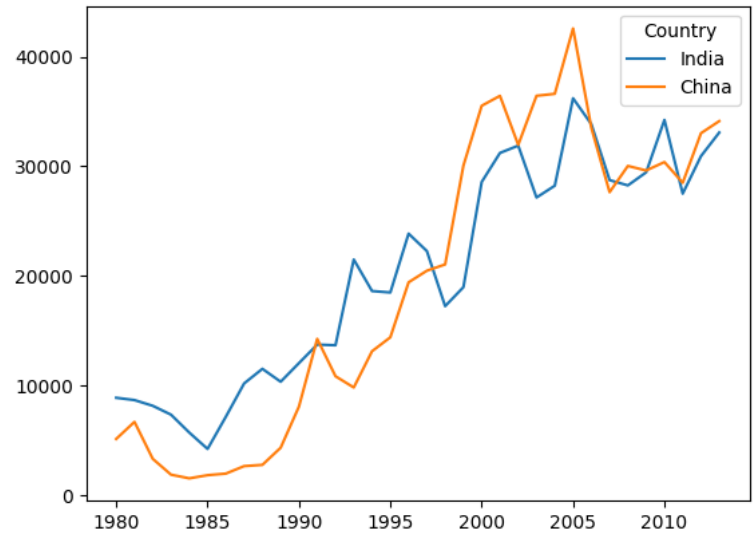
```
df_IC = df_IC.transpose()
```

```
df_IC.head()
```

Country	India	China
1980	8880	5123

```
df_IC.plot(kind='line')
```

```
<Axes: >
```



QUESTION :

Create bubble plots of immigration from China and India to visualize any difference with time from 1980-2013.

```
### type your answer here
df_IC = df_Cnd_Img.loc[['India', 'China'], '1980':'2013']
df_IC.head()
```

Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	...	28235	36210	33848	28742
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	...	36619	42584	33518	27642

2 rows × 34 columns

```
df_IC = df_IC.transpose()
```

```
df_IC.index
```

```
Index(['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'],
      dtype='object')
```

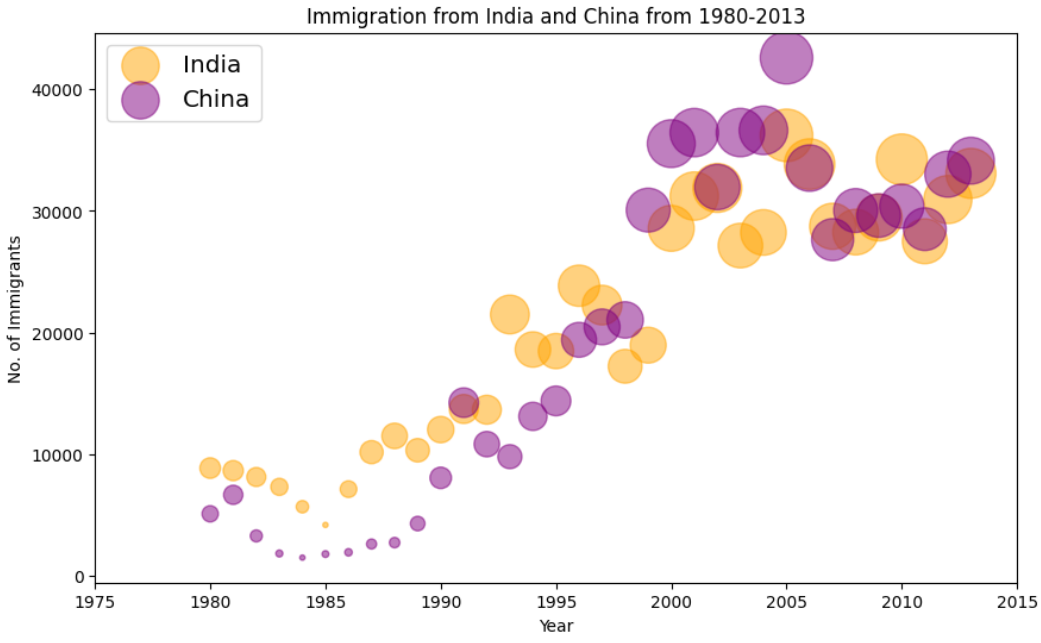
```
#converting/mapping the index (or years) of our df from string to integer
df_IC.index= map(int, df_IC.index)

#lets's label the index. this will automatically be the column name when we reset the index
df_IC.index.name = 'Year'

#reseting the index to bring the year in as column
df_IC.reset_index(inplace=True)

df_IC.head()
```

