In [1]:

```
#pip install openweathermapy
#pip install census
#pip install citipy
#pip install gmaps
#pip install plotly
#pip install chart-studio
import requests
import random
import json
import os
import csv
import matplotlib.pyplot as plt
import pandas as pd
import openweathermapy.core as owm
from citipy import citipy
from pprint import pprint
import numpy as np
from scipy.stats import sem,linregress
import matplotlib.style as style
from census import Census
import gmaps
import chart_studio.plotly as py
import plotly.graph_objs as go
import seaborn as sb
```

# Home Prices

In [2]:

```
# Home PriceFile to Load (Remember to Change These)
home_prices = pd.read_csv("../data/HPI_master.csv")
home_prices.head()
```

Out[2]:

| | hpi_type | hpi_flavor | frequency | level | place_name | place_id | yr | period | index_nsa |
|---|---|---|---|---|---|---|---|---|---|
| **0** | traditional | purchase-only | monthly | USA or Census Division | East North Central Division | DV_ENC | 1991 | 1 | 100.00 |
| **1** | traditional | purchase-only | monthly | USA or Census Division | East North Central Division | DV_ENC | 1991 | 2 | 101.03 |
| **2** | traditional | purchase-only | monthly | USA or Census Division | East North Central Division | DV_ENC | 1991 | 3 | 101.40 |
| **3** | traditional | purchase-only | monthly | USA or Census Division | East North Central Division | DV_ENC | 1991 | 4 | 101.79 |
| **4** | traditional | purchase-only | monthly | USA or Census Division | East North Central Division | DV_ENC | 1991 | 5 | 102.44 |

In [3]:

```
#Clean up dataframe
home_prices_df = home_prices[["place_name","yr","period","price_nsa"]]
home_prices_df = home_prices_df.rename(columns={"place_name":"Place", "yr":"Year", "per
iod":"Month", "price_nsa":"Price"})
home_prices_df.head()
```

Out[3]:

| | Place | Year | Month | Price |
|---|---|---|---|---|
| **0** | East North Central Division | 1991 | 1 | $148,600 |
| **1** | East North Central Division | 1991 | 2 | $150,131 |
| **2** | East North Central Division | 1991 | 3 | $150,680 |
| **3** | East North Central Division | 1991 | 4 | $151,260 |
| **4** | East North Central Division | 1991 | 5 | $152,226 |

In [4]:

```
#Split the Place into city and state
new = home_prices_df['Place'].str.split(", ", n=1,expand=True)
home_prices_df['City']= new[0]
home_prices_df['State']= new[1]

#Clean up the Price data
home_prices_df['Price'] = home_prices_df['Price'].str.replace(r"$","")

# [x.strip('$') for x in home_prices_df.Price]
home_prices_df['Price'] = home_prices_df['Price'].str.replace(",","")
home_prices_df['Price'] = home_prices_df['Price'].str.replace(" ","")
home_prices_df['Price'] = round(pd.to_numeric(home_prices_df['Price']),2)

# Store the mean home prices for each year grouped by Place
mhp_df = home_prices_df.groupby(['State','Year'])['Price'].mean().to_frame().reset_inde
x()
# Keep only 25 years of data
mhp_df = mhp_df[mhp_df['Year'] > 1990]
# Round Price to 2 decimal Places
mhp_df['Price'] = round(mhp_df['Price'],2)

#Drop the row if State characters are more than 2
mhp_df.drop(mhp_df[mhp_df['State'].map(len) > 2].index, inplace=True)

# Preview DataFrame
mhp_df.head()
```

Out[4]:

| | State | Year | Price |
|---|---|---|---|
| **11** | AK | 1991 | 128022.62 |
| **12** | AK | 1992 | 136195.62 |
| **13** | AK | 1993 | 140638.62 |
| **14** | AK | 1994 | 147100.88 |
| **15** | AK | 1995 | 150769.75 |

In [5]:

```
# Minor Data Munging to Re-Format the Data Frames
p_mhp_df = mhp_df.pivot(index='Year', columns='State', values='Price')
# Find the mean value in US across all states and add a Mean column
p_mhp_df['United States'] = round(mhp_df.groupby('Year')['Price'].mean(),2)
p_mhp_df.head()
```

Out[5]:

| State | AK | AL | AR | AZ | CA | CO | CT | DI |
|---|---|---|---|---|---|---|---|---|
| Year | | | | | | | | |
| 1991 | 128022.62 | 132484.49 | 137407.92 | 137249.95 | 154051.50 | 121320.40 | 155192.61 | 147225.2 |
| 1992 | 136195.62 | 137324.69 | 138416.13 | 140997.56 | 153366.52 | 129326.58 | 152757.64 | 149970.5 |
| 1993 | 140638.62 | 143582.16 | 144847.42 | 146291.91 | 149276.88 | 141648.85 | 148877.50 | 153522.2 |
| 1994 | 147100.88 | 148493.75 | 151845.40 | 154051.18 | 145108.32 | 158788.40 | 146122.64 | 151583.2 |
| 1995 | 150769.75 | 154772.63 | 158682.65 | 162792.02 | 143402.81 | 170650.70 | 145388.57 | 152697.7 |

5 rows × 50 columns

In [6]:

```
# Create a dataframe for % change against 1991 for plotting and 100 as a base
pp_mhp_df = pd.DataFrame(p_mhp_df.values / p_mhp_df.loc[:1991].values * 100, index = p_
mhp_df.index, columns=p_mhp_df.columns)
# pp_mhp_df = pd.DataFrame((((p_mhp_df.values / p_mhp_df.loc[:1991].values)-1)*100)+10
0, index = p_mhp_df.index, columns=p_mhp_df.columns)
# pp_mhp_df = pd.DataFrame(((p_mhp_df.values - p_mhp_df.loc[:1991].values) / p_mhp_df.l
oc[:1991].values * 100) + 100, index = p_mhp_df.index, columns=p_mhp_df.columns)
pp_mhp_df.head()
```

Out[6]:

| State | AK | AL | AR | AZ | CA | CO | CT |
|---|---|---|---|---|---|---|---|
| Year | | | | | | | |
| 1991 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| 1992 | 106.384028 | 103.653409 | 100.733735 | 102.730500 | 99.555356 | 106.599203 | 98.431001 |
| 1993 | 109.854509 | 108.376581 | 105.414171 | 106.587951 | 96.900634 | 116.756003 | 95.930792 |
| 1994 | 114.902257 | 112.083875 | 110.507022 | 112.241338 | 94.194682 | 130.883512 | 94.155669 |
| 1995 | 117.768055 | 116.823207 | 115.482899 | 118.609894 | 93.087578 | 140.661175 | 93.682663 |

5 rows × 50 columns

In [7]:

```
#Find standard deviation and sort them
sd_hp_df = pp_mhp_df.std()
sd_hp_df = sd_hp_df.sort_values(ascending=True)
sd_hp_df = sd_hp_df.to_frame('Std').reset_index()

# Return the state with the highest standard deviation and index of this state in the p
ivoted mean home price dataframe
state_h_std = sd_hp_df['State'][len(sd_hp_df)-1]
state_h_std_idx = p_mhp_df.columns.get_loc(state_h_std)

# Return the state with the lowest standard deviation and index of this state in the pi
voted mean home price dataframe
state_l_std = sd_hp_df['State'][0]
state_l_std_idx = p_mhp_df.columns.get_loc(state_l_std)

# Return the NY state
state_ny_idx = p_mhp_df.columns.get_loc('NY')
```

In [8]:

```
#Import of states mapping

states_map = pd.read_csv("../data/states.csv")
states_map.head()
```

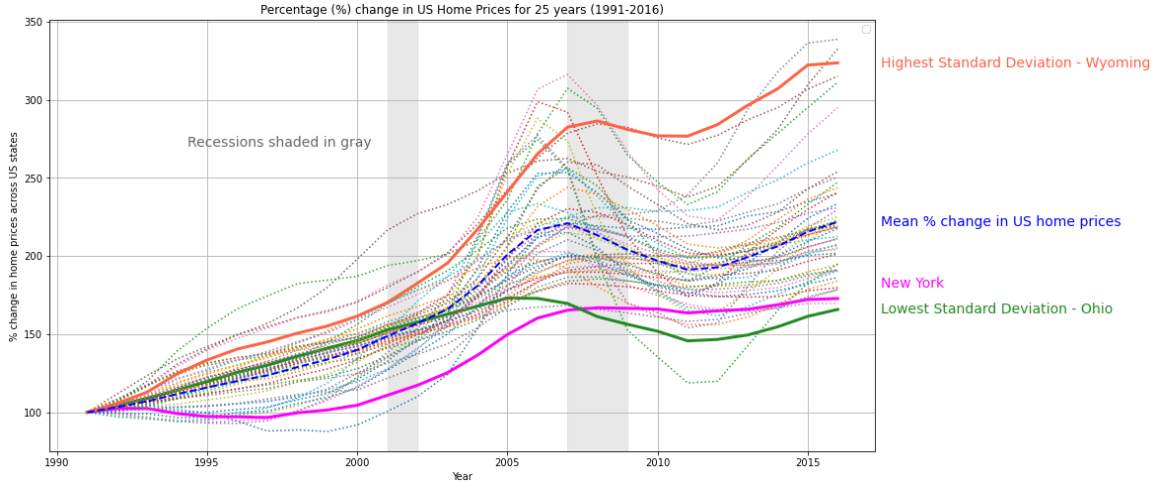
Out[8]:

| | State | Abbreviation |
|---|---|---|
| **0** | Alabama | AL |
| **1** | Alaska | AK |
| **2** | Arizona | AZ |
| **3** | Arkansas | AR |
| **4** | California | CA |

In [9]:

```
# Plot the data
ax = pp_mhp_df.plot(title = "Percentage (%) change in US Home Prices for 25 years (1991
-2016)", grid=1, figsize=(15,8), ls=':')

# Change the line color of the US mean
line = plt.gca().get_lines()[-1]
line.set_color("blue")
line.set_ls("--")
line.set_linewidth(2)
y = line.get_ydata()[-1]
ax.annotate("Mean % change in US home prices", xy=(1,y), xytext=(6,0), color=line.get_c
olor(), xycoords = ax.get_yaxis_transform(), textcoords="offset points", size=14, va="c
enter")

# Change the line color of the state with the highest standard deviation
line = plt.gca().get_lines()[state_h_std_idx]
line.set_color("tomato")
line.set_ls("-")
line.set_linewidth(3)
y = line.get_ydata()[-1]
state_name = states_map['State'].loc[states_map['Abbreviation'] == state_h_std].reset_i
ndex()
label = "Highest Standard Deviation - "+ state_name['State'][0]
ax.annotate(label, xy=(1,y), xytext=(6,0), color=line.get_color(), xycoords = ax.get_ya
xis_transform(), textcoords="offset points", size=14, va="center")

# Change the line color of the state with the lowest standard deviation
line = plt.gca().get_lines()[state_l_std_idx]
line.set_color("forestgreen")
line.set_ls("-")
line.set_linewidth(3)
y = line.get_ydata()[-1]
state_name = states_map['State'].loc[states_map['Abbreviation'] == state_l_std].reset_i
ndex()
label = "Lowest Standard Deviation - "+ state_name['State'][0]
ax.annotate(label, xy=(1,y), xytext=(6,0), color=line.get_color(), xycoords = ax.get_ya
xis_transform(), textcoords="offset points", size=14, va="center")

# Change the line color of the NY state
line = plt.gca().get_lines()[state_ny_idx]
line.set_color("magenta")
line.set_ls("-")
line.set_linewidth(3)
y = line.get_ydata()[-1]
label = "New York"
ax.annotate(label, xy=(1,y+10), xytext=(6,0), color=line.get_color(), xycoords = ax.get
_yaxis_transform(), textcoords="offset points", size=14, va="center")


# Plot the recessions

rec_bgn = [2001,2007]
rec_end = [2002,2009]

for i in range(len(rec_bgn)):
    plt.axvspan(rec_bgn[i], rec_end[i], color='lightgrey', alpha=0.5)

label = "Recessions shaded in gray"
ax.annotate(label, xy=(0.16,y+100), xytext=(6,0), color='dimgray', xycoords = ax.get_ya
```

```
xis_transform(), textcoords="offset points", size=14, va="center")


# Add Labels

plt.xlabel('Year')
plt.ylabel('% change in home prices across US states')
ax.legend("")
#plt.savefig("../figs/1_US_home_prices_pct_change.png", bbox_inches='tight')
plt.show()
```




