In [87]:
```
import pandas as pd
import numpy as np
import warnings
import seaborn as sns
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
data = pd.read_csv('https://raw.githubusercontent.com/fivethirtyeight/guns-da
print(data.shape)
data.head()
```

```
(100798, 10)
```

Out[87]:

	year	month	intent	police	sex	age	race	hispanic	place	education
1	2012	1	Suicide	0	M	34.0	Asian/Pacific Islander	100	Home	BA+
2	2012	1	Suicide	0	F	21.0	White	100	Street	Some college
3	2012	1	Suicide	0	M	60.0	White	100	Other specified	BA+
4	2012	2	Suicide	0	M	64.0	White	100	Home	BA+
5	2012	2	Suicide	0	M	31.0	White	100	Other specified	HS/GED

In [88]:
```
data.index.name = 'Index'
data.head()
```

Out[88]:

Index	year	month	intent	police	sex	age	race	hispanic	place	education
1	2012	1	Suicide	0	M	34.0	Asian/Pacific Islander	100	Home	BA+
2	2012	1	Suicide	0	F	21.0	White	100	Street	Some college
3	2012	1	Suicide	0	M	60.0	White	100	Other specified	BA+
4	2012	2	Suicide	0	M	64.0	White	100	Home	BA+
5	2012	2	Suicide	0	M	31.0	White	100	Other specified	HS/GED

In [89]:
```
# calling out column names
data.columns = map(str.capitalize, data.columns)
data.columns
```

Out[89]:
```
Index(['Year', 'Month', 'Intent', 'Police', 'Sex', 'Age', 'Race', 'Hispanic',
       'Place', 'Education'],
      dtype='object')
```

In [90]:
```
# calling out the column datatypes
data.dtypes
```

Out[90]:
```
Year           int64
Month          int64
Intent        object
Police         int64
Sex           object
Age          float64
Race          object
Hispanic       int64
Place         object
Education     object
dtype: object
```

In [91]:
```
# checking NaN data to check if the columns are clean
data.notnull().sum()
```

Out[91]:
```
Year         100798
Month        100798
Intent       100797
Police       100798
Sex          100798
Age          100780
Race         100798
Hispanic     100798
Place         99414
Education     99376
dtype: int64
```

In [92]:
```
# Checking to see the percentage of valid data:
data.notnull().sum() * 100.0/data.shape[0]
```

Out[92]:
```
Year          100.000000
Month         100.000000
Intent         99.999008
Police        100.000000
Sex           100.000000
Age            99.982143
Race          100.000000
Hispanic      100.000000
Place          98.626957
Education      98.589258
dtype: float64
```

In [93]:
```
# Organizing the data by column value: first by the year, then by month:
data.sort_values(['Year', 'Month'], inplace = True)
data.head(10)
```

Out[93]:

Index	Year	Month	Intent	Police	Sex	Age	Race	Hispanic	Place	Educa
1	2012	1	Suicide	0	M	34.0	Asian/Pacific Islander	100	Home	
2	2012	1	Suicide	0	F	21.0	White	100	Street	S co
3	2012	1	Suicide	0	M	60.0	White	100	Other specified	
12	2012	1	Suicide	0	M	21.0	Native American/Native Alaskan	100	Home	HS/
135	2012	1	Suicide	0	F	59.0	White	100	Home	HS/
136	2012	1	Suicide	0	F	30.0	White	100	Other unspecified	
137	2012	1	Homicide	0	M	58.0	Black	100	Home	Less
138	2012	1	Suicide	0	M	78.0	White	100	Home	
139	2012	1	Suicide	0	M	60.0	White	100	Other unspecified	Less
140	2012	1	Accidental	0	M	61.0	White	100	Home	HS/

In [94]:
```
# Choosing the column we need work on
data.Intent.value_counts(ascending =  False)
```

Out[94]:
```
Suicide         63175
Homicide        35176
Accidental       1639
Undetermined      807
Name: Intent, dtype: int64
```

