Area Plots, Histograms, and Bar Plots

Importing required libraries

In [27]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Loading data

Note: All steps that are performed below are explain in detail in Tutorial

In [7]:
```
df = pd.read_excel(
    'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-Skil
    sheet_name='Canada by Citizenship',
    skiprows=range(20),
    skipfooter=2)

print('Data read into a pandas dataframe!')
```

```
Data read into a pandas dataframe!
```

In [8]:
```
# in pandas axis=0 represents rows (default) and axis=1 represents columns.
df.drop(['AREA','REG','DEV','Type','Coverage'], axis=1, inplace=True)
df.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':'Region'}, inplace=True)
df['Total'] = df.sum(axis=1)
df.set_index('Country', inplace=True)
```

```
C:\Users\Meer Moazzam\AppData\Local\Temp\ipykernel_8116\3820691460.py:4: FutureWarning: Dropping of nuisance co
lumns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise Typ
eError.  Select only valid columns before calling the reduction.
  df['Total'] = df.sum(axis=1)
```

In [9]:
```
df.head()
```

Out[9]:

	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	201
Country																			
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	1758	2203	263
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856	702	560	716	561	539	62
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	4752	4325	377
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1	0	0	0	0	0	
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1	1	0	0	0	0	

5 rows × 38 columns

Area Plots

In the last module, we created a line plot that visualized the top 5 countries that contribued the most immigrants to Canada from 1980 to 2013. With a little modification to the code, we can visualize this plot as a cumulative plot, also knows as a **Stacked Line Plot** or **Area plot**.

In [28]:
```
df.sort_values(['Total'], ascending=False, axis=0, inplace=True)

# get the top 5 entries
df_top5 = df.head()
```

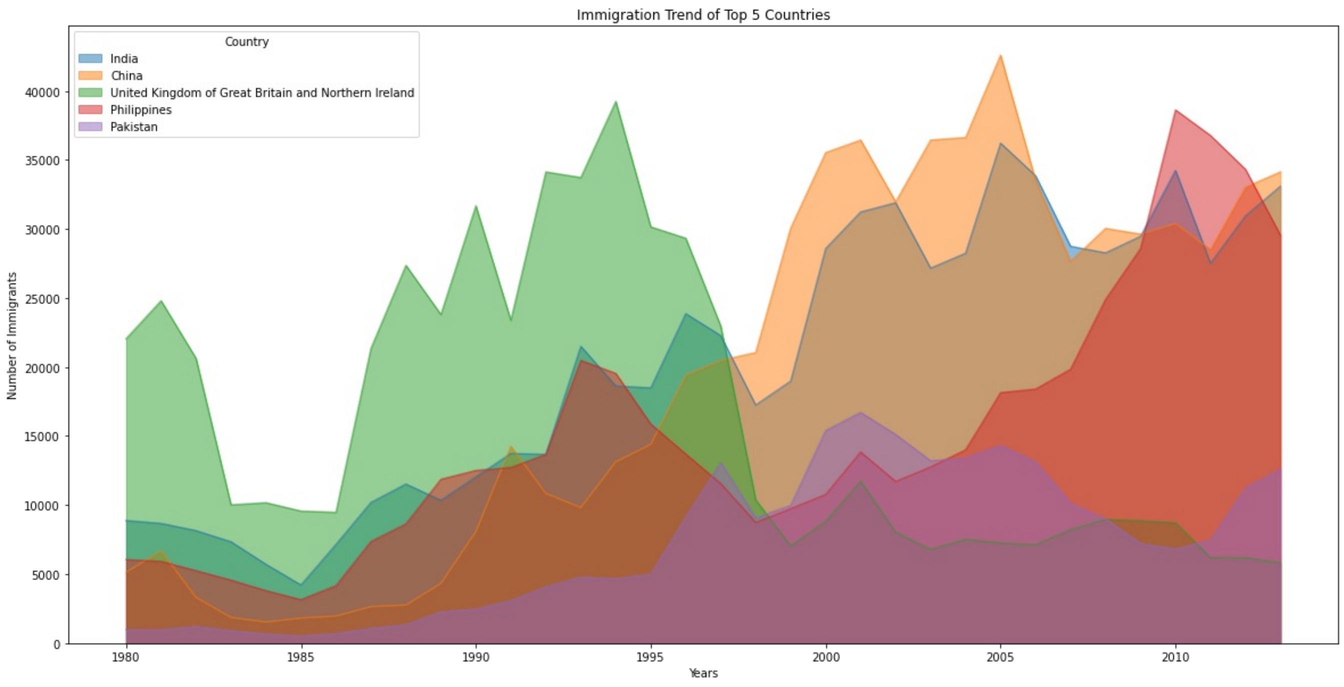
```
years=list(range(1980,2014))
# transpose the dataframe
df_top5 = df_top5[years].transpose()

df_top5.head()
```

Out[28]:

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

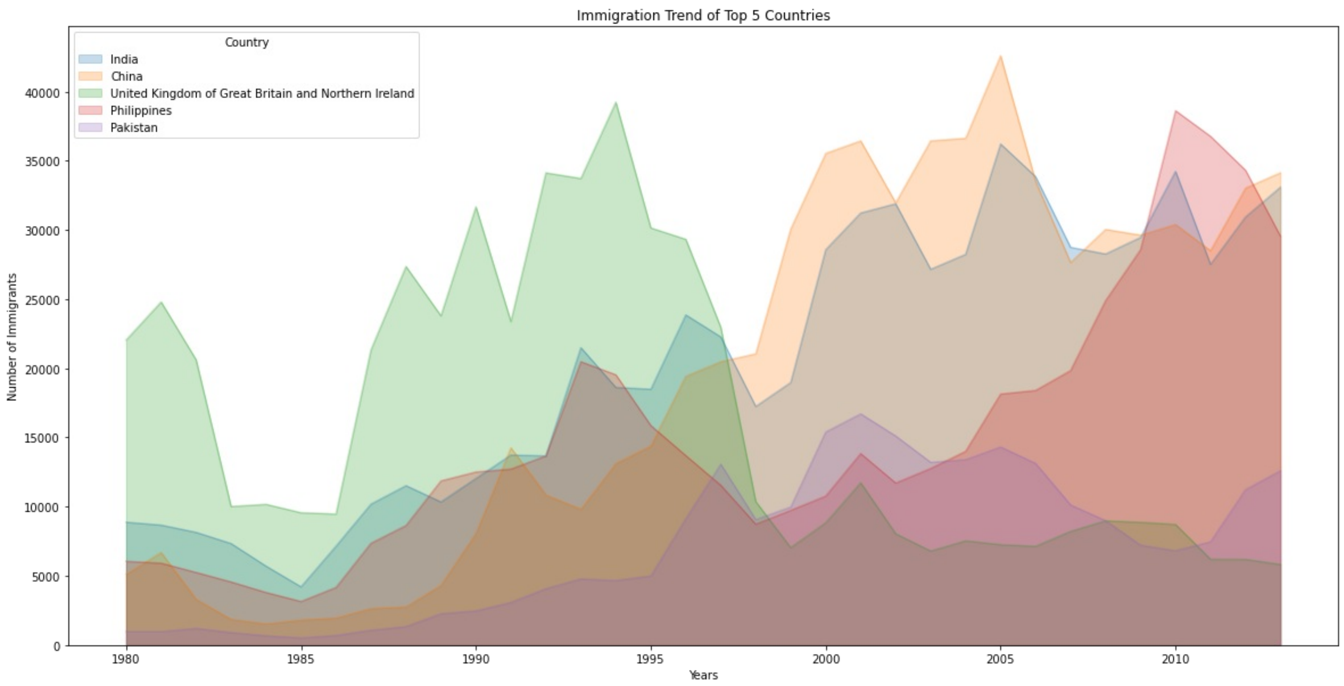
Area plots are stacked by default. And to produce a stacked area plot, each column must be either all positive or all negative values (any `NaN`, i.e. not a number, values will default to 0). To produce an unstacked plot, set parameter `stacked` to value `False`.

In [29]:
```
# let's change the index values of df_top5 to type integer for plotting
df_top5.index = df_top5.index.map(int)
df_top5.plot(kind='area',
             stacked=False,
             figsize=(20, 10))  # pass a tuple (x, y) size

plt.title('Immigration Trend of Top 5 Countries')
plt.ylabel('Number of Immigrants')
plt.xlabel('Years')

plt.show()
```



The unstacked plot has a default transparency (alpha value) at 0.5. We can modify this value by passing in the `alpha` parameter.

In [30]:
```
df_top5.plot(kind='area',
             alpha=0.25,  # 0 - 1, default value alpha = 0.5
             stacked=False,
             figsize=(20, 10))

plt.title('Immigration Trend of Top 5 Countries')
plt.ylabel('Number of Immigrants')
plt.xlabel('Years')

plt.show()
```



Question: Use the scripting layer to create a stacked area plot of the 5 countries that contributed the least to immigration to Canada **from** 1980 to 2013. Use a transparency value of 0.45.

In [58]:
```
### type your answer here
```

▶ Click here for a sample python solution

Histograms

A histogram is a way of representing the *frequency* distribution of numeric dataset. The way it works is it partitions the x-axis into *bins*, assigns each data point in our dataset to a bin, and then counts the number of data points that have been assigned to each bin. So the y-axis is the frequency or the number of data points in each bin. Note that we can change the bin size and usually one needs to tweak it so that the distribution is displayed nicely.

Question: What is the frequency distribution of the number (population) of new immigrants from the various countries to Canada in 2013?

Before we proceed with creating the histogram plot, let's first examine the data split into intervals. To do this, we will us **Numpy**'s `histrogram` method to get the bin ranges and frequency counts as follows:

In [31]:
```
# let's quickly view the 2013 data

df[2013].head()
```

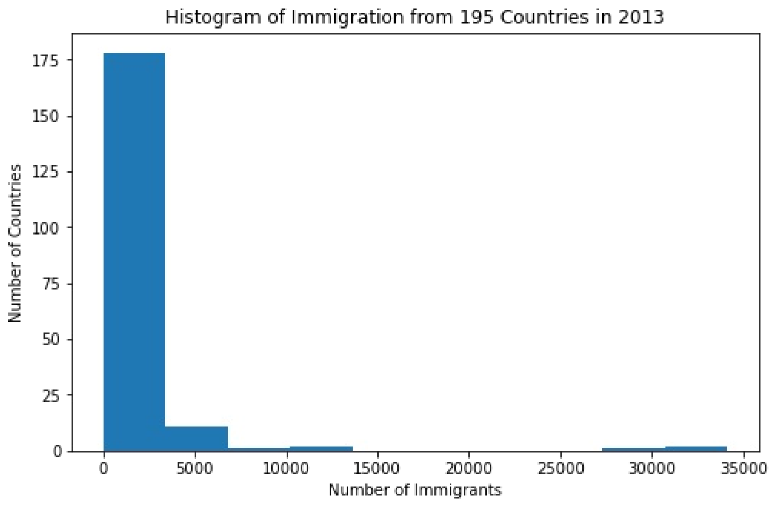
Out[31]:
```
Country
India                                                   33087
China                                                   34129
United Kingdom of Great Britain and Northern Ireland     5827
Philippines                                             29544
Pakistan                                                12603
Name: 2013, dtype: int64
```

In [32]:
```
# np.histogram returns 2 values
count, bin_edges = np.histogram(df[2013])

print(count) # frequency count
print(bin_edges) # bin ranges, default = 10 bins
```

```
[178  11   1   2   0   0   0   0   1   2]
[    0.    3412.9  6825.8 10238.7 13651.6 17064.5 20477.4 23890.3 27303.2
 30716.1 34129. ]
```

We can easily graph this distribution by passing `kind=hist` to `plot()`.

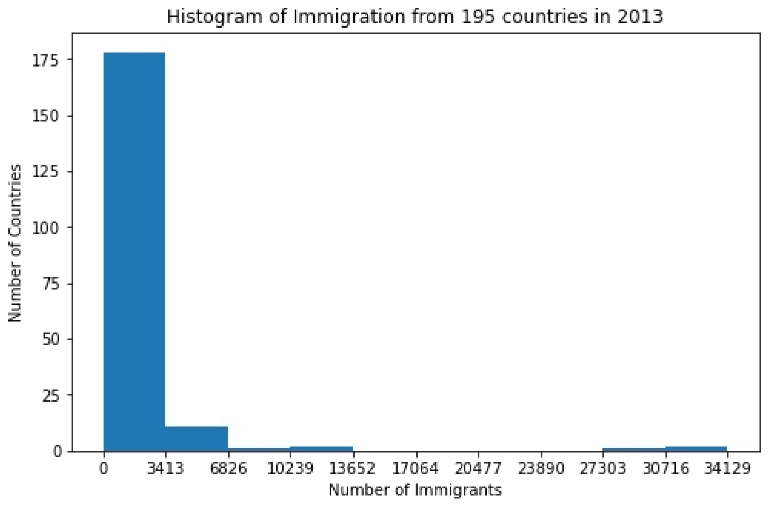
In [33]:
```
df[2013].plot(kind='hist', figsize=(8, 5))

# add a title to the histogram
plt.title('Histogram of Immigration from 195 Countries in 2013')
# add y-label
plt.ylabel('Number of Countries')
# add x-label
plt.xlabel('Number of Immigrants')
plt.show()
```



In the above plot, the x-axis represents the population range of immigrants in intervals of 3412.9. The y-axis represents the number of countries that contributed to the aforementioned population.

Notice that the x-axis labels do not match with the bin size. This can be fixed by passing in a `xticks` keyword that contains the list of the bin sizes, as follows:

In [35]:
```
# 'bin_edges' is a list of bin intervals
count, bin_edges = np.histogram(df[2013])

df[2013].plot(kind='hist', figsize=(8, 5), xticks=bin_edges)

plt.title('Histogram of Immigration from 195 countries in 2013') # add a title to the histogram
plt.ylabel('Number of Countries') # add y-label
plt.xlabel('Number of Immigrants') # add x-label

plt.show()
```



We can also plot multiple histograms on the same plot. For example, let's try to answer the following questions using a histogram.

Question: What is the immigration distribution for Pakistan, China, and India for years 1980 - 2013?

In [38]:
```
# let's quickly view the dataset
df.loc[['Pakistan', 'China', 'India'], years]
```

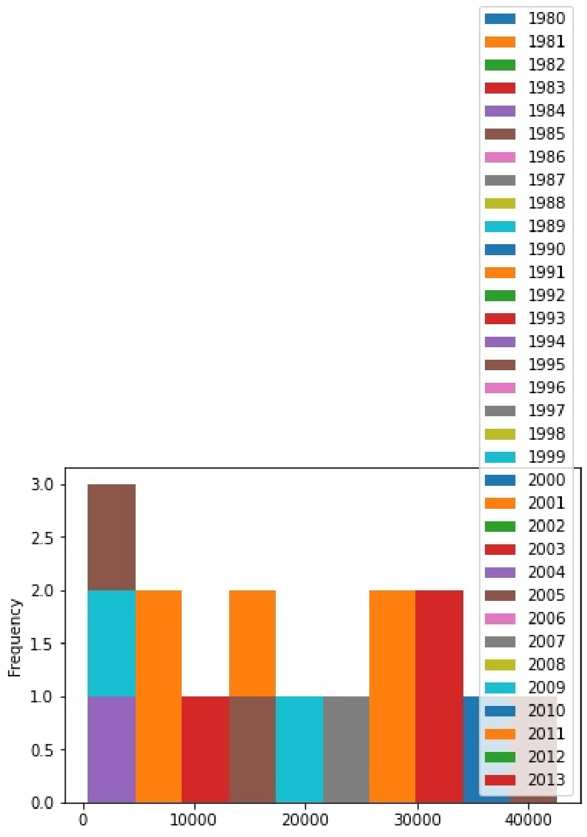
Out[38]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007	2008	2009	2010	2011	20
Country																				
Pakistan	978	972	1201	900	668	514	691	1072	1334	2261	...	13399	14314	13127	10124	8994	7217	6811	7468	112
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	...	36619	42584	33518	27642	30037	29622	30391	28502	330
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	...	28235	36210	33848	28742	28261	29456	34235	27509	30

3 rows × 34 columns

In [39]:
```
df.loc[['Pakistan', 'China', 'India'], years].plot.hist()
```

Out[39]:
```
<AxesSubplot:ylabel='Frequency'>
```



That does not look right!

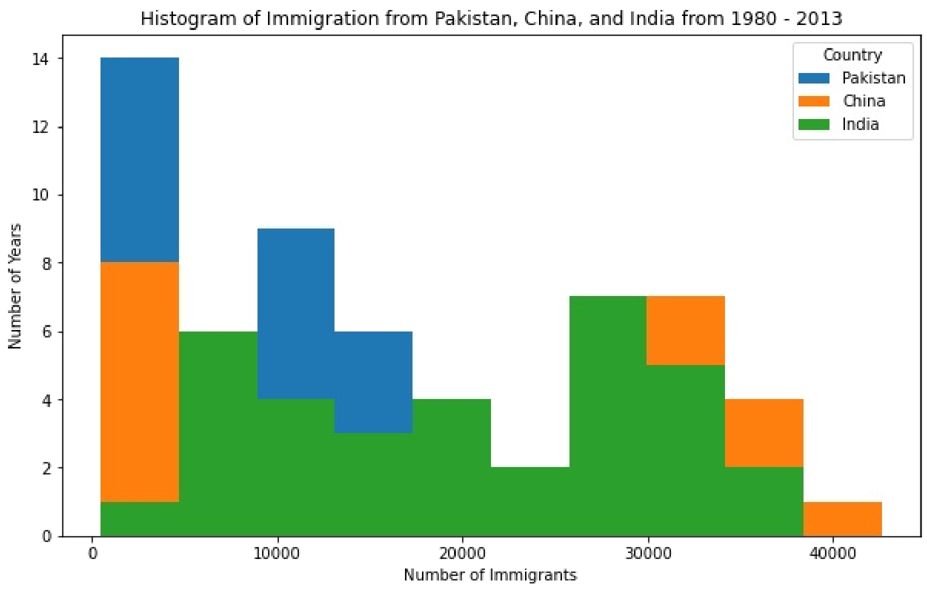
Don't worry, you'll often come across situations like this when creating plots. The solution often lies in how the underlying dataset is structured.

Instead of plotting the population frequency distribution of the population for the 3 countries, *pandas* instead plotted the population frequency distribution for the `years`.

This can be easily fixed by first transposing the dataset, and then plotting as shown below.

In [41]:
```
# transpose dataframe

df_new=df.loc[['Pakistan', 'China', 'India'], years].transpose()
```

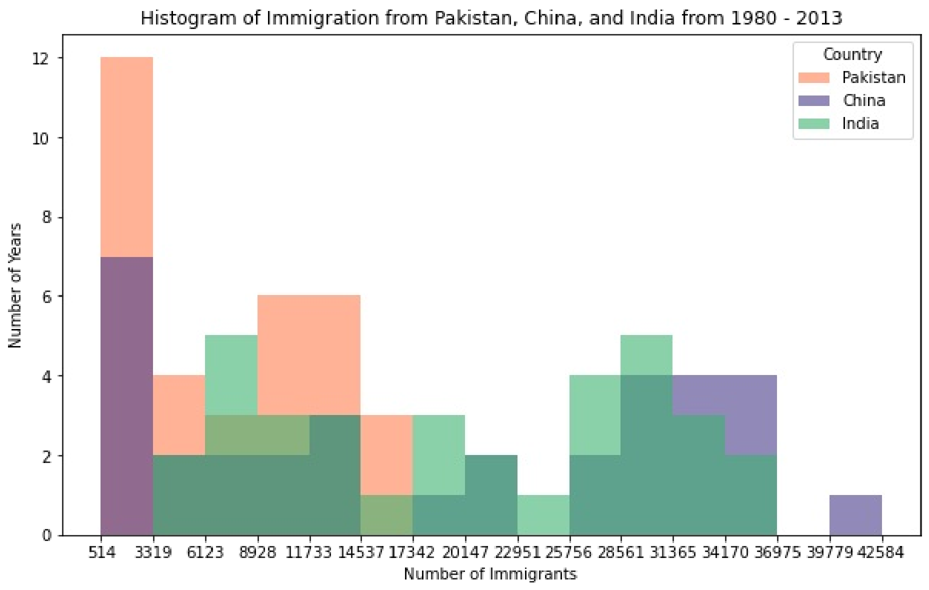
In [43]:
```
# generate histogram
df_new.plot(kind='hist', figsize=(10, 6))

plt.title('Histogram of Immigration from Pakistan, China, and India from 1980 - 2013')
plt.ylabel('Number of Years')
plt.xlabel('Number of Immigrants')

plt.show()
```



Let's make a few modifications to improve the impact and aesthetics of the previous plot:

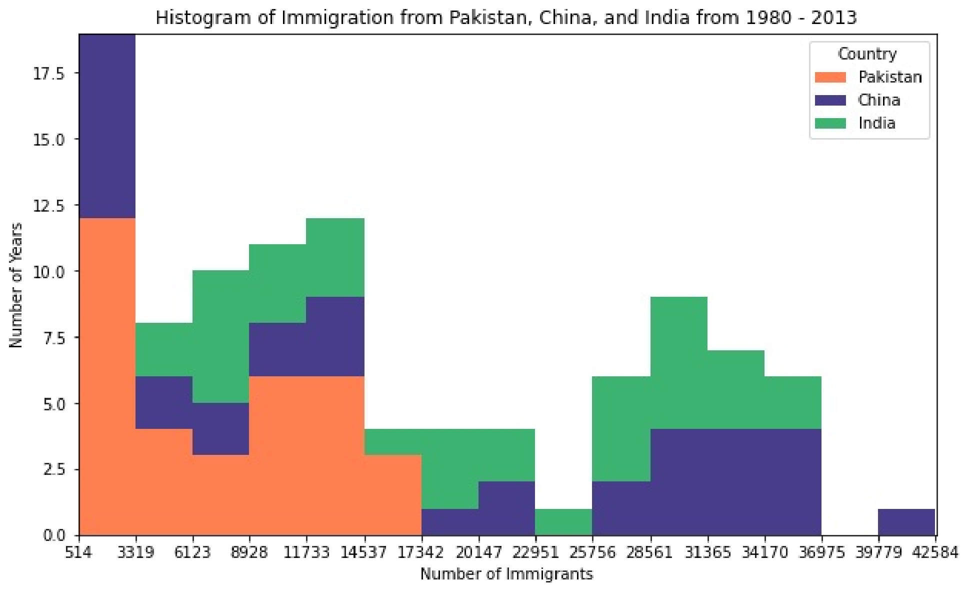
- increase the bin size to 15 by passing in `bins` parameter;
- set transparency to 60% by passing in `alpha` parameter;
- label the x-axis by passing in `x-label` parameter;
- change the colors of the plots by passing in `color` parameter.

In [47]:
```
# let's get the x-tick values
count, bin_edges = np.histogram(df_new, 15)

# un-stacked histogram
df_new.plot(kind ='hist',
          figsize=(10, 6),
          bins=15,
          alpha=0.6,
          xticks=bin_edges,
          color=['coral', 'darkslateblue', 'mediumseagreen']
         )

plt.title('Histogram of Immigration from Pakistan, China, and India from 1980 - 2013')
plt.ylabel('Number of Years')
plt.xlabel('Number of Immigrants')

plt.show()
```



If we do not want the plots to overlap each other, we can stack them using the `stacked` parameter. Let's also adjust the min and max x-axis labels to remove the extra gap on the edges of the plot. We can pass a tuple (min,max) using the `xlim` paramater, as show below.

In [48]:
```
count, bin_edges = np.histogram(df_new, 15)
xmin = bin_edges[0] - 10   #  first bin value is 31.0, adding buffer of 10 for aesthetic purposes
xmax = bin_edges[-1] + 10  #  last bin value is 308.0, adding buffer of 10 for aesthetic purposes

# stacked Histogram
df_new.plot(kind='hist',
          figsize=(10, 6),
          bins=15,
          xticks=bin_edges,
          color=['coral', 'darkslateblue', 'mediumseagreen'],
          stacked=True,
          xlim=(xmin, xmax)
         )

plt.title('Histogram of Immigration from Pakistan, China, and India from 1980 - 2013')
plt.ylabel('Number of Years')
plt.xlabel('Number of Immigrants')

plt.show()
```



Question: Use the scripting layer to display the immigration distribution for Greece, Albania, and Bulgaria for years 1980 - 2013? Use an overlapping plot with 15 bins and a transparency value of 0.35.

```
In [58]: ### type your answer here
```

▶ Click here for a sample python solution

Bar Charts (Dataframe)

A bar plot is a way of representing data where the *length* of the bars represents the magnitude/size of the feature/variable. Bar graphs usually represent numerical and categorical variables grouped in intervals.

To create a bar plot, we can pass one of two arguments via `kind` parameter in `plot()`:

- `kind=bar` creates a *vertical* bar plot
- `kind=barh` creates a *horizontal* bar plot

Vertical bar plot

In vertical bar graphs, the x-axis is used for labelling, and the length of bars on the y-axis corresponds to the magnitude of the variable being measured. Vertical bar graphs are particularly useful in analyzing time series data. One disadvantage is that they lack space for text labelling at the foot of each bar.

Let's start off by analyzing the effect of Iceland's Financial Crisis:

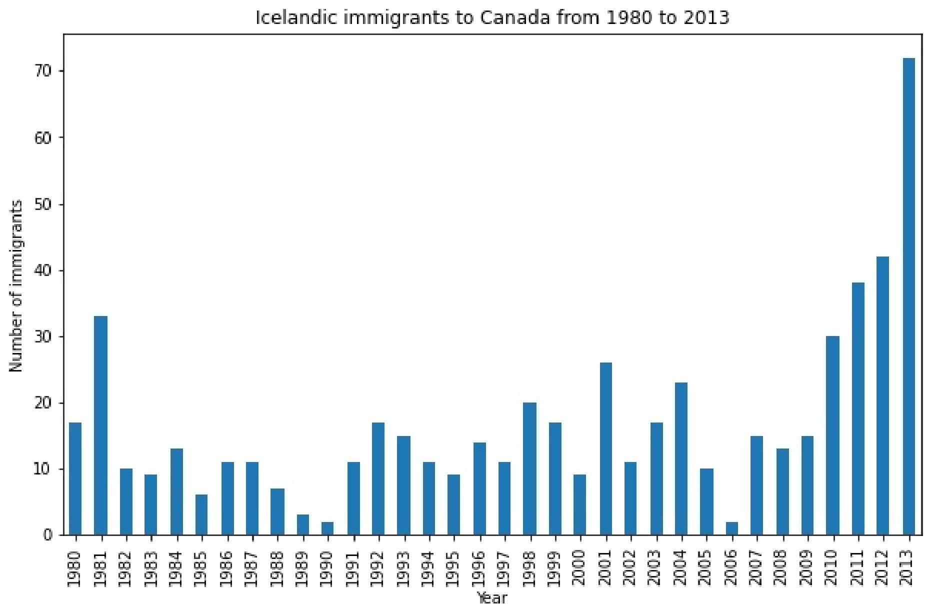
The 2008 - 2011 Icelandic Financial Crisis was a major economic and political event in Iceland. Relative to the size of its economy, Iceland's systemic banking collapse was the largest experienced by any country in economic history. The crisis led to a severe economic depression in 2008 - 2011 and significant political unrest.

Question: Let's compare the number of Icelandic immigrants (country = 'Iceland') to Canada from year 1980 to 2013.

```
In [51]: # step 1: get the data
df_iceland = df.loc['Iceland', years]
df_iceland.head()
```

```
Out[51]:
1980    17
1981    33
1982    10
1983     9
1984    13
Name: Iceland, dtype: object
```

```
In [52]: # step 2: plot data
df_iceland.plot(kind='bar', figsize=(10, 6))

plt.xlabel('Year') # add to x-label to the plot
plt.ylabel('Number of immigrants') # add y-label to the plot
plt.title('Icelandic immigrants to Canada from 1980 to 2013') # add title to the plot

plt.show()
```

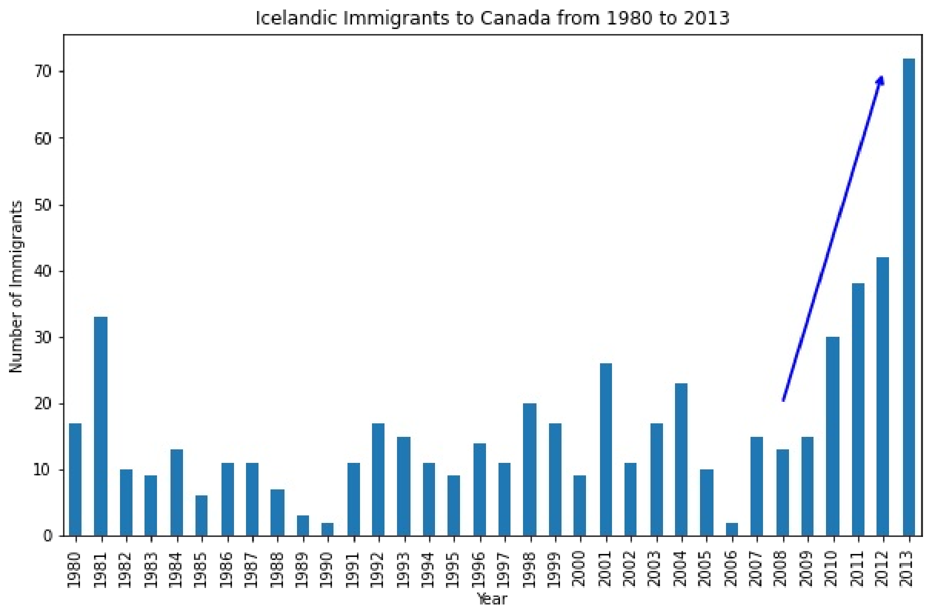


The bar plot above shows the total number of immigrants broken down by each year. We can clearly see the impact of the financial crisis; the number of immigrants to Canada started increasing rapidly after 2008.

Let's annotate this on the plot using the `annotate` method of the **scripting layer** or the **pyplot interface**. We will pass in the following parameters:

- `s` : str, the text of annotation.
- `xy` : Tuple specifying the (x,y) point to annotate (in this case, end point of arrow).
- `xytext` : Tuple specifying the (x,y) point to place the text (in this case, start point of arrow).
- `xycoords` : The coordinate system that xy is given in - 'data' uses the coordinate system of the object being annotated (default).
- `arrowprops` : Takes a dictionary of properties to draw the arrow:
  - `arrowstyle` : Specifies the arrow style, `'->'` is standard arrow.
  - `connectionstyle` : Specifies the connection type. `arc3` is a straight line.
  - `color` : Specifies color of arrow.
  - `lw` : Specifies the line width.

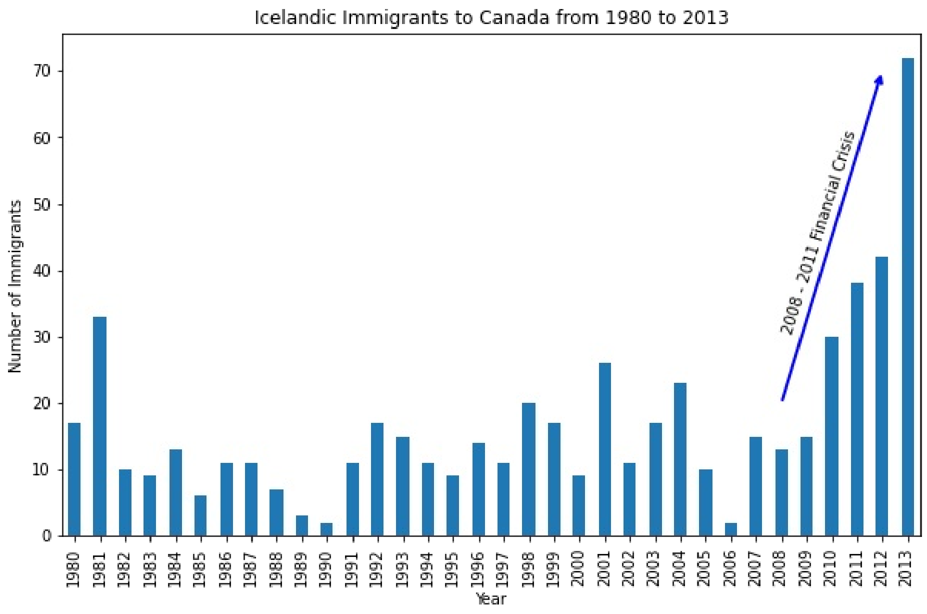
In [54]:
```
df_iceland.plot(kind='bar', figsize=(10, 6), rot=90)  # rotate the xticks(labelled points on x-axis) by 90 degr

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

# Annotate arrow
plt.annotate('',  # s: str. Will leave it blank for no text
             xy=(32, 70),  # place head of the arrow at point (year 2012 , pop 70)
             xytext=(28, 20),  # place base of the arrow at point (year 2008 , pop 20)
             xycoords='data',  # will use the coordinate system of the object being annotated
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2)
             )

plt.show()
```



Let's also annotate a text to go over the arrow. We will pass in the following additional parameters:

- `rotation` : rotation angle of text in degrees (counter clockwise)
- `va` : vertical alignment of text ['center' | 'top' | 'bottom' | 'baseline']
- `ha` : horizontal alignment of text ['center' | 'right' | 'left']

In [55]:
```python
df_iceland.plot(kind='bar', figsize=(10, 6), rot=90)

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

# Annotate arrow
plt.annotate('',  # s: str. will leave it blank for no text
             xy=(32, 70),  # place head of the arrow at point (year 2012 , pop 70)
             xytext=(28, 20),  # place base of the arrow at point (year 2008 , pop 20)
             xycoords='data',  # will use the coordinate system of the object being annotated
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2)
             )

# Annotate Text
plt.annotate('2008 - 2011 Financial Crisis',  # text to display
             xy=(28, 30),  # start the text at at point (year 2008 , pop 30)
             rotation=72.5,  # based on trial and error to match the arrow
             va='bottom',  # want the text to be vertically 'bottom' aligned
             ha='left',  # want the text to be horizontally 'left' algned.
             )

plt.show()
```



Horizontal Bar Plot

Sometimes it is more practical to represent the data horizontally, especially if you need more room for labelling the bars. In horizontal bar graphs, the y-axis is used for labelling, and the length of bars on the x-axis corresponds to the magnitude of the variable being measured. As you will see, there is more room on the y-axis to label categorical variables.

Question: Using the scripting later and the `df_can` dataset, create a *horizontal* bar plot showing the *total* number of immigrants to Canada from the top 15 countries, for the period 1980 - 2013. Label each country with the total immigrant count.

In [58]:
```python
### type your answer here
```

▶ Click here for a sample python solution

Thank you

Author

Moazzam Ali

In [ ]:
```python

```