# Census Data (Population and Household Income)

In [10]:

```
# Read in the csv containing state centroid coordinates
centroids = pd.read_csv("../data/state_centroids.csv")
centroids.head()
```

Out[10]:

| | State | Latitude | Longitude |
|---|---|---|---|
| **0** | Alabama | 32.7794 | -86.8287 |
| **1** | Alaska | 64.0685 | -152.2782 |
| **2** | Arizona | 34.2744 | -111.6602 |
| **3** | Arkansas | 34.8938 | -92.4426 |
| **4** | California | 37.1841 | -119.4696 |

In [11]:

```
# Import Median Household Income
mhi_df = pd.read_csv("../data/MHI_1984_2017.csv")
# Keep only 25 years of data
mhi_df = mhi_df[mhi_df['Year'] > 1990]
mhi_df = mhi_df[mhi_df['Year'] < 2017]
mhi_df = mhi_df.set_index('Year').rename_axis('State', axis=1)
mhi_df = mhi_df.sort_values('Year')
mhi_df.head()
```

Out[11]:

| State | United States | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut | Delav |
|---|---|---|---|---|---|---|---|---|---|
| Year | | | | | | | | | |
| 1991 | 30126 | 24346 | 40612 | 30737 | 23435 | 33664 | 31499 | 42154 | 32 |
| 1992 | 30636 | 25808 | 41802 | 29358 | 23882 | 34903 | 32484 | 40841 | 35 |
| 1993 | 31241 | 25082 | 42931 | 30510 | 23039 | 34073 | 34488 | 39516 | 36 |
| 1994 | 32264 | 27196 | 45367 | 31293 | 25565 | 35331 | 37833 | 41097 | 35 |
| 1995 | 34076 | 25991 | 47954 | 30863 | 25814 | 37009 | 40706 | 40243 | 34 |

5 rows × 52 columns

In [12]:

```
# Create a dataframe for % change against 1991 for plotting
pp_yc_df = pd.DataFrame(mhi_df.values / mhi_df.loc[:1991].values * 100, index = mhi_df.
index, columns=mhi_df.columns)
pp_yc_df.head()
```

Out[12]:

| State | United States | Alabama | Alaska | Arizona | Arkansas | California | Colorado |
|---|---|---|---|---|---|---|---|
| Year | | | | | | | |
| 1991 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| 1992 | 101.692890 | 106.005093 | 102.930168 | 95.513550 | 101.907403 | 103.680490 | 103.127083 |
| 1993 | 103.701122 | 103.023084 | 105.710135 | 99.261476 | 98.310220 | 101.214948 | 109.489190 |
| 1994 | 107.096860 | 111.706235 | 111.708362 | 101.808895 | 109.088969 | 104.951877 | 120.108575 |
| 1995 | 113.111598 | 106.756757 | 118.078400 | 100.409929 | 110.151483 | 109.936431 | 129.229499 |

5 rows × 52 columns

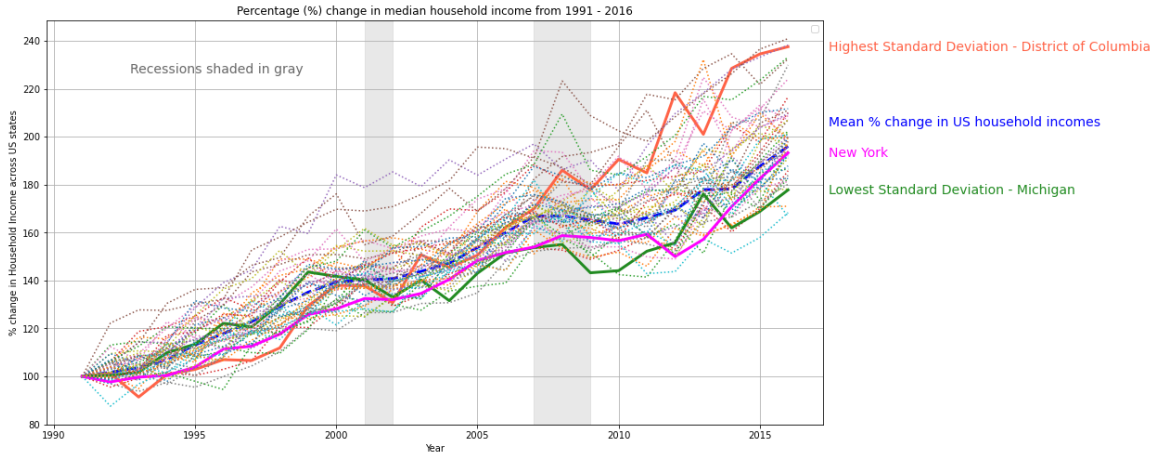
In [13]:

```
#Find standard deviation and sort them
sd_hp_df = pp_yc_df.std()
sd_hp_df = sd_hp_df.sort_values(ascending=True)
sd_hp_df = sd_hp_df.to_frame('Std').reset_index()
sd_hp_df

# Return the state with the highest standard deviation and index of this state in the p
ivoted mean home price dataframe
state_h_std = sd_hp_df['State'][len(sd_hp_df)-1]
state_h_std_idx = pp_yc_df.columns.get_loc(state_h_std)

# # Return the state with the lowest standard deviation and index of this state in the
 pivoted mean home price dataframe
state_l_std = sd_hp_df['State'][1]
state_l_std_idx = pp_yc_df.columns.get_loc(state_l_std)
```

In [14]:

```
# Plot the data

ax1 = pp_yc_df.plot(title = "Percentage (%) change in median household income from 1991
- 2016", grid=1, figsize=(15,8), ls='dotted', legend='False')

# Change the line color of the US mean
line = plt.gca().get_lines()[0]
line.set_color("blue")
line.set_ls("--")
line.set_linewidth(3)
y = line.get_ydata()[-1]
ax1.annotate("Mean % change in US household incomes", xy=(1,y+10), xytext=(6,0), color=
line.get_color(), xycoords = ax1.get_yaxis_transform(), textcoords="offset points", siz
e=14, va="center")

# Change the line color of the state with the highest standard deviation
line = plt.gca().get_lines()[state_h_std_idx]
line.set_color("tomato")
line.set_ls("-")
line.set_linewidth(3)
y = line.get_ydata()[-1]
label = "Highest Standard Deviation - "+ state_h_std
ax1.annotate(label, xy=(1,y), xytext=(6,0), color=line.get_color(), xycoords = ax1.get_
yaxis_transform(), textcoords="offset points", size=14, va="center")

# Change the line color of the state with the lowest standard deviation
line = plt.gca().get_lines()[state_l_std_idx]
line.set_color("forestgreen")
line.set_ls("-")
line.set_linewidth(3)
y = line.get_ydata()[-1]
label = "Lowest Standard Deviation - "+ state_l_std
ax1.annotate(label, xy=(1,y), xytext=(6,0), color=line.get_color(), xycoords = ax1.get_
yaxis_transform(), textcoords="offset points", size=14, va="center")

# Change the line color of the NY state
line = plt.gca().get_lines()[state_ny_idx]
line.set_color("magenta")
line.set_ls("-")
line.set_linewidth(3)
y = line.get_ydata()[-1]
label = "New York"
ax1.annotate(label, xy=(1,y), xytext=(6,0), color=line.get_color(), xycoords = ax1.get_
yaxis_transform(), textcoords="offset points", size=14, va="center")

# Plot the recessions

rec_bgn = [2001,2007]
rec_end = [2002,2009]

for i in range(len(rec_bgn)):
    plt.axvspan(rec_bgn[i], rec_end[i], color='lightgrey', alpha=0.5)

label = "Recessions shaded in gray"
ax1.annotate(label, xy=(0.1,y+35), xytext=(6,0), color='dimgray', xycoords = ax1.get_ya
xis_transform(), textcoords="offset points", size=14, va="center")

# Add labels
```

```
plt.xlabel('Year')
plt.ylabel('% change in Household Income across US states')
plt.legend("")
#plt.savefig("../figs/2_US_household_income_pct_change.png", bbox_inches='tight')
plt.show()
```




# GDP

In [15]:

```
# Load GDP
gdp = pd.read_csv("../data/gdp_growth.csv")
gdp.head()
```

Out[15]:

| | Year | Nominal GDP (trillions) | Real GDP (trillions) | GDP Growth Rate | Events Affecting GDP |
|---|---|---|---|---|---|
| **0** | 1929 | $0.11 | $1.11 | NaN | Depression began. |
| **1** | 1930 | $0.09 | $1.02 | -8.50% | Smoot-Hawley. |
| **2** | 1931 | $0.08 | $0.95 | -6.40% | Dust Bowl. |
| **3** | 1932 | $0.06 | $0.83 | -12.90% | Hoover tax hikes. |
| **4** | 1933 | $0.06 | $0.82 | -1.20% | New Deal. |

In [16]:

```
#Clean up the GDP data
gdp['GDP Growth Rate'] = gdp['GDP Growth Rate'].str.replace(r"%","")
gdp.head()
```

Out[16]:

| | Year | Nominal GDP (trillions) | Real GDP (trillions) | GDP Growth Rate | Events Affecting GDP |
|---|---|---|---|---|---|
| **0** | 1929 | $0.11 | $1.11 | NaN | Depression began. |
| **1** | 1930 | $0.09 | $1.02 | -8.50 | Smoot-Hawley. |
| **2** | 1931 | $0.08 | $0.95 | -6.40 | Dust Bowl. |
| **3** | 1932 | $0.06 | $0.83 | -12.90 | Hoover tax hikes. |
| **4** | 1933 | $0.06 | $0.82 | -1.20 | New Deal. |

In [17]:

```
# Keep only data from 1991 - 2016
gdp = gdp[gdp['Year'] > 1990]
gdp = gdp[gdp['Year'] < 2017]
gdp['GDP Growth Rate'] = pd.to_numeric(gdp['GDP Growth Rate'])
gdp = gdp.reset_index()
yearly_gdp = gdp[["Year","GDP Growth Rate"]]
yearly_gdp.head()
```

Out[17]:

| | Year | GDP Growth Rate |
|---|---|---|
| **0** | 1991 | -0.1 |
| **1** | 1992 | 3.5 |
| **2** | 1993 | 2.8 |
| **3** | 1994 | 4.0 |
| **4** | 1995 | 2.7 |

In [18]:

```
# Adjust GDP growth rate to 1991 as a base
yearly_gdp['GDP Growth Rate'] = yearly_gdp['GDP Growth Rate'] + 0.1
gdp_adj = [100]
for i in range(len(yearly_gdp)):
    if i!=0:
        if i==1:
            gdp_adj.append(yearly_gdp['GDP Growth Rate'][i]+100)
        elif i>1:
            value = gdp_adj[i-1]
            gdp_adj.append(((yearly_gdp['GDP Growth Rate'][i]/100)+1)*value)
years = gdp['Year']
yearly_gdp_adj = pd.DataFrame({'Year':years,'GDP': gdp_adj})
yearly_gdp_adj.head()
```

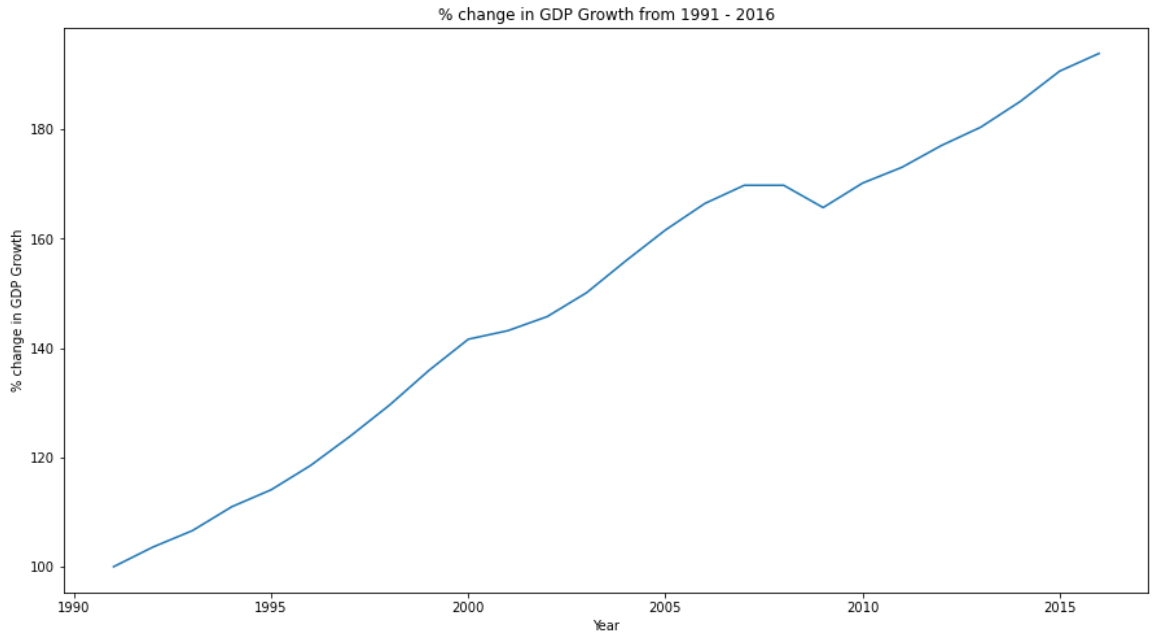
```
<ipython-input-18-ba4eab60b3a6>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  yearly_gdp['GDP Growth Rate'] = yearly_gdp['GDP Growth Rate'] + 0.1
```

Out[18]:

| | Year | GDP |
|---|---|---|
| **0** | 1991 | 100.000000 |
| **1** | 1992 | 103.600000 |
| **2** | 1993 | 106.604400 |
| **3** | 1994 | 110.975180 |
| **4** | 1995 | 114.082485 |

In [19]:

```
#Plot
plt.figure(figsize=(15,8))
plt.plot(yearly_gdp_adj['Year'],yearly_gdp_adj['GDP'])
plt.xlabel('Year')
plt.ylabel('% change in GDP Growth')
plt.title('% change in GDP Growth from 1991 - 2016')
#plt.savefig("../figs/3_GDP_Growth_pct_change.png", bbox_inches='tight')
plt.show()
```




# Interest Rates

In [20]:

```
# Load Interest Rates
ir = pd.read_csv("../data/fed-funds-rate-historical-chart.csv")
ir.head()
```

Out[20]:

| | date | value |
|---|---|---|
| **0** | 1/07/1954 | 1.13 |
| **1** | 2/07/1954 | 1.25 |
| **2** | 3/07/1954 | 1.25 |
| **3** | 4/07/1954 | 1.25 |
| **4** | 5/07/1954 | 0.88 |

In [21]:

```
#Strip the dates, leaving only year and get average
ir['date']= [x[-4:] for x in ir['date']]
yearly_ir = ir.groupby('date').mean().reset_index()

# # Keep only data from 1991 - 2016
yearly_ir = yearly_ir[yearly_ir['date'] > '1990']
yearly_ir = yearly_ir[yearly_ir['date'] < '2017']
yearly_ir = yearly_ir.rename(columns={"date": "Year", " value": "Interest Rate (%)"})
yearly_ir = yearly_ir.reset_index()
yearly_ir = yearly_ir[['Year','Interest Rate (%)']]
yearly_ir['Year'] = pd.to_numeric(yearly_ir['Year'])
yearly_ir.head()
```

Out[21]:

| | Year | Interest Rate (%) |
|---|---|---|
| **0** | 1991 | 5.685014 |
| **1** | 1992 | 3.521066 |
| **2** | 1993 | 3.021342 |
| **3** | 1994 | 4.206329 |
| **4** | 1995 | 5.834301 |

In [22]:

```
yearly_ir['Interest Rate'] = yearly_ir['Interest Rate (%)'] / yearly_ir['Interest Rate
 (%)'][0] * 100
yearly_ir.head()
```

Out[22]:

| | Year | Interest Rate (%) | Interest Rate |
|---|---|---|---|
| **0** | 1991 | 5.685014 | 100.000000 |
| **1** | 1992 | 3.521066 | 61.935921 |
| **2** | 1993 | 3.021342 | 53.145738 |
| **3** | 1994 | 4.206329 | 73.989774 |
| **4** | 1995 | 5.834301 | 102.625986 |

In [23]:

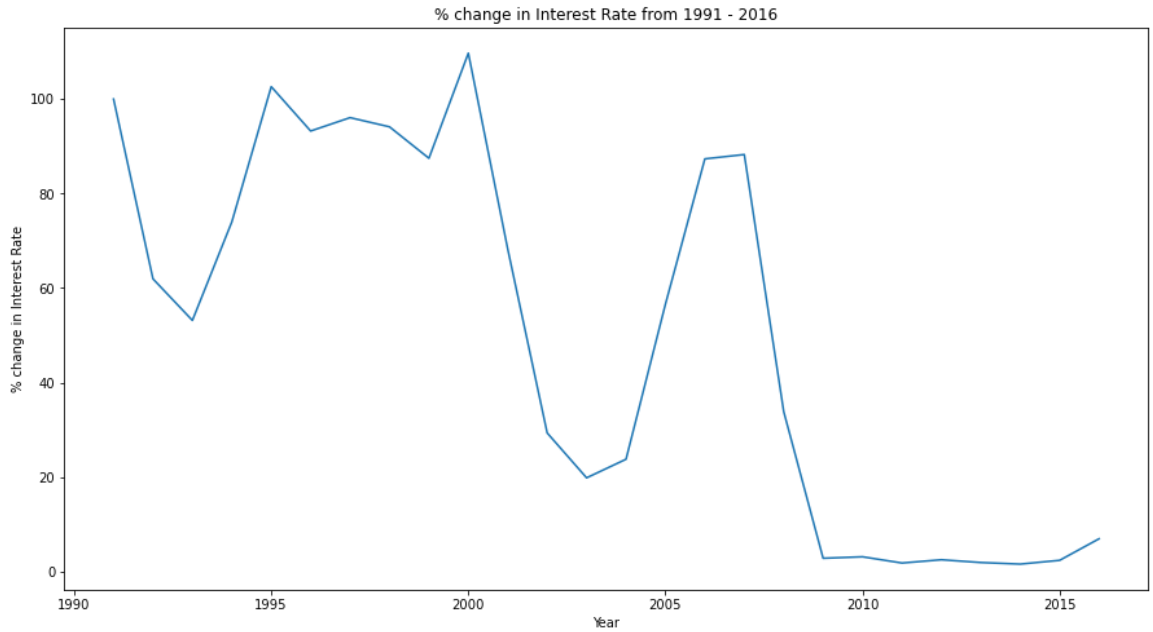
```
yearly_ir = yearly_ir.drop('Interest Rate (%)',axis=1)
yearly_ir.head()
```

Out[23]:

| | Year | Interest Rate |
|---|---|---|
| **0** | 1991 | 100.000000 |
| **1** | 1992 | 61.935921 |
| **2** | 1993 | 53.145738 |
| **3** | 1994 | 73.989774 |
| **4** | 1995 | 102.625986 |

In [24]:

```
#Plot
plt.figure(figsize=(15,8))
#plt.plot(yearly_ir['Year'],yearly_ir['Interest Rate (%)'])
plt.plot(yearly_ir['Year'],yearly_ir['Interest Rate'])
plt.xlabel('Year')
plt.ylabel('% change in Interest Rate')
plt.title('% change in Interest Rate from 1991 - 2016')
#plt.savefig("../figs/4_Interest_Rate_pct_change.png", bbox_inches='tight')
plt.show()
```




# Analysis (Home Prices, Household Income, GDP and Interest Rate)

In [25]:

```
#Create a dataframe that shows the mean of the home prices, household income, gdp and i
nterest rates
main_df = pd.DataFrame({'Home Prices':pp_mhp_df['United States'],'Household Income':pp_
yc_df['United States']})
main_df = main_df.reset_index()
main_df = main_df.merge(yearly_gdp_adj,on='Year')
main_df = main_df.merge(yearly_ir,on='Year')
main_df = main_df.set_index('Year')
main_df.head()
```

Out[25]:

| Year | Home Prices | Household Income | GDP | Interest Rate |
|---|---|---|---|---|
| **1991** | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| **1992** | 103.286975 | 101.692890 | 103.600000 | 61.935921 |
| **1993** | 107.060206 | 103.701122 | 106.604400 | 53.145738 |
| **1994** | 111.693782 | 107.096860 | 110.975180 | 73.989774 |
| **1995** | 115.884853 | 113.111598 | 114.082485 | 102.625986 |

In [26]:

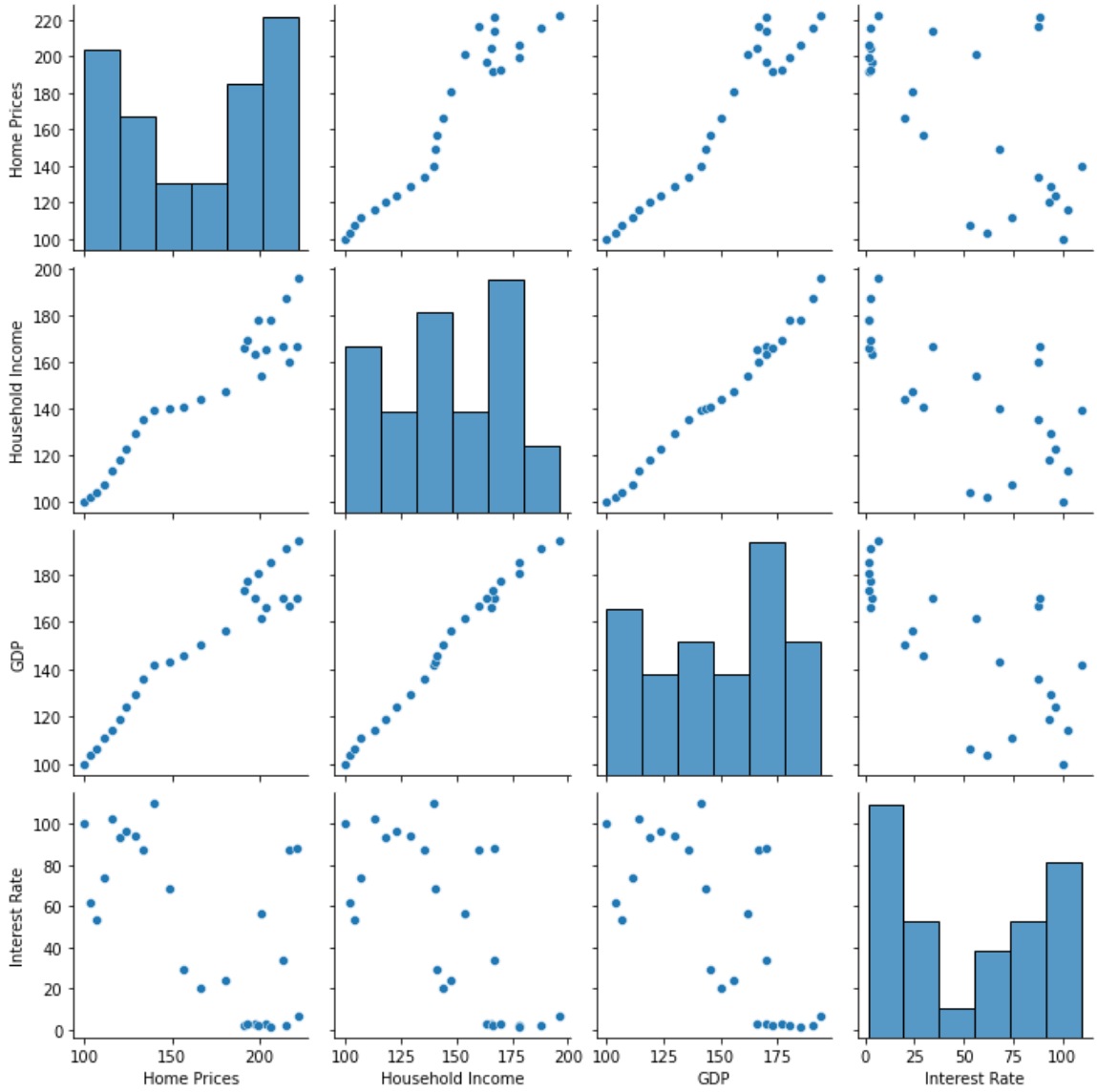
```
main_df.describe()
```

Out[26]:

| | Home Prices | Household Income | GDP | Interest Rate |
|---|---|---|---|---|
| **count** | 26.000000 | 26.000000 | 26.000000 | 26.000000 |
| **mean** | 166.020426 | 145.885231 | 149.363626 | 50.088771 |
| **std** | 42.867809 | 27.814845 | 28.733192 | 40.113475 |
| **min** | 100.000000 | 100.000000 | 100.000000 | 1.556604 |
| **25%** | 124.944562 | 124.394211 | 125.290084 | 4.054240 |
| **50%** | 173.401162 | 145.475669 | 153.055554 | 54.867159 |
| **75%** | 203.273168 | 166.594470 | 170.074805 | 88.057402 |
| **max** | 221.874135 | 195.973578 | 193.897019 | 109.717092 |

In [27]:

```
hp_hi_gdp_ir = sb.pairplot(main_df)
#hp_hi_gdp_ir.savefig("../figs/5_US_pair_plot.png")
```




In [28]:

```
#Linear Regression Result between Home Prices and Household Income
linregress(main_df['Home Prices'], main_df['Household Income'])
```

Out[28]:

```
LinregressResult(slope=0.6180412025442654, intercept=43.277766942769574, r
value=0.952515528541622, pvalue=6.799879346819933e-14, stderr=0.0403285675
0561952)
```

In [29]:

```
#Linear Regression Result between Home Prices and GDP
linregress(main_df['Home Prices'], main_df['GDP'])
```

Out[29]:

```
LinregressResult(slope=0.6443029694740582, intercept=42.396172716258604, r
value=0.9612526376303705, pvalue=6.200383644642083e-15, stderr=0.037716797
50116573)
```

In [30]:

```
#Linear Regression Result between Home Prices and Interest Rate
linregress(main_df['Home Prices'], main_df['Interest Rate'])
```

Out[30]:

```
LinregressResult(slope=-0.6013563387941091, intercept=149.92620682528099,
rvalue=-0.6426476093037841, pvalue=0.0003996975507456346, stderr=0.1463436
6423513998)
```

In [31]:

```
main_corr = main_df.corr()
main_corr
```

Out[31]:

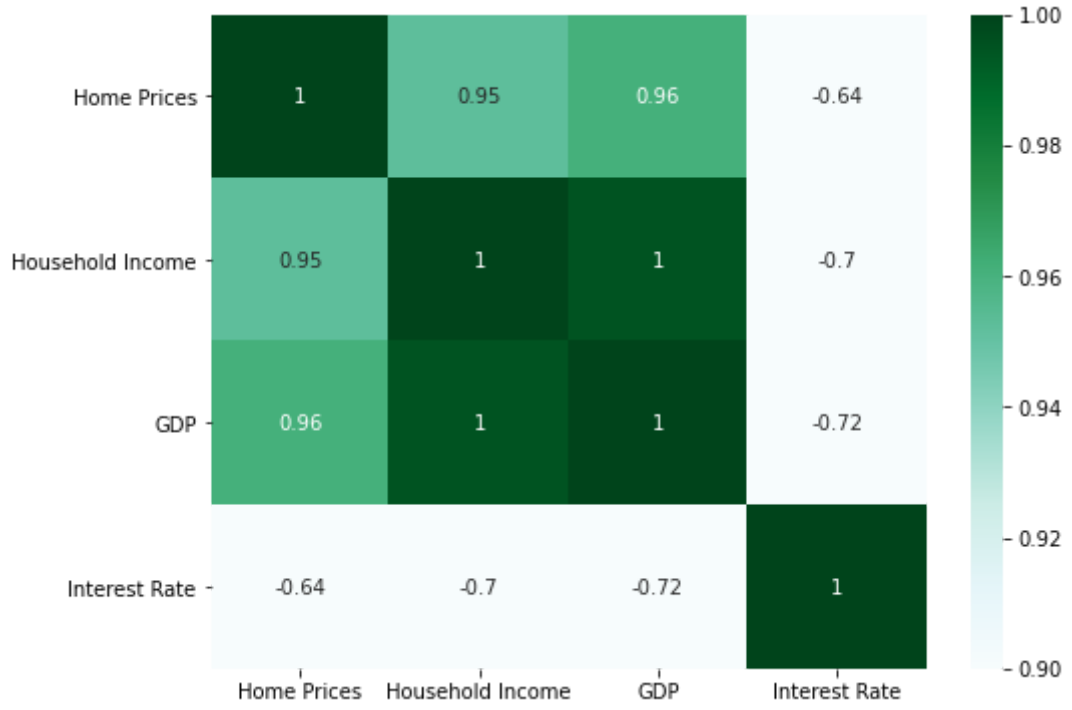
| | Home Prices | Household Income | GDP | Interest Rate |
|---|---|---|---|---|
| **Home Prices** | 1.000000 | 0.952516 | 0.961253 | -0.642648 |
| **Household Income** | 0.952516 | 1.000000 | 0.995202 | -0.696658 |
| **GDP** | 0.961253 | 0.995202 | 1.000000 | -0.716882 |
| **Interest Rate** | -0.642648 | -0.696658 | -0.716882 | 1.000000 |

In [32]:

```
# plot the heatmap
hp_hi_gdp_ir1 = plt.figure(figsize=(8,6))
sb.heatmap(main_corr, xticklabels=main_corr.columns, yticklabels=main_corr.columns, cma
p='BuGn', vmin = 0.9, annot = True)
#hp_hi_gdp_ir1.savefig("../figs/6_US_heatmap.png", bbox_inches='tight')
```

Out[32]:

<AxesSubplot:>




# Crime Rate

In [33]:

```
# Base URL to retrieve crime data
# url = "https://api.usa.gov/crime/fbi/sapi"
```

In [34]:

```
# Parameters for URL
# stateAbbr = ["SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"]
# offense = ["aggravated-assault", "burglary", "larceny", "motor-vehicle-theft", "homic
ide", "rape", "robbery", "arson", "violent-crime", "property-crime"]
# since = "2009"
# until = "2016"
```

In [35]:

```
# Request URL and convert into JSON format (https://crime-data-explorer.fr.cloud.gov/ap
i)
# crime = pd.DataFrame()

# for o in offense:
    # for s in stateAbbr:
        # request_url = url + "/api" + "/summarized" + "/state/" + s + "/" + o + "/" +
 since + "/" + until + "?api_key=" + api_key_crime
        # crime_data = requests.get(request_url).json()
        # c_crime_data = pd.concat([pd.Series(c3) for c3 in crime_data["results"]],1).T
        # crime = crime.append(c_crime_data, ignore_index=True)
```

In [36]:

```
# Export into CSV file
# crime.to_csv("output/crime5.csv")
# crime.head()
```

In [37]:

```
# Import crime CSV file
crime_df = pd.read_csv("../data/crime_states2.csv")
crime_df.head()
```

Out[37]:

| | data_year | offense | state_abbr | cleared | actual |
|---|---|---|---|---|---|
| **0** | 2009 | all | AL | 1982 | 63942 |
| **1** | 2009 | all | AK | 8018 | 29707 |
| **2** | 2009 | all | AZ | 236 | 620 |
| **3** | 2009 | all | AR | 22 | 511 |
| **4** | 2009 | all | CA | 2034 | 11939 |

In [38]:

```
# Drop unnecessary columns
crime_df1 = crime_df.drop(["offense", "cleared"], axis = 1)
crime_df1.head()
```

Out[38]:

| | data_year | state_abbr | actual |
|---|---|---|---|
| **0** | 2009 | AL | 63942 |
| **1** | 2009 | AK | 29707 |
| **2** | 2009 | AZ | 620 |
| **3** | 2009 | AR | 511 |
| **4** | 2009 | CA | 11939 |

In [39]:

```
# Import state data CSV file
state_df = pd.read_csv("../data/states.csv")
state_df.head()
```

Out[39]:

| | State | Abbreviation |
|---|---|---|
| **0** | Alabama | AL |
| **1** | Alaska | AK |
| **2** | Arizona | AZ |
| **3** | Arkansas | AR |
| **4** | California | CA |

In [40]:

```
# Merge crime data with state data
crime_df2 = crime_df1.merge(state_df, left_on="state_abbr", right_on="Abbreviation")
crime_df2 = crime_df2.drop(["Abbreviation"], axis = 1)
crime_df2 = crime_df2.rename(columns={"data_year": "Year"})
crime_df2.head()
```

Out[40]:

| | Year | state_abbr | actual | State |
|---|---|---|---|---|
| **0** | 2009 | AL | 63942 | Alabama |
| **1** | 2010 | AL | 37369 | Alabama |
| **2** | 2011 | AL | 58826 | Alabama |
| **3** | 2012 | AL | 52547 | Alabama |
| **4** | 2013 | AL | 16243 | Alabama |

In [41]:

```
# Import population data CSV file
population_df = pd.read_csv("../output/census_data_states20092016.csv")
population_df.head()
```

Out[41]:

| | State | Household Income | Population | Per Capita Income | State Index | Year |
|---|---|---|---|---|---|---|
| **0** | Alaska | 64635.0 | 683142.0 | 29382.0 | 2 | 2009 |
| **1** | Alabama | 41216.0 | 4633360.0 | 22732.0 | 1 | 2009 |
| **2** | Arkansas | 38542.0 | 2838143.0 | 20977.0 | 5 | 2009 |
| **3** | Arizona | 50296.0 | 6324865.0 | 25203.0 | 4 | 2009 |
| **4** | California | 60392.0 | 36308527.0 | 29020.0 | 6 | 2009 |

