# Fatal_Force_(start)

November 21, 2025

## 1 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.

#### 1.0.1 Upgrade Plotly

Run the cell below if you are working with Google Colab

[111]:
```
# %pip install --upgrade plotly

# # Need this to make map chart to png
# %conda install -c plotly plotly-orca
```

### 1.1 Import Statements

[1]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns

from collections import Counter

import matplotlib.patches as mpatches
import matplotlib.colors as colors
```

```python
import random
```

### 1.2 Notebook Presentation

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

### 1.3 Load the Data

[3]:
```python
df_hh_income = pd.read_csv('Median_Household_Income_2015.csv',␣
    ↪encoding="windows-1252")
df_pct_poverty = pd.read_csv('Pct_People_Below_Poverty_Level.csv',␣
    ↪encoding="windows-1252")
df_pct_completed_hs = pd.read_csv('Pct_Over_25_Completed_High_School.csv',␣
    ↪encoding="windows-1252")
df_share_race_city = pd.read_csv('Share_of_Race_By_City.csv',␣
    ↪encoding="windows-1252")
df_fatalities = pd.read_csv('Deaths_by_Police_US.csv', encoding="windows-1252")
```

## 2 Preliminary Data Exploration

- What is the shape of the DataFrames?
- How many rows and columns do they have?
- What are the column names?
- Are there any NaN values or duplicates?

[4]:
```python
df_fatalities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2535 entries, 0 to 2534
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       2535 non-null   int64
 1   name                     2535 non-null   object
 2   date                     2535 non-null   object
 3   manner_of_death          2535 non-null   object
 4   armed                    2526 non-null   object
 5   age                      2458 non-null   float64
 6   gender                   2535 non-null   object
 7   race                     2340 non-null   object
 8   city                     2535 non-null   object
 9   state                    2535 non-null   object
 10  signs_of_mental_illness  2535 non-null   bool
 11  threat_level             2535 non-null   object
 12  flee                     2470 non-null   object
 13  body_camera              2535 non-null   bool
```

```
dtypes: bool(2), float64(1), int64(1), object(10)
memory usage: 242.7+ KB
```

[5]:
```python
df_hh_income.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29322 entries, 0 to 29321
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Geographic Area  29322 non-null  object
 1   City             29322 non-null  object
 2   Median Income    29271 non-null  object
dtypes: object(3)
memory usage: 687.4+ KB
```

[6]:
```python
df_pct_completed_hs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Geographic Area       29329 non-null  object
 1   City                  29329 non-null  object
 2   percent_completed_hs  29329 non-null  object
dtypes: object(3)
memory usage: 687.5+ KB
```

[7]:
```python
df_pct_poverty.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Geographic Area  29329 non-null  object
 1   City             29329 non-null  object
 2   poverty_rate     29329 non-null  object
dtypes: object(3)
memory usage: 687.5+ KB
```

[8]:
```python
df_share_race_city.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29268 entries, 0 to 29267
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Geographic area        29268 non-null  object
```

```
 1   City                    29268 non-null  object
 2   share_white             29268 non-null  object
 3   share_black             29268 non-null  object
 4   share_native_american   29268 non-null  object
 5   share_asian             29268 non-null  object
 6   share_hispanic          29268 non-null  object
dtypes: object(7)
memory usage: 1.6+ MB
```

### 2.1 Data Cleaning - Check for Missing Values and Duplicates

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

### 2.2 Check for NaN values

[9]:
```
df_fatalities.isna().any()
```

[9]:
```
id                         False
name                       False
date                       False
manner_of_death            False
armed                       True
age                         True
gender                     False
race                        True
city                       False
state                      False
signs_of_mental_illness    False
threat_level               False
flee                        True
body_camera                False
dtype: bool
```

[10]:
```
df_fatalities.isna().sum()
```

[10]:
```
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
```

[11]:
```
df_hh_income.isna().any()
```

[11]:
```
Geographic Area    False
City               False
Median Income       True
dtype: bool
```

[12]:
```
#Replace NaN with 0
df_hh_income['Median Income'] = df_hh_income['Median Income'].replace(np.nan, 0)
```

[13]:
```
df_pct_completed_hs.isna().any()
```

[13]:
```
Geographic Area         False
City                    False
percent_completed_hs    False
dtype: bool
```

[14]:
```
df_pct_poverty.isna().any()
```

[14]:
```
Geographic Area    False
City               False
poverty_rate       False
dtype: bool
```

[15]:
```
df_share_race_city.isna().any()
```

[15]:
```
Geographic area          False
City                     False
share_white              False
share_black              False
share_native_american    False
share_asian              False
share_hispanic           False
dtype: bool
```

### 2.3 Check Duplicate Rows

[127]:
```
df_fatalities.duplicated().any()
```

[127]:
```
False
```

[128]:
```
df_hh_income.duplicated().any()
```

[128]:
```
False
```

[129]:
```
df_pct_completed_hs.duplicated().any()
```

[129]:
```
False
```

[130]:
```
df_pct_poverty.duplicated().any()
```

[130]:
```
False
```

[131]:
```
df_share_race_city.duplicated().any()
```

[131]:
```
False
```

## 3 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

[132]:
```
df_share_race_city.head(10)
```

[132]:
```
  Geographic area                City share_white share_black  \
0              AL          Abanda CDP        67.2        30.2
1              AL      Abbeville city        54.4        41.4
2              AL     Adamsville city        52.3        44.9
3              AL        Addison town        99.1         0.1
4              AL          Akron town        13.2        86.5
5              AL      Alabaster city        79.4        13.5
6              AL    Albertville city        75.9         1.9
7              AL  Alexander City city        62.2          32
8              AL      Alexandria CDP        87.4        10.2
9              AL     Aliceville city        22.6        74.9

  share_native_american share_asian share_hispanic
0                     0           0            1.6
1                   0.1           1            3.1
2                   0.5         0.3            2.3
3                     0         0.1            0.4
4                     0           0            0.3
5                   0.4         0.9              9
6                   0.8         0.5           27.9
7                   0.2         0.9            4.8
8                   0.3         0.5            0.9
9                   0.1           0            1.2
```

[133]:
```
df_hh_income.head(10)
```

[133]:
```
  Geographic Area                City Median Income
0              AL          Abanda CDP         11207
1              AL      Abbeville city         25615
```

```
2               AL      Adamsville city        42575
3               AL         Addison town        37083
4               AL           Akron town        21667
5               AL       Alabaster city        71816
6               AL     Albertville city        32911
7               AL  Alexander City city        29874
8               AL       Alexandria CDP        56058
9               AL      Aliceville city        21131
```

[134]:
```python
df_pct_completed_hs.head(10)
```

[134]:
```
  Geographic Area                 City percent_completed_hs
0               AL          Abanda CDP                 21.2
1               AL       Abbeville city                69.1
2               AL      Adamsville city                78.9
3               AL         Addison town                81.4
4               AL           Akron town                68.6
5               AL       Alabaster city                89.3
6               AL     Albertville city                72.7
7               AL  Alexander City city                78.1
8               AL       Alexandria CDP                88.8
9               AL      Aliceville city                74.3
```

[135]:
```python
df_pct_poverty.head(10)
```

[135]:
```
  Geographic Area                 City poverty_rate
0               AL          Abanda CDP         78.8
1               AL       Abbeville city         29.1
2               AL      Adamsville city         25.5
3               AL         Addison town         30.7
4               AL           Akron town           42
5               AL       Alabaster city         11.2
6               AL     Albertville city         26.7
7               AL  Alexander City city         30.4
8               AL       Alexandria CDP          9.7
9               AL      Aliceville city         41.3
```

[136]:
```python
df_pct_poverty.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Geographic Area  29329 non-null  object
 1   City             29329 non-null  object
 2   poverty_rate     29329 non-null  object
dtypes: object(3)
```

```
memory usage: 687.5+ KB
```

[137]:
```
df_pct_poverty['Geographic Area'].unique()
```

[137]:
```
array(['AL', 'AK', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DE', 'DC', 'FL', 'GA',
       'HI', 'ID', 'IL', 'IN', 'IA', 'KS', 'KY', 'LA', 'ME', 'MD', 'MA',
       'MI', 'MN', 'MS', 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ', 'NM', 'NY',
       'NC', 'ND', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX',
       'UT', 'VT', 'VA', 'WA', 'WV', 'WI', 'WY'], dtype=object)
```

[138]:
```
df_pct_poverty['poverty_rate'].unique()
```

[138]:
```
array(['78.8', '29.1', '25.5', '30.7', '42', '11.2', '26.7', '30.4',
       '9.7', '41.3', '27.7', '27.5', '24.5', '13.8', '31.7', '16.7',
       '19.1', '8.6', '37.6', '31.6', '18.8', '22.4', '28.3', '13',
       '24.7', '20.3', '31.8', '24.1', '22.2', '6.8', '15.7', '28.6',
       '7.1', '38.2', '26.3', '30.1', '25.3', '44.9', '8.1', '28.8',
       '39.1', '42.9', '36.7', '0', '30.9', '32.9', '20.5', '12.7',
       '41.2', '0.5', '4', '19.4', '60.3', '47.6', '18.2', '53.7', '20.7',
       '19.2', '17.3', '27.8', '34.7', '33', '22.5', '39.7', '11.5',
       '10.8', '23.8', '32.4', '5.8', '79.4', '31.9', '36.6', '32.2',
       '30', '17.2', '18.9', '7.4', '39.6', '25.8', '25', '25.1', '19.8',
       '4.9', '19.9', '33.6', '38.6', '14.7', '16.9', '6.6', '16.4',
       '29.3', '15', '31.4', '19.5', '21.2', '10', '32.1', '24.8', '20.1',
       '24', '7.9', '23.5', '32.8', '12.8', '29.8', '10.7', '11.7',
       '18.1', '38.1', '21', '21.9', '8.5', '9.3', '3.6', '12.9', '48',
       '5.1', '28.1', '31', '27.9', '34.9', '30.3', '7.7', '24.2', '30.6',
       '13.4', '26.5', '4.5', '31.1', '15.8', '37.8', '8.2', '16.3',
       '32.3', '32.7', '11.1', '52.6', '18.6', '23.4', '26', '30.2',
       '25.9', '15.3', '15.5', '20.4', '24.6', '17', '48.1', '19', '8',
       '18.4', '31.3', '13.5', '14', '29.4', '40.9', '3.9', '8.3', '19.3',
       '7.8', '35.5', '21.5', '21.4', '13.6', '15.2', '26.9', '39.2',
       '6.4', '7.2', '8.9', '17.4', '39.3', '26.1', '37.5', '16.6',
       '16.2', '30.8', '12', '25.7', '17.8', '26.8', '18.5', '6.9', '5.5',
       '44.4', '14.9', '13.1', '32.5', '22.6', '2.1', '9.2', '2', '17.6',
       '23.2', '13.7', '24.4', '29', '46.4', '6', '10.2', '14.3', '46',
       '15.4', '4.4', '31.2', '22.3', '24.9', '10.9', '17.7', '6.3',
       '2.2', '18', '25.6', '6.7', '16.1', '21.3', '4.8', '11.6', '55.2',
       '14.2', '11.8', '3.4', '20', '62.6', '4.2', '40', '5.3', '63.7',
       '9.9', '10.1', '6.5', '46.9', '60.5', '31.5', '43.1', '8.8',
       '23.3', '9.5', '55.8', '24.3', '18.7', '3', '1.7', '11', '30.5',
       '7', '29.7', '21.7', '2.3', '26.2', '18.3', '1.9', '35', '20.9',
       '14.8', '9.6', '29.5', '72.7', '23', '10.5', '37.3', '23.1',
       '35.6', '40.6', '4.6', '20.8', '39.5', '34.4', '48.5', '44.7',
       '12.1', '36.3', '29.9', '5.9', '9.8', '39.9', '34.6', '35.9',
       '15.9', '10.6', '28.7', '11.4', '5', '17.9', '17.5', '12.5',
       '10.3', '16', '41.9', '45.5', '36.8', '2.6', '49', '7.5', '4.7',
       '1.4', '26.4', '20.6', '1.6', '27.2', '17.1', '19.6', '12.2',
```

```
'13.3', '70.8', '29.6', '33.5', '27.1', '27.6', '16.5', '22.7',
'6.2', '48.4', '12.3', '9', '4.3', '23.9', '39', '22', '5.7',
'43.2', '-', '42.7', '25.2', '26.6', '44.1', '46.8', '40.5',
'44.6', '9.1', '1.5', '33.3', '100', '57.6', '5.2', '34.8', '50.5',
'63.6', '13.2', '55.4', '13.9', '39.8', '65.1', '5.4', '4.1',
'11.9', '3.7', '39.4', '36.2', '12.6', '46.2', '6.1', '21.8',
'2.8', '35.8', '35.1', '49.1', '48.9', '46.3', '46.7', '7.3',
'20.2', '85.7', '35.2', '16.8', '57.3', '53.1', '42.3', '21.1',
'11.3', '37.2', '50', '43.9', '27.3', '33.7', '78', '28.2', '38',
'33.2', '8.4', '42.8', '92.9', '3.1', '72', '62.5', '14.5', '14.6',
'74.7', '7.6', '9.4', '34.3', '57.4', '54.5', '73.2', '49.4',
'40.7', '2.9', '74.8', '43.7', '29.2', '22.1', '45.1', '61',
'21.6', '19.7', '45.3', '56.9', '25.4', '43.8', '65.9', '52.9',
'41', '68.8', '59.2', '38.8', '64.6', '28.5', '67.2', '51.5',
'27.4', '54.7', '1.3', '88.9', '42.2', '41.6', '32', '23.7',
'78.5', '28.4', '47.8', '12.4', '41.8', '52.1', '60.6', '54.1',
'43.4', '83.7', '42.5', '64.5', '8.7', '76.4', '40.2', '45.9',
'35.7', '41.5', '50.1', '43.3', '59.9', '5.6', '74.6', '14.4',
'77', '51', '45.2', '70.2', '88.4', '15.6', '52.3', '45', '79.3',
'37.9', '51.7', '55.7', '51.2', '33.9', '58.8', '28.9', '78.1',
'48.7', '68.5', '57.8', '60.7', '72.5', '27', '41.7', '65.3', '53',
'80.6', '34.1', '63.2', '50.4', '47.9', '22.8', '23.6', '36',
'2.4', '74.4', '3.5', '33.1', '46.5', '22.9', '32.6', '37.1',
'36.5', '36.1', '37.4', '38.4', '45.4', '28', '34', '2.7', '60.4',
'1.8', '47.7', '44.2', '63.3', '67.5', '35.4', '47.4', '37',
'52.8', '0.8', '52.7', '48.2', '61.7', '34.2', '2.5', '34.5',
'61.6', '53.3', '46.1', '36.4', '52.5', '56', '43', '64.9', '64',
'40.3', '40.8', '81.9', '3.2', '49.5', '3.8', '15.1', '33.4',
'0.1', '38.5', '0.6', '59.7', '14.1', '48.8', '45.7', '10.4',
'35.3', '61.4', '47.3', '44', '1', '60.2', '38.3', '68', '1.1',
'1.2', '87', '3.3', '38.9', '57', '47.2', '70.6', '59.6', '83.1',
'45.6', '86', '67.1', '41.4', '37.7', '47.1', '50.2', '66.8',
'58.2', '49.9', '36.9', '83', '67.6', '54.2', '58.7', '38.7',
'40.4', '0.7', '53.4', '50.8', '69', '70.3', '48.6', '42.1',
'85.2', '75', '42.4', '47', '51.8', '44.3', '71.4', '53.6', '33.8',
'57.1', '41.1', '48.3', '52.4', '63.8', '98.6', '76.2', '44.8',
'46.6', '56.1', '81.2', '58', '61.3', '42.6', '40.1', '49.2',
'53.5', '52.2', '43.6', '51.9', '55.6', '47.5', '65.5', '73.1',
'67.3', '85', '63.4', '49.3', '45.8', '0.2', '0.9', '43.5', '55',
'51.1', '59.1', '54', '80.1', '86.6', '82.4', '72.6', '91.2',
'69.1', '74.2', '44.5', '52', '51.3', '53.9', '57.5', '57.9',
'62.4', '70', '51.6', '66.2', '50.9', '0.3', '62.7', '71.7',
'50.3', '49.8', '57.7', '77.4', '54.9', '53.2', '61.9', '57.2',
'54.3', '54.8', '49.6', '62.9', '55.9', '73.8', '92', '66.7',
'73.5', '60.8', '83.3', '60', '65.7', '65.8', '66.3', '71.8',
'56.7', '72.2', '55.3', '74.5', '81.5', '56.6', '68.6', '76.7',
'67.4', '92.3', '80', '58.6', '62.8', '73.9', '65.4', '68.9',
```

```
 '76.6', '69.7', '79.2', '72.4', '92.7', '0.4', '73.7', '56.8',
 '70.5', '87.1', '88.7', '86.1', '87.3', '59.8', '58.1', '77.3',
 '64.3', '68.2', '73', '55.1', '84.9', '55.5', '90.9', '56.5',
 '93.8', '49.7', '69.2', '84.7', '76.3', '56.3', '53.8', '51.4',
 '87.8', '54.4', '94.1', '76.5', '71.6', '71.1', '84.8', '63.1',
 '69.3', '61.5', '89.6', '88', '68.7', '62.3', '58.5', '69.4',
 '80.2', '88.2', '83.9', '89', '86.7', '62.2', '50.7', '81.7',
 '84.3', '59.4', '67.8', '65', '80.7', '81.8', '93.3', '91.9',
 '56.2', '70.9', '70.7', '50.6', '56.4', '78.3', '93.5', '74.1',
 '77.9', '74.9', '73.3', '77.8', '93.4', '74.3', '77.2', '83.6'],
dtype=object)
```

[139]:
```
df_pct_poverty[df_pct_poverty.poverty_rate == "-"]
```

[139]:
```
      Geographic Area               City poverty_rate
573                AL        Whatley CDP            -
608                AK   Attu Station CDP            -
632                AK        Chicken CDP            -
637                AK        Chisana CDP            -
662                AK       Dot Lake CDP            -
…                 …                  …          …
29261              WY         Oakley CDP            -
29266              WY      Owl Creek CDP            -
29273              WY   Powder River CDP            -
29289              WY      Ryan Park CDP            -
29304              WY     Table Rock CDP            -

[201 rows x 3 columns]
```

[140]:
```
len(df_pct_poverty['poverty_rate'].unique())
```

[140]: 771

[141]:
```
df_pct_poverty.poverty_rate.replace('-', np.nan, regex=True, inplace=True)
```

[142]:
```
df_pct_poverty[df_pct_poverty.poverty_rate.isna()]
```

[142]:
```
      Geographic Area               City poverty_rate
573                AL        Whatley CDP          NaN
608                AK   Attu Station CDP          NaN
632                AK        Chicken CDP          NaN
637                AK        Chisana CDP          NaN
662                AK       Dot Lake CDP          NaN
…                 …                  …          …
29261              WY         Oakley CDP          NaN
29266              WY      Owl Creek CDP          NaN
29273              WY   Powder River CDP          NaN
```

```
29289              WY      Ryan Park CDP          NaN
29304              WY     Table Rock CDP          NaN

[201 rows x 3 columns]
```

[143]:
```python
df_pct_poverty.poverty_rate = df_pct_poverty.poverty_rate.astype(float)
```

[144]:
```python
df_pct_poverty.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Geographic Area   29329 non-null  object
 1   City              29329 non-null  object
 2   poverty_rate      29128 non-null  float64
dtypes: float64(1), object(2)
memory usage: 687.5+ KB
```

[145]:
```python
poverty = df_pct_poverty.groupby('Geographic Area')['poverty_rate'].mean().
 ↪sort_values(ascending = False)
```

[146]:
```python
poverty
```

[146]:
```
Geographic Area
MS   26.88
AZ   25.67
GA   23.78
NM   23.08
AR   22.96
LA   22.34
SC   22.16
WV   21.13
OK   20.66
AL   20.65
MO   20.11
KY   20.08
TX   19.92
TN   19.89
AK   19.85
NC   19.75
ID   18.24
DC   18.00
MI   17.90
FL   17.57
CA   17.12
ME   16.89
```

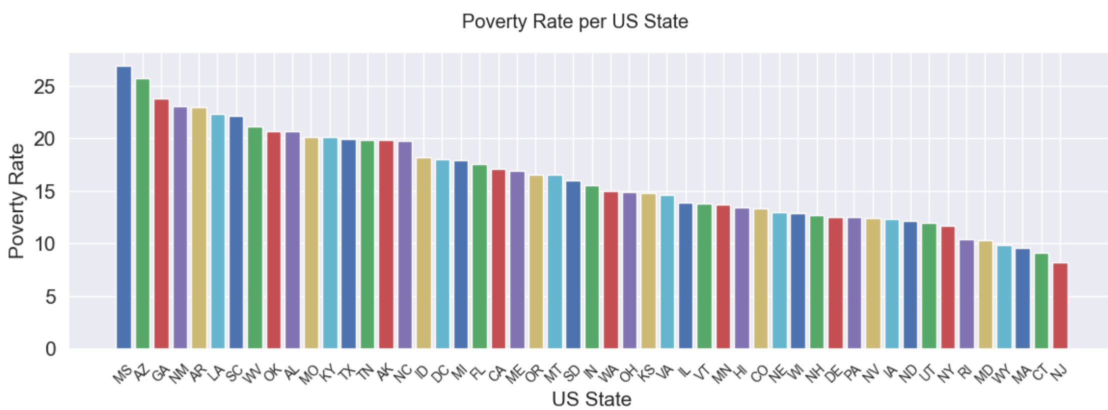
```
OR    16.52
MT    16.51
SD    16.03
IN    15.50
WA    15.02
OH    14.85
KS    14.76
VA    14.59
IL    13.88
VT    13.79
MN    13.75
HI    13.40
CO    13.36
NE    12.98
WI    12.86
NH    12.66
DE    12.56
PA    12.52
NV    12.47
IA    12.29
ND    12.16
UT    11.98
NY    11.67
RI    10.37
MD    10.31
WY     9.89
MA     9.59
CT     9.14
NJ     8.19
Name: poverty_rate, dtype: float64
```

[147]:

```python
plt.style.use('seaborn-deep')

plt.figure(figsize=(14,4))
plt.suptitle('Poverty Rate per US State')
plt.ylabel('Poverty Rate', fontsize = 15)
plt.xlabel('US State', fontsize = 15)

for n in range(len(poverty)):
    plt.xticks(fontsize = 10, rotation = 45)
    plt.yticks(fontsize = 15)
    plt.bar(poverty.index[n], poverty[n])

plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_11456\2074896531.py:1:
MatplotlibDeprecationWarning:
```

```
The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no
longer correspond to the styles shipped by seaborn. However, they will remain
available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API
instead.
```




## 4 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?

[148]:
```
df_pct_completed_hs.percent_completed_hs.replace('-', np.nan, regex = True,␣
    ↪inplace = True)
df_pct_completed_hs.percent_completed_hs = df_pct_completed_hs.
    ↪percent_completed_hs.astype(float)
```

[149]:
```
df_pct_completed_hs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29329 entries, 0 to 29328
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Geographic Area       29329 non-null  object
 1   City                  29329 non-null  object
 2   percent_completed_hs  29132 non-null  float64
dtypes: float64(1), object(2)
memory usage: 687.5+ KB
```

[150]:
```
graduation = df_pct_completed_hs.groupby('Geographic␣
    ↪Area')['percent_completed_hs'].mean().sort_values(ascending = False)
```

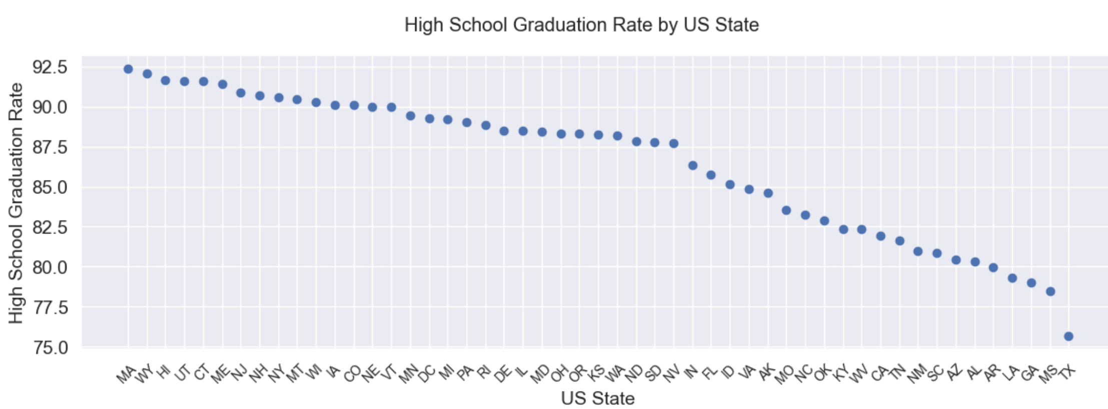
[151]:
```
graduation
```

[151]:

```
Geographic Area
MA    92.40
WY    92.10
HI    91.67
UT    91.62
CT    91.59
ME    91.43
NJ    90.85
NH    90.71
NY    90.61
MT    90.49
WI    90.26
IA    90.11
CO    90.11
NE    89.99
VT    89.98
MN    89.47
DC    89.30
MI    89.21
PA    89.02
RI    88.82
DE    88.52
IL    88.48
MD    88.42
OH    88.34
OR    88.30
KS    88.23
WA    88.20
ND    87.82
SD    87.75
NV    87.72
IN    86.32
FL    85.74
ID    85.17
VA    84.88
AK    84.63
MO    83.52
NC    83.25
OK    82.91
KY    82.37
WV    82.35
CA    81.96
TN    81.63
NM    80.98
SC    80.85
AZ    80.47
AL    80.30
```

```
AR    79.95
LA    79.29
GA    79.01
MS    78.47
TX    75.69
Name: percent_completed_hs, dtype: float64
```

[152]:
```python
plt.figure(figsize=(14,4))
plt.suptitle('High School Graduation Rate by US State')
plt.ylabel('High School Graduation Rate', fontsize=14)
plt.xlabel('US State', fontsize=14)

plt.xticks(fontsize = 10, rotation = 45)
plt.yticks(fontsize = 14)
plt.scatter(graduation.index, graduation)

plt.show()
```




# 5 Visualise the Relationship between Poverty Rates and High School Graduation Rates

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

[153]:
```python
graduation_vs = df_pct_completed_hs.groupby('Geographic␣
↪Area')['percent_completed_hs'].mean()
graduation_vs
```

[153]:
```
Geographic Area
AK    84.63
AL    80.30
AR    79.95
AZ    80.47
```

| | |
|---|---|
| CA | 81.96 |
| CO | 90.11 |
| CT | 91.59 |
| DC | 89.30 |
| DE | 88.52 |
| FL | 85.74 |
| GA | 79.01 |
| HI | 91.67 |
| IA | 90.11 |
| ID | 85.17 |
| IL | 88.48 |
| IN | 86.32 |
| KS | 88.23 |
| KY | 82.37 |
| LA | 79.29 |
| MA | 92.40 |
| MD | 88.42 |
| ME | 91.43 |
| MI | 89.21 |
| MN | 89.47 |
| MO | 83.52 |
| MS | 78.47 |
| MT | 90.49 |
| NC | 83.25 |
| ND | 87.82 |
| NE | 89.99 |
| NH | 90.71 |
| NJ | 90.85 |
| NM | 80.98 |
| NV | 87.72 |
| NY | 90.61 |
| OH | 88.34 |
| OK | 82.91 |
| OR | 88.30 |
| PA | 89.02 |
| RI | 88.82 |
| SC | 80.85 |
| SD | 87.75 |
| TN | 81.63 |
| TX | 75.69 |
| UT | 91.62 |
| VA | 84.88 |
| VT | 89.98 |
| WA | 88.20 |
| WI | 90.26 |
| WV | 82.35 |
| WY | 92.10 |

```
Name: percent_completed_hs, dtype: float64
```

[154]:
```
poverty_vs = df_pct_poverty.groupby('Geographic Area')['poverty_rate'].mean()
```

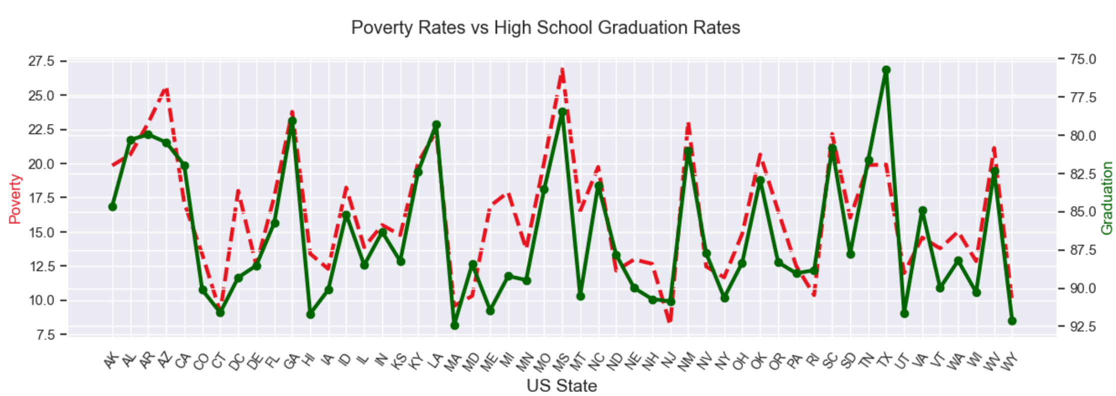
[155]:
```
poverty_vs
```

[155]:
```
Geographic Area
AK   19.85
AL   20.65
AR   22.96
AZ   25.67
CA   17.12
CO   13.36
CT    9.14
DC   18.00
DE   12.56
FL   17.57
GA   23.78
HI   13.40
IA   12.29
ID   18.24
IL   13.88
IN   15.50
KS   14.76
KY   20.08
LA   22.34
MA    9.59
MD   10.31
ME   16.89
MI   17.90
MN   13.75
MO   20.11
MS   26.88
MT   16.51
NC   19.75
ND   12.16
NE   12.98
NH   12.66
NJ    8.19
NM   23.08
NV   12.47
NY   11.67
OH   14.85
OK   20.66
OR   16.52
PA   12.52
RI   10.37
```

```
SC    22.16
SD    16.03
TN    19.89
TX    19.92
UT    11.98
VA    14.59
VT    13.79
WA    15.02
WI    12.86
WV    21.13
WY     9.89
Name: poverty_rate, dtype: float64
```

[156]:

```python
plt.figure(figsize=(14, 4))
plt.suptitle('Poverty Rates vs High School Graduation Rates')
plt.xlabel('US State', fontsize=14)
plt.xticks(fontsize=10, rotation=55)

ax1 = plt.gca()
ax2 = ax1.twinx()

ax1.set_ylabel('Poverty', color='#E5141E')
ax2.set_ylabel('Graduation', color='darkgreen')

ax1.plot(poverty_vs.index, poverty_vs, color='#E5141E', linewidth=3, linestyle='--')
ax2.plot(graduation_vs.index, graduation_vs, color='darkgreen', linewidth=3, marker='o')

# Reverse the y-axis for the graduation rate
ax2.invert_yaxis()

plt.show()
```



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

[157]:
```
df_pct_poverty
```

[157]:
```
      Geographic Area                    City  poverty_rate
0                  AL              Abanda CDP         78.80
1                  AL          Abbeville city         29.10
2                  AL         Adamsville city         25.50
3                  AL            Addison town         30.70
4                  AL              Akron town         42.00
…                 …                      …            …
29324              WY  Woods Landing-Jelm CDP         18.60
29325              WY            Worland city         15.30
29326              WY             Wright town          5.90
29327              WY              Yoder town          5.40
29328              WY            Y-O Ranch CDP          0.00

[29329 rows x 3 columns]
```

[158]:
```
df_pct_poverty.poverty_rate = pd.to_numeric(df_pct_poverty.poverty_rate)
df_pct_poverty.sort_values('poverty_rate', ascending=False, inplace=True)
poverty = df_pct_poverty.groupby('Geographic Area', as_index=False).
 ↪agg({'poverty_rate': pd.Series.mean})
```

[159]:
```
poverty
```

[159]:
```
   Geographic Area  poverty_rate
0               AK         19.85
1               AL         20.65
2               AR         22.96
3               AZ         25.67
4               CA         17.12
5               CO         13.36
6               CT          9.14
7               DC         18.00
8               DE         12.56
9               FL         17.57
10              GA         23.78
11              HI         13.40
12              IA         12.29
13              ID         18.24
14              IL         13.88
15              IN         15.50
16              KS         14.76
17              KY         20.08
18              LA         22.34
19              MA          9.59
```

```
20             MD          10.31
21             ME          16.89
22             MI          17.90
23             MN          13.75
24             MO          20.11
25             MS          26.88
26             MT          16.51
27             NC          19.75
28             ND          12.16
29             NE          12.98
30             NH          12.66
31             NJ           8.19
32             NM          23.08
33             NV          12.47
34             NY          11.67
35             OH          14.85
36             OK          20.66
37             OR          16.52
38             PA          12.52
39             RI          10.37
40             SC          22.16
41             SD          16.03
42             TN          19.89
43             TX          19.92
44             UT          11.98
45             VA          14.59
46             VT          13.79
47             WA          15.02
48             WI          12.86
49             WV          21.13
50             WY           9.89
```

[160]:

```
df_pct_completed_hs.percent_completed_hs = pd.to_numeric(df_pct_completed_hs.
 ↪percent_completed_hs)
df_pct_completed_hs.sort_values('percent_completed_hs', ascending=False,␣
 ↪inplace=True)
hs = df_pct_completed_hs.groupby('Geographic Area', as_index=False).
 ↪agg({'percent_completed_hs': pd.Series.mean})
```

[161]:

```
merged = pd.merge(hs, poverty, on=['Geographic Area'], how='inner')
```

[162]:

```
merged
```

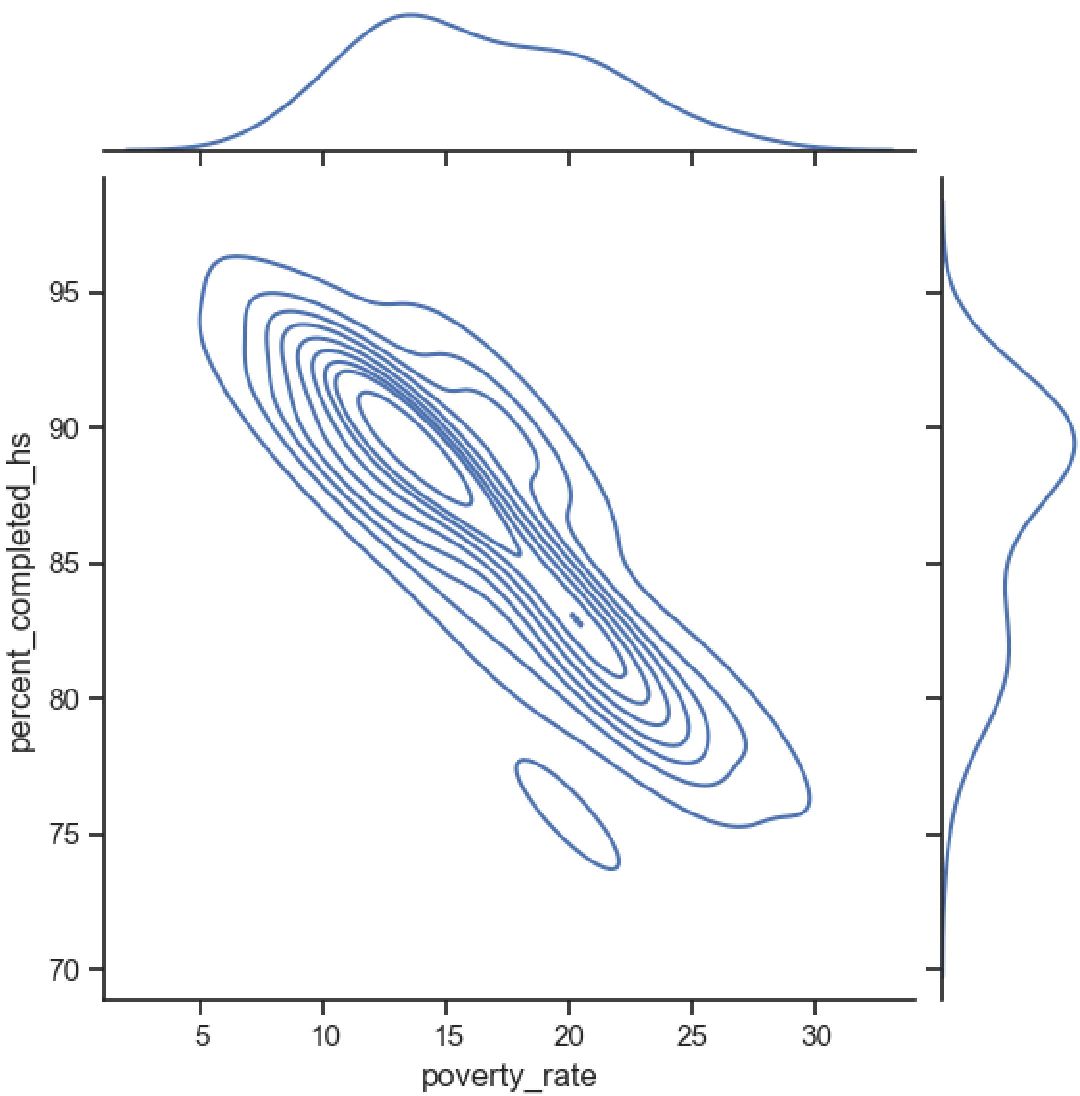
[162]:

```
   Geographic Area  percent_completed_hs  poverty_rate
0               AK                 84.63         19.85
1               AL                 80.30         20.65
2               AR                 79.95         22.96
```

| | | | |
|---|---|---|---|
| 3 | AZ | 80.47 | 25.67 |
| 4 | CA | 81.96 | 17.12 |
| 5 | CO | 90.11 | 13.36 |
| 6 | CT | 91.59 | 9.14 |
| 7 | DC | 89.30 | 18.00 |
| 8 | DE | 88.52 | 12.56 |
| 9 | FL | 85.74 | 17.57 |
| 10 | GA | 79.01 | 23.78 |
| 11 | HI | 91.67 | 13.40 |
| 12 | IA | 90.11 | 12.29 |
| 13 | ID | 85.17 | 18.24 |
| 14 | IL | 88.48 | 13.88 |
| 15 | IN | 86.32 | 15.50 |
| 16 | KS | 88.23 | 14.76 |
| 17 | KY | 82.37 | 20.08 |
| 18 | LA | 79.29 | 22.34 |
| 19 | MA | 92.40 | 9.59 |
| 20 | MD | 88.42 | 10.31 |
| 21 | ME | 91.43 | 16.89 |
| 22 | MI | 89.21 | 17.90 |
| 23 | MN | 89.47 | 13.75 |
| 24 | MO | 83.52 | 20.11 |
| 25 | MS | 78.47 | 26.88 |
| 26 | MT | 90.49 | 16.51 |
| 27 | NC | 83.25 | 19.75 |
| 28 | ND | 87.82 | 12.16 |
| 29 | NE | 89.99 | 12.98 |
| 30 | NH | 90.71 | 12.66 |
| 31 | NJ | 90.85 | 8.19 |
| 32 | NM | 80.98 | 23.08 |
| 33 | NV | 87.72 | 12.47 |
| 34 | NY | 90.61 | 11.67 |
| 35 | OH | 88.34 | 14.85 |
| 36 | OK | 82.91 | 20.66 |
| 37 | OR | 88.30 | 16.52 |
| 38 | PA | 89.02 | 12.52 |
| 39 | RI | 88.82 | 10.37 |
| 40 | SC | 80.85 | 22.16 |
| 41 | SD | 87.75 | 16.03 |
| 42 | TN | 81.63 | 19.89 |
| 43 | TX | 75.69 | 19.92 |
| 44 | UT | 91.62 | 11.98 |
| 45 | VA | 84.88 | 14.59 |
| 46 | VT | 89.98 | 13.79 |
| 47 | WA | 88.20 | 15.02 |
| 48 | WI | 90.26 | 12.86 |
| 49 | WV | 82.35 | 21.13 |

```
50              WY                  92.10           9.89
```

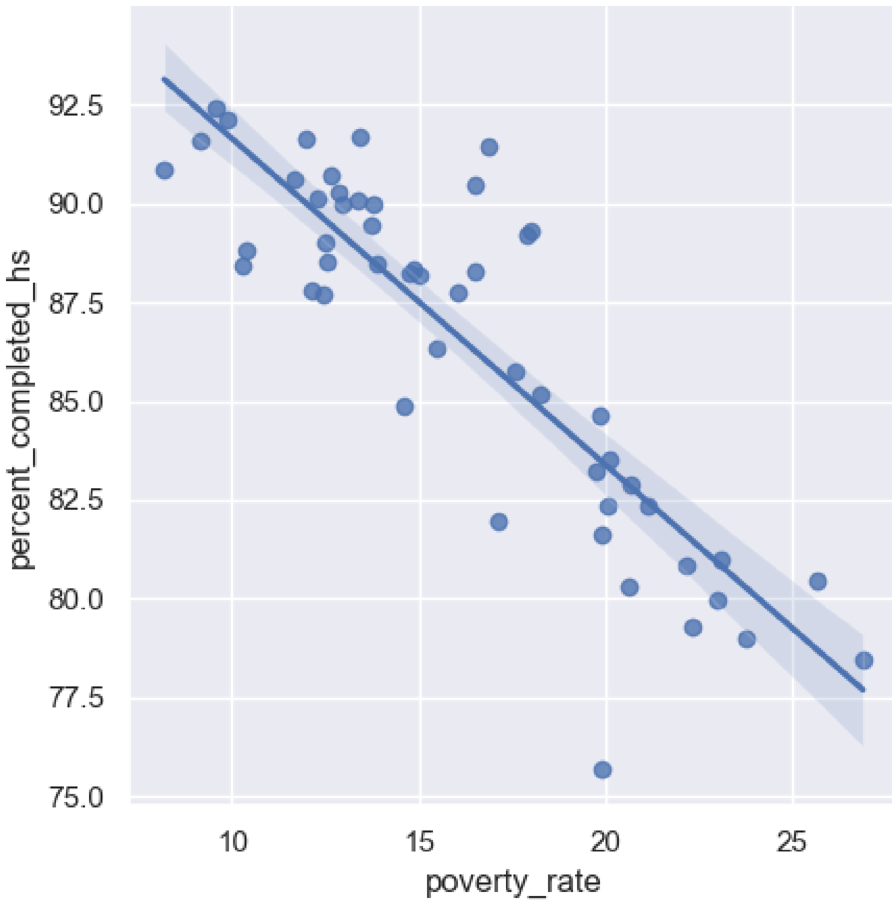
[163]:
```
sns.set_theme(style="ticks")

sns.jointplot(x='poverty_rate', y='percent_completed_hs', data=merged,
    kind='kde')
plt.show()
```



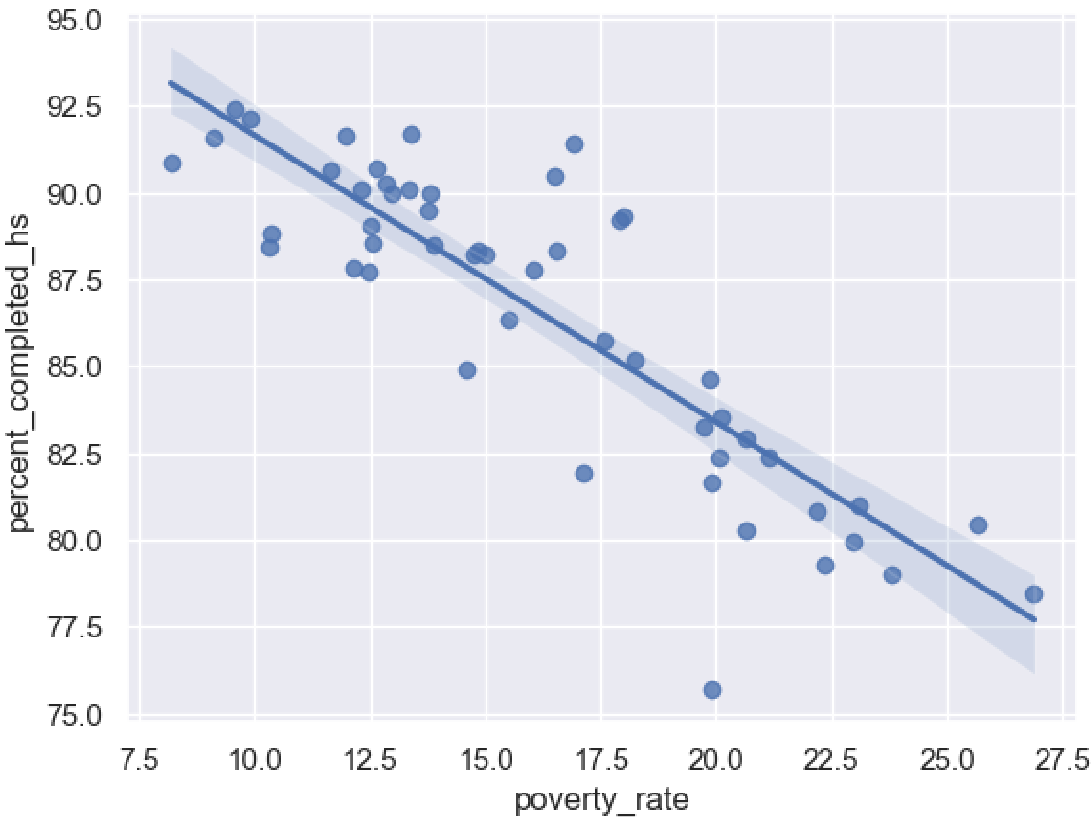

**Seaborn's .lmplot() or .regplot() to show a linear regression between the poverty ratio and the high school graduation ratio.**

[164]:
```
sns.set_theme(color_codes=True)
```

```
sns.lmplot(x='poverty_rate', y='percent_completed_hs', data=merged)
plt.show()
```




[165]:
```
sns.regplot(x='poverty_rate', y='percent_completed_hs', data=merged)
plt.show()
```




# 6 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.

[166]:
```
df_share_race_city
```

[166]:
```
      Geographic area                      City share_white share_black  \
0                  AL                Abanda CDP        67.2        30.2
1                  AL            Abbeville city        54.4        41.4
2                  AL           Adamsville city        52.3        44.9
3                  AL              Addison town        99.1         0.1
4                  AL                Akron town        13.2        86.5
…                 …                        …          …          …
29263              WY  Woods Landing-Jelm CDP        95.9           0
29264              WY              Worland city        89.9         0.3
29265              WY               Wright town        94.5         0.1
29266              WY                Yoder town        97.4           0
29267              WY             Y-O Ranch CDP        92.8         1.5
```

```
       share_native_american share_asian share_hispanic
0                          0           0            1.6
1                        0.1           1            3.1
2                        0.5         0.3            2.3
3                          0         0.1            0.4
4                          0           0            0.3
…                        …           …              …
29263                      0         2.1              0
29264                    1.3         0.6           16.6
29265                    1.4         0.2            6.2
29266                      0           0              4
29267                    2.6           0           11.8

[29268 rows x 7 columns]
```

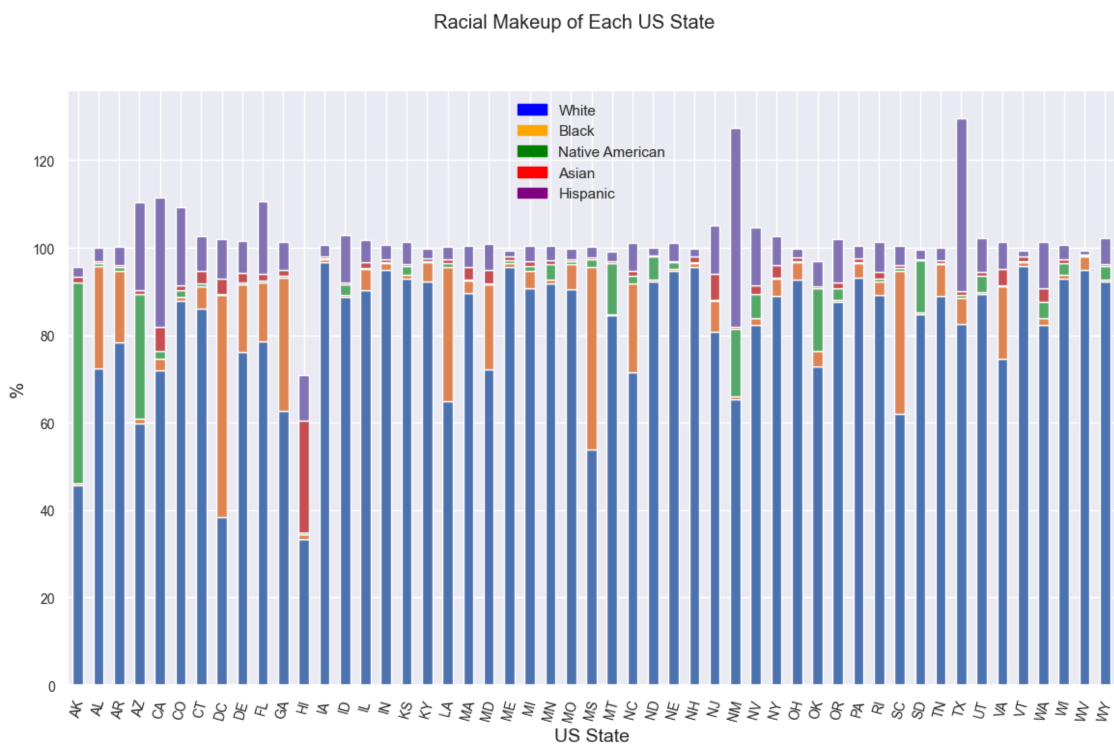
[167]:
```
df_share_race_city[['share_white', 'share_black', 'share_native_american',␣
↪'share_asian', 'share_hispanic']] = df_share_race_city[['share_white',␣
↪'share_black', 'share_native_american', 'share_asian', 'share_hispanic']].
↪apply(pd.to_numeric, errors='coerce')
```

[168]:
```
racial = df_share_race_city.groupby('Geographic area').agg({'share_white':␣
↪'mean', 'share_black': 'mean', 'share_native_american': 'mean',␣
↪'share_asian': 'mean', 'share_hispanic': 'mean'})
racial.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 51 entries, AK to WY
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   share_white            51 non-null     float64
 1   share_black            51 non-null     float64
 2   share_native_american  51 non-null     float64
 3   share_asian            51 non-null     float64
 4   share_hispanic         51 non-null     float64
dtypes: float64(5)
memory usage: 2.4+ KB
```

[169]:
```
racial.plot(kind='bar', stacked=True, figsize=(14,8))
plt.suptitle('Racial Makeup of Each US State')
plt.ylabel('%', fontsize=14)
plt.xlabel('US State', fontsize=14)

white_legend = mpatches.Patch(label='White', color='blue')
black_legend = mpatches.Patch(label='Black', color='orange')
native_american_legend = mpatches.Patch(label='Native American', color='green')
asian_legend = mpatches.Patch(label='Asian', color='red')
```

```
hispanic_legend = mpatches.Patch(label='Hispanic', color='purple')

plt.legend(handles=[white_legend, black_legend, native_american_legend, asian_legend, hispanic_legend], loc='best', frameon=False)

plt.xticks(fontsize=10, rotation=75)
plt.yticks(fontsize=10)

plt.show()
```




# 7 Create Donut Chart by of People Killed by Race

Hint: Use .value_counts()

[170]:
```
killed = df_fatalities.race.value_counts()
```

[171]:
```
killed
```

[171]:
```
W    1201
B     618
H     423
A      39
N      31
```

```
O       28
Name: race, dtype: int64
```

[172]:
```
df_fatalities.race.unique()
```

[172]:
```
array(['A', 'W', 'H', 'B', 'O', nan, 'N'], dtype=object)
```

[173]:
```
label_mapping = {
    'W': 'White',
    'B': 'Black',
    'H': 'Hispanic',
    'A': 'Asian',
    'N': 'Native American',
    'O': 'Other'
}

# Replace the index labels using the map function
killed.index = killed.index.map(label_mapping)
```

[174]:
```
import plotly.graph_objects as go

labels = killed.index
values = killed.values

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.6,␣
 ↪textinfo='label+percent')])

fig.update_layout(title='People Killed by Race')
fig.show()
```

# 8 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.

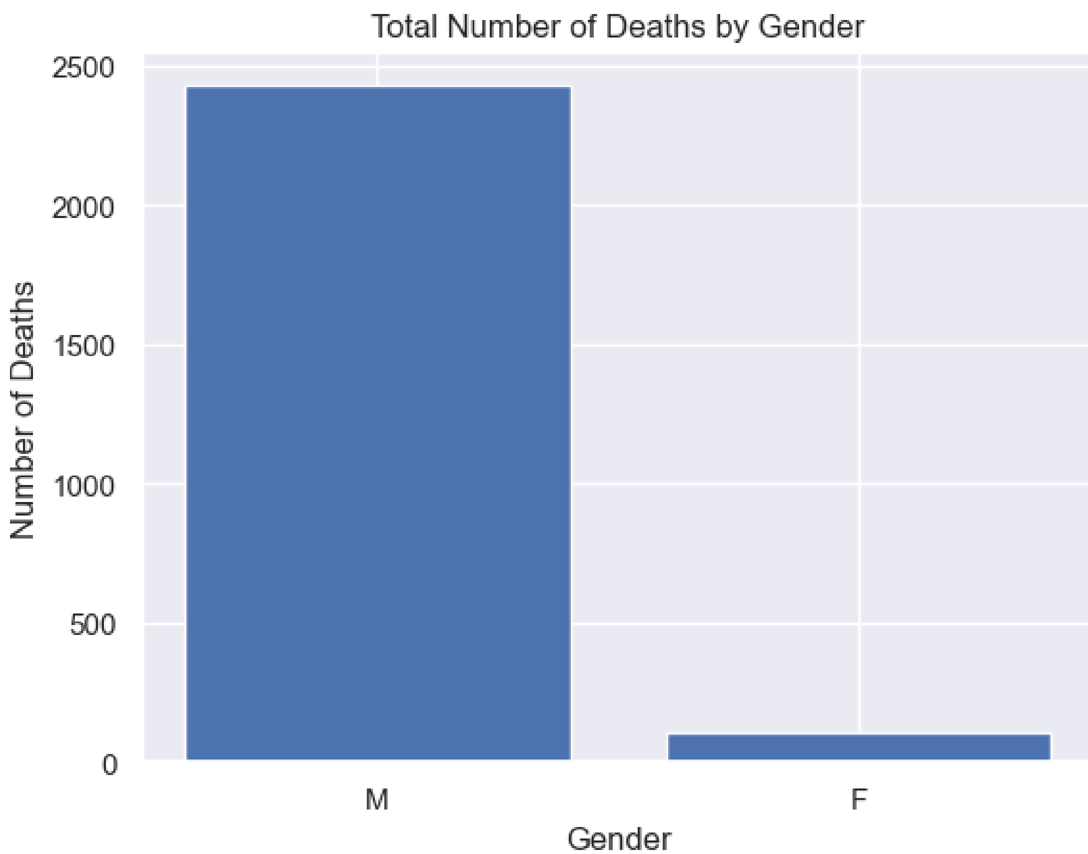
[175]:
```
killed_gender = df_fatalities.gender.value_counts()
killed_gender
```

[175]:
```
M    2428
F     107
Name: gender, dtype: int64
```

[176]:
```
gender = killed_gender.index
count = killed_gender.values
```

[177]:
```
plt.bar(gender, count)
```

```
plt.xlabel("Gender")
plt.ylabel("Number of Deaths")
plt.title("Total Number of Deaths by Gender")


plt.show()
```




## 9 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?

[178]:
```
df_fatalities.age = df_fatalities.age.replace(np.nan, 0)
```

[179]:
```
df_fatalities.age.isnull().any()
```

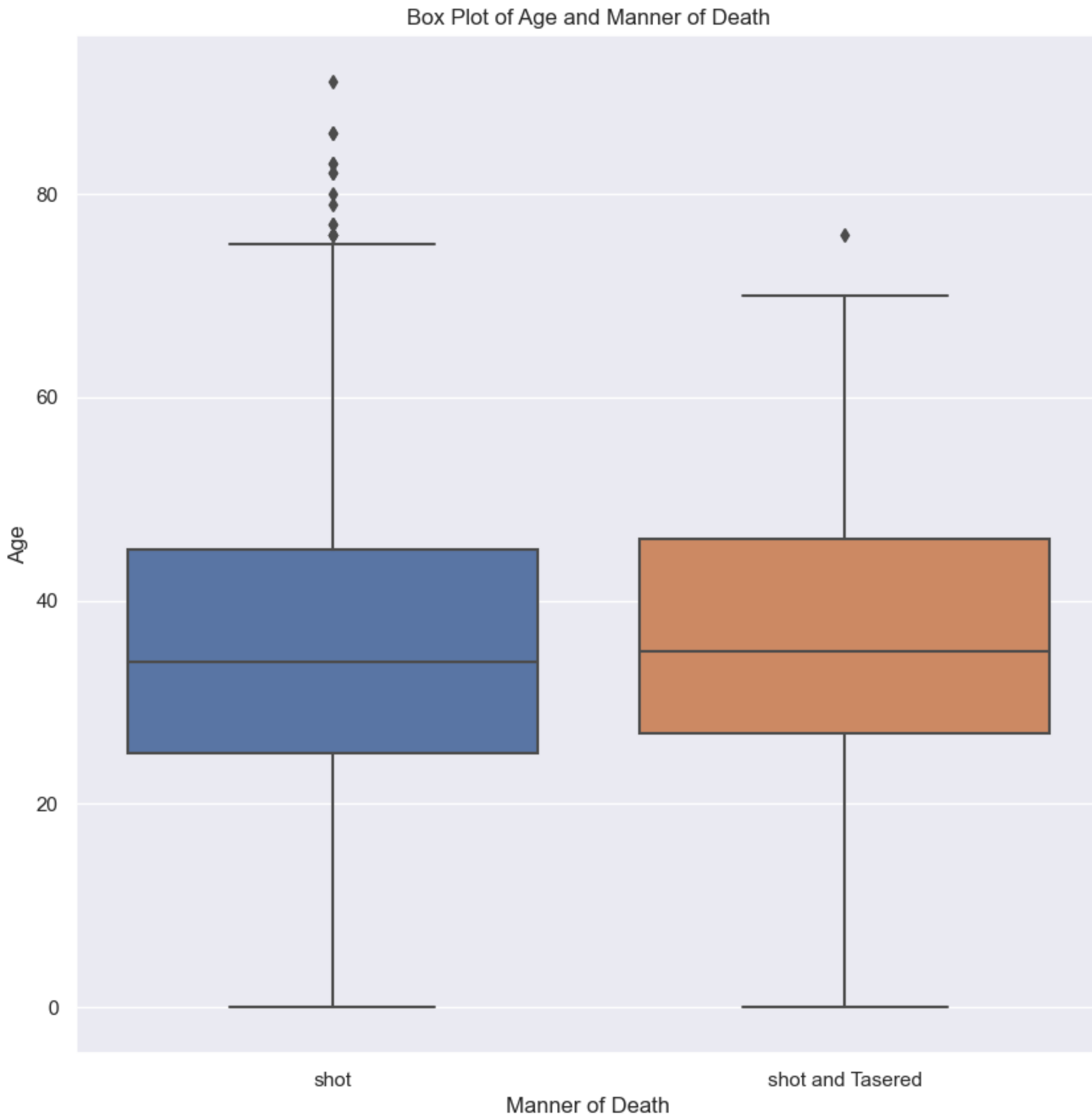
[179]: False

[180]:
```python
death_age = df_fatalities[['age', 'manner_of_death', 'gender']]
```

[181]:
```python
death_age
```

[181]:
```
        age   manner_of_death gender
0     53.00              shot      M
1     47.00              shot      M
2     23.00  shot and Tasered      M
3     32.00              shot      M
4     39.00              shot      M
…     …                   …    …
2530 31.00              shot      M
2531  0.00              shot      M
2532 48.00              shot      M
2533 28.00              shot      M
2534 32.00              shot      M

[2535 rows x 3 columns]
```

[182]:
```python
plt.figure(figsize=(10,10))
sns.boxplot(x="manner_of_death", y="age", data=death_age)

plt.xlabel("Manner of Death")
plt.ylabel("Age")
plt.title("Box Plot of Age and Manner of Death")

plt.show()
```




## 10 Were People Armed?

In what percentage of police killings were people armed? Create chart that show what kind of weapon (if any) the deceased was carrying. How many of the people killed by police were armed with guns versus unarmed?

[183]:
```
weapons = df_fatalities.armed.unique()
```

[184]:
```
weapons
```

[184]:
```
array(['gun', 'unarmed', 'toy weapon', 'nail gun', 'knife', 'vehicle',
       'shovel', 'hammer', 'hatchet', 'undetermined', 'sword', 'machete',
```

```
       'box cutter', 'metal object', 'screwdriver', 'lawn mower blade',
       'flagpole', 'guns and explosives', 'cordless drill', 'crossbow',
       'metal pole', 'Taser', 'metal pipe', 'metal hand tool',
       'blunt object', 'metal stick', 'sharp object', 'meat cleaver', nan,
       'carjack', 'chain', "contractor's level", 'unknown weapon',
       'stapler', 'beer bottle', 'bean-bag gun',
       'baseball bat and fireplace poker', 'straight edge razor',
       'gun and knife', 'ax', 'brick', 'baseball bat', 'hand torch',
       'chain saw', 'garden tool', 'scissors', 'pole', 'pick-axe',
       'flashlight', 'baton', 'spear', 'pitchfork', 'hatchet and gun',
       'rock', 'piece of wood', 'bayonet', 'pipe', 'glass shard',
       'motorcycle', 'metal rake', 'crowbar', 'oar', 'machete and gun',
       'tire iron', 'air conditioner', 'pole and knife',
       'baseball bat and bottle', 'fireworks', 'pen'], dtype=object)
```

[185]:
```
df_fatalities.armed.unique()
```

[185]:
```
array(['gun', 'unarmed', 'toy weapon', 'nail gun', 'knife', 'vehicle',
       'shovel', 'hammer', 'hatchet', 'undetermined', 'sword', 'machete',
       'box cutter', 'metal object', 'screwdriver', 'lawn mower blade',
       'flagpole', 'guns and explosives', 'cordless drill', 'crossbow',
       'metal pole', 'Taser', 'metal pipe', 'metal hand tool',
       'blunt object', 'metal stick', 'sharp object', 'meat cleaver', nan,
       'carjack', 'chain', "contractor's level", 'unknown weapon',
       'stapler', 'beer bottle', 'bean-bag gun',
       'baseball bat and fireplace poker', 'straight edge razor',
       'gun and knife', 'ax', 'brick', 'baseball bat', 'hand torch',
       'chain saw', 'garden tool', 'scissors', 'pole', 'pick-axe',
       'flashlight', 'baton', 'spear', 'pitchfork', 'hatchet and gun',
       'rock', 'piece of wood', 'bayonet', 'pipe', 'glass shard',
       'motorcycle', 'metal rake', 'crowbar', 'oar', 'machete and gun',
       'tire iron', 'air conditioner', 'pole and knife',
       'baseball bat and bottle', 'fireworks', 'pen'], dtype=object)
```

[186]:
```
# Rename the 'armed' column to 'weapon'
df_fatalities.rename(columns={'armed': 'weapon'}, inplace=True)

# Create the 'armed' column
df_fatalities['armed'] = df_fatalities['weapon'] != 'unarmed'

# Print the updated dataframe
print(df_fatalities)
```

```
        id                name      date    manner_of_death      weapon   age  \
0        3          Tim Elliot  02/01/15               shot         gun 53.00
1        4     Lewis Lee Lembke  02/01/15               shot         gun 47.00
2        5   John Paul Quintero  03/01/15   shot and Tasered     unarmed 23.00
3        8      Matthew Hoffman  04/01/15               shot  toy weapon 32.00
```

```
4        9   Michael Rodriguez  04/01/15             shot    nail gun 39.00
…     …                  …       …               …          …    …
2530  2822     Rodney E. Jacobs  28/07/17             shot         gun 31.00
2531  2813                TK TK  28/07/17             shot     vehicle  0.00
2532  2818   Dennis W. Robinson  29/07/17             shot         gun 48.00
2533  2817       Isaiah Tucker  31/07/17             shot     vehicle 28.00
2534  2815         Dwayne Jeune  31/07/17             shot       knife 32.00

     gender race           city state  signs_of_mental_illness threat_level  \
0         M    A        Shelton    WA                     True       attack
1         M    W          Aloha    OR                    False       attack
2         M    H        Wichita    KS                    False        other
3         M    W  San Francisco    CA                     True       attack
4         M    H          Evans    CO                    False       attack
…       …   …            …   …                         …          …
2530      M  NaN    Kansas City    MO                    False       attack
2531      M  NaN    Albuquerque    NM                    False       attack
2532      M  NaN          Melba    ID                    False       attack
2533      M    B        Oshkosh    WI                    False       attack
2534      M    B       Brooklyn    NY                     True       attack

             flee  body_camera  armed
0     Not fleeing        False   True
1     Not fleeing        False   True
2     Not fleeing        False  False
3     Not fleeing        False   True
4     Not fleeing        False   True
…             …          …     …
2530  Not fleeing        False   True
2531          Car        False   True
2532          Car        False   True
2533          Car         True   True
2534  Not fleeing        False   True

[2535 rows x 15 columns]
```

[187]:
```python
armed_pctg = len(df_fatalities[df_fatalities.armed == True]) /
↪len(df_fatalities) * 100
```

[188]:
```python
armed_pctg=round(armed_pctg,2)
```

[189]:
```python
print(f'In police killings, {armed_pctg}% of the victims were armed.')
```

```
In police killings, 93.25% of the victims were armed.
```

[190]:
```python
df_fatalities['armed'] = df_fatalities['armed'].map({True: 'armed', False:
↪'unarmed'})
```

[191]:
```python
df_fatalities.armed
```

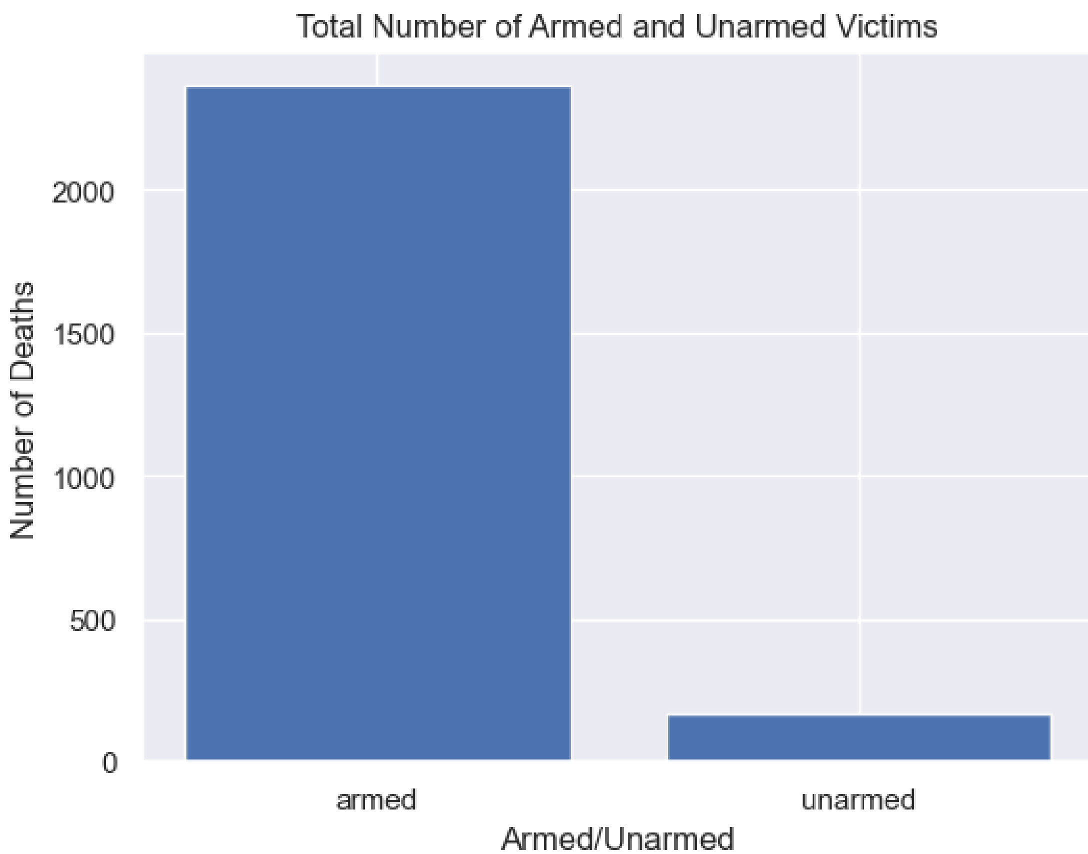
[191]:
```
0         armed
1         armed
2       unarmed
3         armed
4         armed
         …
2530      armed
2531      armed
2532      armed
2533      armed
2534      armed
Name: armed, Length: 2535, dtype: object
```

[192]:
```python
percentage = df_fatalities.armed.value_counts()
```

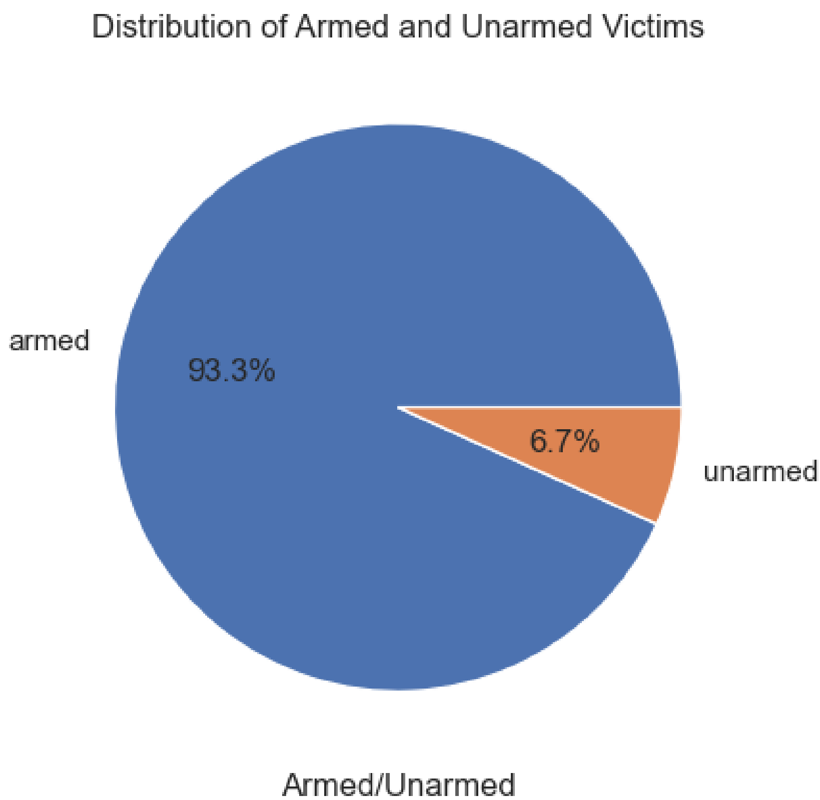
[193]:
```python
percentage
```

[193]:
```
armed      2364
unarmed     171
Name: armed, dtype: int64
```

[194]:
```python
plt.bar(percentage.index, percentage.values)


plt.xlabel("Armed/Unarmed")
plt.ylabel("Number of Deaths")
plt.title("Total Number of Armed and Unarmed Victims")


plt.show()
```




[195]:
```
percentage = df_fatalities['armed'].value_counts() * 100

plt.pie(percentage.values, labels=percentage.index, autopct='%1.1f%%')

plt.title("Distribution of Armed and Unarmed Victims")
plt.xlabel("Armed/Unarmed")

plt.show()
```

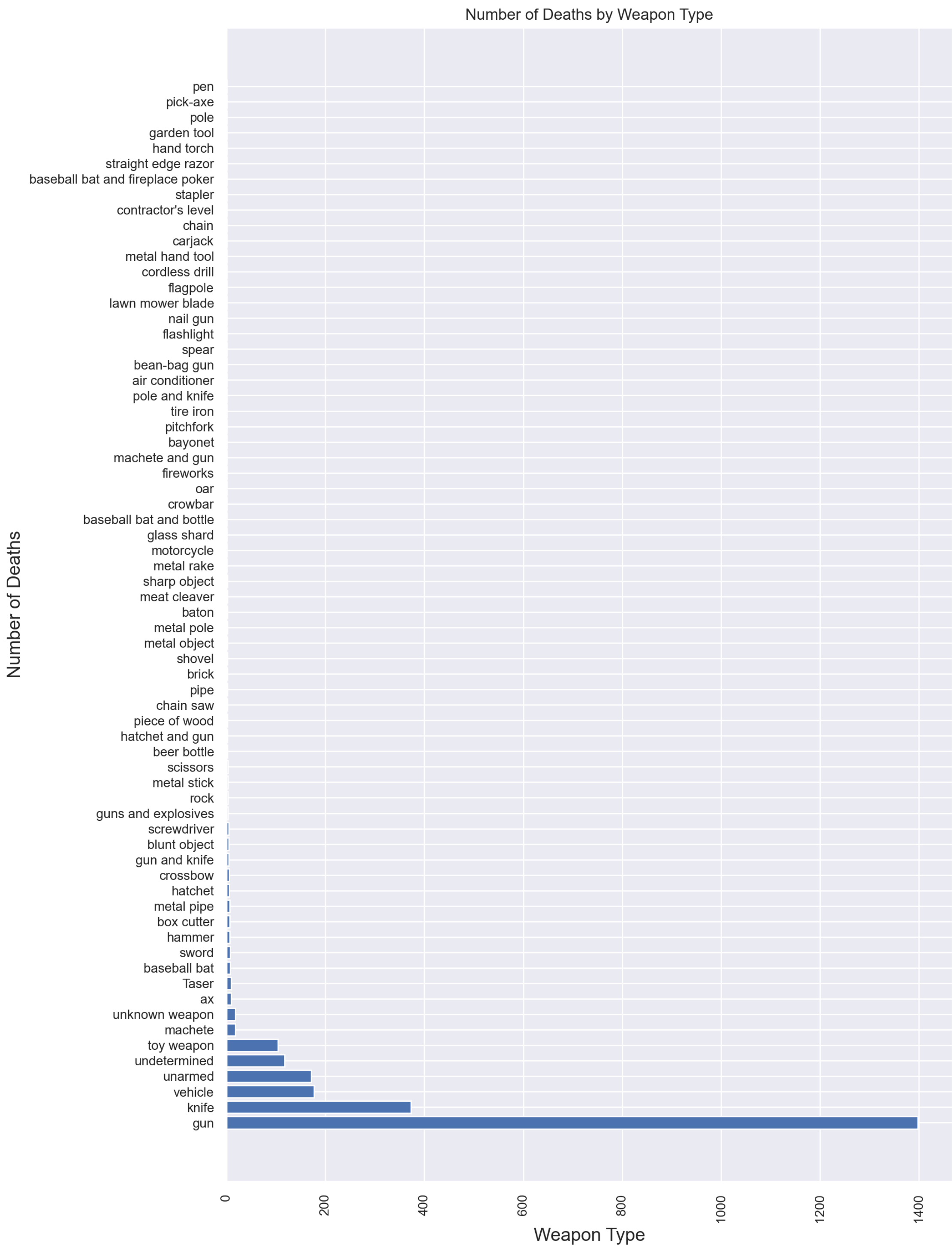



[196]:
```
weapon_counts = df_fatalities['weapon'].value_counts()
weapon_counts
```

[196]:
```
gun             1398
knife            373
vehicle          177
unarmed          171
undetermined     117
                ...
hand torch         1
garden tool        1
pole               1
pick-axe           1
pen                1
Name: weapon, Length: 68, dtype: int64
```

[197]:
```
plt.figure(figsize=(10,16), dpi = 200)
plt.barh(weapon_counts.index, weapon_counts.values)

plt.xlabel("Weapon Type", fontsize = 14)
plt.ylabel("Number of Deaths", fontsize = 14)
```

```
plt.title("Number of Deaths by Weapon Type")

plt.xticks(fontsize=10, rotation=90)
plt.yticks(fontsize=10)

plt.show()
```



## 11 How Old Were the People Killed?

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

[198]:
```
u25 = df_fatalities[df_fatalities.age < 25]
```

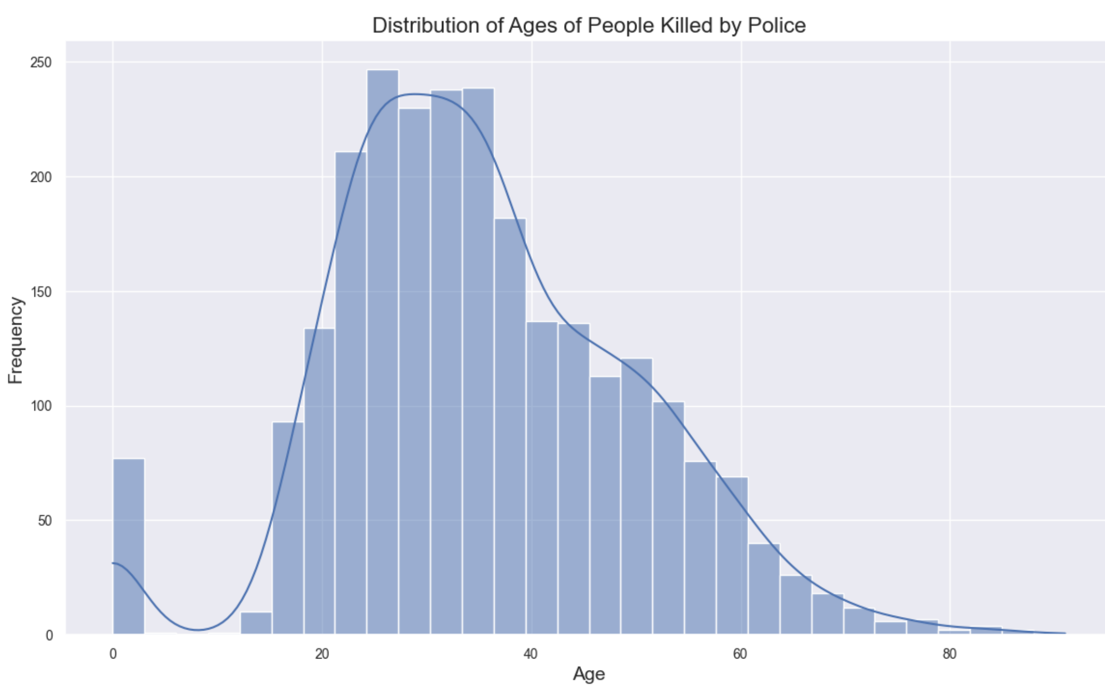
[199]:
```
pct_u25 = round(len(u25) / len(df_fatalities) * 100, 2)
```

[200]:
```
print(f'{pct_u25}% of the people killed by the police, were under 25 years old.
↪')
```

```
20.79% of the people killed by the police, were under 25 years old.
```

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

[201]:
```
plt.figure(figsize = (14,8))
sns.histplot(data=df_fatalities, x='age', bins=30, kde=True)

plt.xlabel("Age", fontsize=14)
plt.ylabel("Frequency", fontsize=14)
plt.title("Distribution of Ages of People Killed by Police", fontsize=16)

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

plt.show()
```



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

[202]:
```
plt.figure(figsize = (14,8))
sns.histplot(data=df_fatalities, x='age', bins=50, kde=True, hue='race',␣
 ↪element='step')

plt.xlabel("Age", fontsize=14)
plt.ylabel("Frequency", fontsize=14)
plt.title("Distribution of Ages of People Killed by Police per Race",␣
 ↪fontsize=16)

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

plt.show()
```

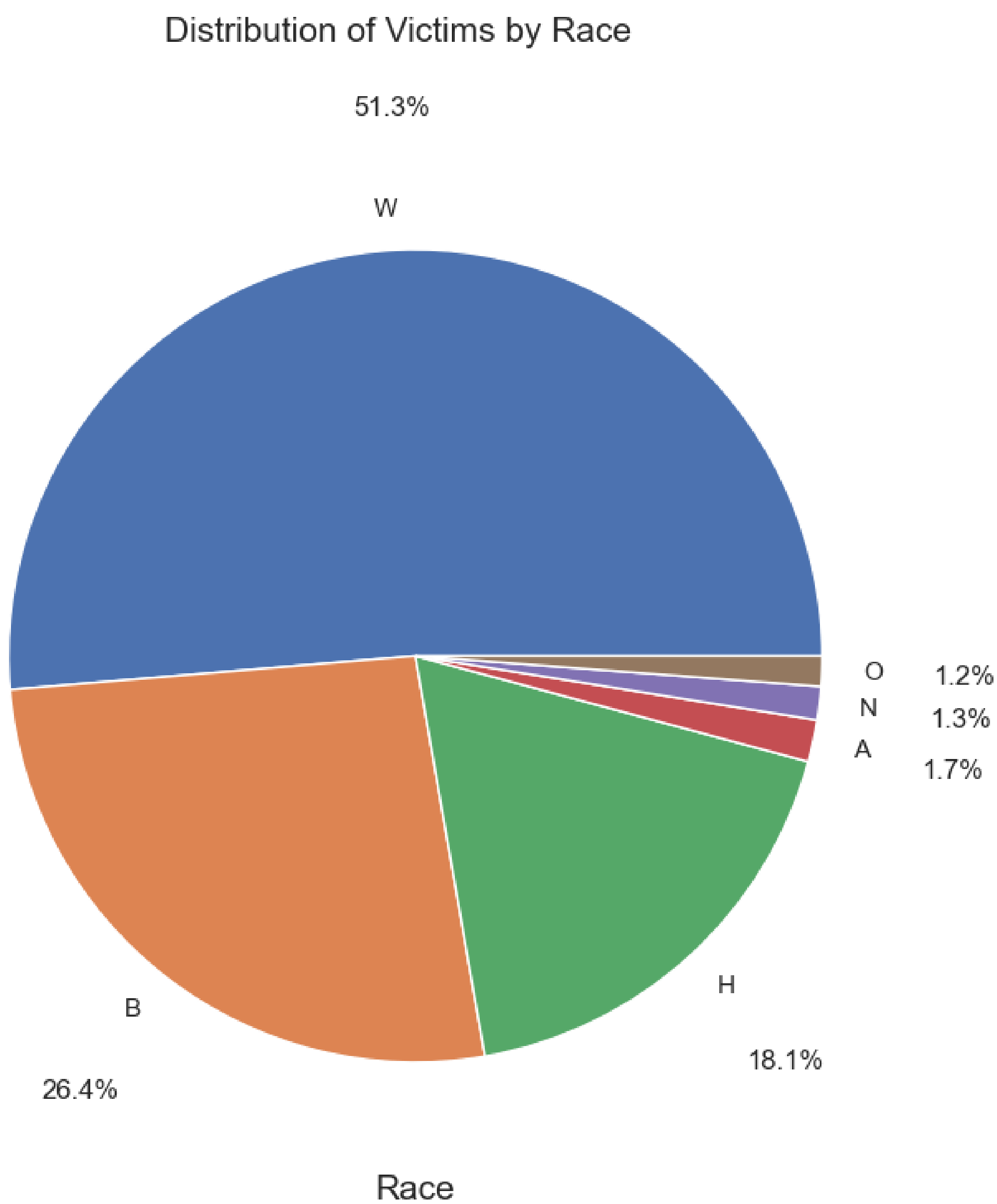

## 12 Race of People Killed

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

[203]:
```
race_kill = df_fatalities.race.value_counts()
race_kill
```

[203]:
```
W    1201
B     618
H     423
A      39
N      31
O      28
Name: race, dtype: int64
```

[204]:
```
plt.figure(figsize = (8,8))
plt.pie(race_kill.values, labels=race_kill.index, autopct='%1.
 ↪1f%%',pctdistance=1.35, labeldistance=1.1)

plt.suptitle("Distribution of Victims by Race", fontsize = 15)
plt.xlabel("Race", fontsize = 15)

plt.show()
```




## 13 Mental Illness and Police Killings

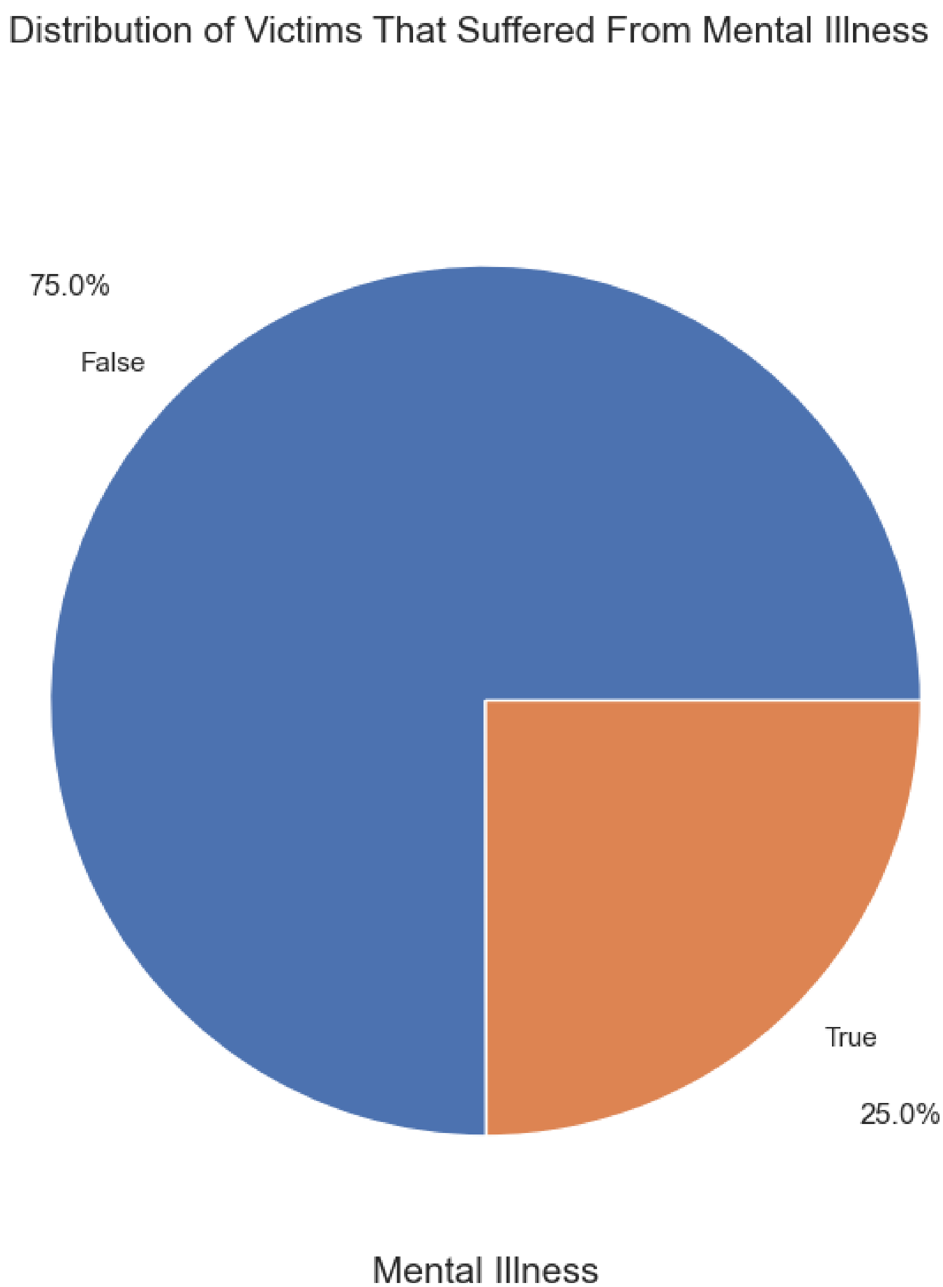
What percentage of people killed by police have been diagnosed with a mental illness?

[205]:
```
mental = df_fatalities.signs_of_mental_illness.value_counts()
mental
```

[205]:
```
False    1902
True      633
Name: signs_of_mental_illness, dtype: int64
```

[206]:

```
plt.figure(figsize = (8,8))
plt.pie(mental.values, labels=mental.index, autopct='%1.1f%%',pctdistance=1.35,␣
 ↪labeldistance=1.1)

plt.suptitle("Distribution of Victims That Suffered From Mental Illness",␣
 ↪fontsize = 15)
plt.xlabel("Mental Illness", fontsize = 15)

plt.show()
```



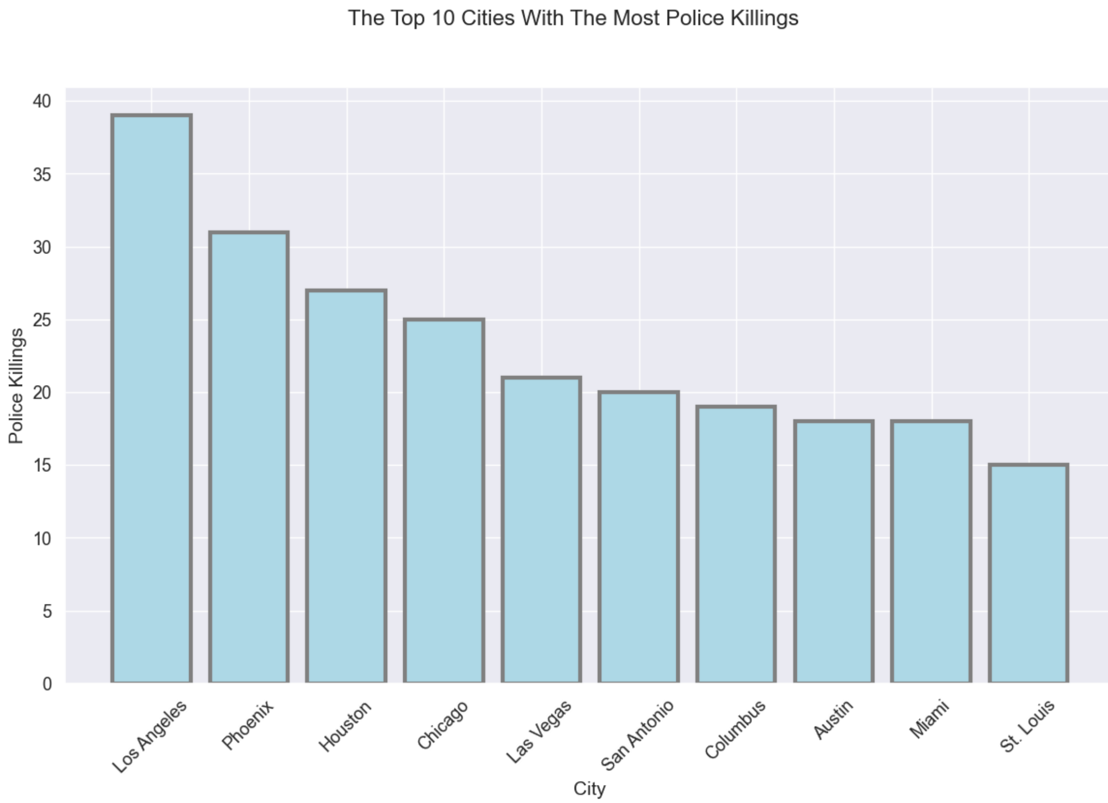
## 14 In Which Cities Do the Most Police Killings Take Place?

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

[207]:
```
cities = df_fatalities.city.value_counts().head(10)
cities
```

[207]:
```
Los Angeles    39
Phoenix        31
Houston        27
Chicago        25
Las Vegas      21
San Antonio    20
Columbus       19
Austin         18
Miami          18
St. Louis      15
Name: city, dtype: int64
```

[208]:
```
plt.figure(figsize=(14,8))
plt.suptitle('The Top 10 Cities With The Most Police Killings', fontsize=16)
plt.ylabel('Police Killings', fontsize=14)
plt.xlabel('City', fontsize=14)
plt.xticks(fontsize=13, rotation=45)
plt.yticks(fontsize=13)
plt.bar(cities.index, cities, label=cities.index, linewidth=3, color =␣
  ↪'lightblue', edgecolor = 'gray')
plt.show()
```




## 15 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.

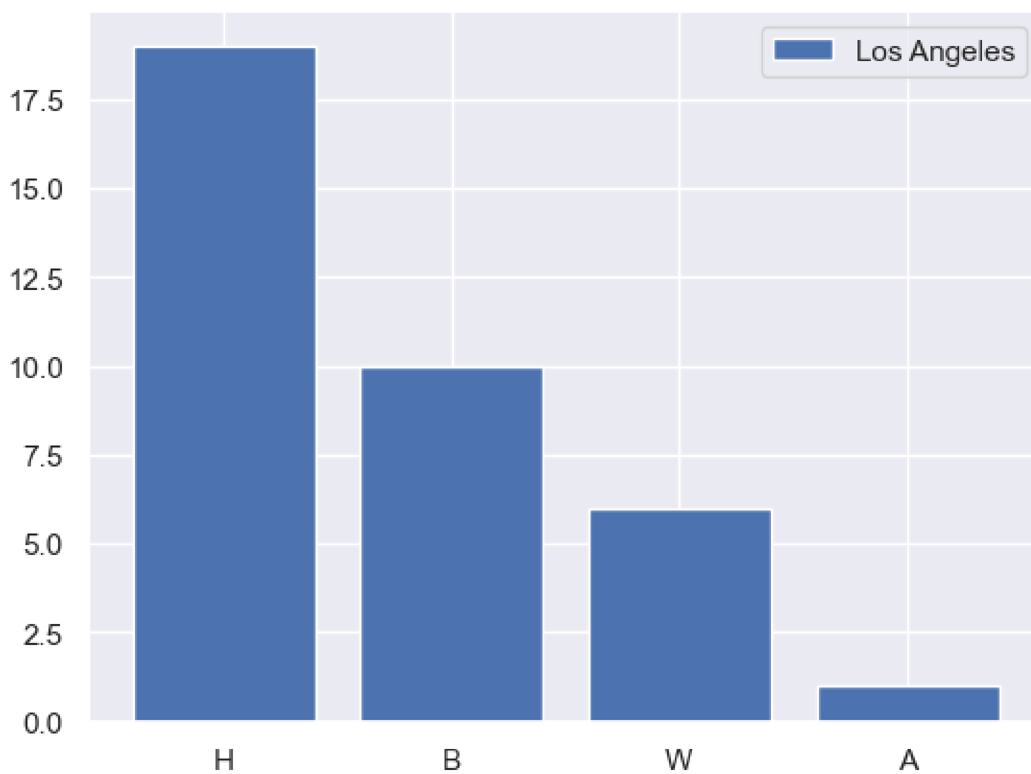
[209]:
```
df_fatalities[['city', 'race']]
```

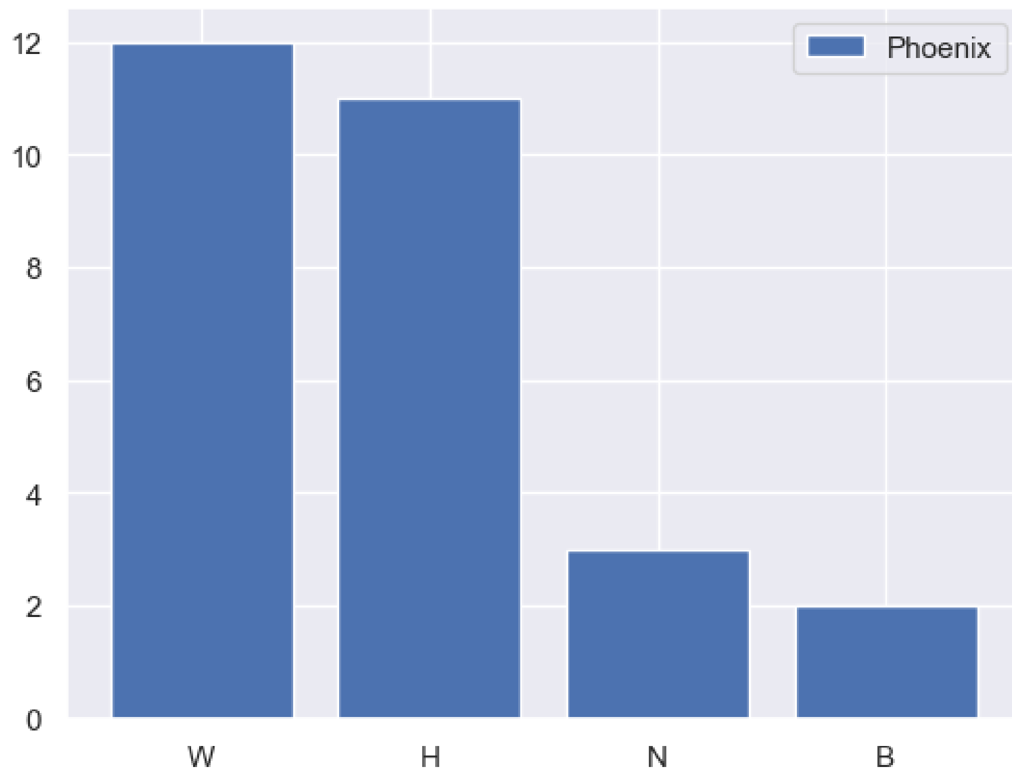
[209]:
```
                 city race
0             Shelton    A
1               Aloha    W
2             Wichita    H
3       San Francisco    W
4               Evans    H
…                 …  …
2530      Kansas City  NaN
2531      Albuquerque  NaN
2532            Melba  NaN
2533          Oshkosh    B
2534         Brooklyn    B

[2535 rows x 2 columns]
```

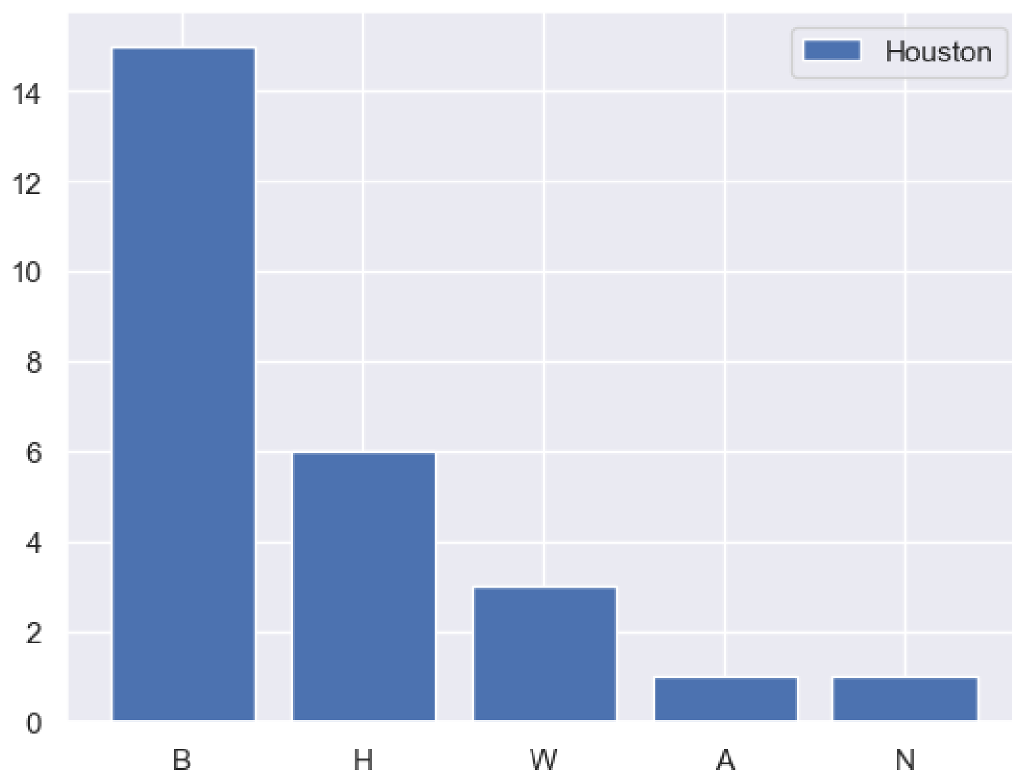
[210]:
```
cities
```

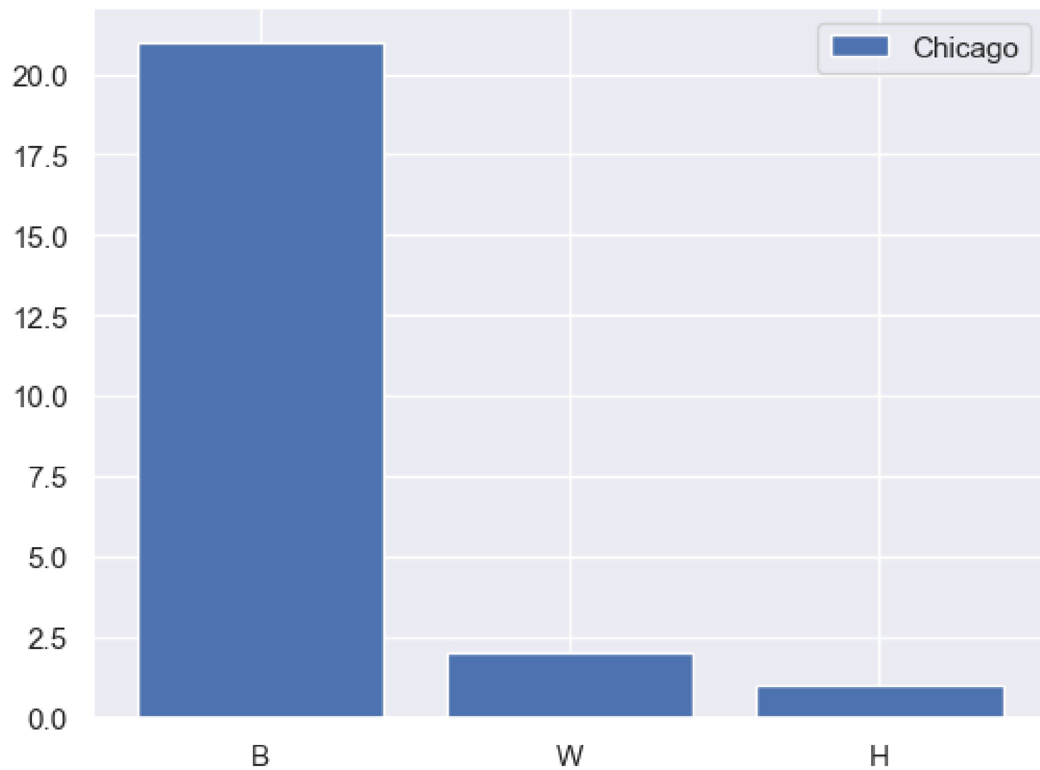
[210]:
```
Los Angeles    39
Phoenix        31
Houston        27
Chicago        25
Las Vegas      21
San Antonio    20
Columbus       19
Austin         18
Miami          18
St. Louis      15
Name: city, dtype: int64
```

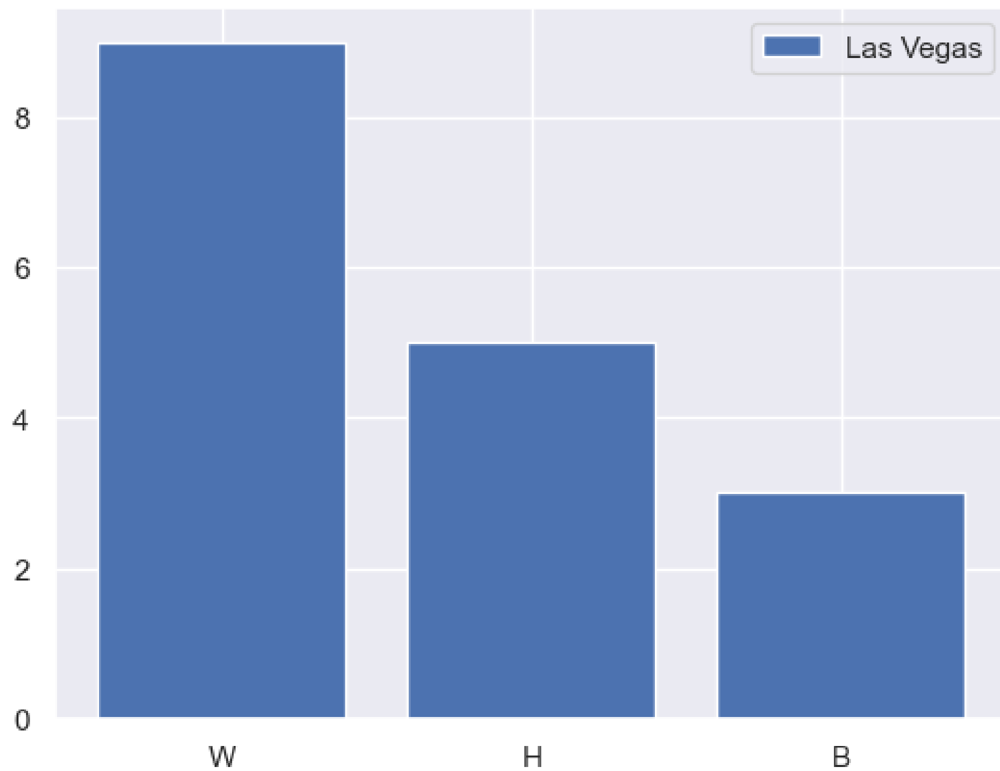
[211]:
```
top = df_fatalities[['city', 'race']]
for c in cities.index:
    top_cities = top.loc[top['city'] == c]
    city = top_cities.race.value_counts()
    plt.bar(city.index, city, label=c)
    plt.legend(loc='best')
    plt.show()
```

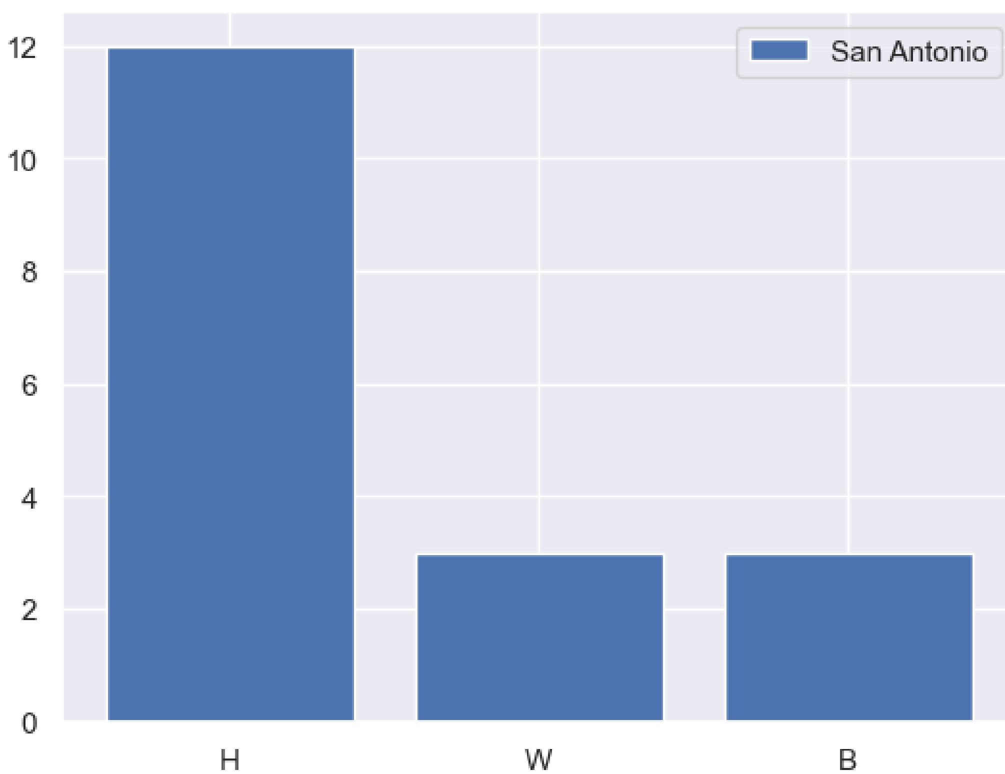


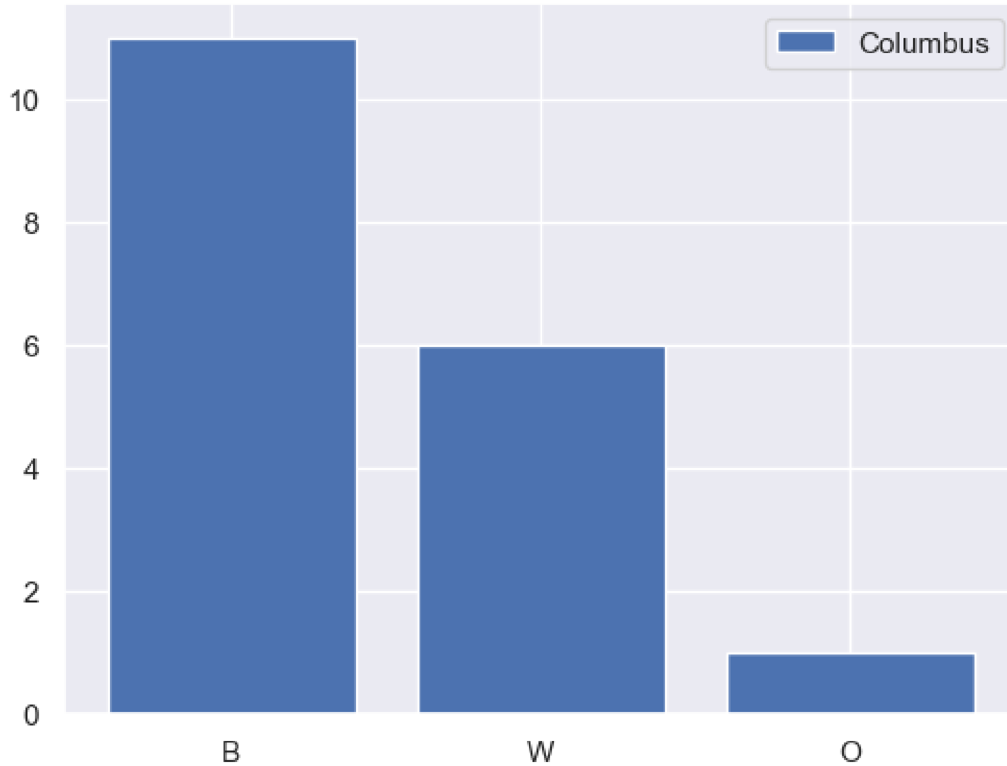

Phoenix
12
10
8
6
4
2
0
W
H
N
B

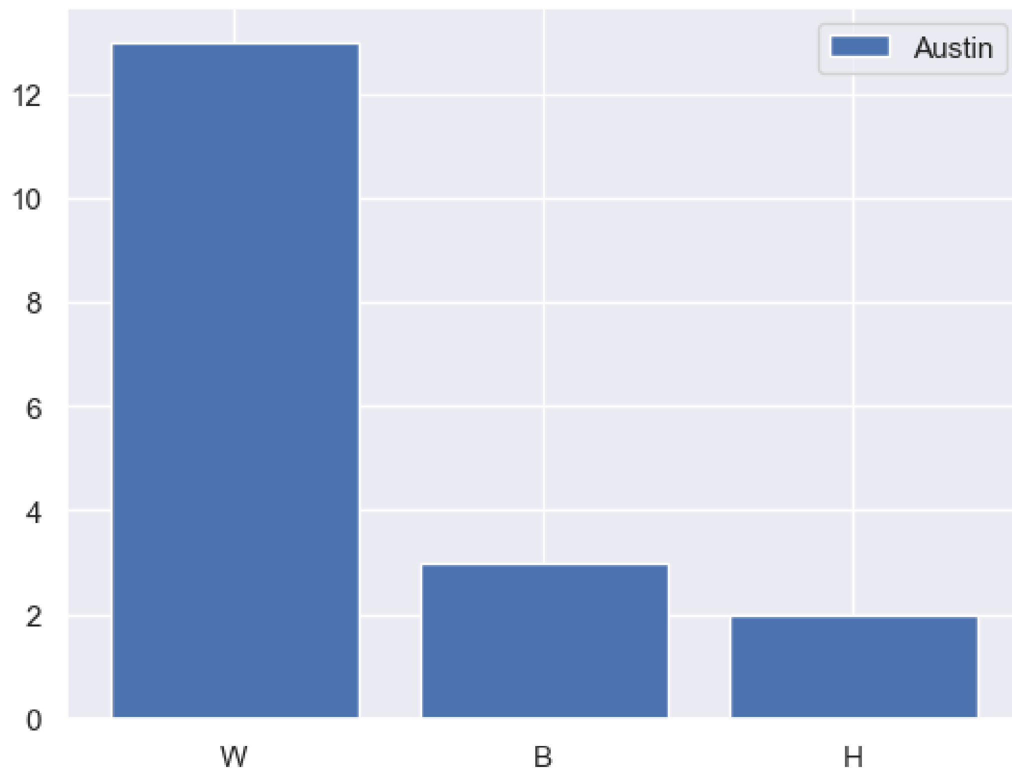

Houston
14
12
10
8
6
4
2
0
B
H
W
A
N

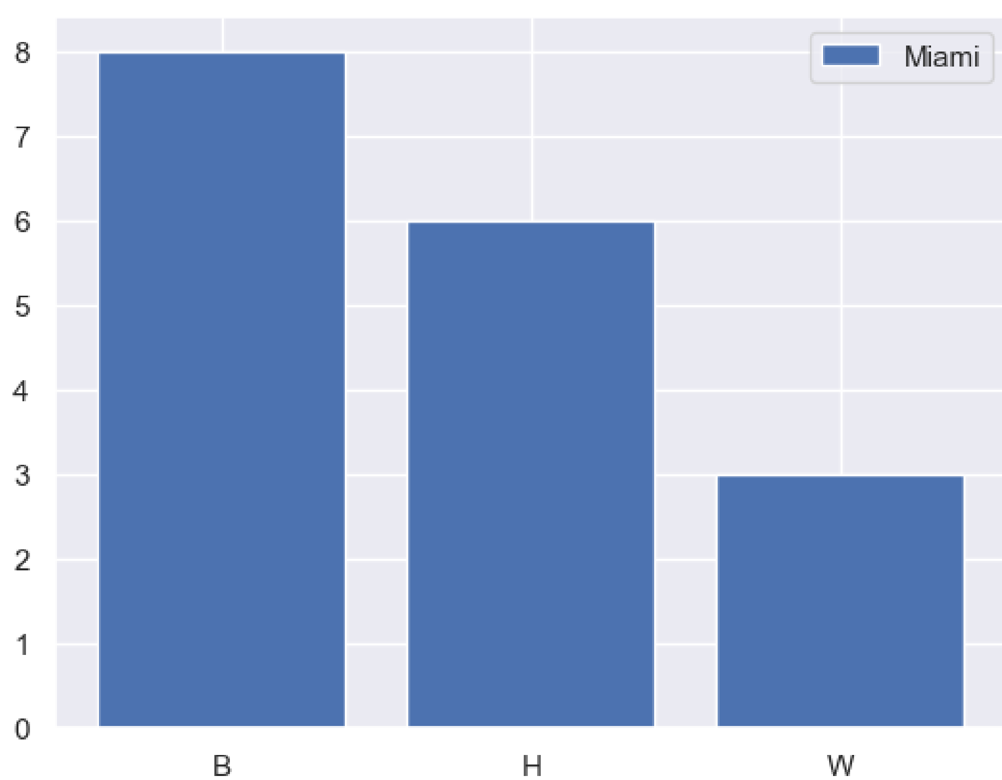

Chicago
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
B
W
H

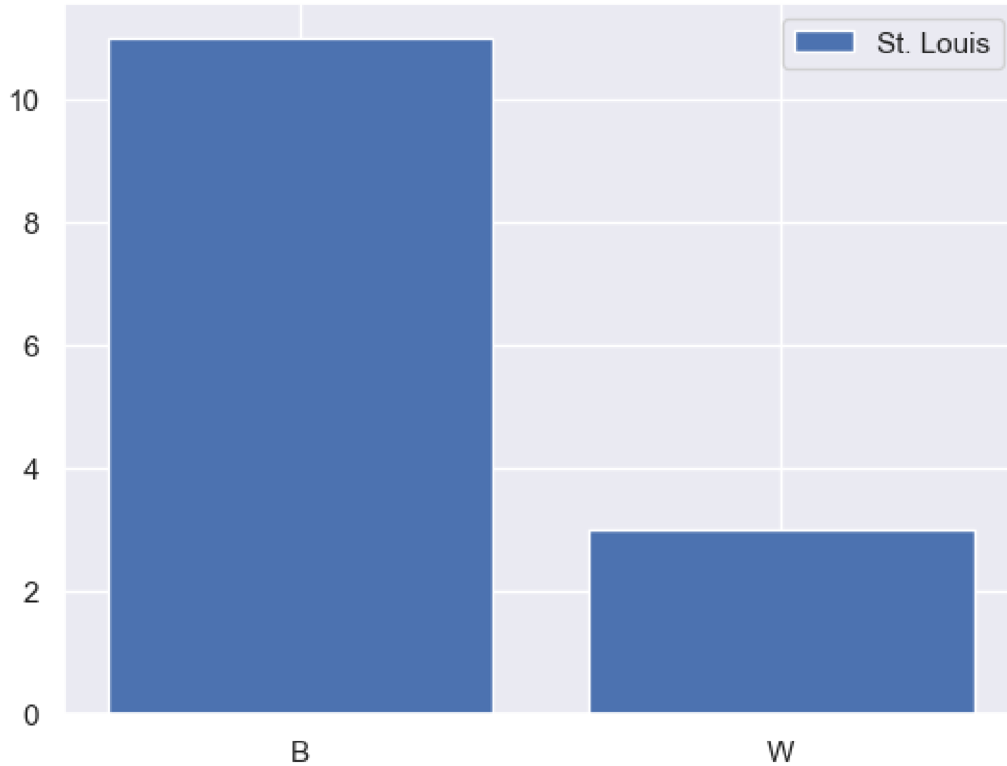

Las Vegas
0
2
4
6
8
W
H
B


San Antonio
0
2
4
6
8
10
12
H
W
B


Columbus
10
8
6
4
2
0
B
W
O


Austin
12
10
8
6
4
2
0
W
B
H


Miami
8
7
6
5
4
3
2
1
0
B
H
W




# 16 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?

[212]:
```
states_to_map = df_fatalities.groupby('state').size().
  ↪sort_values(ascending=False)
states_to_map
```

[212]:
```
state
CA    424
TX    225
FL    154
AZ    118
OH     79
OK     78
CO     74
GA     70
NC     69
```

```
MO      64
IL      62
WA      62
TN      59
LA      57
NM      51
PA      51
AL      50
VA      47
NY      45
SC      44
KY      43
WI      43
IN      43
NV      42
OR      38
MD      38
MI      37
NJ      35
MN      32
WV      27
AR      26
KS      24
MS      23
UT      23
MA      22
ID      17
AK      15
NE      15
ME      13
IA      12
MT      11
HI      11
DC      11
SD      10
CT       9
DE       8
WY       8
NH       7
ND       4
VT       3
RI       2
dtype: int64
```

[214]:

```
fig = go.Figure(data=go.Choropleth(locations=states_to_map.index, z =␣
↪states_to_map, locationmode = 'Europe',
                                    colorscale = 'Oranges', colorbar_title =␣
↪"Police Killings",))

fig.update_layout(title_text = 'Police Killings by US State', geo_scope='usa')

fig.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[214], line 1
----> 1 fig = go.
↪Figure(data=go.Choropleth(locations=states_to_map.index, z = states_to_map, locationmode =
      2␣
↪                                    colorscale = 'Oranges', colorbar_title = "Police Killin
      4 fig.update_layout(title_text = 'Police Killings by US State',␣
↪geo_scope='usa')
      6 fig.show()

File␣
↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\plotly\graph_objs\_choropleth.
↪py:2271, in Choropleth.__init__(self, arg, autocolorscale, coloraxis,␣
↪colorbar, colorscale, customdata, customdatasrc, featureidkey, geo, geojson,␣
↪hoverinfo, hoverinfosrc, hoverlabel, hovertemplate, hovertemplatesrc,␣
↪hovertext, hovertextsrc, ids, idssrc, legend, legendgroup, legendgrouptitle,␣
↪legendrank, legendwidth, locationmode, locations, locationssrc, marker, meta,␣
↪metasrc, name, reversescale, selected, selectedpoints, showlegend, showscale,␣
↪stream, text, textsrc, uid, uirevision, unselected, visible, z, zauto, zmax,␣
↪zmid, zmin, zsrc, **kwargs)
   2269 _v = locationmode if locationmode is not None else _v
   2270 if _v is not None:
-> 2271     self["locationmode"] = _v
   2272 _v = arg.pop("locations", None)
   2273 _v = locations if locations is not None else _v

File␣
↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\plotly\basedatatypes.
↪py:4873, in BasePlotlyType.__setitem__(self, prop, value)
   4869         self._set_array_prop(prop, value)
   4871     # ### Handle simple property ###
   4872     else:
-> 4873         self._set_prop(prop, value)
   4874 else:
   4875     # Make sure properties dict is initialized
   4876     self._init_props()

File␣
↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\plotly\basedatatypes.
↪py:5217, in BasePlotlyType._set_prop(self, prop, val)
```

```
   5215         return
   5216     else:
-> 5217         raise err
   5219 # val is None
   5220 # -----------
   5221 if val is None:
   5222     # Check if we should send null update

File␣
↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\plotly\basedatatypes.
↪py:5212, in BasePlotlyType._set_prop(self, prop, val)
   5209 validator = self._get_validator(prop)
   5211 try:
-> 5212     val = validator.validate_coerce(val)
   5213 except ValueError as err:
   5214     if self._skip_invalid:

File␣
↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\_plotly_utils\basevalidators.
↪py:610, in EnumeratedValidator.validate_coerce(self, v)
    608     v = self.perform_replacemenet(v)
    609     if not self.in_values(v):
--> 610         self.raise_invalid_val(v)
    611 return v

File␣
↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\_plotly_utils\basevalidators.
↪py:287, in BaseValidator.raise_invalid_val(self, v, inds)
    284             for i in inds:
    285                 name += "[" + str(i) + "]"
--> 287         raise ValueError(
    288             """
    289     Invalid value of type {typ} received for the '{name}' property of␣
↪{pname}
    290         Received value: {v}
    291 
    292 {valid_clr_desc}""".format(
    293                 name=name,
    294                 pname=self.parent_name,
    295                 typ=type_str(v),
    296                 v=repr(v),
    297                 valid_clr_desc=self.description(),
    298             )
    299         )

ValueError:
    Invalid value of type 'builtins.str' received for the 'locationmode'␣
↪property of choropleth
```

```
        Received value: 'Europe'

    The 'locationmode' property is an enumeration that may be specified as:
      - One of the following enumeration values:
            ['ISO-3', 'USA-states', 'country names', 'geojson-id']
```

## 17 Number of Police Killings Over Time

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

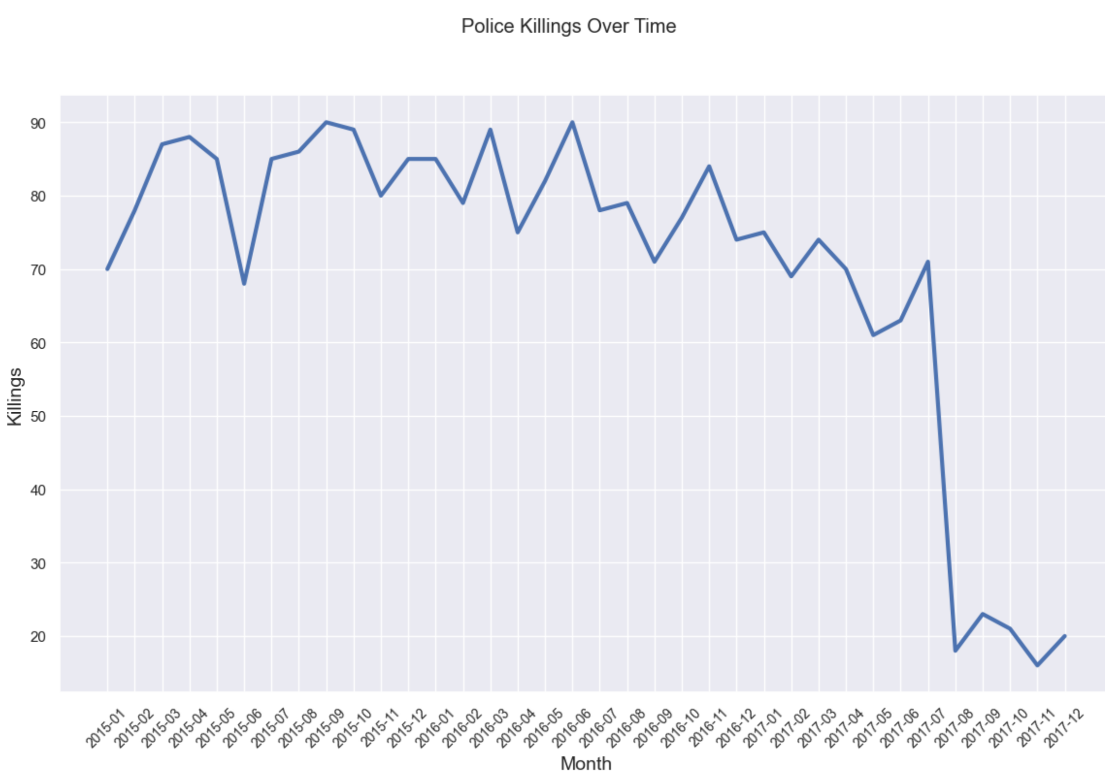
[409]:
```
monthly_fatalities = df_fatalities.copy()
monthly_fatalities.date = pd.to_datetime(monthly_fatalities.date,␣
 ↪infer_datetime_format=True).dt.to_period('m')
monthly_fatalities.date = monthly_fatalities.date.astype(str)
```

[410]:
```
history = monthly_fatalities.groupby('date').size()
history
```

[410]:
```
date
2015-01    70
2015-02    78
2015-03    87
2015-04    88
2015-05    85
2015-06    68
2015-07    85
2015-08    86
2015-09    90
2015-10    89
2015-11    80
2015-12    85
2016-01    85
2016-02    79
2016-03    89
2016-04    75
2016-05    82
2016-06    90
2016-07    78
2016-08    79
2016-09    71
2016-10    77
2016-11    84
2016-12    74
2017-01    75
2017-02    69
2017-03    74
```

```
2017-04    70
2017-05    61
2017-06    63
2017-07    71
2017-08    18
2017-09    23
2017-10    21
2017-11    16
2017-12    20
dtype: int64
```

[413]:
```python
plt.figure(figsize=(14,8))
plt.plot(history.index, history, linewidth=3)
plt.xticks(ticks=history.index, fontsize=10, rotation=45)
plt.suptitle('Police Killings Over Time')
plt.xlabel('Month', fontsize=14)
plt.ylabel('Killings', fontsize=14)

plt.show()
```



## 18 Epilogue

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