In [95]:
```
# Looking at the normalized values makes the picture clearer.
# Note: 'normalize=False' excludes the 'NaN's where here it includes them

data.Intent.value_counts(ascending=False, dropna=False, normalize=True)
```

Out[95]:
```
Suicide         0.626749
Homicide        0.348975
Accidental      0.016260
Undetermined    0.008006
NaN             0.000010
Name: Intent, dtype: float64
```

In [96]:
```
columns = ['Education', 'Age']
for column in columns:
    print(column + ':')
    print(data[column].describe())
    print('-' * 20 + '\n')
```

```
Education:
count      99376
unique         4
top       HS/GED
freq       42927
Name: Education, dtype: object
--------------------

Age:
count    100780.000000
mean         43.857601
std          19.496181
min           0.000000
25%          27.000000
50%          42.000000
75%          58.000000
max         107.000000
Name: Age, dtype: float64
--------------------
```

In [97]:
```
# Calculating the percentage

percentiles = np.arange(0.1,1.1,0.1)
for column in columns:
    print(column + ':')
    print(data[column][data[column].notnull()].describe(percentiles=percentil
    print('-' * 20 + '\n')
```

```
Education:
count      99376
unique         4
top       HS/GED
freq       42927
Name: Education, dtype: object
--------------------

Age:
count    100780.000000
mean         43.857601
std          19.496181
min           0.000000
10%          21.000000
20%          25.000000
30%          29.000000
40%          35.000000
50%          42.000000
60%          49.000000
70%          55.000000
80%          61.000000
90%          72.000000
100%        107.000000
max         107.000000
Name: Age, dtype: float64
--------------------
```

In [98]:
```
# Cleaning 'Education' column & checking how many of the incidents resulted i
data[data['Age'] < 16].shape
```

Out[98]:
```
(1841, 10)
```

In [99]:
```
## More info
data[data['Age'] < 16].head()
```

Out[99]:

Index	Year	Month	Intent	Police	Sex	Age	Race	Hispanic	Place	Education
1012	2012	1	Suicide	0	M	15.0	White	100	Home	Less than HS
2423	2012	1	Homicide	0	M	14.0	Black	100	Home	Less than HS
2492	2012	1	Homicide	0	F	1.0	Hispanic	210	Home	Less than HS
2493	2012	1	Homicide	0	M	3.0	Hispanic	210	Home	Less than HS
2589	2012	1	Homicide	0	M	2.0	Hispanic	210	Home	Less than HS

In [100…

```
# Converting the NaN values in education column.

data[(data['Age'] < 16) & ((data['Education'].isnull()) | (data['Education'] 
```

Out[100…

Index	Year	Month	Intent	Police	Sex	Age	Race	Hispanic	Place	Education
13751	2012	2	Accidental	0	M	4.0	Black	100	Home	NaN
22146	2012	2	Homicide	0	F	1.0	Black	100	Trade/service area	NaN
1626	2012	3	Homicide	0	M	0.0	Hispanic	210	Home	NaN
1627	2012	3	Homicide	0	M	3.0	Hispanic	210	Home	NaN
13968	2012	3	Homicide	0	M	2.0	Black	100	Other unspecified	NaN

In [101…

```
index_temp = data[(data['Age'] < 16) &
                  ((data['Education'].isnull()) | (data['Education'] == "BA+"
data.loc[index_temp, 'Education'] = "Less than HS"
data[data.Education.isnull()].shape
```

Out[101…

```
(1345, 10)
```

In [102…

```
index_temp = data[(data['Age'] < 5)].index
data.loc[index_temp, 'Education'] = "Less than HS"
data.Education.describe()
```

Out[102…

```
count      99453
unique         4
top       HS/GED
freq       42927
Name: Education, dtype: object
```

In [ ]:

In [103…
```
data.dropna(inplace=True)
data = data[data.Education != "BA+"]
data.Education.value_counts()
```

Out[103…
```
HS/GED          42258
Less than HS    21525
Some college    21430
Name: Education, dtype: int64
```

In [104…
```
for column in data.columns:
    if column not in ['Age', '']:
        print(column, ': ', data[column].unique())
```

```
Year :  [2012 2013 2014]
Month :  [ 1  2  3  4  5  6  7  8  9 10 11 12]
Intent :  ['Suicide' 'Homicide' 'Accidental' 'Undetermined']
Police :  [0 1]
Sex :  ['F' 'M']
Race :  ['White' 'Native American/Native Alaskan' 'Black' 'Hispanic'
 'Asian/Pacific Islander']
Hispanic :  [100 211 261 210 282 222 260 200 223 270 226 281 220 275 998 271 2
31 250
 239 280 225 235 234 237 299 227 233 224 221 286 212 242 291 252 232 217
 218 238]
Place :  ['Street' 'Home' 'Other unspecified' 'Other specified'
 'Trade/service area' 'Farm' 'Residential institution'
 'Industrial/construction' 'Sports' 'School/instiution']
Education :  ['Some college' 'HS/GED' 'Less than HS']
```

In [105…
```
data.Sex.value_counts()
```

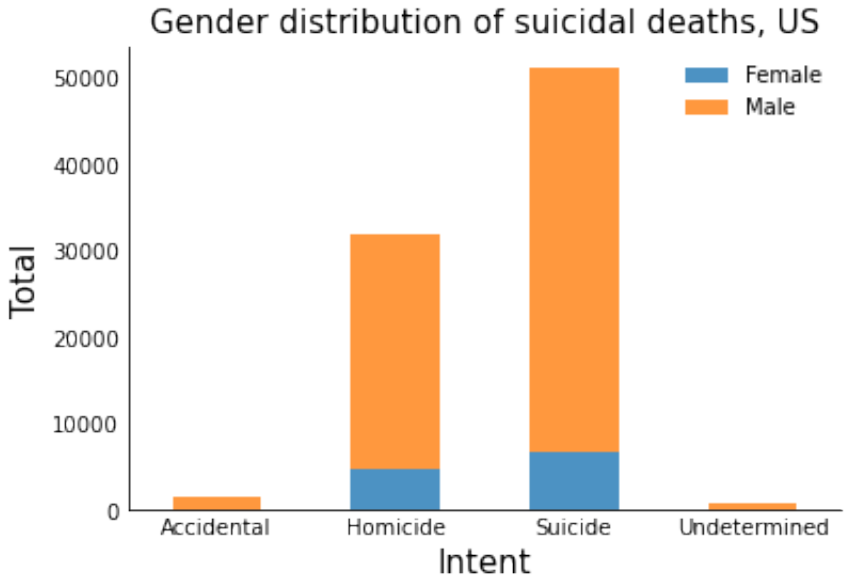
Out[105…
```
M    73347
F    11866
Name: Sex, dtype: int64
```

In [106…
```
data.Sex.value_counts(normalize=True)
```

Out[106…
```
M    0.860749
F    0.139251
Name: Sex, dtype: float64
```

In [107…

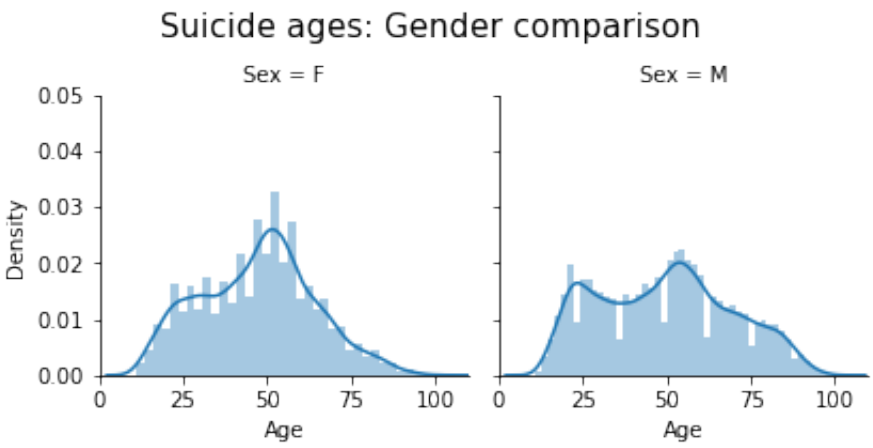
```
# Q. Is one gender more prone to suicide than the other one?
intent_sex = data.groupby(['Intent', 'Sex'])['Intent'].count().unstack('Sex')
ax = intent_sex.plot(kind='bar', stacked=True, alpha=0.8)
ax.set_xlabel('Intent', fontsize=15)
ax.set_ylabel('Total', fontsize=15)
plt.xticks(rotation=0)
plt.tick_params(axis='both', which='both',length=0)
ax.legend(labels=['Female', 'Male'], frameon=False, loc=0)
plt.title('Gender distribution of suicidal deaths, US', fontsize=15, fontweig
sns.despine()
plt.show()
```



In [108…

```
suicide = data[data['Intent'] == 'Suicide']
s = sns.FacetGrid(suicide, col='Sex')
s.map(sns.distplot, 'Age')
plt.subplots_adjust(top=0.8)
s.set(xlim=(0, 110), ylim=(0, 0.05))
s.fig.suptitle('Suicide ages: Gender comparison', fontsize=15, fontweight='li
```

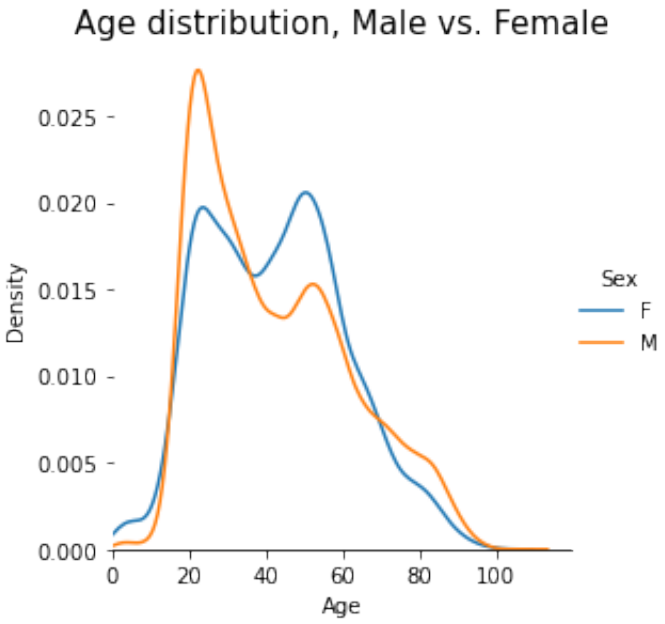
Out[108… Text(0.5, 0.98, 'Suicide ages: Gender comparison')



In [109…

```
# KDE Plot for gender
sns.FacetGrid(data, hue='Sex', size=4).map(sns.kdeplot, 'Age').add_legend()
sns.despine(left=True)
plt.xlim(xmin=0)
plt.title('Age distribution, Male vs. Female', fontsize=15, fontweight='light
```

Out[109… Text(0.5, 1.0, 'Age distribution, Male vs. Female')

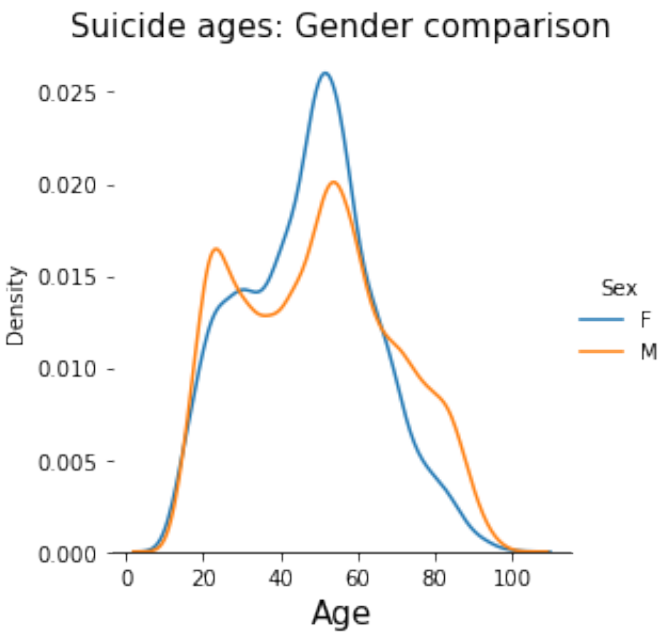


In [110…

```
# # Q.Does age play a role in occurrence of suicides?

sns.FacetGrid(suicide, hue='Sex', size=4).map(sns.kdeplot, 'Age').add_legend(
plt.xlabel('Age', fontsize=15)
sns.despine(left=True)
plt.title('Suicide ages: Gender comparison', fontsize=15, fontweight='light')
```

Out[110… Text(0.5, 1.0, 'Suicide ages: Gender comparison')



In [111…
```
data['Age'] .tail(3)
```

Out[111…
```
Index
100795    19.0
100796    20.0
100797    22.0
Name: Age, dtype: float64
```

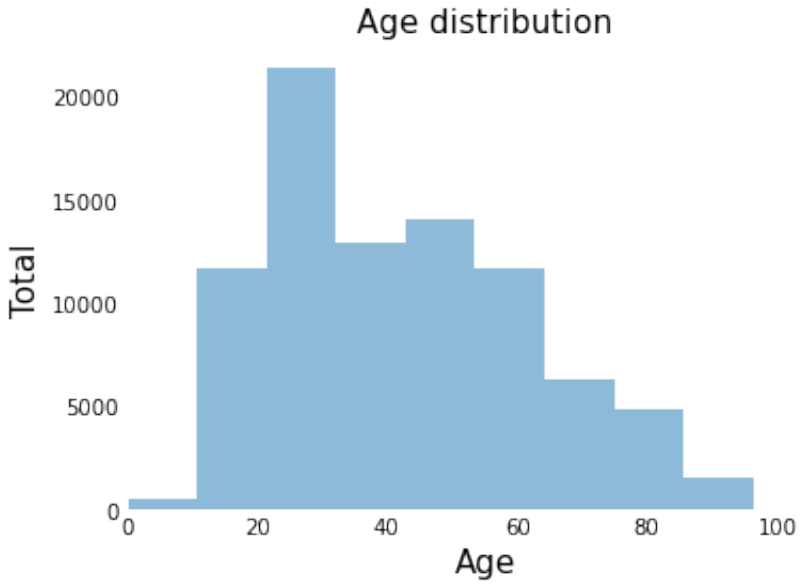
In [112…
```
data.head(3)
```

Out[112…

Index	Year	Month	Intent	Police	Sex	Age	Race	Hispanic	Place	Education
2	2012	1	Suicide	0	F	21.0	White	100	Street	Some college
12	2012	1	Suicide	0	M	21.0	Native American/Native Alaskan	100	Home	HS/GED
135	2012	1	Suicide	0	F	59.0	White	100	Home	HS/GED

In [113…

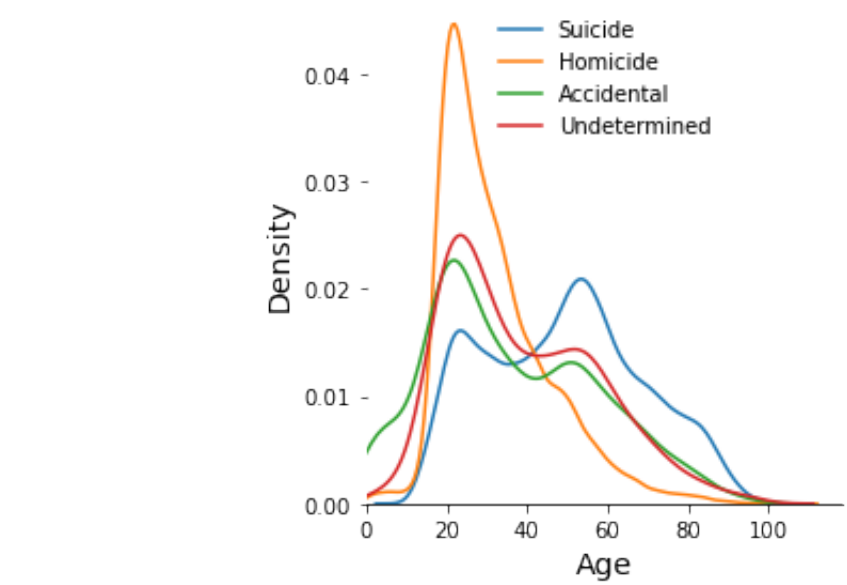
```
age_dist = data.Age.value_counts()
sorted_age_dist = age_dist.sort_index()
sorted_age_dist.head()
plt.hist(data['Age'], range=(0,107), alpha=0.5)
plt.tick_params(axis='both', which='both',length=0)
plt.xlim(xmin=0, xmax=110)
plt.xlabel('Age', fontsize=15)
plt.ylabel('Total', fontsize=15)
plt.title('Age distribution', fontsize=15, fontweight='light')
sns.despine(bottom=True, left=True)
plt.show()
```



In [114…

```
# limit the x-axis
sns.FacetGrid(data, hue='Intent', size=4).map(sns.kdeplot, 'Age')
plt.legend(loc=9, frameon=False)
plt.xlim(xmin=0)
plt.xlabel('Age', fontsize=14)
plt.ylabel('Density', fontsize=14)
sns.despine(left=True)
plt.title('Age distribution, Homicide vs. Suicide vs. Accidental vs. Undeterm
```

Out[114…
```
Text(0.5, 1.0, 'Age distribution, Homicide vs. Suicide vs. Accidental vs. Undetermined')
```

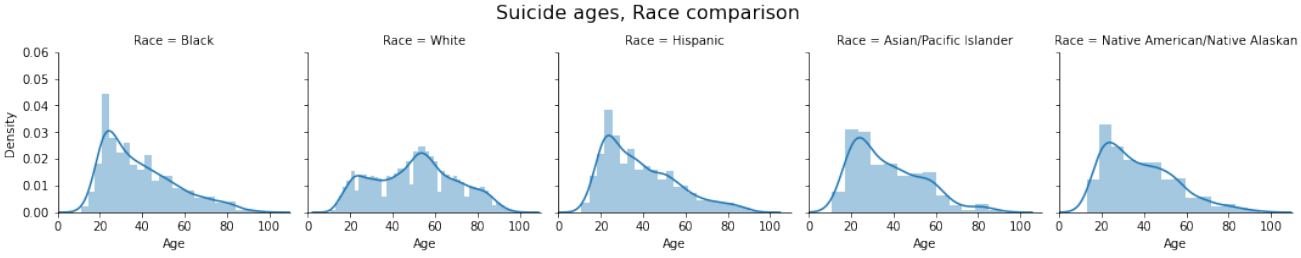


In [115…
```
# Is any combination of factors indicative of more suicides than other?

race_ordered = ['Black', 'White', 'Hispanic', 'Asian/Pacific Islander', 'Nati
data['Race'] = data['Race'].astype('category')
data.Race.cat.set_categories(race_ordered, inplace=True)

suicide = data[data['Intent'] == 'Suicide']
s = sns.FacetGrid(suicide, col='Race')
s.map(sns.distplot, 'Age')
plt.subplots_adjust(top=0.8)
s.set(xlim=(0, 110), ylim=(0, 0.06), xlabel='Age')
s.fig.suptitle('Suicide ages, Race comparison', fontsize=16, fontweight='ligh
```

Out[115…
```
Text(0.5, 0.98, 'Suicide ages, Race comparison')
```



In [ ]:

In [ ]:

In [ ]: