Assignment 4

Before working on this assignment please read these instructions fully. In the submission area, you will notice that you can click the link to **Preview the Grading** for each step of the assignment. This is the criteria that will be used for peer grading. Please familiarize yourself with the criteria before beginning the assignment.

This assignment requires that you to find **at least** two datasets on the web which are related, and that you visualize these datasets to answer a question with the broad topic of **economic activity or measures** (see below) for the region of **Monmouth Junction, New Jersey, United States**, or **United States** more broadly.

You can merge these datasets with data from different regions if you like! For instance, you might want to compare **Monmouth Junction, New Jersey, United States** to Ann Arbor, USA. In that case at least one source file must be about **Monmouth Junction, New Jersey, United States**.

You are welcome to choose datasets at your discretion, but keep in mind **they will be shared with your peers**, so choose appropriate datasets. Sensitive, confidential, illicit, and proprietary materials are not good choices for datasets for this assignment. You are welcome to upload datasets of your own as well, and link to them using a third party repository such as github, bitbucket, pastebin, etc. Please be aware of the Coursera terms of service with respect to intellectual property.

Also, you are welcome to preserve data in its original language, but for the purposes of grading you should provide english translations. You are welcome to provide multiple visuals in different languages if you would like!

As this assignment is for the whole course, you must incorporate principles discussed in the first week, such as having as high data-ink ratio (Tufte) and aligning with Cairo’s principles of truth, beauty, function, and insight.

Here are the assignment instructions:

- State the region and the domain category that your data sets are about (e.g., **Monmouth Junction, New Jersey, United States** and **economic activity or measures**).
- You must state a question about the domain category and region that you identified as being interesting.
- You must provide at least two links to available datasets. These could be links to files such as CSV or Excel files, or links to websites which might have data in tabular form, such as Wikipedia pages.
- You must upload an image which addresses the research question you stated. In addition to addressing the question, this visual should follow Cairo's principles of truthfulness, functionality, beauty, and insightfulness.
- You must contribute a short (1-2 paragraph) written justification of how your visualization addresses your stated research question.

What do we mean by **economic activity or measures**? For this category you might look at the inputs or outputs to the given economy, or major changes in the economy compared to other regions.

Tips

- Wikipedia is an excellent source of data, and I strongly encourage you to explore it for new data sources.
- Many governments run open data initiatives at the city, region, and country levels, and these are wonderful resources for localized data sources.
- Several international agencies, such as the United Nations (http://data.un.org/), the World Bank (http://data.worldbank.org/), the Global Open Data Index (http://index.okfn.org/place/) are other great places to look for data.
- This assignment requires you to convert and clean datafiles. Check out the discussion forums for tips on how to do this from various sources, and share your successes with your fellow students!

Example

Looking for an example? Here's what our course assistant put together for the **Ann Arbor, MI, USA** area using **sports and athletics** as the topic. Example Solution File (./readonly/Assignment4_example.pdf)

Assignment 4: Becoming an Independent Data Scientist

```
@author: Steven Ponce
Date: May 2021
```

1. Region and Domain

```
Country: United States of America
State: New Jersey, USA
County: Middlesex County, NJ
```

2. Research Question

What can the number of residential construction permits can tell us about the New Jersey economy when compared to the rest of the US?

3. Links

The data used in this project was obtained from the SOCDS Building Permits Database.

This database contains data on permits for residential construction issued by about 21,000 jurisdictions collected in the Census Bureau's Building Permits Survey. You can create output tables at the State, County, CBSA or permit-issuing jurisdiction level.

