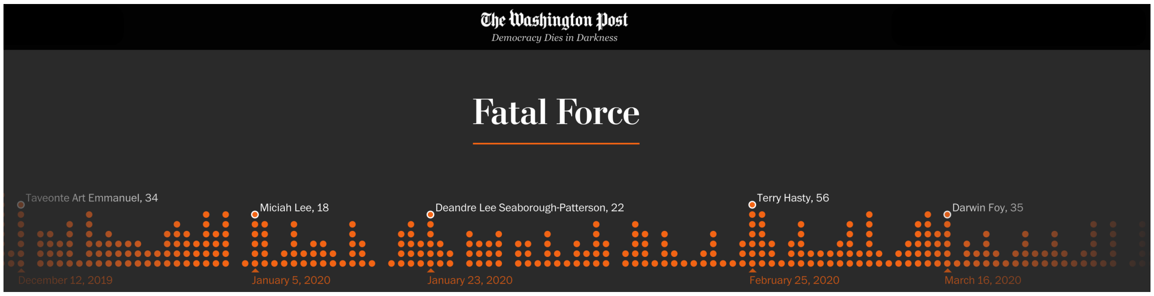
Introduction

Since Jan. 1, 2015, The Washington Post has been compiling a database of every fatal shooting in the US by a police officer in the line of duty.



While there are many challenges regarding data collection and reporting, The Washington Post has been tracking more than a dozen details about each killing. This includes the race, age and gender of the deceased, whether the person was armed, and whether the victim was experiencing a mental-health crisis. The Washington Post has gathered this supplemental information from law enforcement websites, local new reports, social media, and by monitoring independent databases such as "Killed by police" and "Fatal Encounters". The Post has also conducted additional reporting in many cases.

There are 4 additional datasets: US census data on poverty rate, high school graduation rate, median household income, and racial demographics. Source of census data.

Upgrade Plotly

Run the cell below if you are working with Google Colab

In [ ]:
```
# %pip install --upgrade plotly
```

Import Statements

In [ ]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns

# This might be helpful:
from collections import Counter
```

Notebook Presentation

In [ ]:
```
pd.options.display.float_format = '{:,.2f}'.format
```

Load the Data

Make file retrieval process more robust w/ `os` stuff:

In [ ]:
```
import os
```

In [ ]:
```
curr_dir = os.getcwd()
data_dir = os.path.join(curr_dir, 'data')

data_dir
```

Out[ ]:
```
'e:\\nowGitRepos\\dataLawUS\\data'
```

In [ ]:
```
enc = 'windows-1252'

hh_income_csv_path = os.path.join(data_dir, "Median_Household_Income_2015.csv")
df_hh_income = pd.read_csv(hh_income_csv_path, encoding=enc)

pct_poverty_csv_path = os.path.join(data_dir, "Pct_People_Below_Poverty_Level.csv")
df_pct_poverty = pd.read_csv(pct_poverty_csv_path, encoding=enc)

pct_completed_hs_csv_path = os.path.join(data_dir, "Pct_Over_25_Completed_High_Scho
df_pct_completed_hs = pd.read_csv(pct_completed_hs_csv_path, encoding=enc)

share_race_city_csv_path = os.path.join(data_dir, "Share_of_Race_By_City.csv")
df_share_race_city = pd.read_csv(share_race_city_csv_path, encoding=enc)

fatalities_csv_path = os.path.join(data_dir, "Deaths_by_Police_US.csv")
df_fatalities = pd.read_csv(fatalities_csv_path, encoding=enc)
```

Preliminary Data Exploration

- What is the shape of the DataFrames?

In [ ]:
```
print(f"`df_hh_income` shape: {df_hh_income.shape}")
print(f"`df_pct_poverty` shape: {df_pct_poverty.shape}")
print(f"`df_pct_completed_hs` shape: {df_pct_completed_hs.shape}")
print(f"`df_share_race_city` shape: {df_share_race_city.shape}")
print(f"`df_fatalities` shape: {df_fatalities.shape}")
```

```
`df_hh_income` shape: (29322, 3)
`df_pct_poverty` shape: (29329, 3)
`df_pct_completed_hs` shape: (29329, 3)
`df_share_race_city` shape: (29268, 7)
`df_fatalities` shape: (2535, 14)
```

- How many rows and columns do they have?

In [ ]:
```
df_hh_income_columns, df_hh_income_rows = df_hh_income.shape
print(f"df_hh_income cols: {df_hh_income_columns}")
print(f"df_hh_income rows: {df_hh_income_rows}\n")

df_pct_poverty_columns, df_pct_poverty_rows = df_pct_poverty.shape
```

```
print(f"df_pct_poverty cols: {df_pct_poverty_columns}")
print(f"df_pct_poverty rows: {df_pct_poverty_rows}\n")

df_pct_completed_hs_columns, df_pct_completed_hs_rows = df_pct_completed_hs.shape
print(f"df_pct_completed_hs cols: {df_pct_completed_hs_columns}")
print(f"df_pct_completed_hs rows: {df_pct_completed_hs_rows}\n")

df_share_race_city_columns, df_share_race_city_rows = df_share_race_city.shape
print(f"df_share_race_city cols: {df_share_race_city_columns}")
print(f"df_share_race_city rows: {df_share_race_city_rows}\n")

df_fatalities_columns, df_fatalities_rows = df_fatalities.shape
print(f"df_fatalities cols: {df_fatalities_columns}")
print(f"df_fatalities rows: {df_fatalities_rows}\n")
```

```
df_hh_income cols: 29322
df_hh_income rows: 3

df_pct_poverty cols: 29329
df_pct_poverty rows: 3

df_pct_completed_hs cols: 29329
df_pct_completed_hs rows: 3

df_share_race_city cols: 29268
df_share_race_city rows: 7

df_fatalities cols: 2535
df_fatalities rows: 14
```

Analysis:

The first four DFs have similar number of rows.

The second and third DFs have the exact same number of rows.

The first three have the same number of columns.

The last has many fewer rows, but more columns.

- What are the column names?

In [ ]:
```
def format_col_names(dataframe):
    col_list = dataframe.columns
    col_name_str = ", ".join(col_list)
    return col_name_str
```

In [ ]:
```
df_hh_income_cols = format_col_names(df_hh_income)
print(f"`df_hh_income` col names: {df_hh_income_cols}\n")

df_pct_poverty_cols = format_col_names(df_pct_poverty)
print(f"`df_pct_poverty` col names: {df_pct_poverty_cols}\n")

df_pct_completed_hs_cols = format_col_names(df_pct_completed_hs)
print(f"`df_pct_completed_hs` col names: {df_pct_completed_hs_cols}\n")
```

```
df_share_race_city_cols = format_col_names(df_share_race_city)
print(f"`df_share_race_city` col names: {df_share_race_city_cols}\n")

df_fatalities_cols = format_col_names(df_fatalities)
print(f"`df_fatalities` col names: {df_fatalities_cols}\n")
```

```
`df_hh_income` col names: Geographic Area, City, Median Income

`df_pct_poverty` col names: Geographic Area, City, poverty_rate

`df_pct_completed_hs` col names: Geographic Area, City, percent_completed_hs

`df_share_race_city` col names: Geographic area, City, share_white, share_black, sha
re_native_american, share_asian, share_hispanic

`df_fatalities` col names: id, name, date, manner_of_death, armed, age, gender, race
, city, state, signs_of_mental_illness, threat_level, flee, body_camera
```

- Are there any NaN values or duplicates?

In [ ]:
```
print(f"`NaN` info for `df_hh_income`: {df_hh_income.isna().sum()}\n")
print(f"Duplicate # for `df_hh_income`: {df_hh_income.duplicated().sum()}\n")
```

```
`NaN` info for `df_hh_income`: Geographic Area     0
City                0
Median Income      51
dtype: int64

Duplicate # for `df_hh_income`: 0
```

In [ ]:
```
print(f"`NaN` info for `df_pct_poverty`: {df_pct_poverty.isna().sum()}\n")
print(f"Duplicate # for `df_pct_poverty`: {df_pct_poverty.duplicated().sum()}\n")
```

```
`NaN` info for `df_pct_poverty`: Geographic Area    0
City               0
poverty_rate       0
dtype: int64

Duplicate # for `df_pct_poverty`: 0
```

In [ ]:
```
print(f"`NaN` info for `df_pct_completed_hs`: {df_pct_completed_hs.isna().sum()}\n"
print(f"Duplicate # for `df_pct_completed_hs`: {df_pct_completed_hs.duplicated().su
```

```
`NaN` info for `df_pct_completed_hs`: Geographic Area         0
City                    0
percent_completed_hs    0
dtype: int64

Duplicate # for `df_pct_completed_hs`: 0
```

In [ ]:
```
print(f"`NaN` info for `df_share_race_city`: {df_share_race_city.isna().sum()}\n")
print(f"Duplicate # for `df_share_race_city`: {df_share_race_city.duplicated().sum(
```

```
`NaN` info for `df_share_race_city`: Geographic area          0
City                     0
share_white              0
share_black              0
share_native_american    0
share_asian              0
share_hispanic           0
dtype: int64

Duplicate # for `df_share_race_city`: 0
```

In [ ]:
```
print(f"`NaN` info for `df_fatalities`: {df_fatalities.isna().sum()}\n")
print(f"Duplicate # for `df_fatalities`: {df_fatalities.duplicated().sum()}\n")
```

```
`NaN` info for `df_fatalities`: id                           0
name                         0
date                         0
manner_of_death              0
armed                        9
age                         77
gender                       0
race                       195
city                         0
state                        0
signs_of_mental_illness      0
threat_level                 0
flee                        65
body_camera                  0
dtype: int64

Duplicate # for `df_fatalities`: 0
```

Analysis:

df_hh_income and df_fatalities have NaN issues.

None of the DFs have duplicates.

Data Cleaning - Check for Missing Values and Duplicates

Consider how to deal with the NaN values. Perhaps substituting 0 is appropriate.

In [ ]:
```
# Before fixing NaNs:
df_hh_income.isna().sum()
```

Out[ ]:
```
Geographic Area     0
City                0
Median Income      51
dtype: int64
```

In [ ]:
```
df_hh_income['Median Income'] = df_hh_income['Median Income'].fillna(0)
```

In [ ]:
```
# After fixing NaNs:
df_hh_income.isna().sum()
```

Out[ ]:
```
Geographic Area    0
City               0
Median Income      0
dtype: int64
```

In [ ]:
```
# Before fixing NaNs:
df_fatalities.isna().sum()
```

Out[ ]:
```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      9
age                       77
gender                     0
race                     195
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                      65
body_camera                0
dtype: int64
```

In [ ]:
```
df_fatalities['armed'] = df_fatalities['armed'].fillna(0)
df_fatalities['age'] = df_fatalities['age'].fillna(0)
df_fatalities['race'] = df_fatalities['race'].fillna(0)
df_fatalities['flee'] = df_fatalities['flee'].fillna(0)
```

In [ ]:
```
# After fixing NaNs:
df_fatalities.isna().sum()
```

Out[ ]:
```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      0
age                        0
gender                     0
race                       0
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                       0
body_camera                0
dtype: int64
```

Analysis:

All `NaN` s handled.

No duplicates to be handled.

Chart the Poverty Rate in each US State

Create a bar chart that ranks the poverty rate from highest to lowest by US state. Which state has the highest poverty rate? Which state has the lowest poverty rate? Bar Plot

In [ ]:
```
# df_hh_income
# df_pct_poverty
# df_pct_completed_hs
# df_share_race_city
# df_fatalities
```

Check data types for column `poverty_rate`:

In [ ]:
```
df_pct_poverty.dtypes
```

Out[ ]:
```
Geographic Area    object
City               object
poverty_rate       object
dtype: object
```

Check all strings are numbers:

In [ ]:
```
# for i in df_pct_poverty.poverty_rate:
#     print(i)
```

Some values are `-`, let's replace them with `0`:

In [ ]:
```
df_pct_poverty["poverty_rate"] = df_pct_poverty["poverty_rate"].replace("-", "0")
```

Convert that column from str/object to numeric/int:

In [ ]:
```
df_pct_poverty["poverty_rate"] = pd.to_numeric(df_pct_poverty["poverty_rate"])
```

Remove rows w/0 (no data):

In [ ]:
```
df_pct_poverty = df_pct_poverty[df_pct_poverty["poverty_rate"] != 0]
```

Two ways to aggregate data:

In [ ]:
```
df_pov_state = df_pct_poverty.groupby("Geographic Area")["poverty_rate"].mean().res

# Or:
# df_pov_state = df_pct_poverty.groupby("Geographic Area").agg({"poverty_rate": 'me
```

Sort and reset indices:

In [ ]:
```
df_pov_state.sort_values(by="poverty_rate", inplace=True, ascending=False)
df_pov_state.reset_index(inplace=True, drop=True)
```

Verify the new DF:

In [ ]:
```
# df_pov_state
```

Colors I like for a white background:

In [ ]:
```
seaborn_palettes = ['deep', 'muted', 'pastel', 'bright', 'dark', 'colorblind']
```

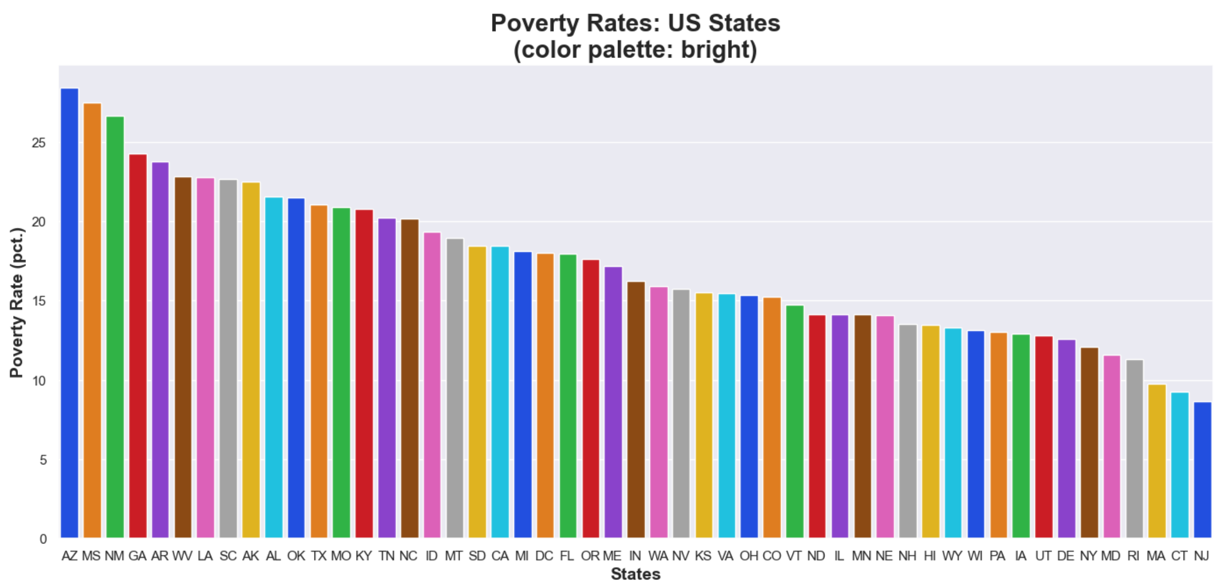
In [ ]:
```
from matplotlib import colormaps
all_palettes = list(colormaps) + seaborn_palettes
```

In [ ]:
```
pov_states = df_pov_state["Geographic Area"]
pov_rate = df_pov_state["poverty_rate"]
```

In [ ]:
```
from random import choice, choices
```

w/Seaborn:

In [ ]:
```
# Initialize MatPlotLib figure
fig, ax = plt.subplots(figsize=(17, 7))

palette = choice(seaborn_palettes)

# Plot the months/counts:
# sns.barplot(x=pov_states, y=pov_rate, hue=pov_states, palette=palette, legend='au
sns.barplot(
    df_pov_state,
    x="Geographic Area",
    y="poverty_rate",
    hue="Geographic Area",
    palette=palette,
)

plt.xlabel("States", size=14, weight='bold')
plt.ylabel("Poverty Rate (pct.)", size=14, weight='bold')

plt.title(f"Poverty Rates: US States\n(color palette: {palette})",
          size=20,
          weight='bold')

plt.show()
```



In [ ]:
```
colors = ["blue",
          "navy",
          "peru",
          "black",
          "brown",
          "green",
```

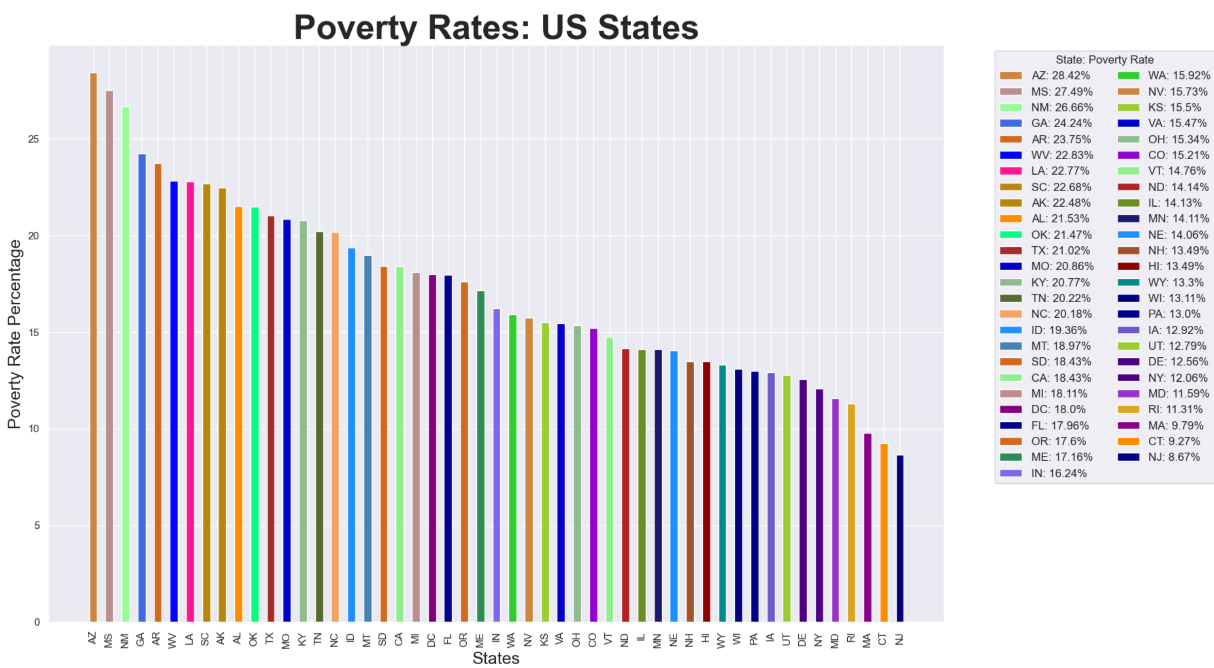
```
          "olive",
          "indigo",
          "maroon",
          "purple",
          "sienna",
          "crimson",
          "darkred",
          "dimgray",
          "darkblue",
          "darkcyan",
          "deeppink",
          "seagreen",
          "chocolate",
          "darkgreen",
          "darkkhaki",
          "firebrick",
          "goldenrod",
          "limegreen",
          "olivedrab",
          "palegreen",
          "rosybrown",
          "royalblue",
          "slateblue",
          "steelblue",
          "darkorange",
          "darkorchid",
          "darksalmon",
          "darkviolet",
          "dodgerblue",
          "lightgreen",
          "mediumblue",
          "sandybrown",
          "darkmagenta",
          "forestgreen",
          "greenyellow",
          "saddlebrown",
          "springgreen",
          "yellowgreen",
          "darkseagreen",
          "midnightblue",
          "darkgoldenrod",
          "darkslateblue",
          "darkslategray",
          "darkslategrey",
          "darkturquoise",
          "rebeccapurple",
          "cornflowerblue",
          "darkolivegreen",
          "mediumseagreen",
          "mediumslateblue",
          "mediumspringgreen"]
```

In [ ]:
```
bar_colors = choices(colors, k=len(df_pov_state))
```

In [ ]:
```
# Create labels for Legend:
state_rate = zip(pov_states, pov_rate)
labels_state_rate = [f"{state}: {round(rate, 2)}%" for state, rate in state_rate]
```

w/Matplotlib and descriptive legend:

```
In [ ]: fig, ax = plt.subplots(figsize=(17, 11))

ax.bar(x=pov_states,
       height=pov_rate,
       label=labels_state_rate,
       color=bar_colors,
       width=0.5)

ax.set_title("Poverty Rates: US States", fontsize=36, weight='bold')
ax.set_xlabel("States", fontsize=18)
ax.set_ylabel("Poverty Rate Percentage", fontsize=18)

plt.legend(ncol=2,
           title="State: Poverty Rate",
           fontsize='medium',
           bbox_to_anchor=(1.05, 1))

plt.xticks(rotation=90)

plt.show()
```



Analysis:

In [ ]:
```
df_pov_state.describe()
```

Out[ ]:

	poverty_rate
count	51.00
mean	17.24
std	4.74
min	8.67
25%	13.49
50%	16.24
75%	20.82
max	28.42

Which state has the highest poverty rate?

- Mississippi (MS): 26.88%

Which state has the lowest poverty rate?

- New Jersey (NJ): 8.16%

Middling:

- Indiana (IN): 15.5%

Chart the High School Graduation Rate by US State

Show the High School Graduation Rate in ascending order of US States.

Which state has the lowest high school graduation rate?

Which state has the highest?

Do the exact same as above:

Clean/aggregate new dataframe:

In [ ]:
```
df_pct_completed_hs.columns
```

Out[ ]:
```
Index(['Geographic Area', 'City', 'percent_completed_hs'], dtype='object')
```

In [ ]:
```
df_pct_completed_hs.dtypes
```

```
Out[ ]: Geographic Area         object
        City                    object
        percent_completed_hs    object
        dtype: object
```

```
In [ ]: for i in df_pct_completed_hs.percent_completed_hs:
            print(i)
```

```
In [ ]: df_pct_completed_hs["percent_completed_hs"] = df_pct_completed_hs["percent_complete
```

```
In [ ]: df_pct_completed_hs["percent_completed_hs"] = pd.to_numeric(df_pct_completed_hs["pe
```

```
In [ ]: df_pct_completed_hs = df_pct_completed_hs[df_pct_completed_hs["percent_completed_hs
```

```
In [ ]: # df_pct_completed_hs
```

```
In [ ]: df_hs_state = df_pct_completed_hs.groupby("Geographic Area")["percent_completed_hs"

        # Or:
        # df_hs_state = df_pct_completed_hs.groupby("Geographic Area").agg({"percent_comple
```

```
In [ ]: # df_hs_state
        # df_pct_completed_hs
        # percent_completed_hs
```

```
In [ ]: df_hs_state.sort_values(by="percent_completed_hs", inplace=True, ascending=True)
        df_hs_state.reset_index(inplace=True, drop=True)
```

```
In [ ]: # df_hs_state
```

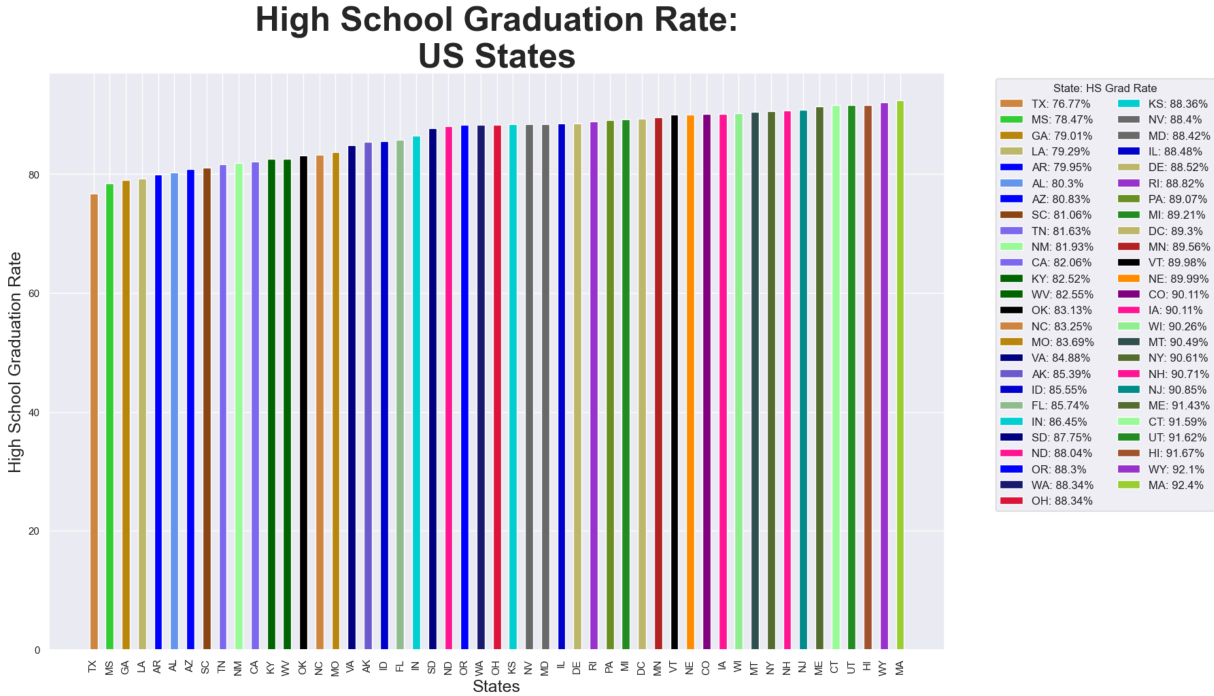
Plot:

```
In [ ]: colors = ["black",
                  "blue",
                  "brown",
                  "chocolate",
                  "cornflowerblue",
                  "crimson",
                  "darkblue",
                  "darkcyan",
                  "darkgoldenrod",
                  "darkgreen",
                  "darkkhaki",
                  "darkmagenta",
                  "darkolivegreen",
                  "darkorange",
                  "darkorchid",
                  "darkred",
                  "darksalmon",
                  "darkseagreen",
                  "darkslateblue",
                  "darkslategray",
                  "darkslategrey",
                  "darkturquoise",
                  "darkviolet",
                  "deeppink",
                  "dimgray",
                  "dodgerblue",
                  "firebrick",
```

```
        "forestgreen",
        "goldenrod",
        "green",
        "greenyellow",
        "indigo",
        "lightgreen",
        "limegreen",
        "maroon",
        "mediumblue",
        "mediumseagreen",
        "mediumslateblue",
        "mediumspringgreen",
        "midnightblue",
        "navy",
        "olive",
        "olivedrab",
        "palegreen",
        "peru",
        "purple",
        "rebeccapurple",
        "rosybrown",
        "royalblue",
        "saddlebrown",
        "sandybrown",
        "seagreen",
        "sienna",
        "slateblue",
        "springgreen",
        "steelblue",
        "yellowgreen"]
```

In [ ]:
```
states = df_hs_state["Geographic Area"]

grad = df_hs_state["percent_completed_hs"]

# Create labels for legend:
state_grad = zip(states, grad)
labels_state_grad = [f"{state}: {round(grad, 2)}%" for state, grad in state_grad]
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(17, 11))

bar_colors = choices(colors, k=len(df_hs_state))

ax.bar(x=states,
       height=grad,
       label=labels_state_grad,
       color=bar_colors,
       width=0.5)

ax.set_title("High School Graduation Rate:\nUS States", fontsize=36, weight='bold')
ax.set_xlabel("States", fontsize=18)
ax.set_ylabel("High School Graduation Rate", fontsize=18)

plt.legend(ncol=2,
           title="State: HS Grad Rate",
           fontsize='medium',
           bbox_to_anchor=(1.05, 1))

plt.xticks(rotation=90)

plt.show()
```



In [ ]:
```
df_pov_state.describe()
```

Out[ ]:

	poverty_rate
count	51.00
mean	17.24
std	4.74
min	8.67
25%	13.49
50%	16.24
75%	20.82
max	28.42

Analysis:

Which state has the lowest high school graduation rate?

- Texas (TX): 76.77%

Which state has the highest?

- Massachusetts (MA): 92.4%

Middle:

- Ohio (OH): 88.34%

Visualise the Relationship between

Poverty Rates and High School Graduation Rates

Create a line chart with two y-axes to show if the ratio of poverty and high school graduation move together.

Create DF with poverty and high school data merged on State names:

In [ ]:
```
df_state_hs_poverty = df_hs_state.merge(df_pov_state, on='Geographic Area')
```

In [ ]:
```
df_state_hs_poverty
```

Out[ ]:

	Geographic Area	percent_completed_hs	poverty_rate
0	TX	76.77	21.02
1	MS	78.47	27.49
2	GA	79.01	24.24
3	LA	79.29	22.77
4	AR	79.95	23.75
5	AL	80.30	21.53
6	AZ	80.83	28.42
7	SC	81.06	22.68
8	TN	81.63	20.22
9	NM	81.93	26.66
10	CA	82.06	18.43
11	KY	82.52	20.77
12	WV	82.55	22.83
13	OK	83.13	21.47
14	NC	83.25	20.18
15	MO	83.69	20.86
16	VA	84.88	15.47
17	AK	85.39	22.48
18	ID	85.55	19.36
19	FL	85.74	17.96
20	IN	86.45	16.24
21	SD	87.75	18.43
22	ND	88.04	14.14
23	OR	88.30	17.60
24	WA	88.34	15.92
25	OH	88.34	15.34
26	KS	88.36	15.50
27	NV	88.40	15.73
28	MD	88.42	11.59
29	IL	88.48	14.13
30	DE	88.52	12.56
31	RI	88.82	11.31
32	PA	89.07	13.00

	Geographic Area	percent_completed_hs	poverty_rate
33	MI	89.21	18.11
34	DC	89.30	18.00
35	MN	89.56	14.11
36	VT	89.98	14.76
37	NE	89.99	14.06
38	CO	90.11	15.21
39	IA	90.11	12.92
40	WI	90.26	13.11
41	MT	90.49	18.97
42	NY	90.61	12.06
43	NH	90.71	13.49
44	NJ	90.85	8.67
45	ME	91.43	17.16
46	CT	91.59	9.27
47	UT	91.62	12.79
48	HI	91.67	13.49
49	WY	92.10	13.30
50	MA	92.40	9.79

Inspect date:

In [ ]:
```
for i in range(len(df_state_hs_poverty)):
    state = df_state_hs_poverty['Geographic Area'][i]
    print(state)
    state_grad_rate = df_state_hs_poverty['percent_completed_hs'][i]
    print(state_grad_rate)
    state_pov_rate = df_state_hs_poverty['poverty_rate'][i]
    print(state_pov_rate)
```

Create column for state names for clarity on plot:

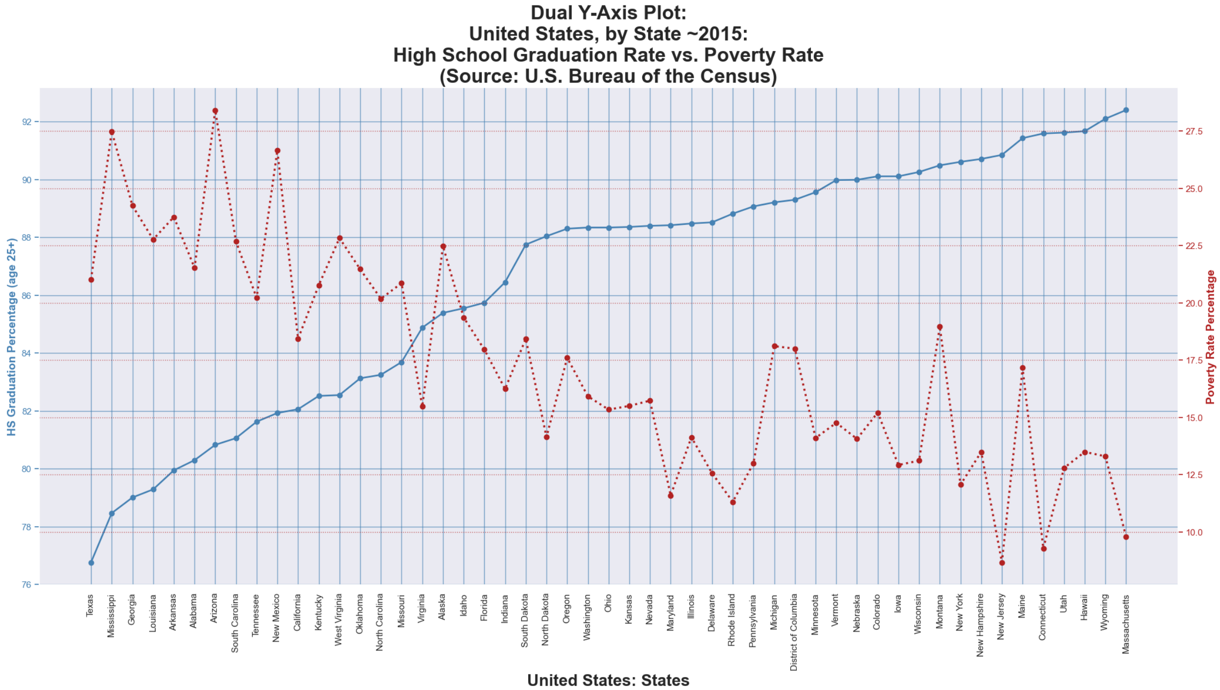
In [ ]:
```
state_dict = {"TX": "Texas", "MS": "Mississippi", "GA": "Georgia", "LA": "Louisiana

state_names= np.array([state_dict.get(i) for i in df_state_hs_poverty['Geographic A
                      dtype='object')

df_state_hs_poverty['State_Names'] = state_names
```

Do the thing:

In [ ]:
```
# Create axes:
fig, ax1 = plt.subplots(figsize=(25, 11))
ax2 = ax1.twinx()
```

```python
# Extract data from DataFrame, using state names instead of abbreviations:
states = df_state_hs_poverty['State_Names']
state_grad_rates = np.round(df_state_hs_poverty['percent_completed_hs'], 2)
state_pov_rates = np.round(df_state_hs_poverty['poverty_rate'], 2)

# Axes colors (find good complements):
ax1_color = 'steelblue'
ax2_color = 'firebrick'

# Plot left y-axis, Graduation Rate:
ax1.plot(states,
         state_grad_rates,
         label=f"{states}: {state_grad_rates}",
         linewidth=2,
         linestyle='-',
         marker='o',
         color=ax1_color)

# Plot right y-axis, Poverty Rate:
ax2.plot(states,
         state_pov_rates,
         label=f"{states}: {state_pov_rates}",
         linewidth=2.5,
         linestyle=':',
         marker='o',
         color=ax2_color)

# Make a title:
title = "Dual Y-Axis Plot:\n"
title += "United States, by State ~2015:\n"
title += "High School Graduation Rate vs. Poverty Rate\n"
title += "(Source: U.S. Bureau of the Census)"
ax1.set_title(title, fontsize=24, weight='bold')


# Make x-axis label:
ax1.set_xlabel("United States: States", fontsize=20, weight='bold')

# Make y-axes labels:
ax1.set_ylabel("HS Graduation Percentage (age 25+)",
                fontsize=14,
                weight='bold')

ax2.set_ylabel("Poverty Rate Percentage",
               fontsize=14,
               weight='bold')

# Customize each axes grid for ease of plot visualization:
ax1.grid(color=ax1_color, linestyle='-', alpha=0.7)
ax1.yaxis.label.set_color(ax1_color)
ax1.tick_params(axis='y', colors=ax1_color)

ax2.grid(color=ax2_color, linestyle=':', alpha=0.7)
ax2.yaxis.label.set_color(ax2_color)
ax2.tick_params(axis='y', colors=ax2_color)

# Rotate x-axis tick labels:
ax1.set_xticks(states)
ax1.set_xticklabels(states, rotation=90)

# Show the thing:
```

```python
plt.show()
```



Analysis:

There is a general trend when viewing the Graduation and Poverty data in this dual plot:

- The upward graduation rate trend is in stark contrast to a trend of falling poverty rates.
- My conclusion is that states with higher high school graduation rates tend to have lower poverty.

There are specific exceptions to this contrasting trend, but the general pattern is obvious.

Now use a Seaborn .jointplot() with a Kernel Density Estimate (KDE) and/or scatter plot to visualise the same relationship

In [ ]:
```python
df_state_hs_poverty
```

Out[ ]:

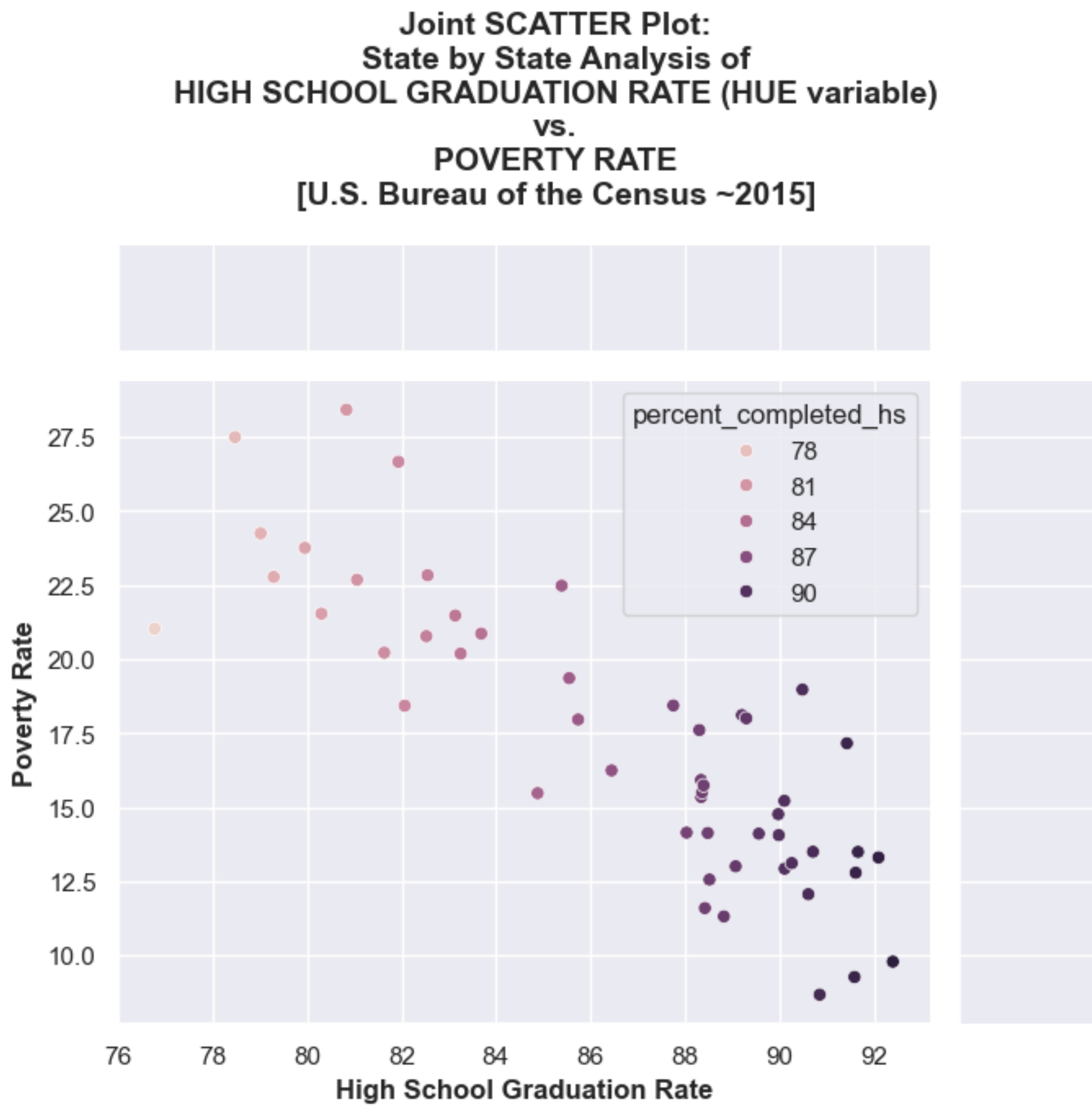
	Geographic Area	percent_completed_hs	poverty_rate	State_Names
0	TX	76.77	21.02	Texas
1	MS	78.47	27.49	Mississippi
2	GA	79.01	24.24	Georgia
3	LA	79.29	22.77	Louisiana
4	AR	79.95	23.75	Arkansas
5	AL	80.30	21.53	Alabama
6	AZ	80.83	28.42	Arizona
7	SC	81.06	22.68	South Carolina
8	TN	81.63	20.22	Tennessee
9	NM	81.93	26.66	New Mexico
10	CA	82.06	18.43	California
11	KY	82.52	20.77	Kentucky
12	WV	82.55	22.83	West Virginia
13	OK	83.13	21.47	Oklahoma
14	NC	83.25	20.18	North Carolina
15	MO	83.69	20.86	Missouri
16	VA	84.88	15.47	Virginia
17	AK	85.39	22.48	Alaska
18	ID	85.55	19.36	Idaho
19	FL	85.74	17.96	Florida
20	IN	86.45	16.24	Indiana
21	SD	87.75	18.43	South Dakota
22	ND	88.04	14.14	North Dakota
23	OR	88.30	17.60	Oregon
24	WA	88.34	15.92	Washington
25	OH	88.34	15.34	Ohio
26	KS	88.36	15.50	Kansas
27	NV	88.40	15.73	Nevada
28	MD	88.42	11.59	Maryland
29	IL	88.48	14.13	Illinois
30	DE	88.52	12.56	Delaware
31	RI	88.82	11.31	Rhode Island
32	PA	89.07	13.00	Pennsylvania

	Geographic Area	percent_completed_hs	poverty_rate	State_Names
33	MI	89.21	18.11	Michigan
34	DC	89.30	18.00	District of Columbia
35	MN	89.56	14.11	Minnesota
36	VT	89.98	14.76	Vermont
37	NE	89.99	14.06	Nebraska
38	CO	90.11	15.21	Colorado
39	IA	90.11	12.92	Iowa
40	WI	90.26	13.11	Wisconsin
41	MT	90.49	18.97	Montana
42	NY	90.61	12.06	New York
43	NH	90.71	13.49	New Hampshire
44	NJ	90.85	8.67	New Jersey
45	ME	91.43	17.16	Maine
46	CT	91.59	9.27	Connecticut
47	UT	91.62	12.79	Utah
48	HI	91.67	13.49	Hawaii
49	WY	92.10	13.30	Wyoming
50	MA	92.40	9.79	Massachusetts

Seaborn `joinplot()` : `kind=scatter` and `hue` set:

In [ ]:
```
with sns.axes_style('darkgrid'):
    grad_pov = sns.jointplot(
        df_state_hs_poverty,
        x='percent_completed_hs',
        y='poverty_rate',
        height=7,
        kind='scatter',
        hue='percent_completed_hs',
        legend='brief'
    )

sup_title = "Joint SCATTER Plot:\nState by State Analysis of\n"
sup_title += "HIGH SCHOOL GRADUATION RATE (HUE variable)\nvs.\nPOVERTY RATE\n"
sup_title += "[U.S. Bureau of the Census ~2015]"

grad_pov.figure.suptitle(sup_title,
                         size=14,
                         weight='bold')
grad_pov.figure.tight_layout()
# grad_pov.figure.subplots_adjust(top=0.865)

grad_pov.ax_joint.set_xlabel('High School Graduation Rate',
                             size=12,
```

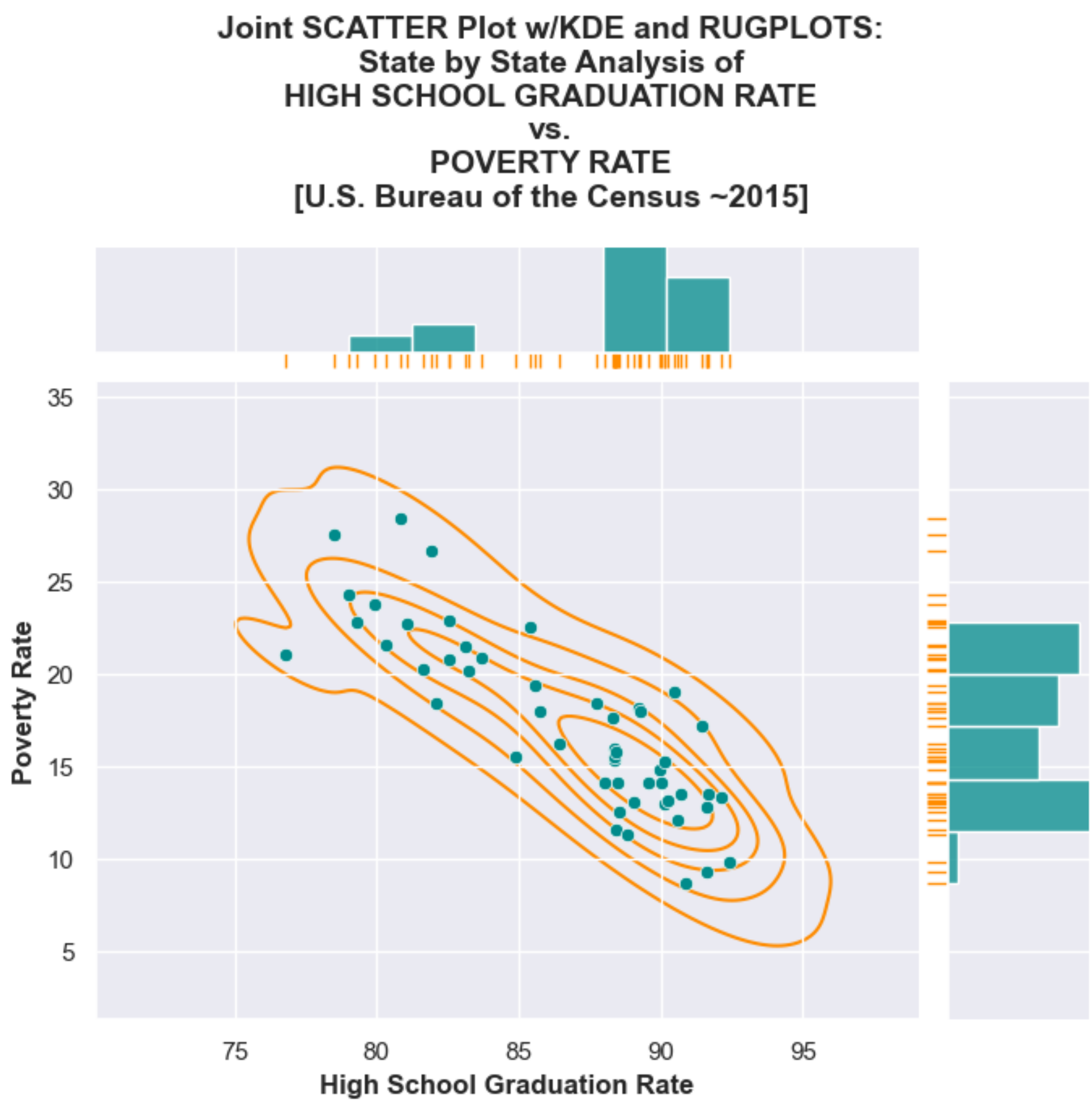
```
                                weight='bold')
grad_pov.ax_joint.set_ylabel('Poverty Rate',
                                size=12,
                                weight='bold')

plt.show()
```



Seaborn joinplot(): kind=scatter w/ kdeplot() & rugplot():

In [ ]:
```
with sns.axes_style('darkgrid'):
    grad_pov = sns.jointplot(
        df_state_hs_poverty,
        x='percent_completed_hs',
        y='poverty_rate',
        height=7,
        kind='scatter',
        color='darkcyan',
    )

grad_pov.plot_joint(sns.kdeplot, color="darkorange", zorder=0, levels=6)
grad_pov.plot_marginals(sns.rugplot, color="darkorange", height=-.15, clip_on=False

sup_title = "Joint SCATTER Plot w/KDE and RUGPLOTS:\n"
sup_title += "State by State Analysis of\n"
```

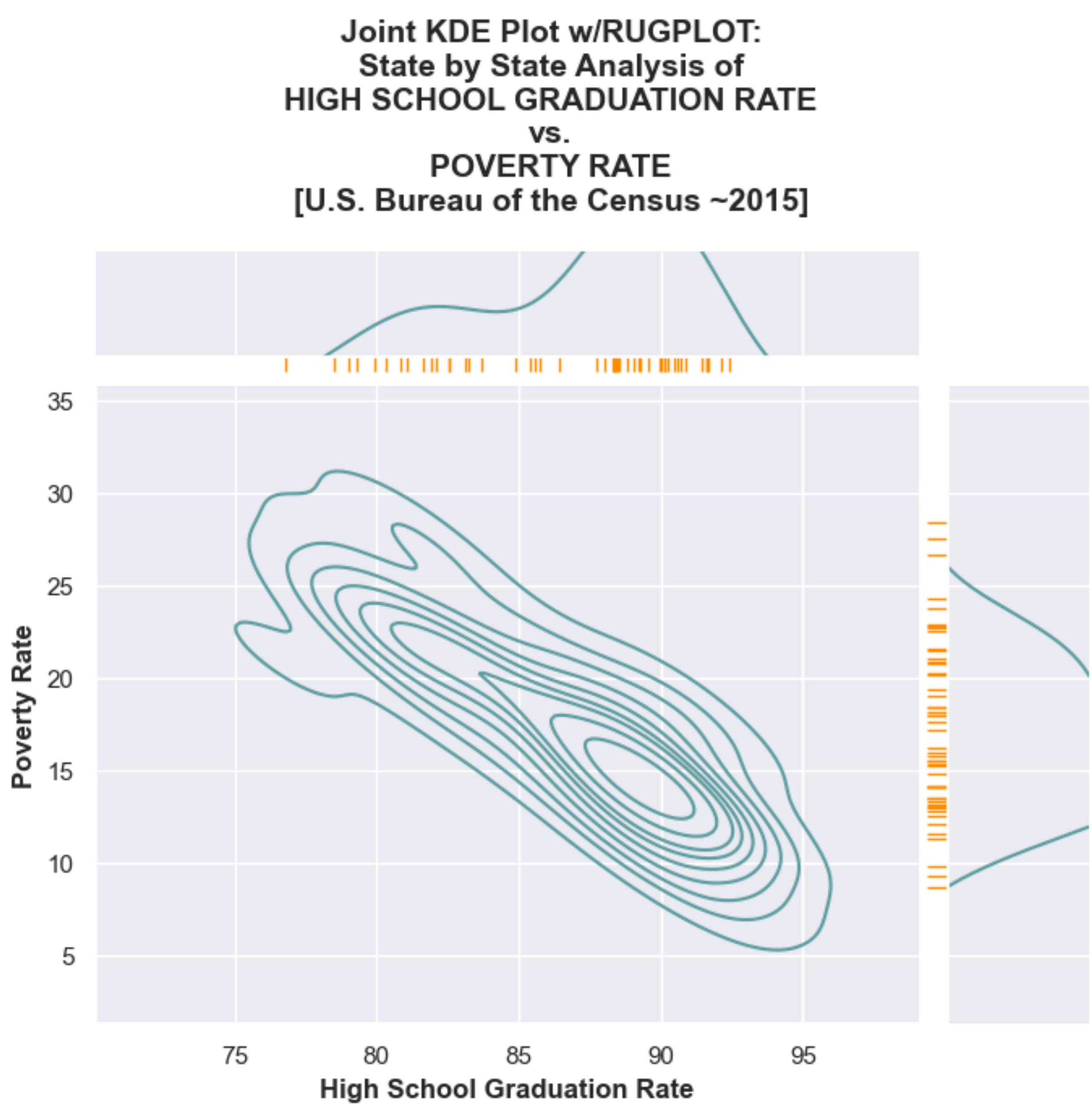
```
sup_title += "HIGH SCHOOL GRADUATION RATE\nvs.\nPOVERTY RATE\n"
sup_title += "[U.S. Bureau of the Census ~2015]"

grad_pov.figure.suptitle(sup_title,
                         size=14,
                         weight='bold')
grad_pov.figure.tight_layout()

grad_pov.ax_joint.set_xlabel('High School Graduation Rate',
                             size=12,
                             weight='bold')
grad_pov.ax_joint.set_ylabel('Poverty Rate',
                             size=12,
                             weight='bold')

plt.show()
```



Seaborn joinplot() : kind=kde w/ rugplot() :

In [ ]:
```
with sns.axes_style('darkgrid'):
    grad_pov = sns.jointplot(
        df_state_hs_poverty,
        x='percent_completed_hs',
        y='poverty_rate',
```

```
        height=7,
        kind='kde',
        color='cadetblue',
    )

grad_pov.plot_marginals(sns.rugplot, color="darkorange", height=-.15, clip_on=False

sup_title = "Joint KDE Plot w/RUGPLOT:\n"
sup_title += "State by State Analysis of\n"
sup_title += "HIGH SCHOOL GRADUATION RATE\nvs.\nPOVERTY RATE\n"
sup_title += "[U.S. Bureau of the Census ~2015]"

grad_pov.figure.suptitle(sup_title,
                         size=14,
                         weight='bold')
grad_pov.figure.tight_layout()

grad_pov.ax_joint.set_xlabel('High School Graduation Rate',
                             size=12,
                             weight='bold')
grad_pov.ax_joint.set_ylabel('Poverty Rate',
                             size=12,
                             weight='bold')

plt.show()
```



Analysis:

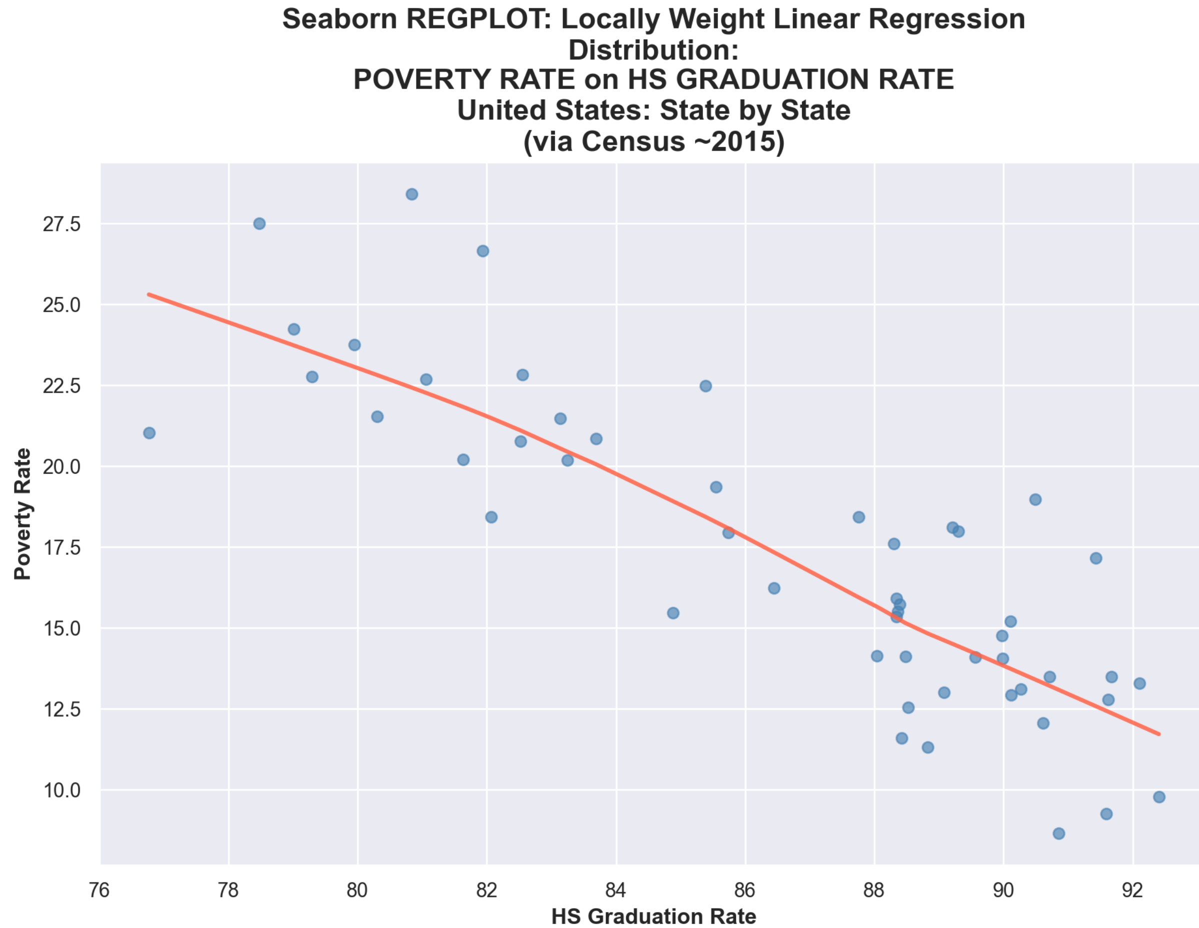
The heaviest clustering is around states with high graduation rates and low poverty rates.

This means there are more states with this combination of positive attributes, and that there are fewer states that have a combination of the more upsetting high poverty and low graduation rates. I would guess, and we'll see upon further analysis, that the instance of more violence upon civilians by police may be more concentrated in these states that have lower grad rates and high poverty rates...we'll see...

Seaborn's `.lmplot()` or `.regplot()` to show a linear regression between the poverty ratio and the high school graduation ratio.

In [ ]:

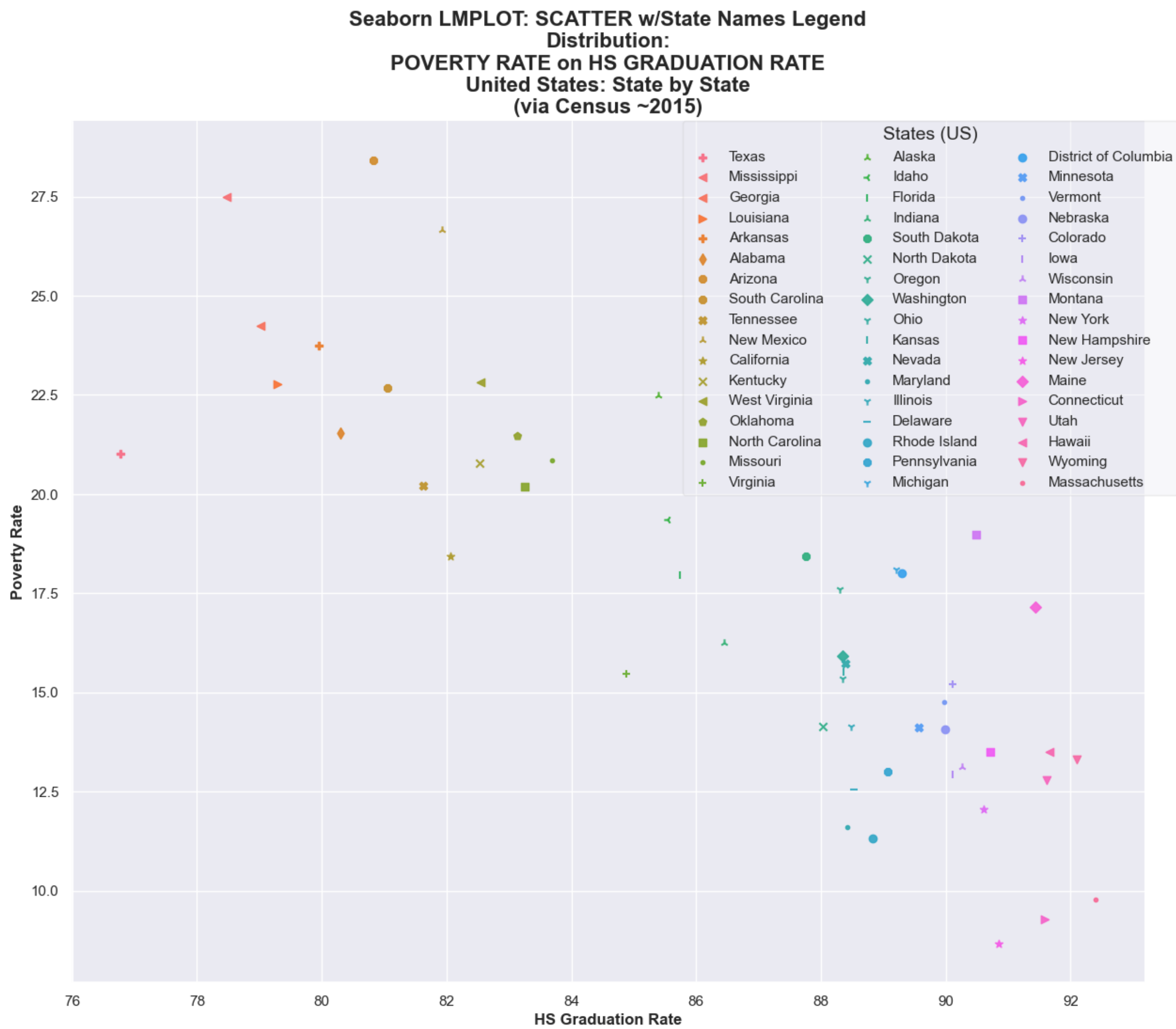
```
plt.figure(figsize=(11, 7), dpi=200)

with sns.axes_style('darkgrid'):
    sns.regplot(data=df_state_hs_poverty,
                x='percent_completed_hs',
                y='poverty_rate',
                lowess=True,
                scatter_kws={'color': 'steelblue', 'alpha': 0.65},
                line_kws={'color': 'tomato', 'alpha': 0.85})

title = "Seaborn REGPLOT: Locally Weight Linear Regression\n"
title += "Distribution:\nPOVERTY RATE on HS GRADUATION RATE\n"
title += "United States: State by State\n(via Census ~2015)"
plt.title(title, size=16, weight='bold')

plt.xlabel("HS Graduation Rate", size=12, weight='bold')
plt.ylabel("Poverty Rate", size=12, weight='bold')

plt.show()
```



In [ ]:
```
from random import choices

seaborn_markers = ['o', '^', 'v', '<', '>', 's', 'D', 'd', 'p', 'h', 'H', '8',
                   'X', '*', '.', 'P', 'x', '+', '1', '2', '3', '4', '|', '_']

plot_markers = choices(seaborn_markers,
                       k=len(df_state_hs_poverty['State_Names']))
```

In [ ]:
```
with sns.axes_style('darkgrid'):
    ax = sns.lmplot(data=df_state_hs_poverty,
                    x='percent_completed_hs',
                    y='poverty_rate',
                    hue='State_Names',
                    height=10,
                    legend=True,
                    aspect=1.25,
                    markers=plot_markers,
                    fit_reg=True,
                    scatter_kws={'alpha': 0.95},
                    facet_kws={'legend_out': True})

sns.move_legend(ax,
                1,
                title='States (US)',
                title_fontsize=14,
                bbox_to_anchor=(.89, .99),
                ncols=3,
                frameon=True,
                fancybox=True,
                framealpha=.35)

title = "Seaborn LMPLOT: SCATTER w/State Names Legend\n"
```

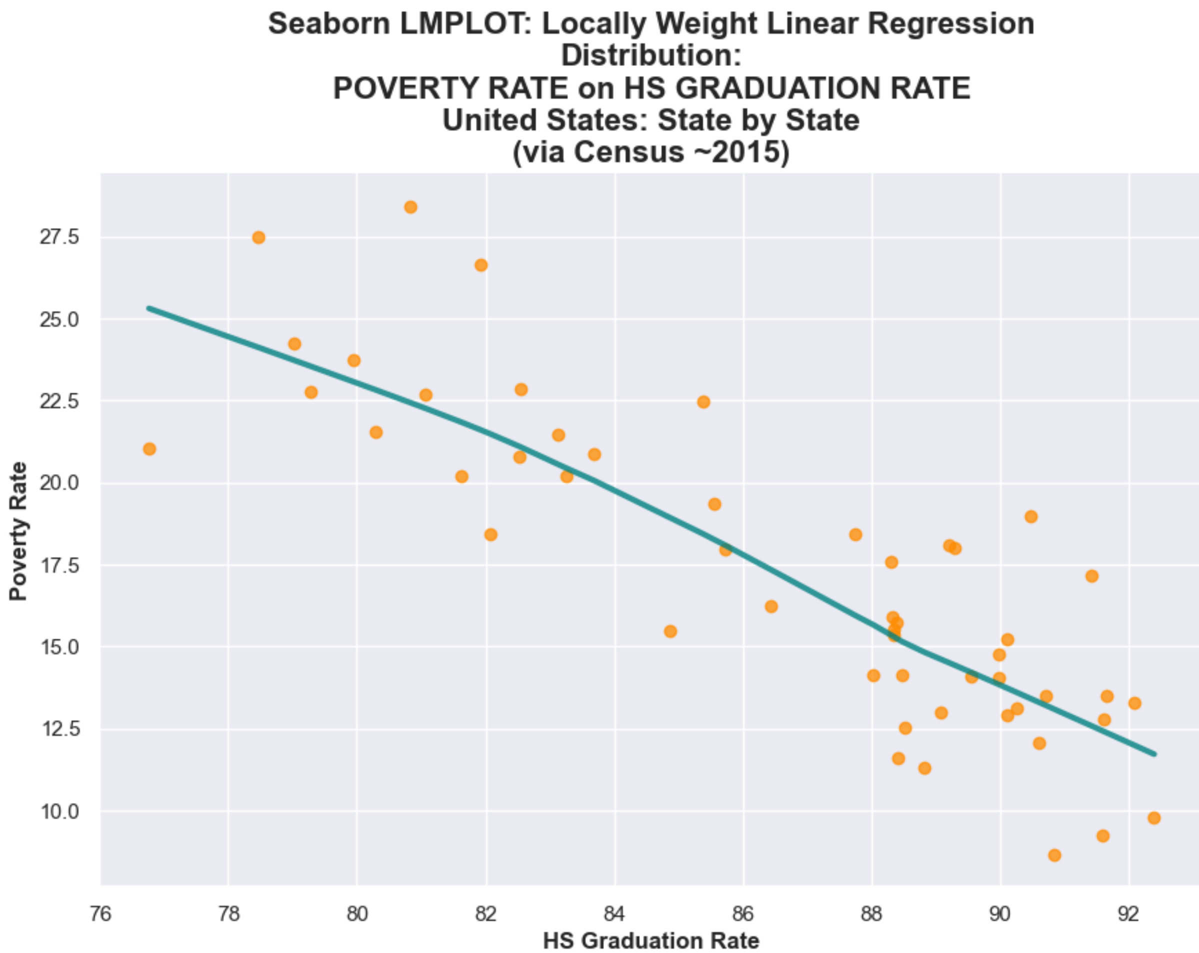
```
title += "Distribution:\nPOVERTY RATE on HS GRADUATION RATE\n"
title += "United States: State by State\n(via Census ~2015)"

plt.title(title, size=16, weight='bold')

plt.xlabel("HS Graduation Rate", size=12, weight='bold')
plt.ylabel("Poverty Rate", size=12, weight='bold')

plt.show()
```



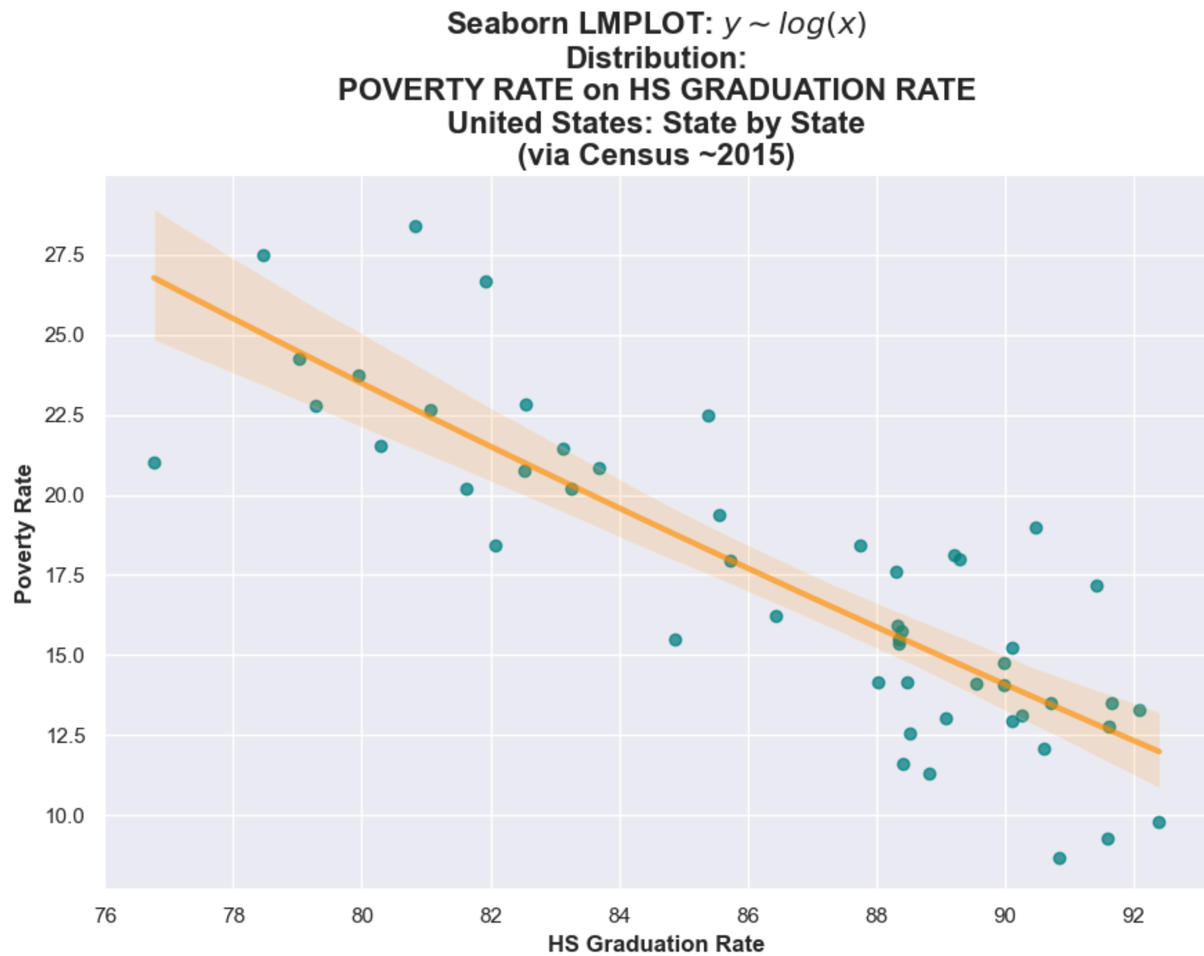
In [ ]:

```
with sns.axes_style('darkgrid'):
    sns.lmplot(
        data=df_state_hs_poverty,
        x='percent_completed_hs',
        y='poverty_rate',
        height=6,
        lowess=True,
        aspect=1.5,
        scatter_kws={'alpha': 0.75, 'color': 'darkorange'},
        line_kws={'alpha': .8, 'linewidth': 3, 'color': 'teal'},
    )

title = "Seaborn LMPLOT: Locally Weight Linear Regression\n"
title += "Distribution:\nPOVERTY RATE on HS GRADUATION RATE\n"
title += "United States: State by State\n(via Census ~2015)"
plt.title(title, size=16, weight='bold')

plt.xlabel("HS Graduation Rate", size=12, weight='bold')
plt.ylabel("Poverty Rate", size=12, weight='bold')
```

```
plt.show()
```



In [ ]:
```
with sns.axes_style('darkgrid'):
    sns.lmplot(
        data=df_state_hs_poverty,
        x='percent_completed_hs',
        y='poverty_rate',
        height=6,
        logx=True,
        aspect=1.5,
        fit_reg=True,
        scatter_kws={'alpha': 0.75, 'color': 'teal'},
        line_kws={'alpha': .65, 'linewidth': 3, 'color': 'darkorange'},
    )

title = "Seaborn LMPLOT: $y \\sim log(x)$\n"
title += "Distribution:\nPOVERTY RATE on HS GRADUATION RATE\n"
title += "United States: State by State\n(via Census ~2015)"
plt.title(title, size=16, weight='bold')

plt.xlabel("HS Graduation Rate", size=12, weight='bold')
plt.ylabel("Poverty Rate", size=12, weight='bold')

plt.show()
```



Analysis:

Based on the fact that all lot of the points in the scatterplots are close to the linear regression line plot, it appears there is a relationship between poverty level and high school graduation level.

Create a Bar Chart...

...with Subsections Showing the Racial Makeup of Each US State

Visualise the share of the white, black, hispanic, asian and native american population in each US State using a bar chart with sub sections.

Inspect and transform `df_share_race_city`***:***

In [ ]:
```
# Create copy of original DF in case we make a mistake:
df_race_city = df_share_race_city.copy()
```

In [ ]:
```
print(df_race_city.columns)
print(df_race_city.dtypes)
```

```
Index(['Geographic area', 'City', 'share_white', 'share_black',
       'share_native_american', 'share_asian', 'share_hispanic'],
      dtype='object')
Geographic area          object
City                     object
share_white              object
share_black              object
share_native_american    object
share_asian              object
share_hispanic           object
dtype: object
```

In [ ]:
```
# Inspect:
# df_race_city
```

In [ ]:
```
race_cols = ['share_white',
             'share_black',
             'share_native_american',
             'share_asian',
             'share_hispanic']

# Inspect all columns to see if data has non-digit strings:
for col in race_cols:
    for i in df_race_city[col]:
        if not i.isdigit():
            print(col)
            break
```

```
share_white
share_black
share_native_american
share_asian
share_hispanic
```

In [ ]:
```
# Entries w/non-digit strings are meant to be '0', let's replace them:
for col in race_cols:
    df_race_city[col] = df_race_city[col].replace(r'[^\d\.]', '0', regex=True)
```

In [ ]:
```
# Convert data to numeric values:
for col in race_cols:
    df_race_city[col] = pd.to_numeric(df_race_city[col])
```

In [ ]:
```
# Inspect the cleaned, transformed DF:
print(df_race_city.dtypes)
print(df_race_city.shape)
df_race_city.describe()
```

```
Geographic area           object
City                      object
share_white              float64
share_black              float64
share_native_american    float64
share_asian              float64
share_hispanic           float64
dtype: object
(29268, 7)
```

Out[ ]:

	share_white	share_black	share_native_american	share_asian	share_hispanic
count	29,268.00	29,268.00	29,268.00	29,268.00	29,268.00
mean	83.16	6.83	2.87	1.54	9.32
std	21.76	15.61	12.67	4.29	17.57
min	0.00	0.00	0.00	0.00	0.00
25%	78.50	0.10	0.10	0.00	1.20
50%	92.50	0.80	0.30	0.40	2.90
75%	96.90	4.20	0.90	1.20	8.00
max	100.00	100.00	100.00	67.10	100.00

In [ ]:

```
# Drop City column so next cell will work:
df_race_city = df_race_city.drop(columns=['City'])
```

In [ ]:

```
df_race_state = df_race_city.groupby('Geographic area').mean().reset_index()
```

In [ ]:

```
# Inspect new DF:
display(df_race_state)
print(df_race_state.dtypes)
print(df_race_state.shape)
df_race_state.describe()
```

	Geographic area	share_white	share_black	share_native_american	share_asian	share_hispanic
0	AK	45.26	0.56	45.48	1.38	2.13
1	AL	72.51	23.32	0.66	0.48	2.98
2	AR	78.45	16.30	0.76	0.48	4.27
3	AZ	59.93	0.95	28.59	0.73	20.14
4	CA	71.54	2.68	1.72	5.54	29.51
5	CO	87.77	0.92	1.62	1.15	17.90
6	CT	86.11	4.99	0.66	2.99	7.98
7	DC	38.50	50.70	0.30	3.50	9.10
8	DE	76.25	15.39	0.52	2.00	7.42
9	FL	78.67	13.37	0.46	1.62	16.53
10	GA	62.59	30.63	0.30	1.49	6.42
11	HI	33.37	1.07	0.39	25.65	10.36
12	IA	96.71	0.56	0.27	0.40	2.82
13	ID	88.82	0.30	2.52	0.49	10.70
14	IL	90.36	4.70	0.26	1.34	5.17
15	IN	94.82	1.69	0.28	0.59	3.32
16	KS	92.96	0.96	1.87	0.43	5.07
17	KY	92.23	4.42	0.21	0.71	2.23
18	LA	64.81	30.78	0.96	0.79	2.98
19	MA	89.30	2.79	0.27	2.84	4.93
20	MD	72.12	19.46	0.34	2.98	5.93
21	ME	95.69	0.82	0.55	1.03	1.31
22	MI	90.67	4.12	1.08	0.95	3.54
23	MN	91.80	1.00	3.36	1.03	3.15
24	MO	90.18	5.86	0.54	0.55	2.36
25	MS	53.80	41.83	1.61	0.55	2.32
26	MT	84.48	0.23	11.87	0.32	2.19
27	NC	71.52	20.40	1.79	0.93	6.41
28	ND	92.29	0.36	5.33	0.29	1.82
29	NE	94.72	0.42	1.56	0.29	4.07
30	NH	95.68	0.72	0.29	1.34	1.74
31	NJ	80.89	6.98	0.26	5.94	11.12
32	NM	65.42	0.67	15.34	0.49	45.43

	Geographic area	share_white	share_black	share_native_american	share_asian	share_hispanic
33	NV	82.33	1.57	5.58	2.01	13.17
34	NY	88.88	4.01	0.36	2.72	6.63
35	OH	92.80	3.96	0.22	0.75	2.13
36	OK	72.93	3.38	14.38	0.43	5.70
37	OR	87.39	0.53	2.58	1.38	9.75
38	PA	93.25	3.27	0.16	0.99	2.73
39	RI	89.23	2.99	0.67	1.69	6.67
40	SC	61.98	32.83	0.46	0.74	4.48
41	SD	84.82	0.29	12.03	0.32	2.04
42	TN	88.95	7.30	0.32	0.64	2.86
43	TX	82.40	5.87	0.66	1.01	39.27
44	UT	89.44	0.38	3.87	0.77	7.66
45	VA	74.60	16.51	0.34	3.67	6.15
46	VT	95.87	0.77	0.37	0.91	1.38
47	WA	82.38	1.43	3.75	3.07	10.61
48	WI	92.96	0.94	2.62	0.79	3.32
49	WV	95.04	2.92	0.20	0.34	0.90
50	WY	91.92	0.40	3.08	0.39	5.99

```
Geographic area           object
share_white              float64
share_black              float64
share_native_american    float64
share_asian              float64
share_hispanic           float64
dtype: object
(51, 6)
```

Out[ ]:

	share_white	share_black	share_native_american	share_asian	share_hispanic
count	51.00	51.00	51.00	51.00	51.00
mean	80.93	7.83	3.60	1.84	7.74
std	15.20	11.67	7.88	3.63	8.92
min	33.37	0.23	0.16	0.29	0.90
25%	72.72	0.80	0.33	0.52	2.78
50%	87.39	2.92	0.66	0.93	5.07
75%	92.07	7.14	2.60	1.65	8.54
max	96.71	50.70	45.48	25.65	45.43

In [ ]:
```
# Rename 'Geographic area' to 'state_abbr':
df_race_state.rename({'Geographic area': 'state_abbr'}, axis=1, inplace=True)
```

In [ ]:
```
# Add a column with state names:
state_dict = {"TX": "Texas", "MS": "Mississippi", "GA": "Georgia", "LA": "Louisiana

state_names= np.array([state_dict.get(i) for i in df_race_state['state_abbr']],
                      dtype='object')

df_race_state.insert(1, 'state_name', state_names)
```

In [ ]:
```
# Verify the rows each add up to about `100`:
df_race_state
```

Out[ ]:

	state_abbr	state_name	share_white	share_black	share_native_american	share_asian
0	AK	Alaska	45.26	0.56	45.48	1.38
1	AL	Alabama	72.51	23.32	0.66	0.48
2	AR	Arkansas	78.45	16.30	0.76	0.48
3	AZ	Arizona	59.93	0.95	28.59	0.73
4	CA	California	71.54	2.68	1.72	5.54
5	CO	Colorado	87.77	0.92	1.62	1.15
6	CT	Connecticut	86.11	4.99	0.66	2.99
7	DC	District of Columbia	38.50	50.70	0.30	3.50
8	DE	Delaware	76.25	15.39	0.52	2.00
9	FL	Florida	78.67	13.37	0.46	1.62
10	GA	Georgia	62.59	30.63	0.30	1.49
11	HI	Hawaii	33.37	1.07	0.39	25.65
12	IA	Iowa	96.71	0.56	0.27	0.40
13	ID	Idaho	88.82	0.30	2.52	0.49
14	IL	Illinois	90.36	4.70	0.26	1.34
15	IN	Indiana	94.82	1.69	0.28	0.59
16	KS	Kansas	92.96	0.96	1.87	0.43
17	KY	Kentucky	92.23	4.42	0.21	0.71
18	LA	Louisiana	64.81	30.78	0.96	0.79
19	MA	Massachusetts	89.30	2.79	0.27	2.84
20	MD	Maryland	72.12	19.46	0.34	2.98
21	ME	Maine	95.69	0.82	0.55	1.03
22	MI	Michigan	90.67	4.12	1.08	0.95
23	MN	Minnesota	91.80	1.00	3.36	1.03
24	MO	Missouri	90.18	5.86	0.54	0.55
25	MS	Mississippi	53.80	41.83	1.61	0.55
26	MT	Montana	84.48	0.23	11.87	0.32
27	NC	North Carolina	71.52	20.40	1.79	0.93
28	ND	North Dakota	92.29	0.36	5.33	0.29
29	NE	Nebraska	94.72	0.42	1.56	0.29
30	NH	New Hampshire	95.68	0.72	0.29	1.34
31	NJ	New Jersey	80.89	6.98	0.26	5.94

	state_abbr	state_name	share_white	share_black	share_native_american	share_asian
32	NM	New Mexico	65.42	0.67	15.34	0.49
33	NV	Nevada	82.33	1.57	5.58	2.01
34	NY	New York	88.88	4.01	0.36	2.72
35	OH	Ohio	92.80	3.96	0.22	0.75
36	OK	Oklahoma	72.93	3.38	14.38	0.43
37	OR	Oregon	87.39	0.53	2.58	1.38
38	PA	Pennsylvania	93.25	3.27	0.16	0.99
39	RI	Rhode Island	89.23	2.99	0.67	1.69
40	SC	South Carolina	61.98	32.83	0.46	0.74
41	SD	South Dakota	84.82	0.29	12.03	0.32
42	TN	Tennessee	88.95	7.30	0.32	0.64
43	TX	Texas	82.40	5.87	0.66	1.01
44	UT	Utah	89.44	0.38	3.87	0.77
45	VA	Virginia	74.60	16.51	0.34	3.67
46	VT	Vermont	95.87	0.77	0.37	0.91
47	WA	Washington	82.38	1.43	3.75	3.07
48	WI	Wisconsin	92.96	0.94	2.62	0.79
49	WV	West Virginia	95.04	2.92	0.20	0.34
50	WY	Wyoming	91.92	0.40	3.08	0.39

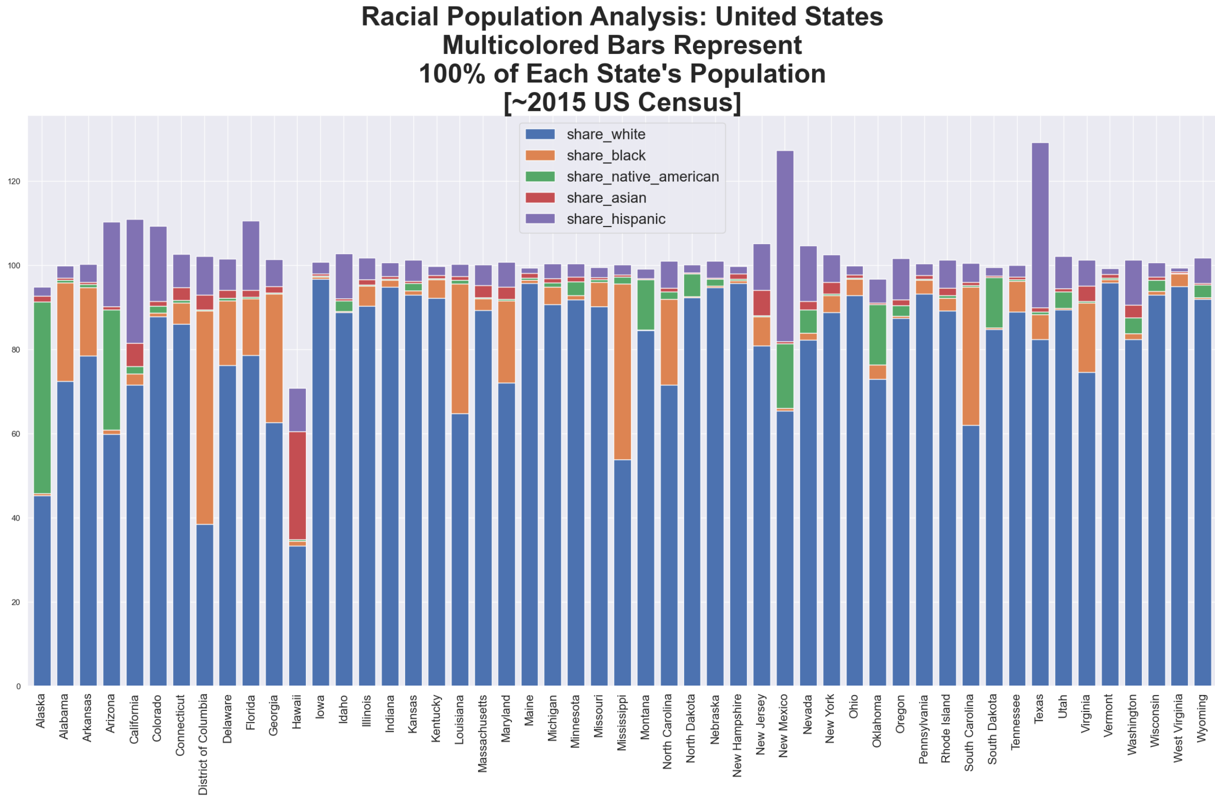
The new df `df_race_state` is good to go!

Now the plot:

Visualise the share of the white, black, hispanic, asian and native american population in each US State using a bar chart with sub sections.

In [ ]:
```
ax = df_race_state.plot(figsize=(29, 14),
                        stacked=True,
                        width=0.75,
                        kind='bar',
                        legend=True,
                        xticks=df_race_state.index)

ax.set_xticklabels(df_race_state['state_name'])


title = "Racial Population Analysis: United States\n"
title += "Multicolored Bars Represent\n100% of Each State's Population\n"
```

```python
title += "[~2015 US Census]"

plt.title(title, size=36, weight='bold')
ax.tick_params(axis='x', which='major', labelsize=16)

plt.legend(fontsize=20)

plt.show()
```



Analysis:

First of all...

...the bars are not all equal to 100 -- which probably means:

- Taking the average of each column's values throws off the aggregate to 100 -- since we're taking the average along Axis 0 for an aggregate to 100 which would be along Axis 1
  - I think the general result should still be representative of the shares represented by each column, but upon a refactor will find a way to even out this phenomenon.

Since all the bars are hovering around 100, though, I feel satisfied that the data and this plot are showing a good representation of the share of data for each state.

I'll ignore this issue, since it seems negligible/a matter for future fine-tuning, and likely doesn't affect the accuracy of the overall accuracy of the racial breakdown for each state.

And B...

We can sort the plots by which percentage of each state is highest for each race for futher analysis, but for now we can clearly which states have the highest population of each race:

- African American: Washington DC

- Caucasian: Maine/New Hampshire/West Virginia
- Asian: Hawaii
- Latino: New Mexico or Texas
- Native American: Alaska

Create Donut Chart...

...of People Killed by Race

Hint: Use `.value_counts()`

Data Cleaning and Transformation:

In [ ]:
```
race_counts = df_fatalities['race'].value_counts()

# Inspect:
# df_fatalities['race'].value_counts().nlargest(10)
# df_fatalities['race'].value_counts().nlargest(10).index
```

In [ ]:
```
race_counts
```

Out[ ]:
```
race
W    1201
B     618
H     423
0     195
A      39
N      31
O      28
Name: count, dtype: int64
```

In [ ]:
```
# Replace `0` with `O` (undetermined):
df_fatalities['race'] = df_fatalities['race'].replace(0, 'O')
```

In [ ]:
```
race_counts = df_fatalities['race'].value_counts()
```

In [ ]:
```
# Now we have one metric for undetermined:
race_counts
```

Out[ ]:
```
race
W    1201
B     618
H     423
O     223
A      39
N      31
Name: count, dtype: int64
```

In [ ]:
```
# Though, for clarity, let's rename to words rather than abbreviations:
race_dict = {'W': 'Caucasian', 'B': 'African American', 'H': 'Latino',
             'A': 'Asian American', 'N': 'Native American', 'O': 'Undetermined'}
```

```
df_fatalities['race'] = df_fatalities['race'].replace(race_dict)
```

In [ ]:
```
# Drop the `id` column (we're not working w/relational databases):
df_fatalities.drop(columns=['id'], inplace=True)
```

In [ ]:
```
# Now create `race_counts` again:
race_counts = df_fatalities['race'].value_counts()
```

In [ ]:
```
# This is much clearer than before:
print(race_counts)
print(race_counts.index)
print(race_counts.values)
```

```
race
Caucasian           1201
African American     618
Latino               423
Undetermined         223
Asian American        39
Native American       31
Name: count, dtype: int64
Index(['Caucasian', 'African American', 'Latino', 'Undetermined',
       'Asian American', 'Native American'],
      dtype='object', name='race')
[1201  618  423  223   39   31]
```

Create Donut Chart of People Killed by Race:

Hint: Use `.value_counts()`

In [ ]:
```
from random import choices

colors = ['sienna', 'steelblue', 'olivedrab',
          'darkslategrey', 'firebrick', 'rebeccapurple']

title = "People Killed by Police in the US by Race, 2015-2017"

fig = px.pie(
    labels=race_counts.index,
    values=race_counts.values,
    title=title,
    names=race_counts.index,
    hole=0.25,
    width=1280,
    height=1280,
    color_discrete_sequence=colors
)

fig.update_traces(
    textposition='inside',
    textfont=dict(family='tahoma', size=16, color='gainsboro'),
    textinfo='percent',
)

fig.update_layout(
    title=dict(font=dict(size=36)),
)
```

```
fig.show()
```

Analysis:

In order from highest percentage to lowest:

- Caucasian: 47.4%
- African American: 24.4%
- Latino: 16.7%
- Undetermined: 8.8%
- Asian American: 1.54%
- Native American: 1.22%

Create a Chart Comparing the Total Number of Deaths of Men and Women

Use `df_fatalities` to illustrate how many more men are killed compared to women.

In [ ]:
```
df_fatalities.sample()
```

Out[ ]:

	name	date	manner_of_death	armed	age	gender	race	city	sta
1724	Donte T. Jones	02/10/16	shot	unknown weapon	36.00	M	African American	Markham	

In [ ]:
```
gender_counts = df_fatalities['gender'].value_counts()
```

In [ ]:
```
# Replace abbreviations with full words:
gender_dict = {'M': 'Male', 'F': 'Female'}

df_fatalities['gender'] = df_fatalities['gender'].replace(gender_dict)
```

In [ ]:
```
gender_counts = df_fatalities['gender'].value_counts()
```

In [ ]:
```
gender_counts
```

Out[ ]:
```
gender
Male      2428
Female     107
Name: count, dtype: int64
```

In [ ]:
```
from random import choices
```

```python
colors = ['steelblue', 'firebrick']

title = "People Killed by Police in the US by Gender, 2015-2017"

fig = px.pie(
    labels=gender_counts.index,
    values=gender_counts.values,
    title=title,
    names=gender_counts.index,
    hole=0.25,
    width=840,
    height=840,
    color_discrete_sequence=colors
)

fig.update_traces(
    textposition='inside',
    textfont=dict(family='tahoma', size=12, color='gainsboro'),
    textinfo='percent',
)

fig.update_layout(
    title=dict(font=dict(size=24)),
)

fig.show()
```

Analysis:

Men killed by police: 95.8%

Women killed by police: 4.22%

Create a Box Plot Showing the Age and Manner of Death

Break out the data by gender using `df_fatalities`. Is there a difference between men and women in the manner of death?

In [ ]:
```python
df_fatalities.dtypes
```

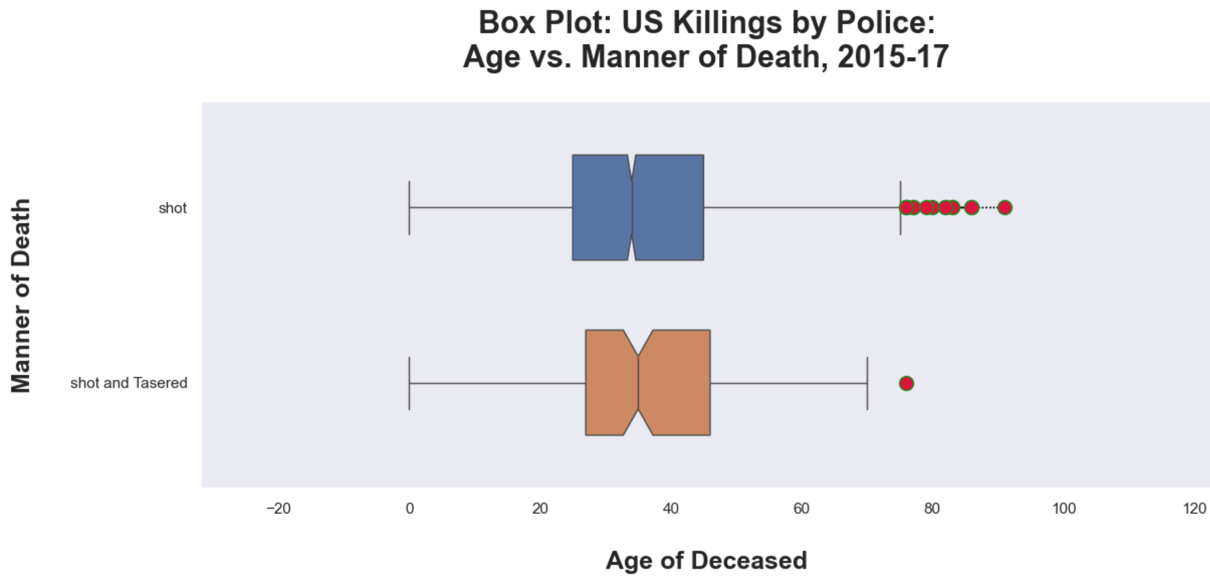
Out[ ]:
```
name                          object
date                          object
manner_of_death               object
armed                         object
age                          float64
gender                        object
race                          object
city                          object
state                         object
signs_of_mental_illness         bool
threat_level                  object
flee                          object
body_camera                     bool
dtype: object
```

In [ ]:
```
# Inspect for non-numeric values:
for i in df_fatalities['age'].value_counts().index:
    print(i)

# All good
```

In [ ]:
```
# Inspect:
df_fatalities['manner_of_death'].value_counts()
```

Out[ ]:
```
manner_of_death
shot                2363
shot and Tasered     172
Name: count, dtype: int64
```

In [ ]:
```
plt.figure(figsize=(13, 5))

sns.boxplot(
    data=df_fatalities,
    x='age',
    y='manner_of_death',
    width=0.6,
    notch=True,
    flierprops={"marker": "o",
                "markersize": 10,
                'markerfacecolor': 'crimson',
                'markeredgecolor': 'forestgreen',
                'linestyle': 'dotted'},
    hue='manner_of_death'
)

plt.title('Box Plot: US Killings by Police:\nAge vs. Manner of Death, 2015-17',
          fontdict={'size': 22, 'weight': 'bold'},
          pad=25.0)

plt.xlabel('Age of Deceased',
           fontdict={'size': 18, 'weight': 'bold'},
           labelpad=25.0)
plt.ylabel('Manner of Death',
           fontdict={'size': 18, 'weight': 'bold'},
           labelpad=25.0)

plt.grid(axis='x')
plt.margins(0.35)

plt.show()
```



Analysis:

Most victims were between the ages of 25 and 45, with the age range slightly higher for `shot and Tasered`.

More outliers of higher age (above 75) for `shot`.

Were People Armed?

In what percentage of police killings were people armed?

Create a chart that shows what kind of weapon (if any) the deceased was carrying.

How many of the people killed by police were armed with guns versus unarmed?

In what percentage of police killings were people armed?

In [ ]:
```
df_armed = df_fatalities[['armed']]
```

In [ ]:
```
# Search all entries and determine what might be considered 'unarmed':
for i in df_armed.value_counts().index:
    print(i)

# I'll use:
unarmed = ['unarmed', 'undetermined', 'unknown weapon', 0]
```

In [ ]:
```
# Change all 'unarmed'-type to explicitly 'unarmed':
for i in unarmed:
    df_armed['armed'] = df_armed['armed'].replace(i, 'unarmed')
```

```
C:\Users\anb20\AppData\Local\Temp\ipykernel_28868\3150729996.py:3: SettingWithCopyWa
rning:


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-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
```

In [ ]:

```python
df_armed['armed'] = df_armed['armed'].mask(df_armed['armed'] != 'unarmed', 'armed')
```

```
C:\Users\anb20\AppData\Local\Temp\ipykernel_28868\2536365033.py:1: SettingWithCopyWa
rning:


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-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
```

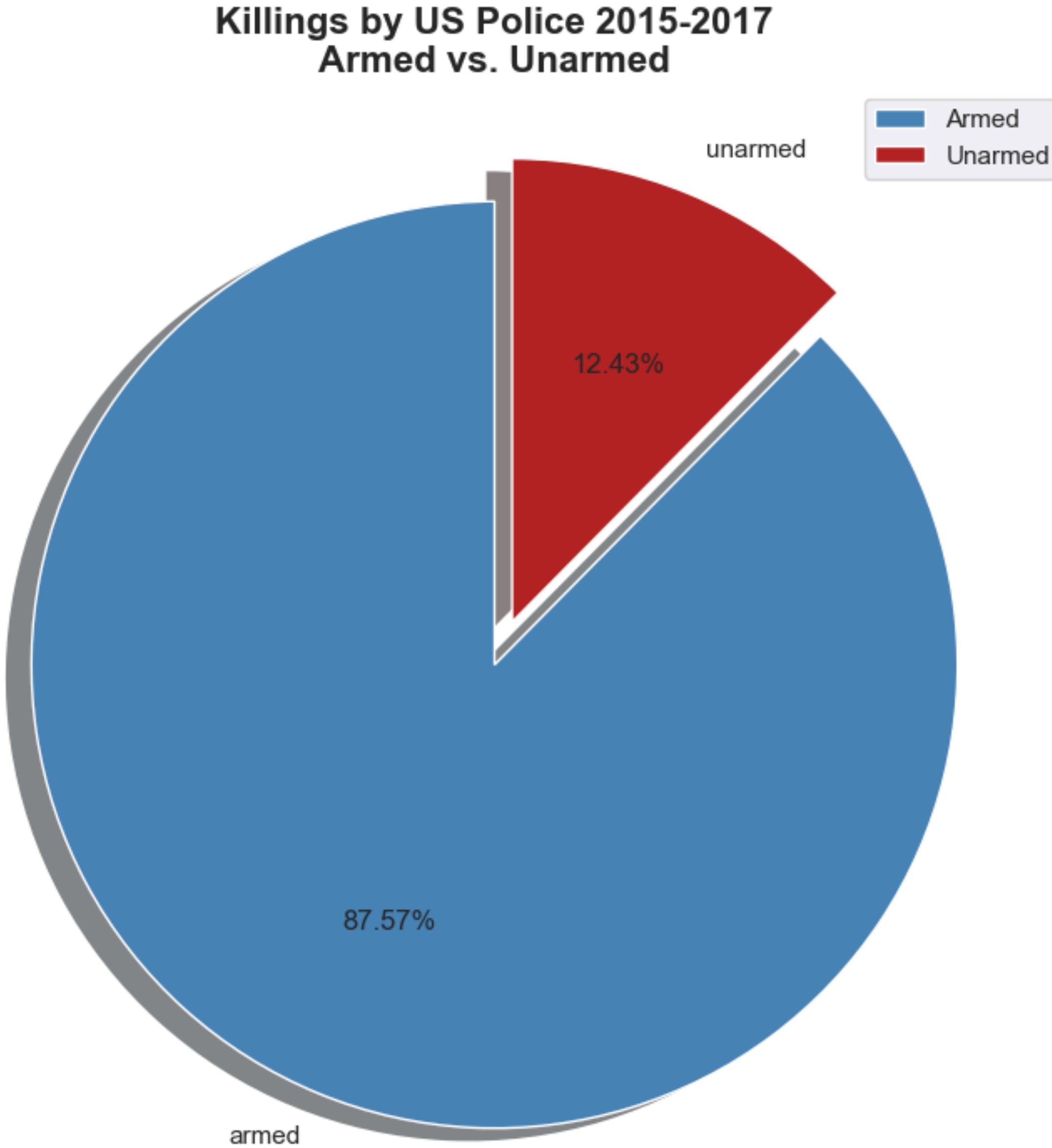
In [ ]:

```python
# Create series and check results:
armed_counts = df_armed.value_counts()
print(armed_counts)
```

```
armed
armed      2220
unarmed     315
Name: count, dtype: int64
```

In [ ]:

```python
fig, ax = plt.subplots(figsize=(9, 9))

labels = 'armed', 'unarmed'
explode = (0, 0.1)

ax.pie(
    armed_counts,
    explode=explode,
    labels=labels,
    autopct='%1.2f%%',
    shadow={'ox': -0.04, 'edgecolor': 'none', 'shade': 0.9},
    startangle=90,
    colors=['steelblue', 'firebrick'],
)

title = "Killings by US Police 2015-2017"
title += "\nArmed vs. Unarmed"
ax.set_title(title, size=16, weight='bold')

ax.legend(['Armed', 'Unarmed'])

plt.show()
```



Analysis:

87.57% Armed

12.43% Unarmed

Create a chart that shows what kind of weapon (if any) the deceased was carrying.

In [ ]:
```
armed_counts = df_fatalities['armed'].value_counts()
```

In [ ]:
```
# Make descriptive labels:
armed_zipped = list(zip(armed_counts.index, armed_counts))
armed_labels = [f"{i}: {j}" for i, j in armed_zipped]
```

In [ ]:
```
colors = [
    "black", "blue", "brown", "chocolate", "cornflowerblue", "crimson", "darkblue",
    "darkcyan", "darkgoldenrod", "darkgreen", "darkkhaki", "darkmagenta", "darkoliv
```

```
    "darkorange", "darkorchid", "darkred", "darksalmon", "darkseagreen", "darkslate
    "darkslategray", "darkslategrey", "darkturquoise", "darkviolet", "deeppink", "d
    "dodgerblue", "firebrick", "forestgreen", "goldenrod", "green", "greenyellow",
    "lightgreen", "limegreen", "maroon", "mediumblue", "mediumseagreen", "mediumsla
    "mediumspringgreen", "midnightblue", "navy", "olive", "olivedrab", "palegreen",
    "purple", "rebeccapurple", "rosybrown", "royalblue", "saddlebrown", "sandybrown
    "seagreen", "sienna", "slateblue", "springgreen", "steelblue", "yellowgreen"
]
```

In [ ]:
```
# Plot it:
title = "Police Killings in the US: Type of Weapon Carried by Deceased , 2015-2017"

fig = px.pie(
    labels=armed_counts.index,
    values=armed_counts.values,
    title=title,
    names=armed_labels,
    hole=0.25,
    width=1920,
    height=1920,
    color_discrete_sequence=choices(colors, k=len(armed_counts.index)),
)

# Make labels outside if too small for text:

fig.update_traces(
    textposition='inside',
    textfont=dict(family='tahoma', size=14, color='gainsboro'),
    textinfo='percent',
    insidetextorientation='radial',
)

fig.update_layout(
    title=dict(font=dict(size=42)),
    legend=dict(entrywidth=0.3,
                entrywidthmode='fraction',
                orientation='h',
                y=-0.05,
                xanchor='center',
                x=0.6)
)

fig.show()
```

Analysis:

Most common weapon: *Gun*

Least common weapon: *Pen*

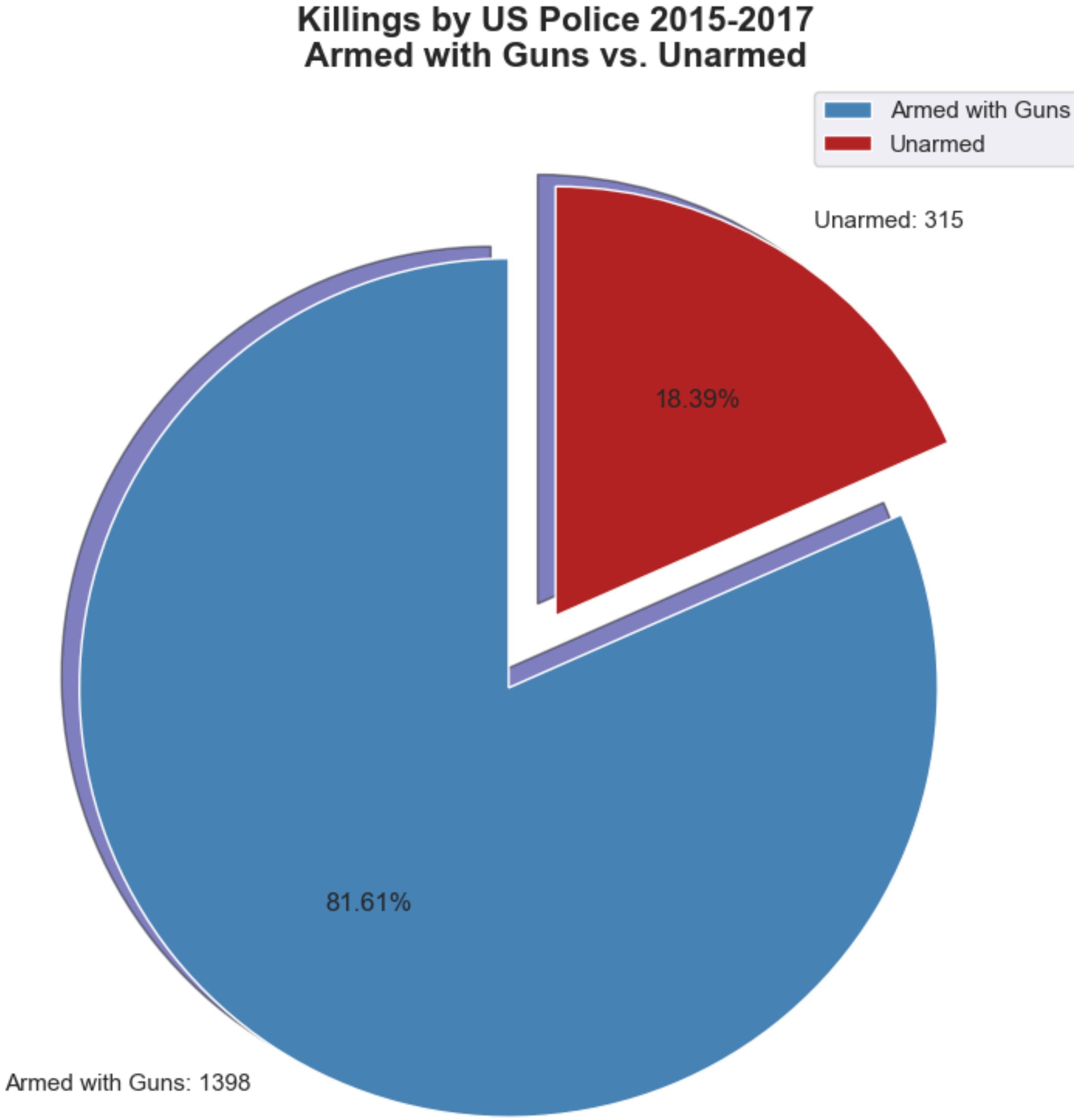
How many of the people killed by police were armed with guns versus unarmed?

In [ ]:
```
num_unarmed = df_armed.value_counts()['unarmed']
```

In [ ]:
```
num_guns = df_fatalities['armed'].value_counts()['gun']
```

In [ ]:
```
labels = [f"Armed with Guns: {num_guns}", f"Unarmed: {num_unarmed}"]
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(9, 9))

explode = [0.2, 0]

ax.pie(
    [num_guns, num_unarmed],
    explode=explode,
    labels=labels,
    autopct='%1.2f%%',
    shadow={'ox': -0.03, 'oy': 0.02, 'edgecolor': 'k', 'facecolor': 'navy'},
    startangle=90,
    colors=['steelblue', 'firebrick'],
)

title = "Killings by US Police 2015-2017"
title += "\nArmed with Guns vs. Unarmed"
ax.set_title(title, size=16, weight='bold')

ax.legend(['Armed with Guns', 'Unarmed'])

plt.show()
```



Analysis:

Armed with guns:

- $1,398$
- 81.61%

Unarmed:

- 315
- 18.39%

How Old Were the People Killed?

Work out what percentage of people killed were under 25 years old.

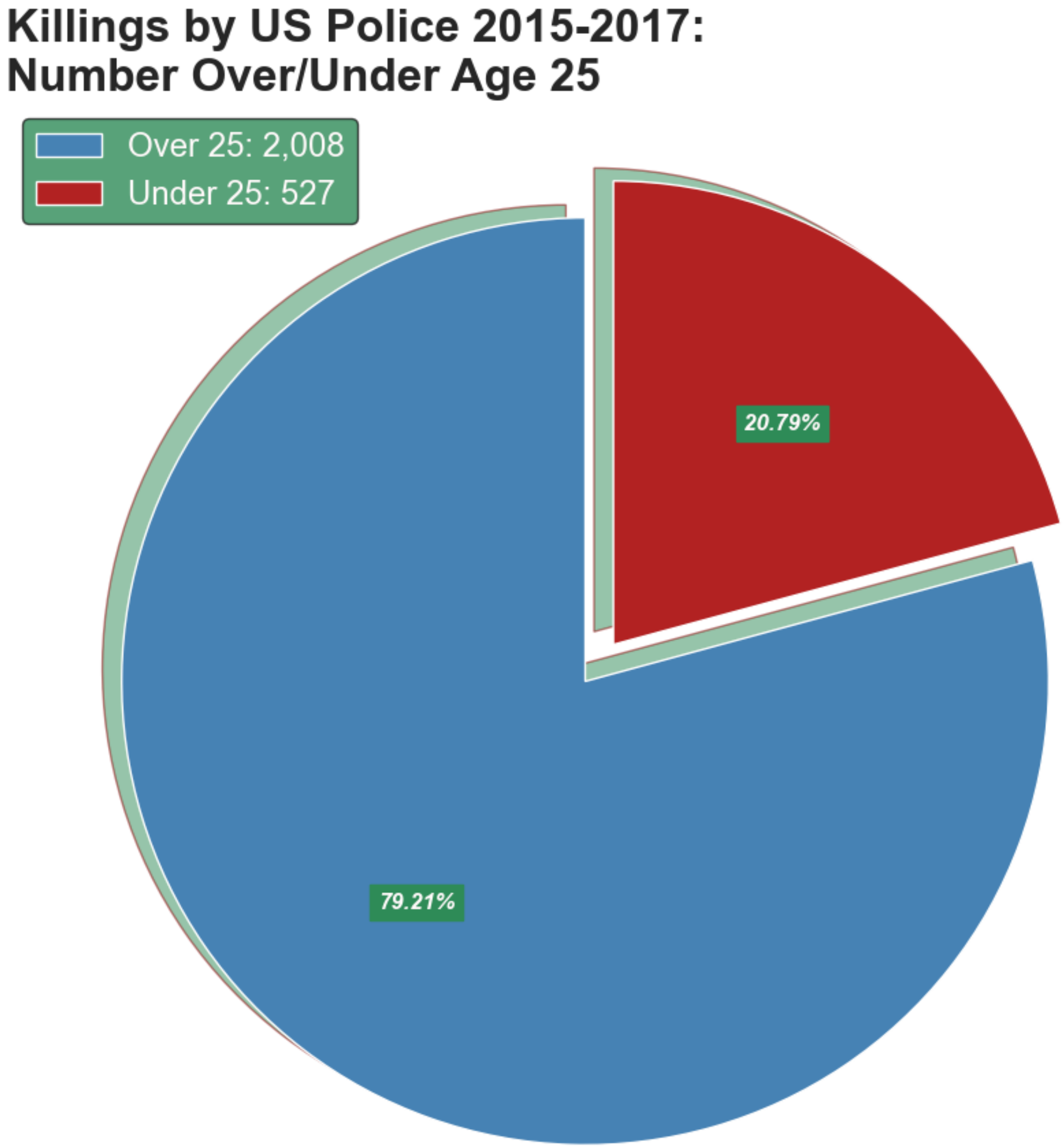
In [ ]:
```
# Create a new row based on bool for `age < 25`
series_under_25 = df_fatalities['age'].apply(lambda x: 'under_25' if x < 25 else 'o
```

In [ ]:
```
# Add row to the DF:
df_fatalities.insert(5, '>25<', series_under_25)
```

In [ ]:
```
# Create series to count age groups:
series_count_25 = df_fatalities['>25<'].value_counts()
```

In [ ]:
```
val_over = series_count_25['over_25']
val_under = series_count_25['under_25']
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(11, 11))

labels = (f"Over 25: {val_over:,}", f"Under 25: {val_under:,}")
explode = (0, 0.1)

ax.pie(
    series_count_25,
    explode=explode,
    # labels=labels,
    autopct='%1.2f%%',
    shadow={'ox': -0.03,
            'oy': 0.02,
            'edgecolor': 'darkred',
            'facecolor': 'seagreen'},
    startangle=90,
    colors=['steelblue', 'firebrick'],
    textprops=dict(
        color='snow',
        backgroundcolor='seagreen',
        size=12,
        style='italic',
        weight='bold'
    ),
)

title = "Killings by US Police 2015-2017:\n"
title += "Number Over/Under Age 25"
ax.set_title(title, size=24, weight='bold', loc='left')

ax.legend(
    labels=labels,
    edgecolor='k',
    mode=None, # 'expand'
    facecolor='seagreen',
    fontsize='x-large',
    labelcolor='snow'
)

plt.show()
```



Analysis:

Over 25:

- 79.21%
- $2,008$ total

Under 25:

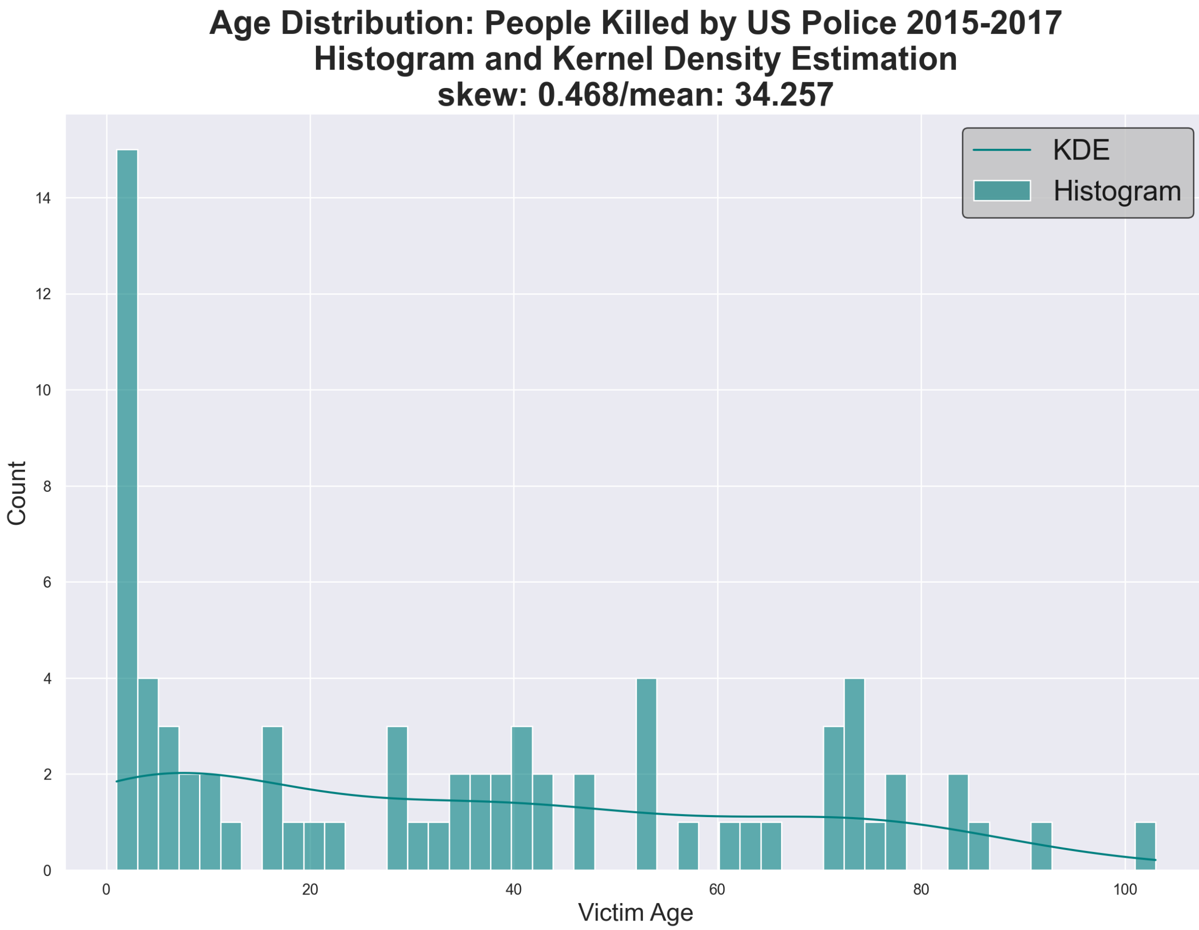
- 20.79%
- 527 total

Create a histogram and KDE plot that shows the distribution of ages of the people killed by police.

In [ ]:
```
series_age = df_fatalities['age'].value_counts()
```

In [ ]:
```
for i in series_age:
    print(i)
```

In [ ]:
```
series_age_skew = np.round(series_age.skew(), 3)
series_age_mean = np.round(series_age.mean(), 3)

fig, ax = plt.subplots(figsize=(15, 10), dpi=200)

sns.histplot(
    series_age,
    kde=True,
    color='teal',
    legend=False,
    ax=ax,
    alpha=0.6,
    bins=50,
    # line_kws={'color': 'crimson', 'lw': 5, 'ls': ':'}
)

sns.set_theme(style='darkgrid')

title = f"Age Distribution: People Killed by US Police 2015-2017\n"
title += f"Histogram and Kernel Density Estimation\n"
title += f"skew: {series_age_skew}/mean: {series_age_mean}"

plt.title(title, size=24, weight='bold')
plt.xlabel("Victim Age", fontsize=18)
plt.ylabel("Count", fontsize=18)

plt.legend(
    labels=['KDE', 'Histogram'],
    edgecolor='k',
    mode=None, # 'expand'
    facecolor='silver',
    fontsize='xx-large',
    labelcolor='k',
)

plt.show()
```



Analysis:

A skew of .468 indicates there are slightly more data points w/lower values.

The slightly decreasing KDE indicates there are slightly fewer data points in this dataset with higher values.

The histogram bars are much higher to the left, so this bears out the above conclusions.

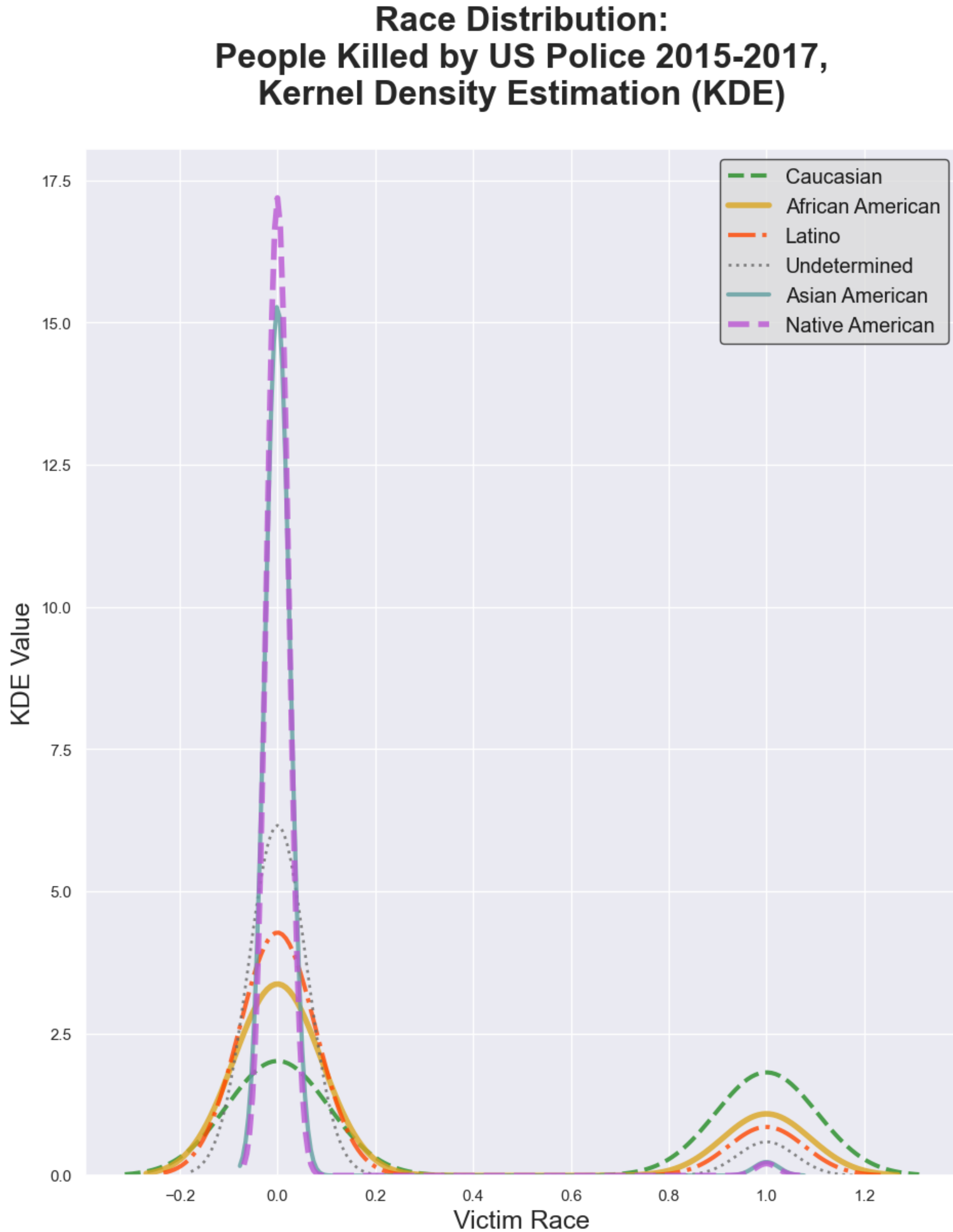
Overall, the spread of data points is pretty even, but there is a spike of data points with smaller values.

Create a seperate KDE plot for each race. Is there a difference between the distributions?

In [ ]:
```
series_races = df_fatalities['race'].value_counts()
```

In [ ]:
```
plt.figure(figsize=(11, 13))

widths = [3, 4, 3, 2, 3, 4]
linestyles = ['--', '-', '-.', ':', '-', '--']
colors = ['forestgreen', 'goldenrod', 'orangered', 'dimgrey', 'cadetblue', 'mediumo

for race in series_races.index:
    sns.kdeplot(
        df_fatalities['race'] == race,
        linewidth=widths.pop(0),
        linestyle=linestyles.pop(0),
```

```python
        legend=True,
        color=colors.pop(0),
        alpha=0.8,
        # fill=True,
    )

sns.set_theme(style='darkgrid')

title = f"Race Distribution:\nPeople Killed by US Police 2015-2017,\n"
title += f"Kernel Density Estimation (KDE)\n"

plt.title(title, size=24, weight='bold')
plt.xlabel("Victim Race", fontsize=18)
plt.ylabel("KDE Value", fontsize=18)

plt.legend(
    labels=series_races.index,
    edgecolor='k',
    facecolor='gainsboro',
    fontsize='large',
    labelcolor='k',
)

plt.show()
```



Analysis:

Caucasians have a higher `1` spike. `1` represents the `True` bool, meaning that more Caucasians are found in the DataFrame than any other race.

This bears out since Caucasians have the lowest `0` spike.

Also, Asian Americans and Native Americans have the highest `0` spikes and lowest `1` spikes, meaning they populate the DataFrame less.

A plot of this simple statistic (portion of each element) confirms which element is most/least present in the dataset.

Race of People Killed

Create a chart that shows the total number of people killed by race.

In [ ]:
```
colors = ['forestgreen', 'darkgoldenrod', 'orangered', 'darkslateblue', 'cadetblue'

title = "Killings by US Police 2015-2017 by Race"

labels = (
    f"Caucasian: {series_races['Caucasian']:,}",
    f"African American: {series_races['African American']:,}",
    f"Latino: {series_races['Latino']:,}",
    f"Undetermined: {series_races['Undetermined']:,}",
    f"Asian American: {series_races['Asian American']:,}",
    f"Native American: {series_races['Native American']:,}",
)

fig = px.pie(
    labels=series_races.index,
    values=series_races.values,
    title=title,
    names=labels,
    hole=0.25,
    width=950,
    height=750,
    color_discrete_sequence=colors
)

fig.update_traces(
    textposition='inside',
    textfont=dict(family='tahoma', size=14, color='aliceblue'),
    textinfo='percent',
)

fig.update_layout(
    title=dict(font=dict(size=30)),
)

fig.show()
```

Analysis:

Caucasians comprise the most of any race killed, Native Americans the fewest.

Mental Illness and Police Killings

What percentage of people killed by police have been diagnosed with a mental illness?

In [ ]:
```
# Create Series to count 'signs_of_mental_illness' counts:
```

```
series_mental_illness = df_fatalities['signs_of_mental_illness'].value_counts()
```

In [ ]:
```
# Create labels, indicate colors and title:
label_no_signs = series_mental_illness[False]
label_signs = series_mental_illness[True]

labels = (
    f"No Signs of Mental Illness: {label_no_signs:,}",
    f"Signs of Mental Illness: {label_signs:,}",
)

colors = ['forestgreen', 'firebrick']

title = "Killings by US Police 2015-2017 by Signs of Mental Illness"
```

In [ ]:
```
# Plot pie:
fig = px.pie(
    labels=series_mental_illness.index,
    values=series_mental_illness.values,
    title=title,
    names=labels,
    hole=0.25,
    width=950,
    height=750,
    color_discrete_sequence=colors
)

fig.update_traces(
    textposition='inside',
    textfont=dict(family='tahoma', size=14, color='aliceblue'),
    textinfo='percent',
)

fig.update_layout(
    title=dict(font=dict(size=30)),
)

fig.show()
```

Analysis:

25% of people killed had signs of mental illness.

In Which Cities Do the Most Police Killings Take Place?

Create a chart ranking the top 10 cities with the most police killings. Which cities are the most dangerous?

In [ ]:
```
# In case same city names have different states:
city_state_series = df_fatalities['city'] + ', ' + df_fatalities['state']
```

In [ ]:
```
# Insert 'city_state' into DF:
```

```
df_fatalities.insert(10, 'city_state', city_state_series)
```

In [ ]:
```
series_cities_10_largest = df_fatalities['city_state'].value_counts().nlargest(10)
```

In [ ]:
```
series_cities_10_largest
```

Out[ ]:
```
city_state
Los Angeles, CA    39
Phoenix, AZ        31
Houston, TX        26
Chicago, IL        25
Las Vegas, NV      21
San Antonio, TX    20
Columbus, OH       17
Miami, FL          17
Austin, TX         16
St. Louis, MO      15
Name: count, dtype: int64
```

In [ ]:
```
# Figure out how to insert these into legend:
labels = [f"{i}: {j}" for i, j in series_cities_10_largest.items()]

title = "Killings by US Police 2015-2017: Cities with Most Shootings"
```

In [ ]:
```
fig = px.bar(
    x=series_cities_10_largest.index,
    y=series_cities_10_largest.values,
    title=title,
    width=1100,
    height=700,
    color=series_cities_10_largest.index,
)

fig.update_layout(
    title_font_family="tahoma",
    title_font_size=36,
    showlegend=False,
)

fig.update_xaxes(
    title_text="Cities with Most Shootings",
    title_font_family="Arial",
    title_font_size=22,
    tickangle=60,
)

fig.update_yaxes(
    title_text="Number of Shootings",
    title_font_family="Arial",
    title_font_size=22
)

fig.show()
```

Analysis:

Most shootings to least amongst cities with most shootings:

- L.A.

- Phoenix
- Houston
- Chicago
- Las Vegas
- San Antonio
- Columbus
- Miami
- Austin
- St. Louis

Rate of Death by Race

Find the share of each race in the top 10 cities. Contrast this with the top 10 cities of police killings to work out the rate at which people are killed by race for each city.

In [ ]:
```
# Create DF foreach of the 10 cities with most shootings:
df_los_angeles = df_fatalities[df_fatalities['city_state'] == 'Los Angeles, CA']
df_phoenix = df_fatalities[df_fatalities['city_state'] == 'Phoenix, AZ']
df_houston = df_fatalities[df_fatalities['city_state'] == 'Houston, TX']
df_chicago = df_fatalities[df_fatalities['city_state'] == 'Chicago, IL']
df_las_vegas = df_fatalities[df_fatalities['city_state'] == 'Las Vegas, NV']
df_san_antonio = df_fatalities[df_fatalities['city_state'] == 'San Antonio, TX']
df_columbus = df_fatalities[df_fatalities['city_state'] == 'Columbus, OH']
df_miami = df_fatalities[df_fatalities['city_state'] == 'Miami, FL']
df_austin = df_fatalities[df_fatalities['city_state'] == 'Austin, TX']
df_st_louis = df_fatalities[df_fatalities['city_state'] == 'St. Louis, MO']
```

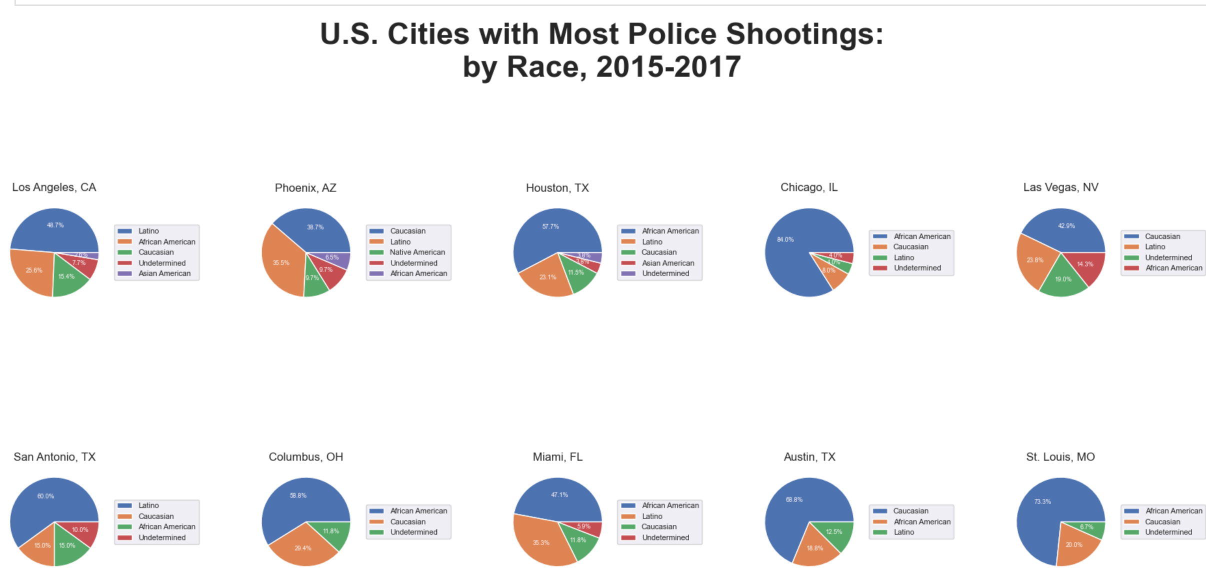
In [ ]:
```
# Create series to graph pie charts for each city by race:
los_angeles_race_counts = df_los_angeles['race'].value_counts()
phoenix_race_counts = df_phoenix['race'].value_counts()
houston_race_counts = df_houston['race'].value_counts()
chicago_race_counts = df_chicago['race'].value_counts()
las_vegas_race_counts = df_las_vegas['race'].value_counts()
san_antonio_race_counts = df_san_antonio['race'].value_counts()
columbus_race_counts = df_columbus['race'].value_counts()
miami_race_counts = df_miami['race'].value_counts()
austin_race_counts = df_austin['race'].value_counts()
st_louis_race_counts = df_st_louis['race'].value_counts()
```

In [ ]:
```
# Create list to plot each city:
```

```python
city_tuples = [(los_angeles_race_counts, 'Los Angeles, CA'),
               (phoenix_race_counts, 'Phoenix, AZ'),
               (houston_race_counts, 'Houston, TX'),
               (chicago_race_counts, 'Chicago, IL'),
               (las_vegas_race_counts, 'Las Vegas, NV'),
               (san_antonio_race_counts, 'San Antonio, TX'),
               (columbus_race_counts, 'Columbus, OH'),
               (miami_race_counts, 'Miami, FL'),
               (austin_race_counts, 'Austin, TX'),
               (st_louis_race_counts, 'St. Louis, MO')]
```

In [ ]:

```python
# fig = plt.figure(figsize=(18, 10), dpi=1600)  # High DPI for saving image
fig = plt.figure(figsize=(18, 10))

cols = 2
rows = 5
ind = 0

for col in range(cols):
    for row in range(rows):
        city = city_tuples[ind]
        ax1 = plt.subplot2grid((cols, rows), (col, row))
        # Plot each sub-pie chart and customize text:
        plt.pie(city[0],
                autopct='%1.1f%%',
                textprops={'color': 'snow', 'size': 6.5})
        plt.title(city[1])
        # Add legend and customize location/text:
        ax1.legend(labels=city[0].index,
                   loc='center left',
                   bbox_to_anchor=(1, 0.5),
                   prop={'size': 8})
        ind += 1

title = "U.S. Cities with Most Police Shootings:"
title += "\nby Race, 2015-2017"

plt.suptitle(title, size=32, weight='bold')

# Optimize plot/legend layout:
plt.tight_layout(w_pad=4.5)  # (Causes padding issue beneath title...)

plt.show()
```



Analysis:

These conclusions do not account for portions *Undetermined*:

- Columbus had killings of only *Caucasians* and *African Americans*.
- *Austin* had killings of only *Caucasians*, *African Americans* and *Latinos*.
- The most killed in Los Angeles and *San Antonio* were *Latino*.
- Caucasians were the highest number killed in *Austin*, *Las Vegas* and *Phoenix*.
- African Americans were the most killed in *Houston*, *Chicago*, *Columbus*, *Miami* and *St. Louis*.

Create a Choropleth Map of Police Killings by US State

Which states are the most dangerous? Compare your map with your previous chart. Are these the same states with high degrees of poverty?

In [ ]:
```
df_state = df_fatalities.groupby('state').size().reset_index(name='Number Killed')
```

In [ ]:
```
df_state.sort_values(by='Number Killed', ascending=False, inplace=True)
```

In [ ]:
```
df_state.reset_index(inplace=True, drop=True)
```

In [ ]:
```
# Try different color scales:
color_scales = ['aggrnyl', 'agsunset', 'blackbody', 'bluered', 'blues', 'blugrn',
                 'bluyl', 'brwnyl', 'bugn', 'bupu', 'burg', 'burgyl', 'cividis',
                 'darkmint', 'electric', 'emrld', 'gnbu', 'greens', 'greys', 'hot',
                 'inferno', 'jet', 'magenta', 'magma', 'mint', 'orrd', 'oranges', 'o
                 'peach', 'pinkyl', 'plasma', 'plotly3', 'pubu', 'pubugn', 'purd', '
                 'purples', 'purpor', 'rainbow', 'rdbu', 'rdpu', 'redor', 'reds', 's
                 'sunsetdark  teal', 'tealgrn', 'turbo', 'viridis', 'ylgn', 'ylgnbu'
                 'ylorbr', 'ylorrd', 'algae', 'amp', 'deep', 'dense', 'gray', 'halin
                 'ice', 'matter', 'solar', 'speed', 'tempo', 'thermal', 'turbid',
                 'armyrose', 'brbg', 'earth', 'fall', 'geyser', 'prgn', 'piyg', 'pic
                 'portland', 'puor', 'rdgy', 'rdylbu', 'rdylgn', 'spectral', 'tealro
                 'temps', 'tropic', 'balance', 'curl', 'delta', 'oxy', 'edge', 'hsv'
                 'icefire', 'phase', 'twilight', 'mrybm', 'mygbm']
```

In [ ]:
```
fig = px.choropleth(
    df_state,
    locations='state',
    color='Number Killed',
    locationmode='USA-states',
    scope='usa',
    color_continuous_scale='aggrnyl',
    width=1000,
    height=650,
)
```

```
title = f"Choropleth Map: U.S. Police Killings 2015-2017"

fig.update_layout(
    title_text=title,
    title_font_size=28,
    title_font_family='Courier New',
    title_xref='paper',
)

fig.show()
```

Analysis:

- California, Texas and Florida have the most killings.
- The Upper Midwest and Northeast have the fewest.

In [ ]:
```
df_pov_state['Geographic Area'].values
```

Out[ ]:
```
array(['AZ', 'MS', 'NM', 'GA', 'AR', 'WV', 'LA', 'SC', 'AK', 'AL', 'OK',
       'TX', 'MO', 'KY', 'TN', 'NC', 'ID', 'MT', 'SD', 'CA', 'MI', 'DC',
       'FL', 'OR', 'ME', 'IN', 'WA', 'NV', 'KS', 'VA', 'OH', 'CO', 'VT',
       'ND', 'IL', 'MN', 'NE', 'NH', 'HI', 'WY', 'WI', 'PA', 'IA', 'UT',
       'DE', 'NY', 'MD', 'RI', 'MA', 'CT', 'NJ'], dtype=object)
```

- California doesn't have the most poverty, so that doesn't exactly match up with the killings by state statistic -- but it does have a very high population, so we should consider that, and perphaps bring that statistic into our analysis in the next refactor...
- Massachusetts and Connecticut seem to abide by the cross-analysis of low-poverty and fewer number of shootings.

Number of Police Killings Over Time

Analyse the Number of Police Killings over Time. Is there a trend in the data?

Clean/transform data:

In [ ]:
```
# Make a copy in case of mistakes:
df_fatalities_months = df_fatalities.filter(['date'], axis=1)
df_fatalities_days = df_fatalities.filter(['date'], axis=1)
```

In [ ]:
```
# Convert to datetime objects:
df_fatalities_months['date'] = pd.to_datetime(df_fatalities_months['date'], format=
df_fatalities_days['date'] = pd.to_datetime(df_fatalities_months['date'], format='%
```

In [ ]:
```
# Make the data for `df_fatalities_months` scale on months, not days:
```

```
df_fatalities_months['date'] = df_fatalities_months['date'].dt.strftime('%m/%y')
```

In [ ]:
```
# Make DF grouped by date (months) and their counts for ease/clarity of plotting:
df_fatalities_months = df_fatalities_months.groupby('date').size().reset_index(name

# Make DF grouped by date (days):
df_fatalities_days = df_fatalities_days.groupby('date').size().reset_index(name='co
```

We now have a DataFrame based on months and another on days.

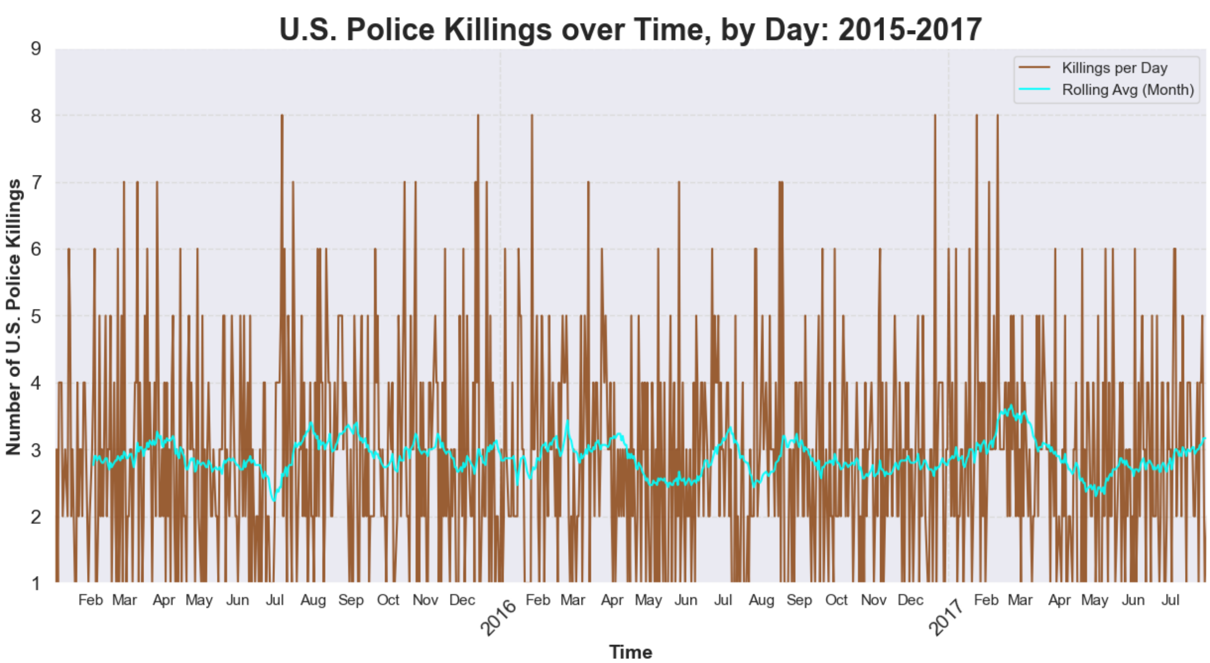
We can look at plots for each and compare. The months should be better for 'at-a-glance', the days for a more in-depth analysis.

Plot:

In [ ]:
```
# Import and define matplotlib.dates helpers:
import matplotlib.dates as mdates

years = mdates.YearLocator()
year_fmt = mdates.DateFormatter('%Y')

months = mdates.MonthLocator()
month_fmt = mdates.DateFormatter('%b')
```

In [ ]:
```
# Plot months:
fig, ax = plt.subplots(figsize=(15, 7))

title = "U.S. Police Killings over Time, by Day: 2015-2017"

ax.set_title(title,
             fontsize=22,
             weight='bold')

ax.tick_params(axis='both', labelsize=10)

ax.set_xlabel('Time', fontsize=14, weight='bold')
ax.set_ylabel('Number of U.S. Police Killings', fontsize=14, weight='bold')

plt.xticks(fontsize=14, rotation=45)
plt.yticks(fontsize=14)

ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(year_fmt)
ax.xaxis.set_minor_locator(months)
ax.xaxis.set_minor_formatter(month_fmt)

all_dates = df_fatalities_days['date']
all_counts = df_fatalities_days['count']

ax.set_xlim([all_dates.min(), all_dates.max()])
ax.set_ylim([all_counts.min(), all_counts.max() + 1])

ax.grid(color='gainsboro', linestyle='--')

ax.plot(
    all_dates,
    all_counts,
    label="Killings per Day",
```

```
    color='saddlebrown',
    alpha=.85,
)

# Optional rolling windows:
# rolling_2_days = df_fatalities_days['count'].rolling(window=2).mean()
# ax.plot(
#     all_dates,
#     rolling_2_days,
#     label="Killings per 2-Days",
#     color='black',
#     alpha=.85,
# )

# rolling_week = df_fatalities_days['count'].rolling(window=7).mean()
# ax.plot(
#     all_dates,
#     rolling_week,
#     label=f"Killings per Week",
#     color='crimson',
#     alpha=.9,
# )

rolling_month = df_fatalities_days['count'].rolling(window=30).mean()
ax.plot(
    all_dates,
    rolling_month,
    label=f"Rolling Avg (Month)",
    color='cyan',
    alpha=.9,
)

plt.legend()
plt.show()
```



Analysis:

The consistencies I see:

- Winter tends to have more killings.

- May-June tends to have fewer killings.

We should isolate each year to get more specific analysis -- I'll do this in the next draft.

Epilogue

Now that you have analysed the data yourself, read The Washington Post's analysis here.

Analysis and Conclusion:

At the outset I scoured the internet searching for updated data -- upon looking at this link from the Washington Post more clearly, I see they have the updated data there!

I'll factor this in during my next pass over this project.

It pays to pay attention!!

The Washington Post makes it clear that African Americans are killed at a disproportionate rate.

In my next refactor, I need to do a better analysis of population by race and killings by race.

Either I mis-read the instructions for this project/assignment or there wasn't enough emphasis placed on comparing these important two factors.

In any case, that's on me, since this is a project which allows freedom in how I analyze, so I will do that, along with updating the dataset, on my next pass.

For now I'm satisfied with my ability to answer this projects challenges, and feel like my handle on Pandas, Matplotlib, Seaborn and Plotly has vastly improved.

In conclusion, my technical abilities seem good for this project, but I need a bit more work on the creative/intellectual side of how to analyze and compare the data points, and should feel more free to go outside the box when it comes to finding ways to look at the data that might benefit the analysis.

A lot that required research/looking up has now become a lot more natural.

I'm almost finished with this course.

This is the final day's project, but I waited to finish the day where we add a 'store' to our website -- this will involve adding a database and login functionality for users.

I purposefully left this for last, since it will be very challenging, meaningful and a continuing project for me after this course is finished.