	Country	Year	India	China
0		1980	8880	5123

```
#normalize Brazil data
norm_India = (df_IC['India']- df_IC['India'].min()) / (df_IC['India'].max()-df_IC['India'].min())

#normalize Argentina data
norm_China = (df_IC['China']- df_IC['China'].min()) / (df_IC['China'].max()-df_IC['China'].min())
```

4		1984	5704	1527

```
#Brazil

ax0 = df_IC.plot(kind='scatter',
                 x='Year',
                 y='India',
                 figsize=(10,6),
                 alpha=0.5,
                 color='orange',
                 s= norm_India * 1000 + 10,
                 xlim=(1975, 2015)
                )

#Argentina
ax1 = df_IC.plot(kind='scatter',
                 x='Year',
                 y='China',
                 figsize=(10,6),
                 alpha=0.5,
                 color='purple',
                 s= norm_China * 1000 + 10,
                 ax = ax0
                )

ax0.set_ylabel("No. of Immigrants")
ax0.set_title('Immigration from India and China from 1980-2013')
ax0.legend(['India', 'China'], loc = 'upper left', fontsize='x-large')
plt.show()


#around year 1999, when brazilian curency dropped...no. of immigrants moving to canada increases as there are financial crisis going on i
#we can see that around 2010 ; there is a peak increase in no. of immigrants from Brazil
#NOTE : larger the bubble ,the  more no. of immigrants in that year
```



QUESTION :

Compare the trend of top 5 countries that contributed the most to immigration to Canada.

#We will also add a 'Total' column that sums up the total immigrants by country over the entire period 1980 - 2013, as follows:

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

```
<ipython-input-35-d31dc5743041>:3: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') i
  df_Cnd_Img['Total'] = df_Cnd_Img.sum(axis=1)
```

```
df_Cnd_Img.head(10)
```

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1
Angola	Africa	Middle Africa	Developing regions	1	3	6	6	4	3	5	...	295	184
Antigua and Barbuda	Latin America and the Caribbean	Caribbean	Developing regions	0	0	0	0	42	52	51	...	24	32
Argentina	Latin America and the Caribbean	South America	Developing regions	368	426	626	241	237	196	213	...	1153	847
Armenia	Asia	Western Asia	Developing regions	0	0	0	0	0	0	0	...	224	218
Australia	Oceania	Australia and New Zealand	Developed regions	702	639	484	317	317	319	356	...	909	875

```
top_5_countries = df_Cnd_Img.sort_values(by = 'Total', axis=0, ascending=False).head(5)
```

```
top_5_countries
```