Annual data are available from 1980 through the most recent reporting year, and may also contain imputed values.

https://socds.huduser.gov/permits (https://socds.huduser.gov/permits)

The database was query to obtain:

```
1) New Jersey and Middlesex county building permits from 1980 - current
https://github.com/poncest/Coursera-Applied-Data-Science-With-Python/blob/main/Course2/NJ%20versus%20County%20building%20permits%20data.csv

2) Rest of the country (USA) building permits from 1980 - current
https://github.com/poncest/Coursera-Applied-Data-Science-With-Python/blob/main/Course2/US%20building%20permits%20data.csv
```

To retrieve data:

- On the left side of the page, select the desired "Geography" (the "sub-criteria" displayed in the upper right frame will adjust accordingly).
- Select the desired "Periodicity" (Note that the choice of years, as found in the lower left side of the right frame, will change according to their availability in the relevant monthly and annual datasets).
- In the upper right frame, select a State or CBSA. If you wish to select counties and/or permitting jurisdictions, click on the hyperlink below the State or CBSA listbox.
- Select individual jurisdictions or use the checkboxes to select State, County, or Metropolitan Area totals, County or Jurisdiction group sums, or all jurisdictions within the Counties or Metropolitan Areas you selected. For selected County or Jurisdiction group sums, you may optionally assign a group name, which will appear as the output table title.
- Select the desired years to include in your output (lower left side of the right frame).
- Select the series you wish to display in your output (lower right side of the right frame).
- Click the "Get Data" button (lower portion of left frame) to see your output. Data are provided initially in tabular form, but a button at the bottom of the output page enables you to access the same data in comma-delimited form. Note: Data may not be available for all jurisdictions for all years.

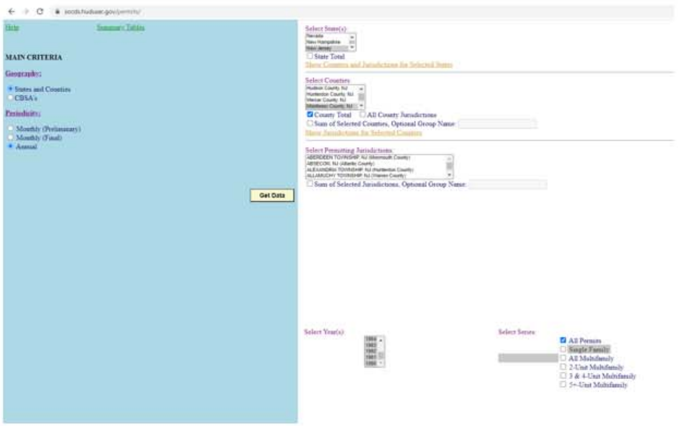
In [1]:
```
from IPython.display import Image
Image('Img/Database query.png')
```

Out[1]:

Query: SOCDS Building Permits Database

https://socds.huduser.gov/permits/

State and Country



socds.huduser.gov/permits/

Help Summary Tables

MAIN CRITERIA

Geography:

● States and Counties
○ CBSA's

Periodicity:

○ Monthly (Preliminary)
○ Monthly (Final)
● Annual

Get Data

Select State(s):
Nevada
New Hampshire
New Jersey

□ State Total

Show Counties and Jurisdictions for Selected States

Select Counties:
Hudson County, NJ
Hunterdon County, NJ
Mercer County, NJ
Middlesex County, NJ

☑ County Total □ All County Jurisdictions

□ Sum of Selected Counties, Optional Group Name:

Show Jurisdictions for Selected Counties

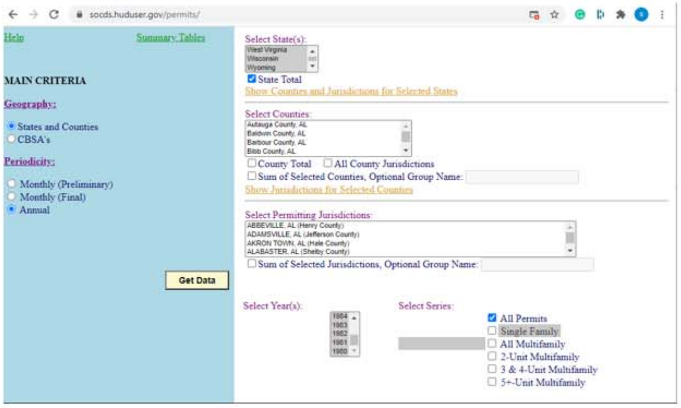
Select Permitting Jurisdictions:
ABERDEEN TOWNSHIP, NJ (Monmouth County)
ABSECON, NJ (Atlantic County)
ALEXANDRIA TOWNSHIP, NJ (Hunterdon County)
ALLAMUCHY TOWNSHIP, NJ (Warren County)

□ Sum of Selected Jurisdictions, Optional Group Name:

Select Year(s):
1984
1983
1982
1981
1980

Select Series:
☑ All Permits
□ Single Family
□ All Multifamily
□ 2-Unit Multifamily
□ 3 & 4-Unit Multifamily
□ 5+-Unit Multifamily

Country



socds.huduser.gov/permits/

Help Summary Tables

MAIN CRITERIA

Geography:

● States and Counties
○ CBSA's

Periodicity:

○ Monthly (Preliminary)
○ Monthly (Final)
● Annual

Get Data

Select State(s):
West Virginia
Wisconsin
Wyoming

☑ State Total

Show Counties and Jurisdictions for Selected States

Select Counties:
Autauga County, AL
Baldwin County, AL
Barbour County, AL
Bibb County, AL

□ County Total □ All County Jurisdictions

□ Sum of Selected Counties, Optional Group Name:

Show Jurisdictions for Selected Counties

Select Permitting Jurisdictions:
ABBEVILLE, AL (Henry County)
ADAMSVILLE, AL (Jefferson County)
AKRON TOWN, AL (Hale County)
ALABASTER, AL (Shelby County)

□ Sum of Selected Jurisdictions, Optional Group Name:

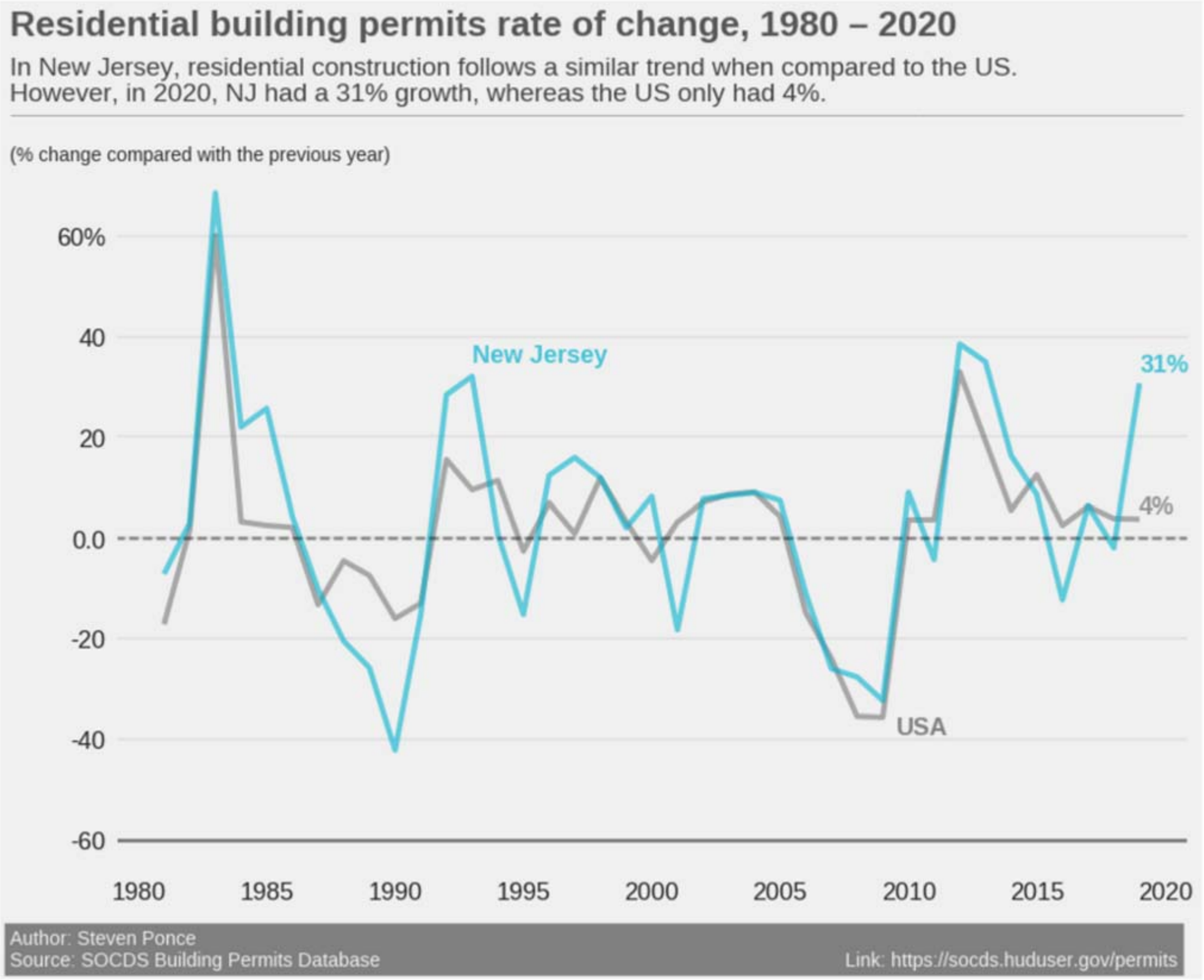
Select Year(s):
1984
1983
1982
1981
1980

Select Series:
☑ All Permits
□ Single Family
□ All Multifamily
□ 2-Unit Multifamily
□ 3 & 4-Unit Multifamily
□ 5+-Unit Multifamily

4. Final Image

In [2]:
```
from IPython.display import Image
Image('Img/Final Image.png')
```

Out[2]:



Residential building permits rate of change, 1980 – 2020

In New Jersey, residential construction follows a similar trend when compared to the US. However, in 2020, NJ had a 31% growth, whereas the US only had 4%.

(% change compared with the previous year)

Author: Steven Ponce
Source: SOCDS Building Permits Database
Link: https://socds.huduser.gov/permits

5. Discussion

The purpose of this visualization was to answer our research question - **what can the number of residential construction permits tell us about the New Jersey economy when compared to the rest of the US?**

The data used for this analysis was obtained from the SOCDS Building Permits Database. This database contains data on permits for residential construction issued by about 21,000 jurisdictions collected in the Census Bureau's Building Permits Survey.

The final figure indicates residential building permits rate of change since 1980. In New Jersey, residential construction follows a similar trend when compared to the US. However, in 2020, NJ had a 31% growth, whereas the US only had 4%.

Since the construction industry creates jobs, income, and tax revenue, therefore, construction trends are important indicators of the health of a state's economy.

Import Libraries

In [3]:

```
'''
@author:  Steven Ponce
Date:     11 May 2021
'''
# Importing important libraries
import sys
import pandas as pd
import numpy as np
import seaborn as sns
from datetime import datetime as dt
from IPython.display import display, HTML

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.style as style
style.use('fivethirtyeight')

# Hide warnings
import warnings
warnings.filterwarnings('ignore')

print('You\'re running python %s' % sys.version.split(' ')[0])
print('You\'re running matplotlib: {}'.format(matplotlib.__version__))

# The following lines adjust the granularity of reporting
pd.options.display.max_rows = 10
pd.options.display.float_format = '{:.1f}'.format
```

```
You're running python 3.6.2
You're running matplotlib: 2.0.0
```

Loading the data

In [4]:

```
# Import New Jersey and Middlesex County data
NJ = pd.read_csv('datasets/NJ versus County building permits data.csv')

# Import the rest of the US data
US = pd.read_csv('datasets/US building permits data.csv')
```

Examining and cleaning the data

In [5]:

```
'''
@author:  Steven Ponce
Date:     11 May 2021
'''
def quick_analysis(df):
    '''
    1. Print number of rows, number of columns, columns names, non-null count, and data type
    '''
    print('\n 1. Dataset Information:')
    print('-'*40)
    print(df.info())
    '''
    2. Data shape = numbers of rows and columns
    '''
    print('\n 2. Number of Rows and Columns:', df.shape)
    print('-'*40)
    '''
    3. How many null we have per columns. Python uses the keyword None to define null objects and variables.
    '''
    print('\n 3. Null Count:')
    print('-'*40)
    print(df.isnull().sum())

    return quick_analysis
    raise NotImplementedError()
```

In [6]:

```
quick_analysis(NJ);
```

```
 1. Dataset Information:
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80 entries, 0 to 79
Data columns (total 5 columns):
Location       80 non-null object
Year           80 non-null int64
Series         80 non-null object
Series Code    80 non-null int64
Permits        80 non-null int64
dtypes: int64(3), object(2)
memory usage: 3.2+ KB
None

 2. Number of Rows and Columns: (80, 5)
----------------------------------------

 3. Null Count:
----------------------------------------
Location       0
Year           0
Series         0
Series Code    0
Permits        0
dtype: int64
```

In [7]:
```
quick_analysis(US);
```

```
 1. Dataset Information:
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2027 entries, 0 to 2026
Data columns (total 5 columns):
Location       2027 non-null object
Year           2027 non-null int64
Series         2027 non-null object
Series Code    2027 non-null int64
Permits        2027 non-null int64
dtypes: int64(3), object(2)
memory usage: 79.3+ KB
None

 2. Number of Rows and Columns: (2027, 5)
----------------------------------------

 3. Null Count:
----------------------------------------
Location       0
Year           0
Series         0
Series Code    0
Permits        0
dtype: int64
```

In [8]:
```
# Lets drop unnecessary columns from the DF
NJ.drop(['Series', 'Series Code'], axis=1, inplace=True)
US.drop(['Series', 'Series Code'], axis=1, inplace=True)
```

In [9]:
```
# Remaming 'Permits' columns
NJ.rename(columns={'Permits':'Total Permits'}, inplace=True)
US.rename(columns={'Permits':'Total Permits'}, inplace=True)
```

In [10]:
```
# Select 'New Jersey' state data from the NJ dataset
State = NJ.loc[NJ['Location']=='New Jersey']
State.head()
```

Out[10]:

	Location	Year	Total Permits
0	New Jersey	1980	22270
1	New Jersey	1981	20676
2	New Jersey	1982	21297
3	New Jersey	1983	35897
4	New Jersey	1984	43787

In [11]:
```
# Select 'Middlesex County' data from the NJ dataset
County = NJ.loc[NJ['Location']=='Middlesex County']
County.head()
```

Out[11]:

	Location	Year	Total Permits
40	Middlesex County	1980	2219
41	Middlesex County	1981	2793
42	Middlesex County	1982	3565
43	Middlesex County	1983	6419
44	Middlesex County	1984	7155

In [12]:

```
# Remove 'New Jersey' state data from the US dataset
Country = US.loc[US['Location']!='New Jersey']
Country.head()
```

Out[12]:

	Location	Year	Total Permits
0	Alabama	1980	15998
1	Alabama	1981	9885
2	Alabama	1982	8732
3	Alabama	1983	17389
4	Alabama	1984	15297

In [13]:

```
print('Number of Rows and Columns: ')
print('-'*26)
print('Country:', Country.shape)
print('State:  ', State.shape)
print('County: ', County.shape)
```

```
Number of Rows and Columns: 
--------------------------
Country: (1987, 3)
State:   (40, 3)
County:  (40, 3)
```

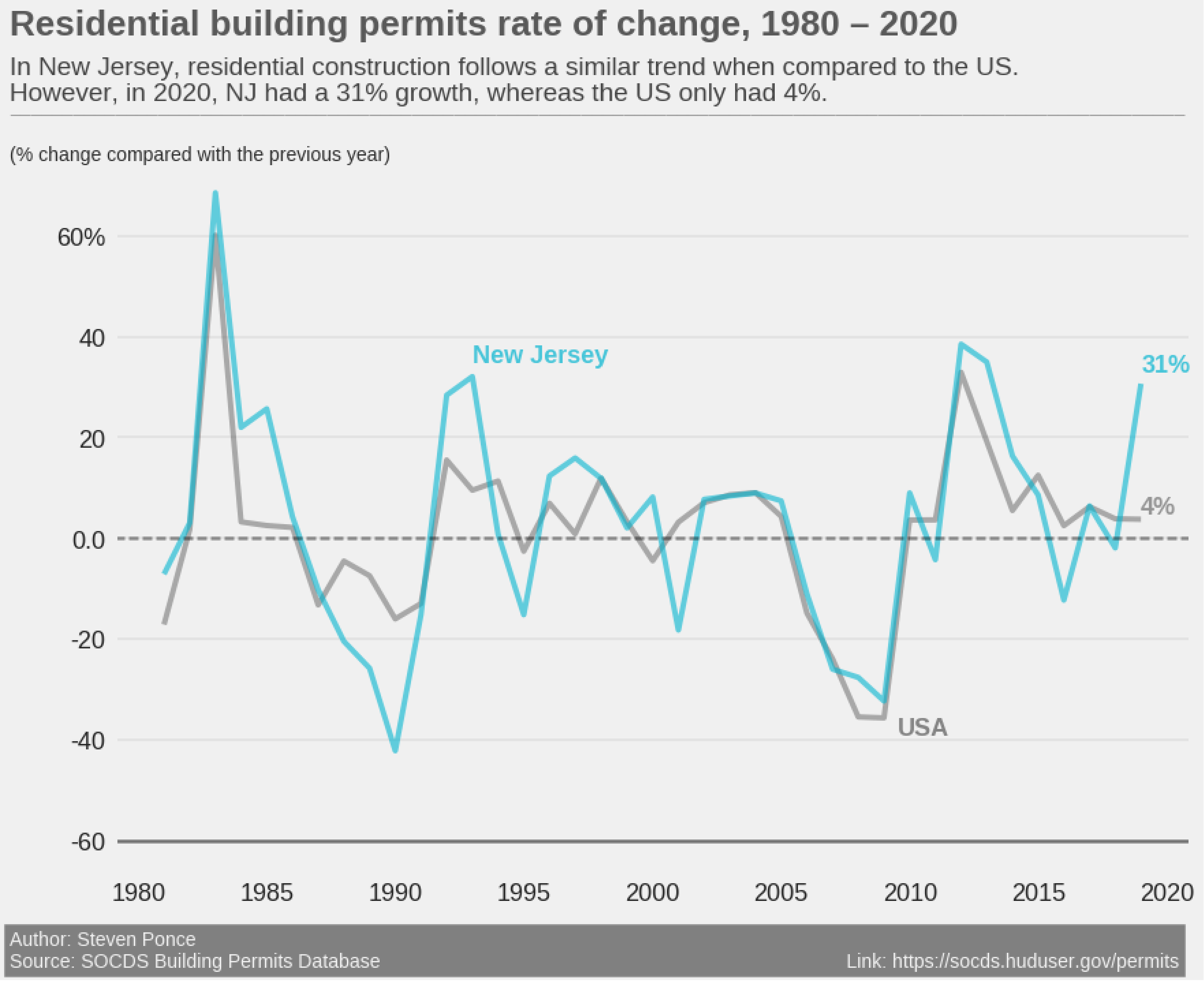
In [14]:

```
# Lets calculate Percent Chage (year-over-year)
USA = (Country.groupby("Year").sum()).pct_change().rename(columns={'Total Permits':'% Change'})
New_Jersey = (State.groupby("Year").sum()).pct_change().rename(columns={'Total Permits':'% Change'})
Middlesex_County = (County.groupby("Year").sum()).pct_change().rename(columns={'Total Permits':'% Change'})
```

```
In [15]: '''
         @author:  Steven Ponce
         Date:     13 May 2021
         '''
         def create_fig():
             # Final figure
             # removed County data for better storytelling

             # adjust the figure
             fig, ax = plt.subplots(figsize = (12,8))

             # line plot
             plt.plot(USA, linewidth=4, linestyle='solid', label='USA', color='#3C3C3C', alpha = 0.4)
             plt.plot(New_Jersey, linewidth=4, linestyle='solid', label='New Jersey', color='#00B4D0', alpha = 0.6)
             #plt.plot(Middlesex_County,linewidth=3, linestyle='solid', label='Middlesex County', color='#606060', alpha = 0.5)

             # Format ticks
             ax.tick_params(axis = 'both', which = 'major', labelsize = 18)

             # Customizing the tick labels of the y-axis
             ax.set_yticklabels(labels = ['-80', '-60', '-40', '-20', '0.0', '20', '40', '60%'])

             # Add a bolded horizontal line at y = 0
             ax.axhline(y = 0, color = 'black', linestyle='dashed', linewidth = 2.5, alpha = 0.4)

             # Add a bolded horizontal line at y = -0.6
             ax.axhline(y = -0.6, color = 'black', linestyle='solid', linewidth = 2.5, alpha = 0.5)

             # Remove the label of the x-axis
             ax.xaxis.label.set_visible(False)

             # format grid
             plt.grid(False)
             ax.yaxis.grid(True, color ="black", linewidth=0.1)

             # axis labels
             plt.xlabel('Year', fontsize=16)

             # Adding a title and a subtitle
             ax.text(x = 1975, y = 1, s = 'Residential building permits rate of change, 1980 – 2020',
                            fontsize = 26, weight = 'bold', alpha = 0.75)
             ax.text(x = 1975, y = 0.87,
                            s = 'In New Jersey, residential construction follows a similar trend when compared to the US.\nHowever,
          in 2020, NJ had a 31% growth, whereas the US only had 4%.',
                           fontsize = 19, alpha = 0.85)

             # y-axis label
             ax.text(x = 1975, y = 0.75, s = '(% change compared with the previous year)', color = '#303030',
                     weight = 'light', fontsize=14)

             # The top bar
             ax.text(x = 1975, y = 0.85,
                 s = '_____________________________________________________________________________________________________________
         _',
                 color = 'grey', alpha = .7)

             # The bottom bar
             ax.text(x = 1975, y = -0.85,
                     s = '''Author: Steven Ponce
         Source: SOCDS Building Permits Database
         Link: https://socds.huduser.gov/permits''',
                     fontsize = 14, color = '#f0f0f0', backgroundcolor = 'grey')

             # Custom Legend
             ax.text(x = 1993, y = 0.35, s = 'New Jersey', color = '#00B4D0', weight = 'bold', fontsize=18, alpha = 0.7)
             ax.text(x = 2009.5, y = -0.39, s = 'USA', color = '#3C3C3C', weight = 'bold', fontsize=18, alpha = 0.6)
             ax.text(x = 2019, y = 0.33, s = '31%', color = '#00B4D0', weight = 'bold', fontsize=18, alpha = 0.7)
             ax.text(x = 2019, y = 0.05, s = '4%', color = '#3C3C3C', weight = 'bold', fontsize=18, alpha = 0.5)

             plt.tight_layout()
             plt.show();

             return create_fig
```

In [16]:

```
create_fig();
```



In [ ]:

```
```