In [42]:

```
# Merge crime data with population data
crime_df3 = pd.merge(crime_df2, population_df, how="inner", on=["Year", "State"])
crime_df3 = crime_df3.drop(["Household Income", "Per Capita Income", "State Index"], ax
is = 1)
crime_df3 = crime_df3.rename(columns={"state_abbr": "StateAbbr", "actual": "Number of C
rime"})
crime_df3.head()
```

Out[42]:

| | Year | StateAbbr | Number of Crime | State | Population |
|---|---|---|---|---|---|
| **0** | 2009 | AL | 63942 | Alabama | 4633360.0 |
| **1** | 2010 | AL | 37369 | Alabama | 4712651.0 |
| **2** | 2011 | AL | 58826 | Alabama | 4747424.0 |
| **3** | 2012 | AL | 52547 | Alabama | 4777326.0 |
| **4** | 2013 | AL | 16243 | Alabama | 4799277.0 |

In [43]:

```
# Calculate crime rate, i.e. (Number of Crime / Population) * 100
nb_crime = crime_df3["Number of Crime"]
pop = crime_df3["Population"]
crime_rate = (nb_crime / pop) * 100
crime_rate.head()
```

Out[43]:

```
0    1.380035
1    0.792951
2    1.239114
3    1.099925
4    0.338447
dtype: float64
```

In [44]:

```
# Create a dataframe to hold crime rate
year = crime_df3["Year"]
stateabbr = crime_df3["StateAbbr"]
state = crime_df3["State"]

crime_df4 = pd.DataFrame({"Year": year, "StateAbbr": stateabbr,
                          "State": state, "Number of Crime": nb_crime,
                          "Population": pop, "Crime Rate": crime_rate})
crime_df4.head()
```

Out[44]:

| | Year | StateAbbr | State | Number of Crime | Population | Crime Rate |
|---|---|---|---|---|---|---|
| **0** | 2009 | AL | Alabama | 63942 | 4633360.0 | 1.380035 |
| **1** | 2010 | AL | Alabama | 37369 | 4712651.0 | 0.792951 |
| **2** | 2011 | AL | Alabama | 58826 | 4747424.0 | 1.239114 |
| **3** | 2012 | AL | Alabama | 52547 | 4777326.0 | 1.099925 |
| **4** | 2013 | AL | Alabama | 16243 | 4799277.0 | 0.338447 |

In [45]:

```
# Check which state has highest crime rate
max_state = crime_df4[crime_df4["Crime Rate"]==crime_df4["Crime Rate"].max()]["State"]
max_state.values[0]
```

Out[45]:

```
'Hawaii'
```

In [46]:

```
# Minor Data Munging to Re-Format the Data Frames
crime_df5 = crime_df4.pivot(index="Year", columns="State", values="Crime Rate")
crime_df5
```

Out[46]:

| State | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut | Delaware |
|---|---|---|---|---|---|---|---|---|
| Year | | | | | | | | |
| **2009** | 1.380035 | 4.348583 | 0.009803 | 0.018005 | 0.032882 | 0.655970 | 0.050937 | 0.654873 |
| **2010** | 0.792951 | 3.752519 | 0.011254 | 0.019007 | 0.031048 | 0.644968 | 0.051243 | 0.650192 |
| **2011** | 1.239114 | 3.498629 | 0.012182 | 0.017369 | 0.029990 | 0.616585 | 0.055113 | 0.704154 |
| **2012** | 1.099925 | 3.790117 | 0.013087 | 0.016253 | 0.031863 | 0.607672 | 0.055680 | 0.680679 |
| **2013** | 0.338447 | 4.160674 | 0.012269 | 0.010841 | 0.027250 | 0.586034 | 0.032258 | 0.559967 |
| **2014** | 0.263882 | 5.118358 | 0.011872 | 0.009908 | 0.024037 | 0.542541 | 0.032572 | 0.494733 |
| **2015** | 0.304329 | 5.331106 | 0.007407 | 0.012339 | 0.023763 | 0.578529 | 0.029361 | 0.521774 |
| **2016** | 0.269832 | 6.249669 | 0.006658 | 0.009432 | 0.025744 | 0.618328 | 0.027504 | 0.499200 |

8 rows × 50 columns

In [47]:

```
# Find the index number for maximum crime value
crime_max_idx = crime_df5.columns.get_loc(max_state.values[0])
crime_max_idx
```

Out[47]:

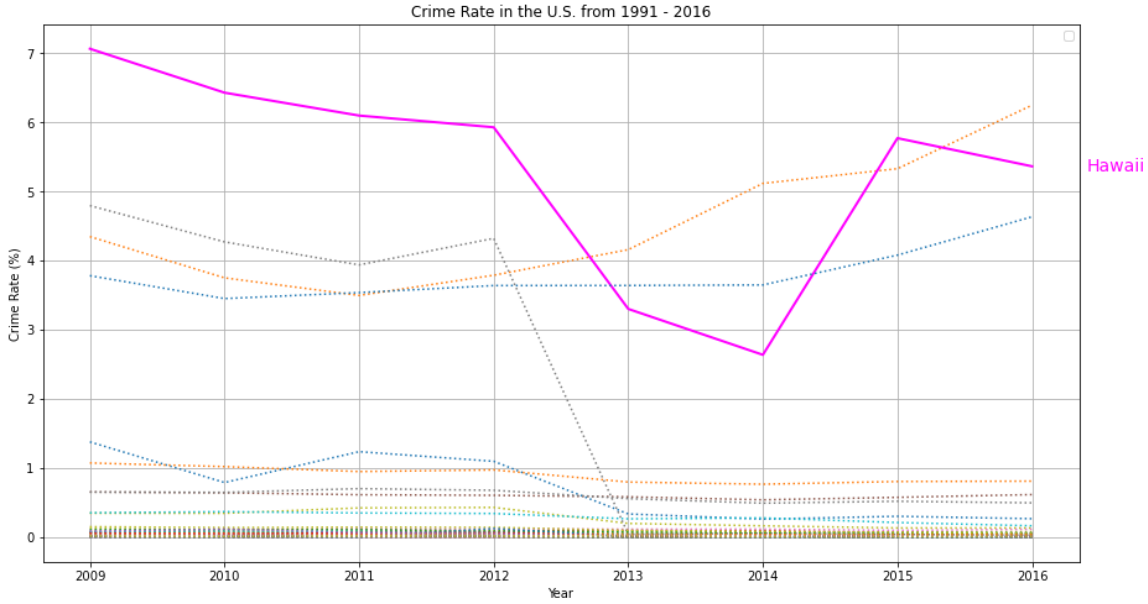
```
10
```

In [48]:

```
# Filter only NY data and export into CSV file
crime_df4_ny = crime_df4.loc[crime_df4["StateAbbr"] == "NY", :]
crime_df4_ny
crime_df4_ny.to_csv("../output/crime_df4_ny.csv")
```

In [49]:

```
# Plot the crime rate

crime_plot = crime_df5.plot(title = "Crime Rate in the U.S. from 1991 - 2016", grid=1,
figsize=(15,8), ls="dotted")

# Change the plot design
line = plt.gca().get_lines()[crime_max_idx]
line.set_color("magenta")
line.set_ls("-")
line.set_linewidth(2)
y = line.get_ydata()[-1]
label = max_state.values[0]
crime_plot.annotate(label, xy=(1,y), xytext=(6,0), color=line.get_color(), xycoords = c
rime_plot.get_yaxis_transform(), textcoords="offset points", size=14, va="center")
plt.xlabel("Year")
plt.ylabel("Crime Rate (%)")
plt.legend("")
#plt.savefig("../figs/7_crime_plot", bbox_inches="tight")
plt.show()
```




In [50]:

```
# Get 2016 Crime Rate for all states

crime_df_hm = crime_df4.loc[crime_df4['Year']==2016]
crime_df_hm.head()
```

Out[50]:

| | Year | StateAbbr | State | Number of Crime | Population | Crime Rate |
|---|---|---|---|---|---|---|
| **7** | 2016 | AL | Alabama | 13063 | 4841164.0 | 0.269832 |
| **15** | 2016 | AK | Alaska | 46051 | 736855.0 | 6.249669 |
| **23** | 2016 | AZ | Arizona | 448 | 6728577.0 | 0.006658 |
| **31** | 2016 | AR | Arkansas | 280 | 2968472.0 | 0.009432 |
| **39** | 2016 | CA | California | 9951 | 38654206.0 | 0.025744 |

In [51]:

```
# Merge it with coordinates
crime_df_hm1 = pd.merge(crime_df_hm, centroids, on="State")
crime_df_hm1.head()
```

Out[51]:

| | Year | StateAbbr | State | Number of Crime | Population | Crime Rate | Latitude | Longitude |
|---|---|---|---|---|---|---|---|---|
| **0** | 2016 | AL | Alabama | 13063 | 4841164.0 | 0.269832 | 32.7794 | -86.8287 |
| **1** | 2016 | AK | Alaska | 46051 | 736855.0 | 6.249669 | 64.0685 | -152.2782 |
| **2** | 2016 | AZ | Arizona | 448 | 6728577.0 | 0.006658 | 34.2744 | -111.6602 |
| **3** | 2016 | AR | Arkansas | 280 | 2968472.0 | 0.009432 | 34.8938 | -92.4426 |
| **4** | 2016 | CA | California | 9951 | 38654206.0 | 0.025744 | 37.1841 | -119.4696 |

# Weather data

In [52]:

```
def find_all(a_str, sub):
    return [i for i in range(len(a_str)) if a_str.startswith(sub, i)]

def get_temp(start_ind):
    end_ind = data.find('</td', start_ind)
    return data[start_ind + len('class="align_right temperature_red">') : end_ind]
```

In [53]:

```
def get_temp(start_ind):
    end_ind = data.find('</td', start_ind)
    return data[start_ind + len('class="align_right temperature_red">') : end_ind]
```

In [54]:

```
def get_state_url(start_ind, data):
    end_ind = data.find('"', start_ind + len('<a class="province" href="'))
    return data[start_ind + len('<a class="province" href="') : end_ind]
```

In [55]:

```
def find_all(a_str, sub):
    return [i for i in range(len(a_str)) if a_str.startswith(sub, i)]

def get_temp(start_ind):
    end_ind = data.find('</td', start_ind)
    return data[start_ind + len('class="align_right temperature_red">') : end_ind]

columns = ['state']
for i in range(12):
    columns.append(str(i+1))
```

In [56]:

```
r = requests.get(url='https://www.usclimatedata.com/climate/united-states/us')
d = r.text
state_url_indexes = find_all(d, '<a class="province" href="')
state_urls = [get_state_url(i, d) for i in state_url_indexes]
```

In [57]:

```
state_urls
```

Out[57]:

```
[]
```

In [58]:

```
all_state_data = pd.DataFrame(columns=columns)
ct = 0
for state_url in state_urls:
    row = pd.Series()
    api_url='https://www.usclimatedata.com' + state_url
    r = requests.get(url=api_url)
    data = r.text
    state_name = state_url.split('/')[2]
    state_name = state_name.replace('-', ' ')
    row['state'] = state_name
    print ("Processing state " + state_name)
    avg_temp_indexes = find_all(data, 'class="align_right temperature_red">')
    temps = [get_temp(i) for i in avg_temp_indexes]
    for index, temp in enumerate(temps):
#         print ("Month", index, temp)
        index_str = str(index + 1)
        row[index_str] = temp
    all_state_data.loc[ct] = row
    ct += 1
```

In [59]:

```
all_state_data
```

Out[59]:

| state | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

In [60]:

```
US_cold = pd.read_csv("../data/US_cold.csv")
US_hot = pd.read_csv("../data/US_hot.csv")
US_Summer = pd.read_csv("../data/US_Summer.csv")
US_Winter = pd.read_csv("../data/US_Winter.csv")
```

In [61]:

```
US_Winter.head(2)
```

Out[61]:

| | State | state | Housing Price | Cold Winter |
|---|---|---|---|---|
| **0** | SD | south dakota | 325129.25 | 22 |
| **1** | AK | alaska | 307330.75 | 23 |

In [62]:

```
US_hot.head(2)
```

Out[62]:

| | State | state | Housing Price | Hot Summer |
|---|---|---|---|---|
| **0** | UT | utah | 349480.5 | 90 |
| **1** | CO | colorado | 403398.6 | 90 |

In [63]:

```
US_cold.head(2)
US_Winter.head(2)
%matplotlib notebook
import matplotlib.pyplot as plt

import numpy as np

State_Winter = US_Winter.iloc[:, 3]
State_Housing = US_Winter.iloc[:, 2]
State_Summer = US_Summer.iloc[:, 3]
q1_cold = np.percentile(State_Winter, 25)
q3_cold = np.percentile(State_Winter,75)

iqr_cold = (q3_cold - q1_cold)
```

In [64]:

```
lower_boundary_cold = q1_cold - (1.5 * iqr_cold)
lower_boundary_cold
```

Out[64]:

```
7.5
```

In [65]:

```
upper_boundary_cold = q3_cold + (1.5 * iqr_cold)
upper_boundary_cold
```

Out[65]:

```
75.5
```

In [66]:

```
State_Winter[State_Winter <= lower_boundary_cold]
```

Out[66]:

```
Series([], Name: Cold Winter, dtype: int64)
```

In [67]:

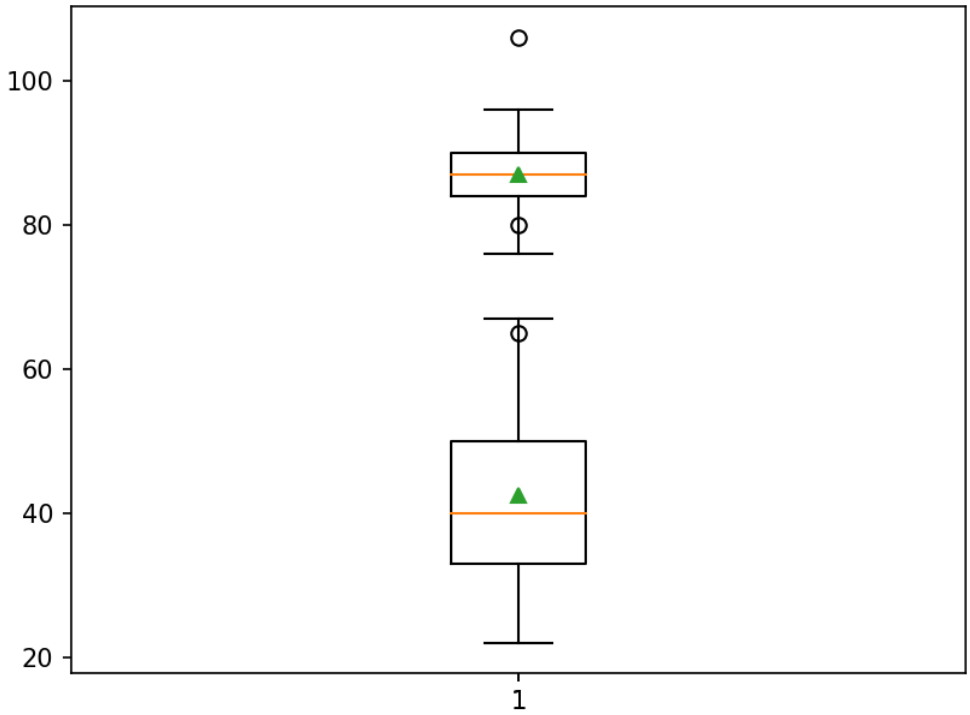
```
State_Winter[State_Winter >= upper_boundary_cold]
```

Out[67]:

```
48    80
Name: Cold Winter, dtype: int64
```

In [68]:

```
plt.boxplot(State_Winter, showmeans=True)
plt.grid()
#plt.savefig("../figs/8_US_Temp_Range_Winter_Summer.png", bbox_inches='tight')
plt.show()
```




In [69]:

```
q1_hot = np.percentile(State_Summer, 25)
q3_hot = np.percentile(State_Summer,75)
iqr_hot = (q3_hot - q1_hot)
```

In [70]:

```
lower_boundary_hot = q1_hot - (1.5 * iqr_hot)
lower_boundary_hot
```

Out[70]:

```
75.0
```

In [71]:

```
upper_boundary_hot = q3_hot + (1.5 * iqr_hot)
upper_boundary_hot
```

Out[71]:

```
99.0
```

In [72]:

```
State_Summer[State_Summer <= lower_boundary_hot]
```

Out[72]:

```
0    65
Name: Hot Summer, dtype: int64
```

In [73]:

```
State_Summer[State_Summer >= upper_boundary_hot]
```

Out[73]:

```
48    106
Name: Hot Summer, dtype: int64
```

In [74]:

```
plt.boxplot(State_Summer, showmeans=True)
plt.grid()
plt.show()
```

In [75]:

```
State_cold = US_cold.iloc[:, 0]
Housing_cold = US_cold.iloc[:,2]
Winter_Temp = US_cold.iloc[:,3]
```

In [76]:

```
State_cold.head()
Winter_Temp.head()
Housing_cold.head(2)
```

Out[76]:

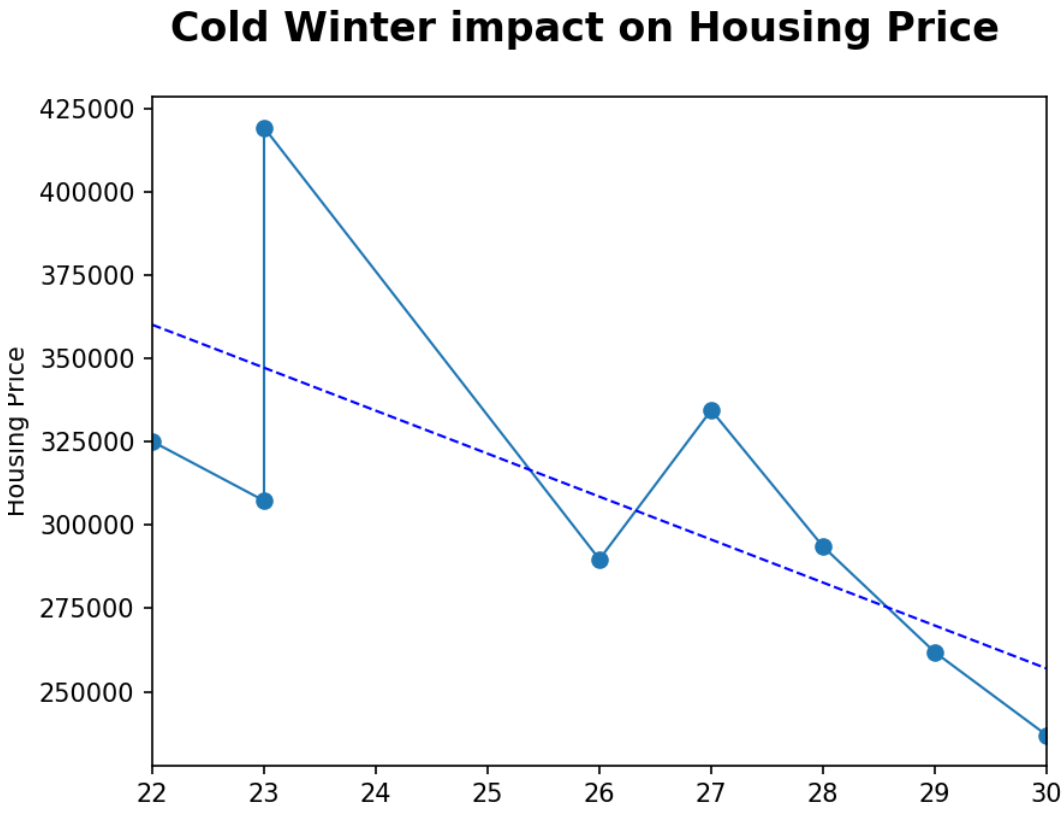
```
0    325129.25
1    307330.75
Name: Housing Price, dtype: float64
```

In [77]:

```
from scipy import stats
vc_slope, vc_int, vc_r, vc_p, vc_std_err = stats.linregress(
    Winter_Temp, Housing_cold)
vc_fit = vc_slope * Winter_Temp + vc_int
```

In [78]:

```
fig, (ax20) = plt.subplots(1, sharex=True)
fig.suptitle("Cold Winter impact on Housing Price", fontsize=16, fontweight="bold")

ax20.set_xlim(min(Winter_Temp), max(Winter_Temp))
ax20.plot(Winter_Temp, Housing_cold, linewidth=1, marker="o")
ax20.plot(Winter_Temp, vc_fit, "b--", linewidth=1)
ax20.set_ylabel("Housing Price")
#plt.savefig("../figs/9_Cold_Winter.png", bbox_inches='tight')
```

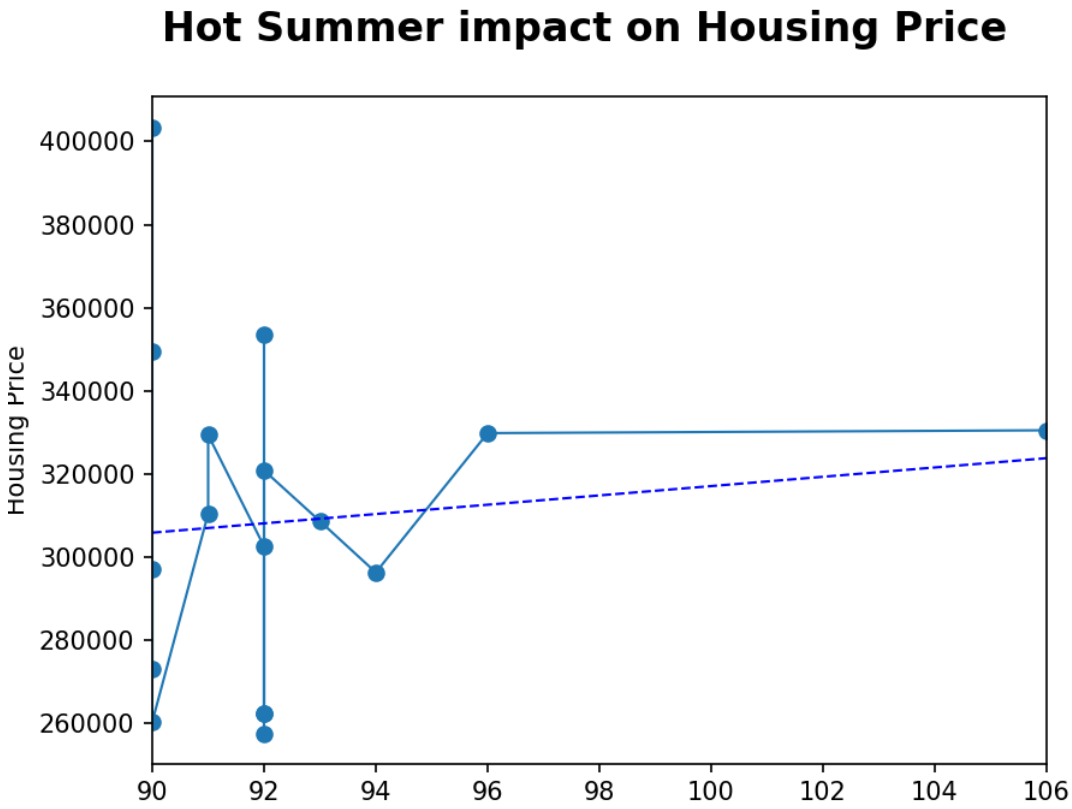



Out[78]:

```
Text(0, 0.5, 'Housing Price')
```

In [79]:

```
State_hot = US_hot.iloc[:, 0]
Housing_hot = US_hot.iloc[:,2]
Summer_Temp = US_hot.iloc[:,3]

vc_slope, vc_int, vc_r, vc_p, vc_std_err = stats.linregress(
    Summer_Temp, Housing_hot)
vc_fit = vc_slope * Summer_Temp + vc_int

fig, (ax20) = plt.subplots(1, sharex=True)
fig.suptitle("Hot Summer impact on Housing Price", fontsize=16, fontweight="bold")

ax20.set_xlim(min(Summer_Temp), max(Summer_Temp))
ax20.plot(Summer_Temp, Housing_hot, linewidth=1, marker="o")
ax20.plot(Summer_Temp, vc_fit, "b--", linewidth=1)
ax20.set_ylabel("Housing Price")
#plt.savefig("../figs/10_Hot_Summer.png", bbox_inches='tight')
```




Out[79]:

```
Text(0, 0.5, 'Housing Price')
```

# Analysis (Home Prices, Household Income, Crime Rate, Weather in NY)

In [80]:

```
main_ny_df = pd.DataFrame({'Home Prices':p_mhp_df['NY'],'Household Income':mhi_df['New
 York']})
main_ny_df = main_ny_df.reset_index()
# Keep only data from 2009 - 2016
main_ny_df = main_ny_df[main_ny_df['Year'] > 2008]
main_ny_df = main_ny_df[main_ny_df['Year'] < 2017]
main_ny_df.head()
```

Out[80]:

| | Year | Home Prices | Household Income |
|---|---|---|---|
| **18** | 2009 | 256457.10 | 50216 |
| **19** | 2010 | 255806.25 | 49781 |
| **20** | 2011 | 251890.58 | 50636 |
| **21** | 2012 | 254002.98 | 47680 |
| **22** | 2013 | 255519.72 | 49966 |

In [81]:

```
# Read in crime data and merge
ny_crimes = pd.read_csv("../output/crime_df4_ny.csv")
ny_crimes = ny_crimes [['Year','Crime Rate']]
main_ny_df = main_ny_df.merge(ny_crimes, on='Year')
main_ny_df2 = main_ny_df
main_ny_df.head()
```

Out[81]:

| | Year | Home Prices | Household Income | Crime Rate |
|---|---|---|---|---|
| **0** | 2009 | 256457.10 | 50216 | 0.060312 |
| **1** | 2010 | 255806.25 | 49781 | 0.063891 |
| **2** | 2011 | 251890.58 | 50636 | 0.062831 |
| **3** | 2012 | 254002.98 | 47680 | 0.058073 |
| **4** | 2013 | 255519.72 | 49966 | 0.051501 |

In [82]:

```
# Read in weather data and merge

ny_weather = pd.read_csv("../data/New_York__City_10_years__temperature.csv")
# Keep only data from 2009 - 2016
ny_weather = ny_weather[ny_weather['Year'] > 2008]
ny_weather = ny_weather[ny_weather['Year'] < 2017]
main_ny_df = main_ny_df.merge(ny_weather, on='Year')
main_ny_df = main_ny_df.set_index('Year')
main_ny_df
```

Out[82]:

| Year | Home Prices | Household Income | Crime Rate | Annual Temperature |
|---|---|---|---|---|
| **2009** | 256457.10 | 50216 | 0.060312 | 54.0 |
| **2010** | 255806.25 | 49781 | 0.063891 | 56.7 |
| **2011** | 251890.58 | 50636 | 0.062831 | 56.4 |
| **2012** | 254002.98 | 47680 | 0.058073 | 57.3 |
| **2013** | 255519.72 | 49966 | 0.051501 | 55.3 |
| **2014** | 259950.50 | 54310 | 0.048723 | 54.4 |
| **2015** | 265114.58 | 58005 | 0.041803 | 56.7 |
| **2016** | 266112.37 | 61437 | 0.045407 | 57.2 |

In [83]:

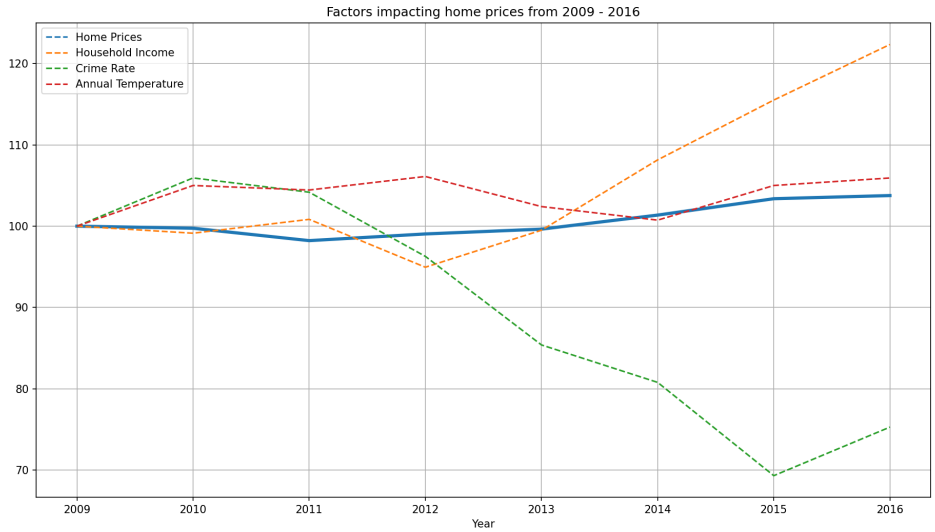
```
# Normalize the data for plotting
main_ny_df1 =  pd.DataFrame(main_ny_df / main_ny_df.loc[2009,:] * 100, index = main_ny_
df.index, columns=main_ny_df.columns)
main_ny_df1
```

Out[83]:

| Year | Home Prices | Household Income | Crime Rate | Annual Temperature |
|---|---|---|---|---|
| **2009** | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| **2010** | 99.746215 | 99.133742 | 105.932904 | 105.000000 |
| **2011** | 98.219383 | 100.836387 | 104.176760 | 104.444444 |
| **2012** | 99.043068 | 94.949817 | 96.286521 | 106.111111 |
| **2013** | 99.634489 | 99.502151 | 85.390300 | 102.407407 |
| **2014** | 101.362177 | 108.152780 | 80.784968 | 100.740741 |
| **2015** | 103.375800 | 115.510993 | 69.311089 | 105.000000 |
| **2016** | 103.764867 | 122.345468 | 75.286253 | 105.925926 |

In [84]:

```
#Plot all the factors on one graph
ax4 = main_ny_df1.plot(title = "Factors impacting home prices from 2009 - 2016", grid=1
, ls='--', figsize=(15,8))
line = plt.gca().get_lines()[0]
line.set_ls("-")
line.set_linewidth(3)
#plt.savefig("../figs/11_factors_2009_2016.png", bbox_inches='tight')
plt.show()
```




In [85]:

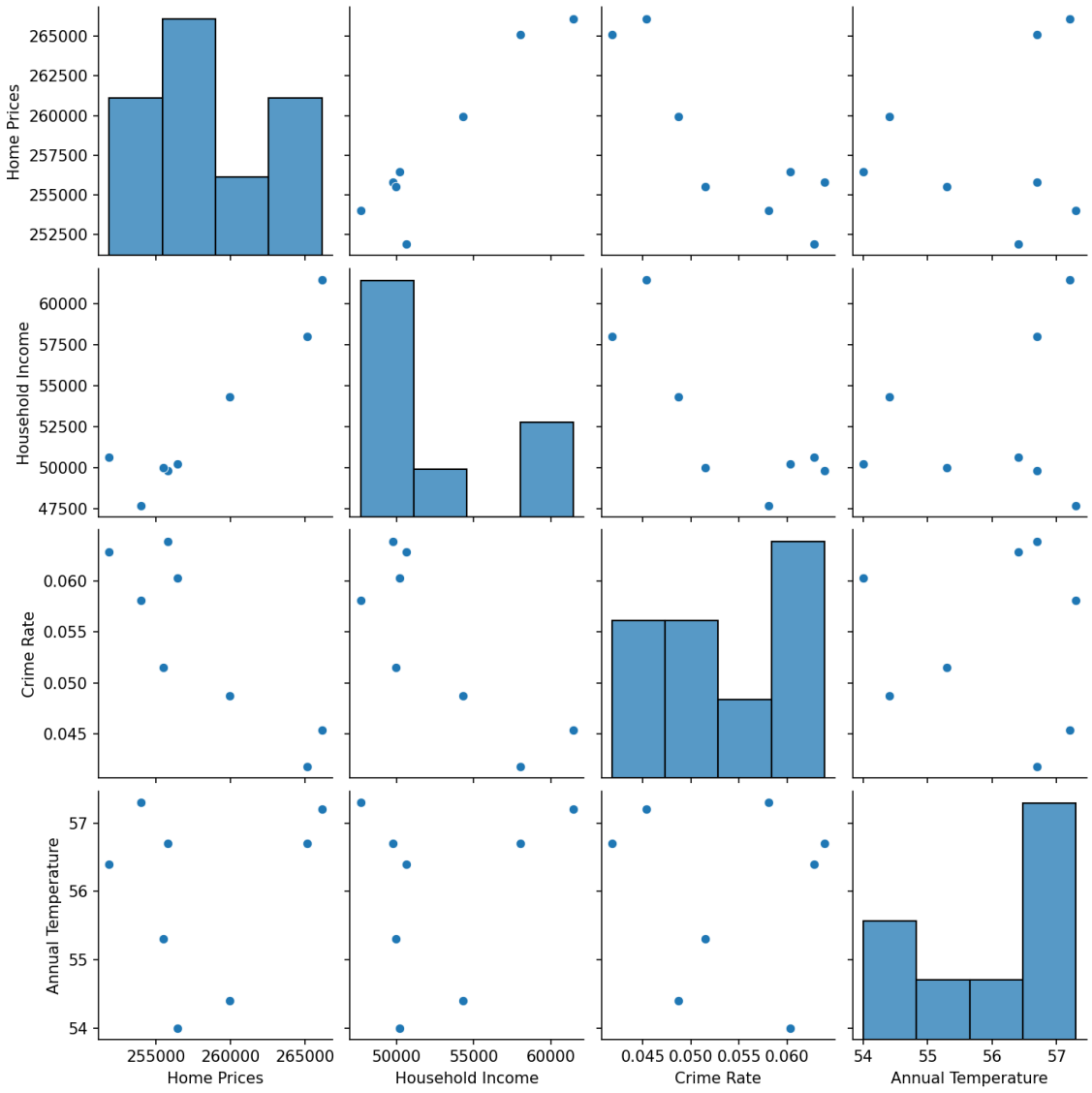
```
main_ny_df.describe()
```

Out[85]:

| | Home Prices | Household Income | Crime Rate | Annual Temperature |
|---|---|---|---|---|
| **count** | 8.000000 | 8.000000 | 8.000000 | 8.000000 |
| **mean** | 258106.760000 | 52753.875000 | 0.054068 | 56.000000 |
| **std** | 5165.643567 | 4761.049447 | 0.008357 | 1.271669 |
| **min** | 251890.580000 | 47680.000000 | 0.041803 | 54.000000 |
| **25%** | 255140.535000 | 49919.750000 | 0.047894 | 55.075000 |
| **50%** | 256131.675000 | 50426.000000 | 0.054787 | 56.550000 |
| **75%** | 261241.520000 | 55233.750000 | 0.060942 | 56.825000 |
| **max** | 266112.370000 | 61437.000000 | 0.063891 | 57.300000 |