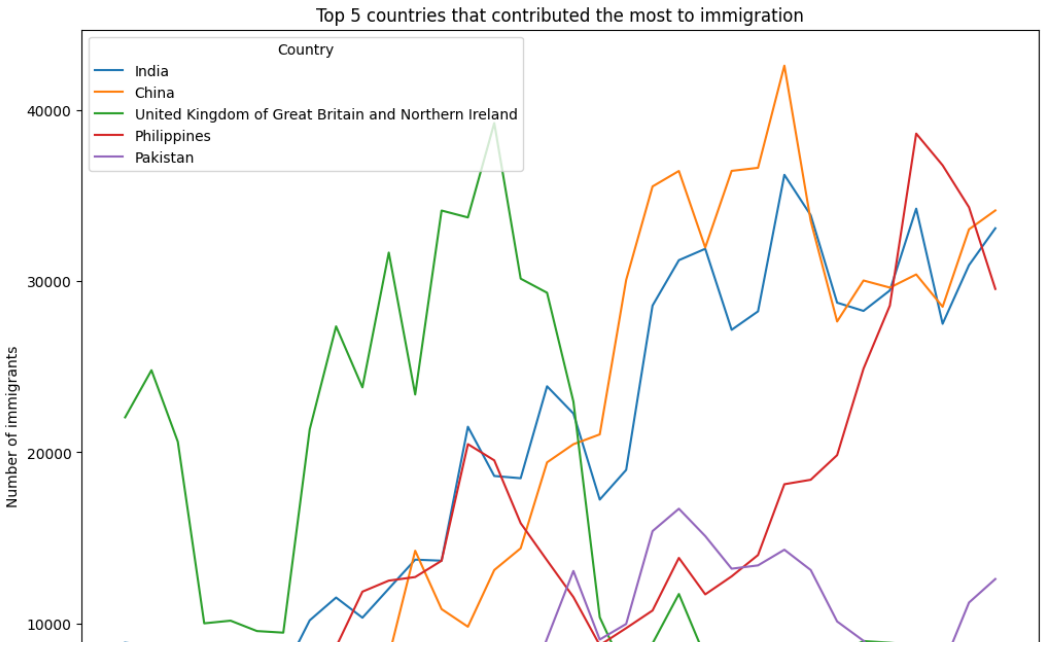
Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	
India	Asia	Southern Asia	Developing regions	8880	8670	8147	7338	5704	4211	7150	...	36210	3
China	Asia	Eastern Asia	Developing regions	5123	6682	3308	1863	1527	1816	1960	...	42584	3
United Kingdom of Great Britain and Northern Ireland	Europe	Northern Europe	Developed regions	22045	24796	20620	10015	10170	9564	9470	...	7258	
Philippines	Asia	South-Eastern Asia	Developing regions	6051	5921	5249	4562	3801	3150	4166	...	18139	1
Pakistan	Asia	Southern Asia	Developing regions	978	972	1201	900	668	514	691	...	14314	1

```
top_5_countries = top_5_countries.drop(['Continent', 'Region', 'DevName', 'Total'], axis=1).transpose()
```

```
top_5_countries
```

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
1985	4211	1816	9564	3150	514
1986	7150	1960	9470	4166	691
1987	10189	2643	21337	7360	1072
1988	11522	2758	27359	8639	1334
1989	10343	4323	23795	11865	2261
1990	12041	8076	31668	12509	2470
1991	13734	14255	23380	12718	3079
1992	13673	10846	34123	13670	4071
1993	21496	9817	33720	20479	4777
1994	18620	13128	39231	19532	4666
1995	18489	14398	30145	15864	4994
1996	23859	19415	29322	13692	9125
1997	22268	20475	22965	11549	13073
1998	17241	21049	10367	8735	9068
1999	18974	30069	7045	9734	9979
2000	28572	35529	8840	10763	15400
2001	31223	36434	11728	13836	16708
2002	31889	31961	8046	11707	15110
2003	27155	36439	6797	12758	13205
2004	28235	36619	7533	14004	13399
2005	36210	42584	7258	18139	14314
2006	33848	33518	7140	18400	13127
2007	28742	27642	8216	19837	10124
2008	28261	30037	8979	24887	8994
2009	29456	29622	8876	28573	7217
2010	34235	30391	8724	38617	6811

```
top_5_countries.plot(kind='line', figsize=(12,10))


plt.title('Top 5 countries that contributed the most to immigration')
plt.ylabel('Number of immigrants')
plt.xlabel('Years', loc = 'left', fontsize='x-large')

plt.show() # need this line to show the updates made to the figure
```



QUESTION :

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

```
df_Cnd_Img
```

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
Viet Nam	Asia	South-Eastern Asia	Developing regions	1191	1829	2162	3404	7583	5907	2741	...	1852	3153
Western Sahara	Africa	Northern Africa	Developing regions	0	0	0	0	0	0	0	...	0	1

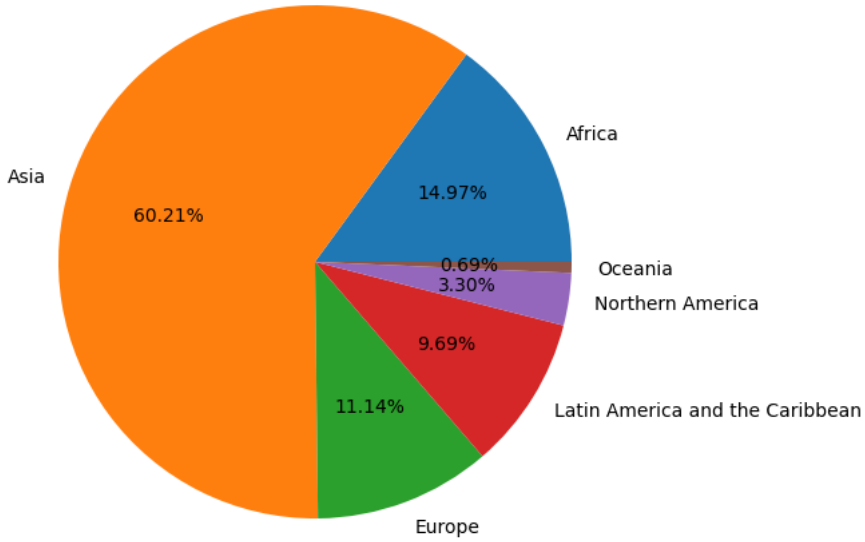
```
Img_via_continent = df_Cnd_Img[['Continent', '2013']].groupby('Continent').sum().reset_index()
```

```
Img_via_continent
```

	Continent	2013
0	Africa	38543
1	Asia	155075
2	Europe	28691
3	Latin America and the Caribbean	24950
4	Northern America	8503
5	Oceania	1775

```
Img_via_continent = Img_via_continent.rename(columns={'2013': 'year2013'})
```

```
#setting Labels and Values

my_labels = Img_via_continent.Continent
my_values = Img_via_continent.year2013


#visualizing the pie chart

fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
ax.axis('equal')

ax.pie(my_values, labels = my_labels, autopct='%1.2f%%')
plt.show()

#plot shows that -> Continent with highest no. of immigrants is : Asia with 60.21%
```



QUESTION :

Create a box plot to visualize the distribution of top 15 countries (on basis of total immigration) grouped by the decades 1980s, 1990s, 2000s .

```
top_15 = df_Cnd_Img.sort_values(by = 'Total', axis=0, ascending=False).head(15)

top_15
```

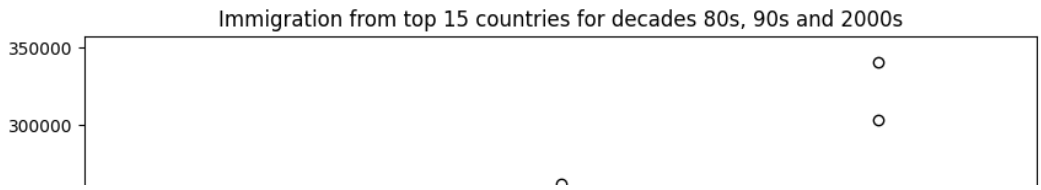
Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005
India	Asia	Southern Asia	Developing regions	8880	8670	8147	7338	5704	4211	7150	...	36210
China	Asia	Eastern Asia	Developing regions	5123	6682	3308	1863	1527	1816	1960	...	42584
United Kingdom of Great Britain and Northern Ireland	Europe	Northern Europe	Developed regions	22045	24796	20620	10015	10170	9564	9470	...	7258
Philippines	Asia	South-Eastern Asia	Developing regions	6051	5921	5249	4562	3801	3150	4166	...	18139
Pakistan	Asia	Southern Asia	Developing regions	978	972	1201	900	668	514	691	...	14314
United	Northern	Northern	Developed									

```
# create a list of all years in decades 80's, 90's, and 00's
years_80s = list(map(str, range(1980, 1990)))
years_90s = list(map(str, range(1990, 2000)))
years_00s = list(map(str, range(2000, 2010)))
```

```
# slice the original dataframe df_Cnd_Img to create a series for each decade
df_80s = top_15.loc[:, years_80s].sum(axis=1)
df_90s = top_15.loc[:, years_90s].sum(axis=1)
df_00s = top_15.loc[:, years_00s].sum(axis=1)
 # merge the three series into a new data frame
new_df = pd.DataFrame({'1980s': df_80s, '1990s': df_90s, '2000s':df_00s})

# display dataframe
new_df.head()
```

Country	1980s	1990s	2000s
India	82154	180395	303591
China	32003	161528	340385
United Kingdom of Great Britain and Northern Ireland	179171	261966	83413
Philippines	60764	138482	172904
Pakistan	10591	65302	127598

```
import matplotlib.pyplot as plt
#The correct answer is:
new_df.plot(kind='box', figsize=(10, 6))

plt.title('Immigration from top 15 countries for decades 80s, 90s and 2000s')

plt.show()
```



Note how the box plot differs from the summary table created. The box plot scans the data and identifies the outliers. In order to be an outlier, the data value must be:

- larger than Q3 by at least 1.5 times the interquartile range (IQR), or,
- smaller than Q1 by at least 1.5 times the IQR.

Let's look at decade 2000s as an example:

- Q1 (25%) = 36,101.5
- Q3 (75%) = 105,505.5
- IQR = Q3 - Q1 = 69,404

Using the definition of outlier, any value that is greater than Q3 by 1.5 times IQR will be flagged as outlier.

Outlier > 105,505.5 + (1.5 * 69,404)
Outlier > 209,611.5

```
# let's check how many entries fall above the outlier threshold
new_df[new_df['2000s']> 209611.5]
```

Country	1980s	1990s	2000s
India	82154	180395	303591
China	32003	161528	340385

China and India are both considered as outliers since their population for the decade exceeds 209,611.5.

QUESTION :

Create scatter plot of total immigration from Denmark, Sweden, Norway to canada from 1980 to 2013.

```
df_Cnd_Img
```

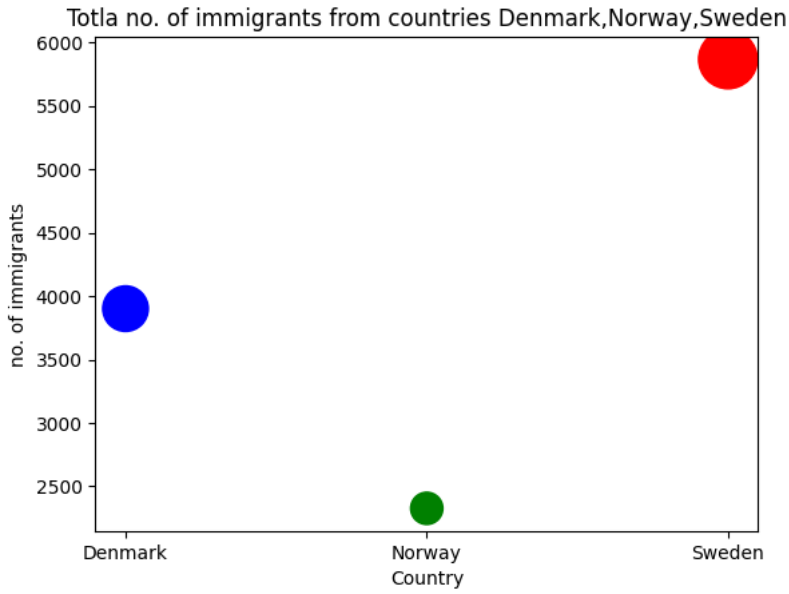
Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
Viet Nam	Asia	South-Eastern Asia	Developing regions	1191	1829	2162	3404	7583	5907	2741	...	1852	3153
Western Sahara	Africa	Northern Africa	Developing regions	0	0	0	0	0	0	0	...	0	1

```
Img_from_DNS =  df_Cnd_Img.loc[['Denmark', 'Norway', 'Sweden'], 'Total'].reset_index()
```

```
Img_from_DNS
```

	Country	Total
0	Denmark	3901
1	Norway	2327

```
country= Img_from_DNS['Country']
total_img = Img_from_DNS['Total']
colors = ['blue', 'green', 'red']
sizes = [600, 300, 1000]
plt.scatter(country, total_img, c=colors, s=sizes)
plt.title("Totla no. of immigrants from countries Denmark,Norway,Sweden")
plt.xlabel('Country')
plt.ylabel('no. of immigrants')
plt.show()
```



QUESTION :

Create bubble plot to visualize no. of immigrants from Brazil and argentina from year 1980 -2013.

```
### type your answer here
df_AB = df_Cnd_Img.loc[['Argentina', 'Brazil'], '1980':'2013']
df_AB.head()
```

Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007	2008
Argentina	368	426	626	241	237	196	213	519	374	538	...	1591	1153	847	620	540
Brazil	211	220	192	139	145	130	205	244	394	650	...	917	969	1181	1746	2138

2 rows × 34 columns

```
df_AB = df_AB.transpose()
```

```
df_AB.index
```

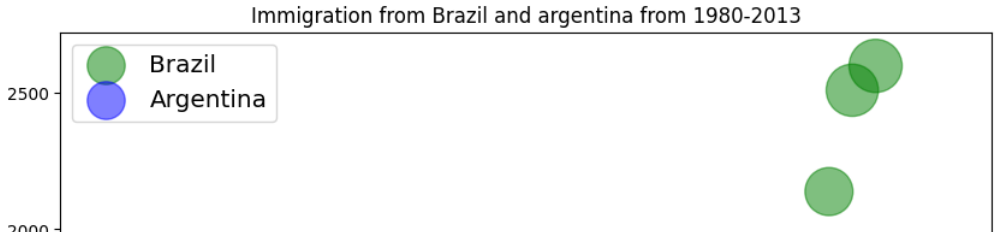
```
Index(['1980', '1981', '1982', '1983', '1984', '1985', '1986', '1987', '1988',
       '1989', '1990', '1991', '1992', '1993', '1994', '1995', '1996', '1997',
       '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006',
       '2007', '2008', '2009', '2010', '2011', '2012', '2013'],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

```
#converting/mapping the index (or years) of our df from string to integer
df_AB.index= map(int, df_AB.index)

#lets's label the index. this will automatically be the column name when we reset the index
df_AB.index.name = 'Year'

#reseting the index to bring the year in as column
df_AB.reset_index(inplace=True)
```

```
df_AB.head()
```

	Country	Year	Argentina	Brazil
0		1980	368	211
1		1981	426	220
2		1982	626	192
3		1983	241	139
4		1984	237	145

```
#normalize Brazil data
norm_brazil = (df_AB['Brazil']- df_AB['Brazil'].min()) / (df_AB['Brazil'].max()-df_AB['Brazil'].min())

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

#If we want to make two bubble charts in one plot, we should be using ax parameter.
#We will also pass weights using s parameter. Given that noralized weights are in range 0-1; they will not be visible on plot.
#Therefore we will multiply weights by 1000 to scale it up on the graph, and, add 10 to compensate for the min value.


#Brazil

ax0 = df_AB.plot(kind='scatter',
                 x='Year',
                 y='Brazil',
                 figsize=(10,8),
                 alpha=0.5,
                 color='green',
                 s= norm_brazil * 1000 + 10,
                 xlim=(1975, 2015)
                )

#Argentina
ax1 = df_AB.plot(kind='scatter',
                 x='Year',
                 y='Argentina',
                 figsize=(10,8),
                 alpha=0.5,
                 color='blue',
                 s= norm_brazil * 1000 + 10,
                 ax = ax0
                )

ax0.set_ylabel("No. of Immigrants")
ax0.set_title('Immigration from Brazil and argentina from 1980-2013')
ax0.legend(['Brazil', 'Argentina'], loc = 'upper left', fontsize='x-large')
plt.show()
```



Around year 1999, when brazilian curency dropped...no. of immigrants moving to canada increases as there are financial crisis going on in Brazil and we can see that around 2010 ; there is a peak increase in no. of immigrants from Brazil NOTE : larger the bubble ,the more no. of immigrants in that year