In [86]:

```
ny_pairplot = sb.pairplot(main_ny_df)
#ny_pairplot.savefig("../figs/12_NY_pair_plot.png")
```




In [87]:

```
#Linear Regression Result between Home Prices and Household Income
linregress(main_ny_df['Home Prices'], main_ny_df['Household Income'])
```

Out[87]:

```
LinregressResult(slope=0.8641227974702729, intercept=-170282.0604971883, r
value=0.9375559777960977, pvalue=0.0005805612168909876, stderr=0.130880575
06870468)
```

In [88]:

```
#Linear Regression Result between Home Prices and Crime Rate
linregress(main_ny_df['Home Prices'], main_ny_df['Crime Rate'])
```

Out[88]:

```
LinregressResult(slope=-1.4043131257419601e-06, intercept=0.41653034247653
31, rvalue=-0.8680087138616346, pvalue=0.005194712596140597, stderr=3.2796
41128850178e-07)
```

In [89]:

```
#Linear Regression Result between Home Prices and Weather
linregress(main_ny_df['Home Prices'], main_ny_df['Annual Temperature'])
```

Out[89]:

```
LinregressResult(slope=3.876459588682571e-05, intercept=45.99459575294209,
rvalue=0.1574655316150324, pvalue=0.7095963486644558, stderr=9.92480548812
3337e-05)
```

In [90]:

```
main_ny_corr = main_ny_df.corr()
main_ny_corr
```

Out[90]:

| | Home Prices | Household Income | Crime Rate | Annual Temperature |
|---|---|---|---|---|
| **Home Prices** | 1.000000 | 0.937556 | -0.868009 | 0.157466 |
| **Household Income** | 0.937556 | 1.000000 | -0.806740 | 0.214925 |
| **Crime Rate** | -0.868009 | -0.806740 | 1.000000 | -0.074356 |
| **Annual Temperature** | 0.157466 | 0.214925 | -0.074356 | 1.000000 |

In [91]:

```
# plot the heatmap
ny_heatmap = plt.figure(figsize=(8,6))
sb.heatmap(main_ny_corr, xticklabels=main_ny_corr.columns, yticklabels=main_ny_corr.col
umns, cmap='GnBu', annot = True)
#ny_heatmap.savefig("../figs/13_NY_heatmap.png", bbox_inches='tight')
```


Out[91]:

<AxesSubplot:>

# Prediction

In [92]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [93]:

```
train1 = main_ny_df2.drop(['Home Prices','Year'],axis=1)
train1 = train1.rename(columns={'Household Income':'HI','Crime Rate':'CR'})
train1
```

Out[93]:

| | HI | CR |
|---|---|---|
| **0** | 50216 | 0.060312 |
| **1** | 49781 | 0.063891 |
| **2** | 50636 | 0.062831 |
| **3** | 47680 | 0.058073 |
| **4** | 49966 | 0.051501 |
| **5** | 54310 | 0.048723 |
| **6** | 58005 | 0.041803 |
| **7** | 61437 | 0.045407 |

In [94]:

```
# For simplicity, I'll be using X and y to denote the feature and target variables.
X = train1
y = main_ny_df2['Home Prices']
y
```

Out[94]:

```
0    256457.10
1    255806.25
2    251890.58
3    254002.98
4    255519.72
5    259950.50
6    265114.58
7    266112.37
Name: Home Prices, dtype: float64
```

In [95]:

```
# Add some extra constant term to allow statsmodels to calculate the bias.
X_constant = sm.add_constant(X)
X_constant
```

Out[95]:

| | const | HI | CR |
|---|---|---|---|
| **0** | 1.0 | 50216 | 0.060312 |
| **1** | 1.0 | 49781 | 0.063891 |
| **2** | 1.0 | 50636 | 0.062831 |
| **3** | 1.0 | 47680 | 0.058073 |
| **4** | 1.0 | 49966 | 0.051501 |
| **5** | 1.0 | 54310 | 0.048723 |
| **6** | 1.0 | 58005 | 0.041803 |
| **7** | 1.0 | 61437 | 0.045407 |

In [96]:

```
# Instantiate and fit our model with an ordinary least square model
model = sm.OLS(y, X_constant)
lin_reg = model.fit()
```

In [97]:

```
lin_reg.summary()
```

```
C:\Users\yinwe\Anaconda3\lib\site-packages\scipy\stats\stats.py:1603: User
Warning: kurtosistest only valid for n>=20 ... continuing anyway, n=8
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[97]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Home Prices | **R-squared:** | 0.915 |
| **Model:** | OLS | **Adj. R-squared:** | 0.881 |
| **Method:** | Least Squares | **F-statistic:** | 26.81 |
| **Date:** | Mon, 14 Dec 2020 | **Prob (F-statistic):** | 0.00212 |
| **Time:** | 23:14:10 | **Log-Likelihood:** | -69.369 |
| **No. Observations:** | 8 | **AIC:** | 144.7 |
| **Df Residuals:** | 5 | **BIC:** | 145.0 |
| **Df Model:** | 2 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **const** | 2.299e+05 | 1.91e+04 | 12.020 | 0.000 | 1.81e+05 | 2.79e+05 |
| **HI** | 0.7374 | 0.240 | 3.075 | 0.028 | 0.121 | 1.354 |
| **CR** | -1.976e+05 | 1.37e+05 | -1.447 | 0.208 | -5.49e+05 | 1.54e+05 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.209 | **Durbin-Watson:** | 2.412 |
| **Prob(Omnibus):** | 0.331 | **Jarque-Bera (JB):** | 0.722 |
| **Skew:** | -0.732 | **Prob(JB):** | 0.697 |
| **Kurtosis:** | 2.843 | **Cond. No.** | 1.16e+07 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.16e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [98]:

```
f_model = smf.ols(formula = 'y ~ HI + CR', data=train1)
f_lin_reg = f_model.fit()
f_lin_reg.summary()
```

Out[98]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **R-squared:** | 0.915 |
| **Model:** | OLS | **Adj. R-squared:** | 0.881 |
| **Method:** | Least Squares | **F-statistic:** | 26.81 |
| **Date:** | Mon, 14 Dec 2020 | **Prob (F-statistic):** | 0.00212 |
| **Time:** | 23:14:10 | **Log-Likelihood:** | -69.369 |
| **No. Observations:** | 8 | **AIC:** | 144.7 |
| **Df Residuals:** | 5 | **BIC:** | 145.0 |
| **Df Model:** | 2 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 2.299e+05 | 1.91e+04 | 12.020 | 0.000 | 1.81e+05 | 2.79e+05 |
| **HI** | 0.7374 | 0.240 | 3.075 | 0.028 | 0.121 | 1.354 |
| **CR** | -1.976e+05 | 1.37e+05 | -1.447 | 0.208 | -5.49e+05 | 1.54e+05 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.209 | **Durbin-Watson:** | 2.412 |
| **Prob(Omnibus):** | 0.331 | **Jarque-Bera (JB):** | 0.722 |
| **Skew:** | -0.732 | **Prob(JB):** | 0.697 |
| **Kurtosis:** | 2.843 | **Cond. No.** | 1.16e+07 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.16e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [99]:

```
# Predict house prices for 10 years (2009 - 2018)
predict_lin_reg = lin_reg.predict(X_constant[:10])
print(predict_lin_reg)
predict_f_lin_reg = f_lin_reg.predict(X_constant[:10])
print(predict_f_lin_reg)
```

```
0    255001.273666
1    253973.334129
2    254813.104519
3    253573.968562
4    256558.375831
5    260310.409276
6    264402.611256
7    266221.002762
dtype: float64
0    255001.273666
1    253973.334129
2    254813.104519
3    253573.968562
4    256558.375831
5    260310.409276
6    264402.611256
7    266221.002762
dtype: float64
```

In [100]:

```
from sklearn.metrics import r2_score
```

In [101]:

```
# BASE REGRESSION
linear_reg = smf.ols(formula = 'y ~ HI + CR', data=train1)
base = linear_reg.fit()
print(r2_score(y, base.predict(train1)))
```

```
0.9147090389953483
```

In [102]:

```
# WITHOUT HOUSEHOLD INCOME
linear_reg = smf.ols(formula = 'y ~ CR', data=train1)
base = linear_reg.fit()
print(r2_score(y, base.predict(train1)))
```

```
0.7534391273397281
```

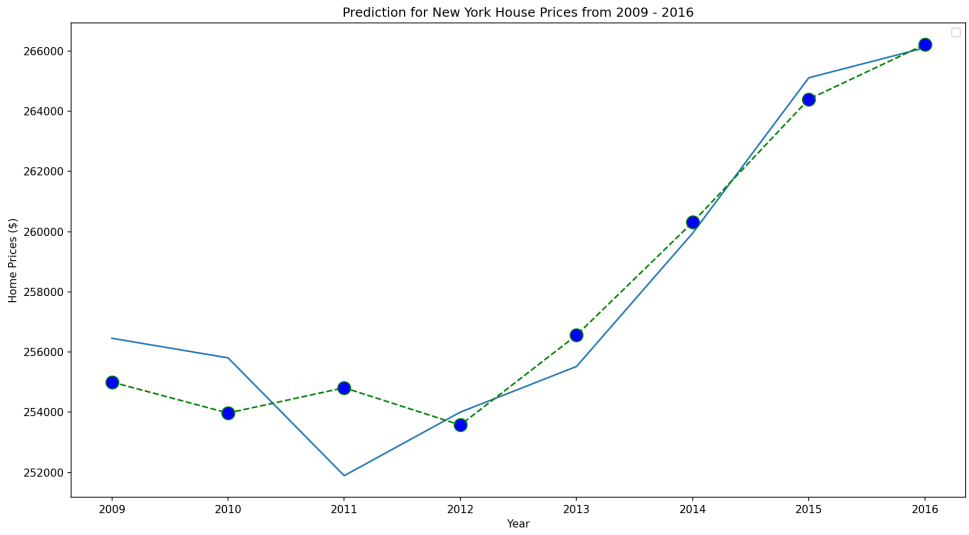
In [103]:

```
# WITHOUT CRIME RATE
linear_reg = smf.ols(formula = 'y ~ HI', data=train1)
base = linear_reg.fit()
print(r2_score(y, base.predict(train1)))
```

```
0.8790112115011977
```

In [104]:

```
#Plot the prediction and actual house prices on the same plot
main_ny_df2['Predicted Home Prices'] = predict_lin_reg
plt.figure(figsize=(15,8))
plt.plot(main_ny_df2['Year'],main_ny_df2['Home Prices'])
plt.plot(main_ny_df2['Year'],main_ny_df2['Predicted Home Prices'], color='green', lines
tyle='dashed', marker='o', markerfacecolor='blue', markersize=12)
plt.xlabel('Year')
plt.ylabel('Home Prices ($)')
plt.title('Prediction for New York House Prices from 2009 - 2016')
plt.legend()
#plt.savefig("../figs/14_predict_ny_house_prices_2009_2016.png", bbox_inches='tight')
plt.show()
```




No handles with labels found to put in legend.

In [ ]: