```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from wordcloud import STOPWORDS
from matplotlib.pyplot import figure
import seaborn as sns
%matplotlib inline


df=pd.read_csv("/content/County_Health_Rankings.csv")


df.head()
```

	State	County	State code	County code	Year span	Measure name	Measure id	Numerator	Denomina
0	US	United States	0.0	0.0	2003-2005	Violent crime rate	43.0	1328750.667	2748771
1	US	United States	0.0	0.0	2004-2006	Violent crime rate	43.0	1340928.667	2776127
2	US	United States	0.0	0.0	2005-2007	Violent crime rate	43.0	1355853.167	2804076
3	US	United	0.0	0.0	2006-	Violent crime	43.0	1366928.333	2876145

```
df.shape
```

```
(303864, 14)
```

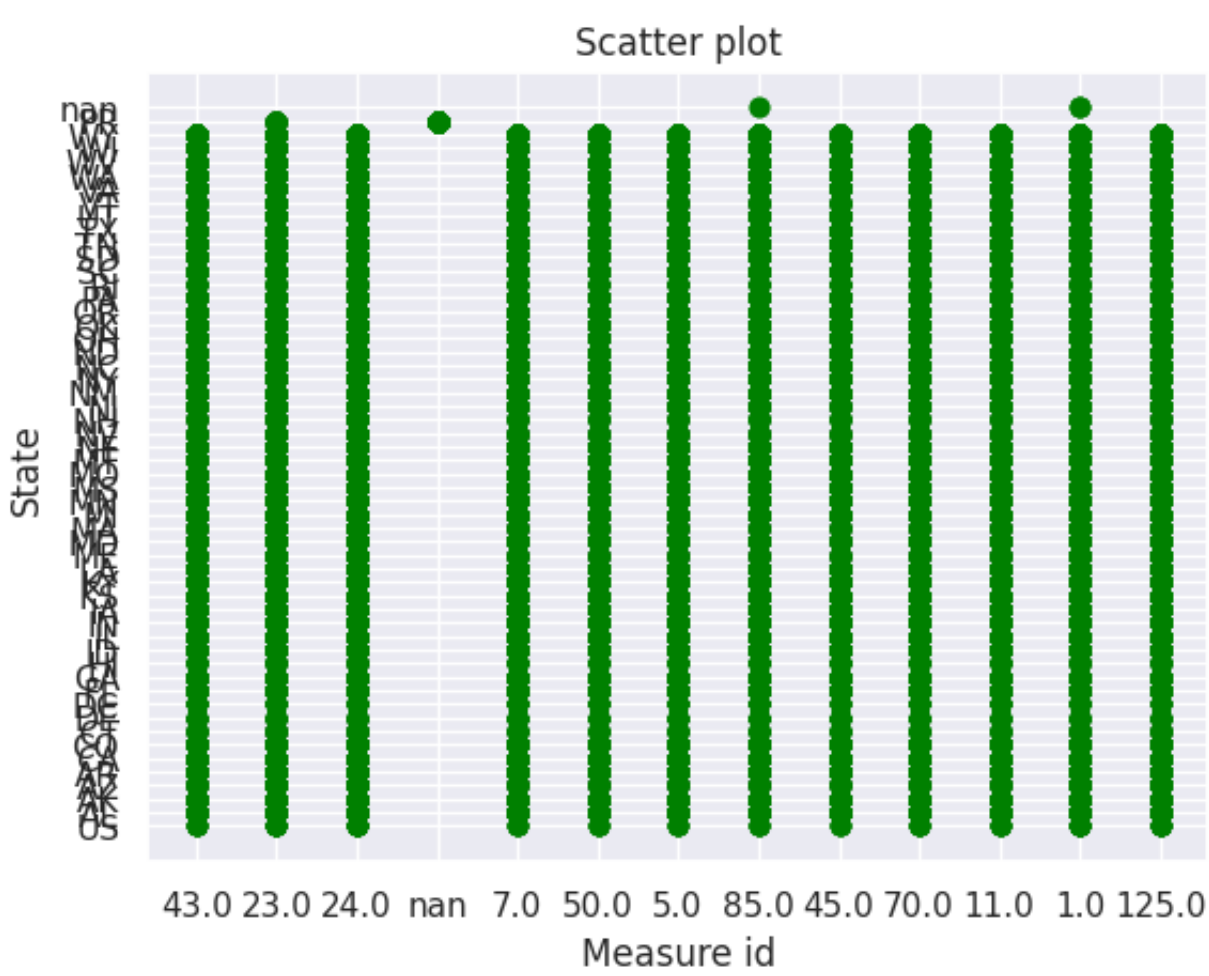
```
ndf=df.dropna()
ndf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 25221 entries, 156696 to 236744
Data columns (total 14 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   State                            25221 non-null  object 
 1   County                           25221 non-null  object 
 2   State code                       25221 non-null  float64
 3   County code                      25221 non-null  float64
 4   Year span                        25221 non-null  object 
 5   Measure name                     25221 non-null  object 
 6   Measure id                       25221 non-null  object 
 7   Numerator                        25221 non-null  float64
 8   Denominator                      25221 non-null  float64
 9   Raw value                        25221 non-null  float64
 10  Confidence Interval Lower Bound  25221 non-null  float64
 11  Confidence Interval Upper Bound  25221 non-null  float64
 12  Data Release Year                25221 non-null  float64
 13  fipscode                         25221 non-null  float64
dtypes: float64(9), object(5)
memory usage: 2.9+ MB
```

```
sns.set()
```

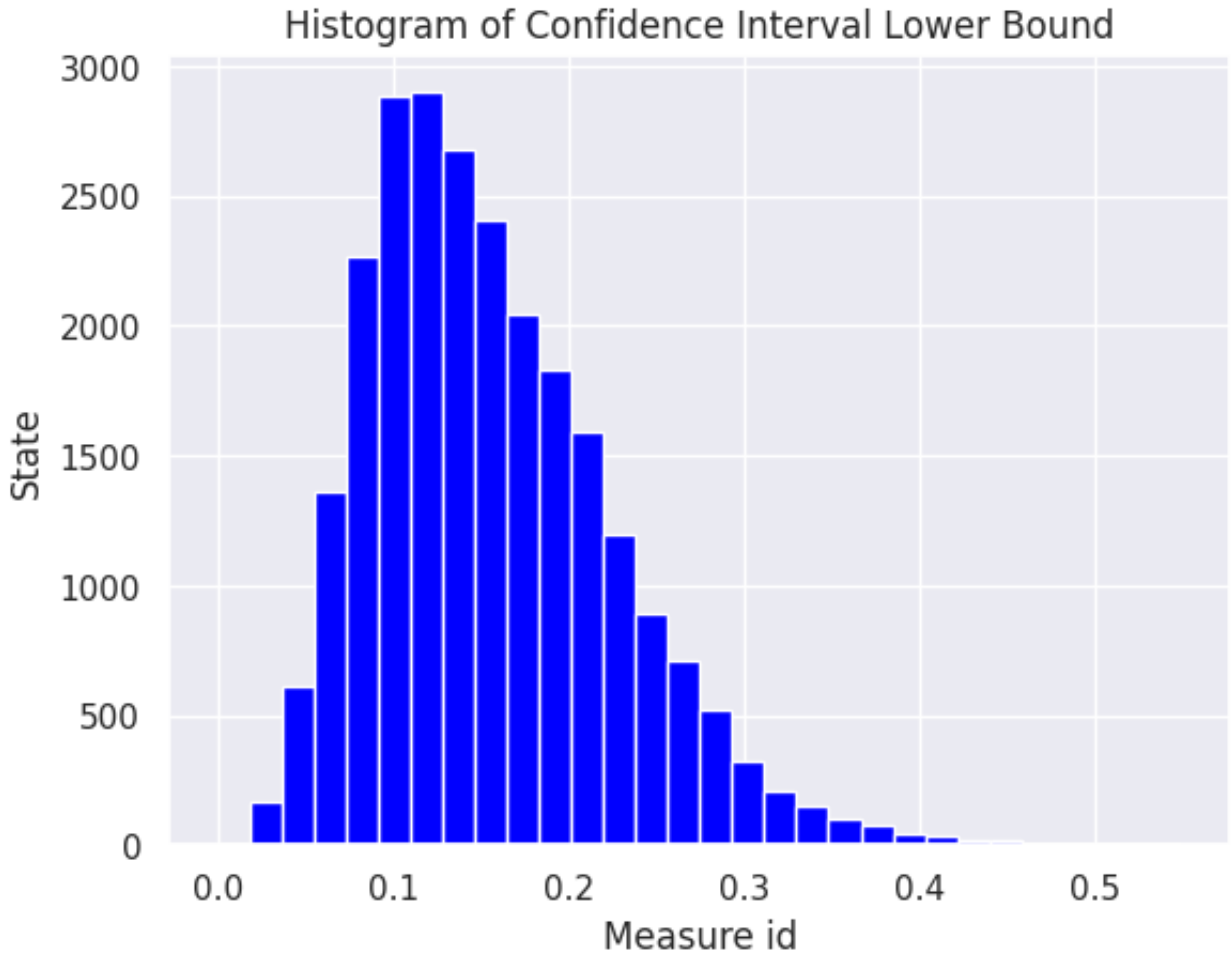
SCATTER PLOT

```
import pandas as pd
import matplotlib.pyplot as plt


# Convert 'State' column to string and handle NaN values
df['Measure id'] = df['Measure id'].astype(str).fillna('')

# Now plot the scatter plot
plt.scatter(df['Measure id'], df['State'], color='Green')
plt.title('Scatter plot')
plt.xlabel('Measure id')
plt.ylabel('State')
plt.show()
```



HISTOGRAM

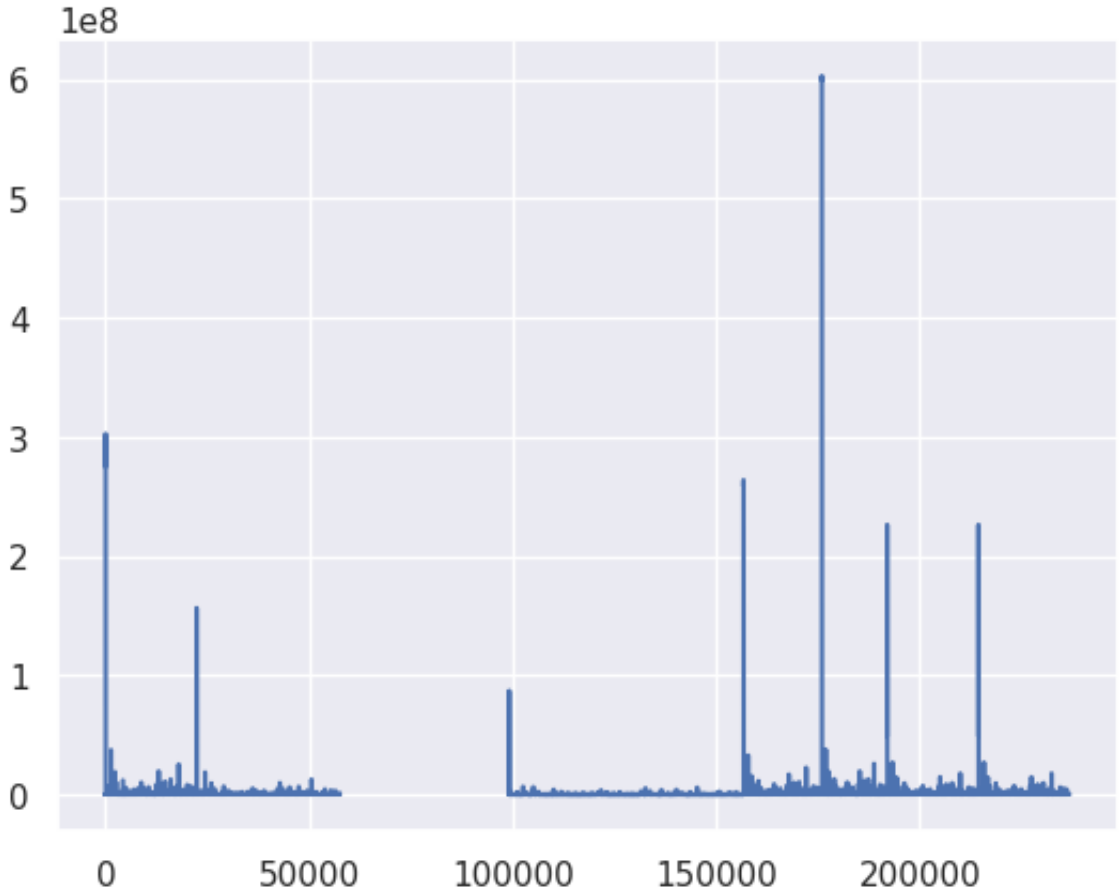
```
plt.hist(df['Confidence Interval Lower Bound'],color='blue',edgecolor='white',b
plt.title('Histogram of Confidence Interval Lower Bound')
plt.xlabel('Measure id')
plt.ylabel('State')
plt.show()
```



LINE PLOT

```
df['Denominator'].plot()
```

```
<Axes: >
```



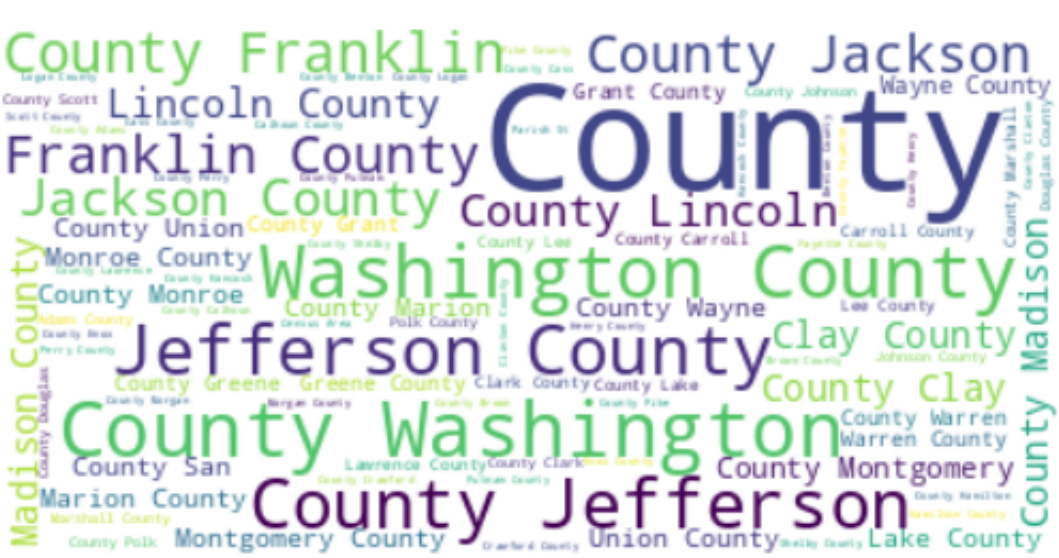

```
pip install wordcloud matplotlib
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
```

```
text= " ".join(item for item in ndf['County'])
print(text)
```

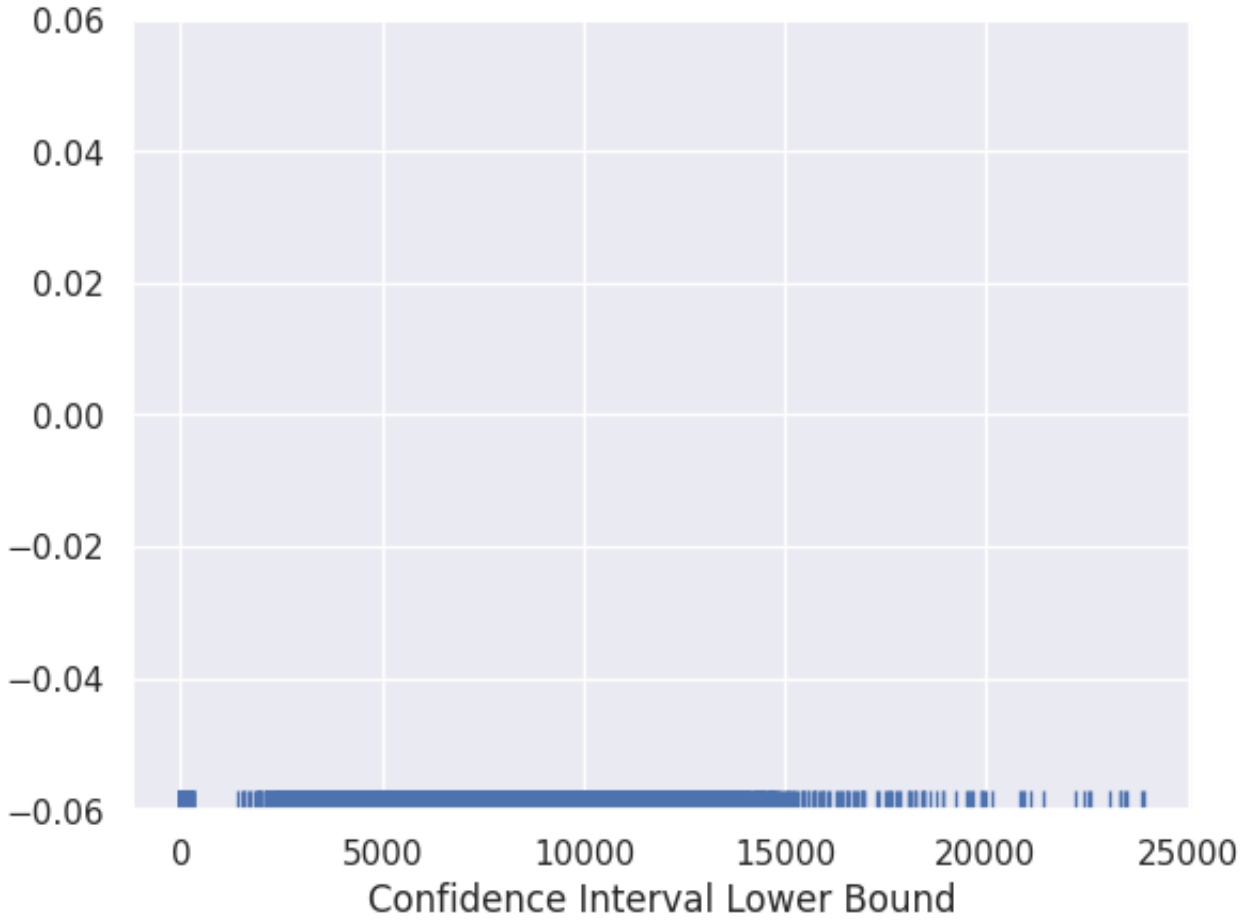
```
Alabama Alabama Autauga County Autauga County Baldwin County Baldwin County
```

```
stopwords = set(STOPWORDS)


wordcloud = WordCloud(background_color="White").generate(text)
plt.imshow(wordcloud, interpolation= 'bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```



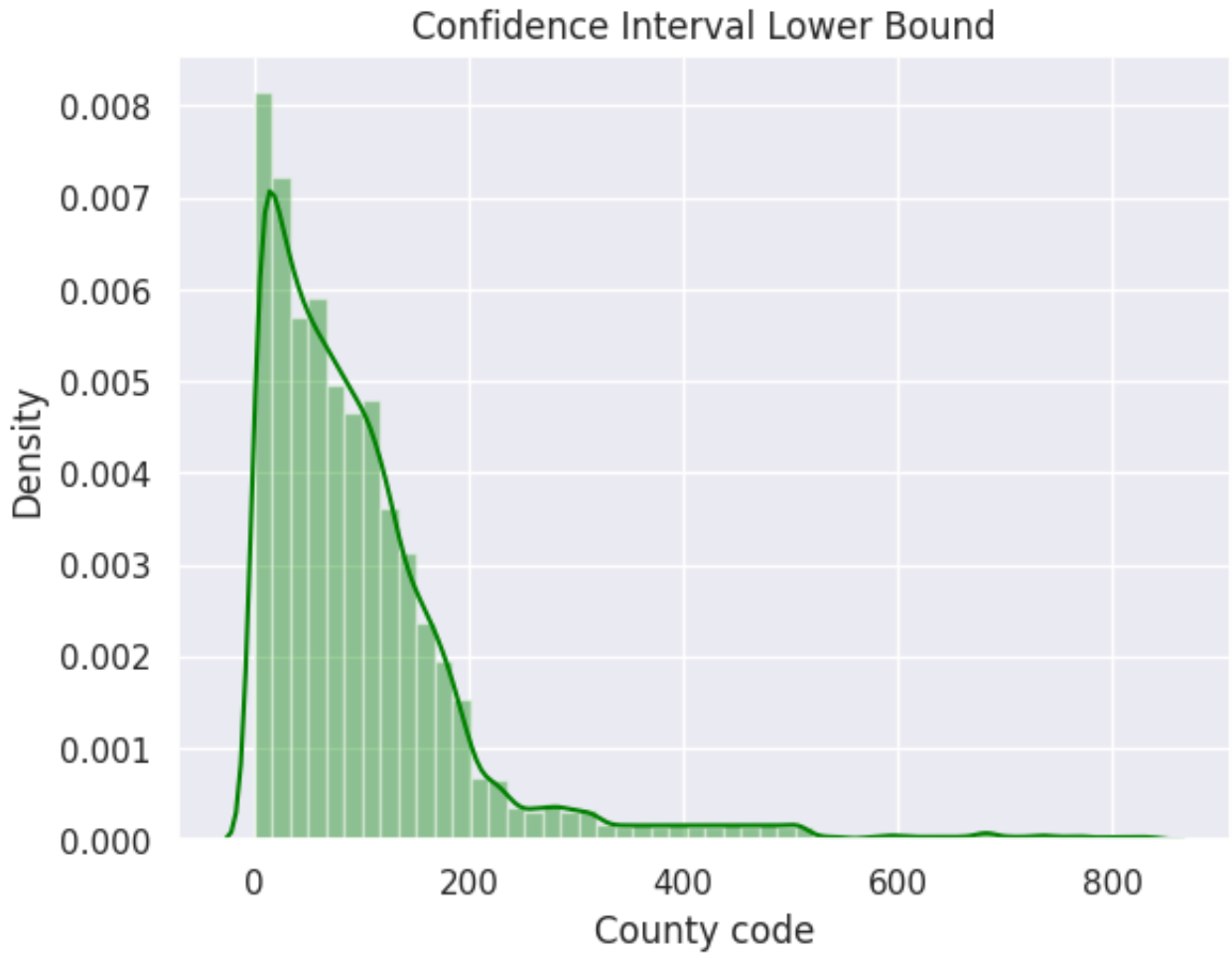
```
sns.rugplot(df['Confidence Interval Lower Bound'])
```

```
<Axes: xlabel='Confidence Interval Lower Bound'>
```



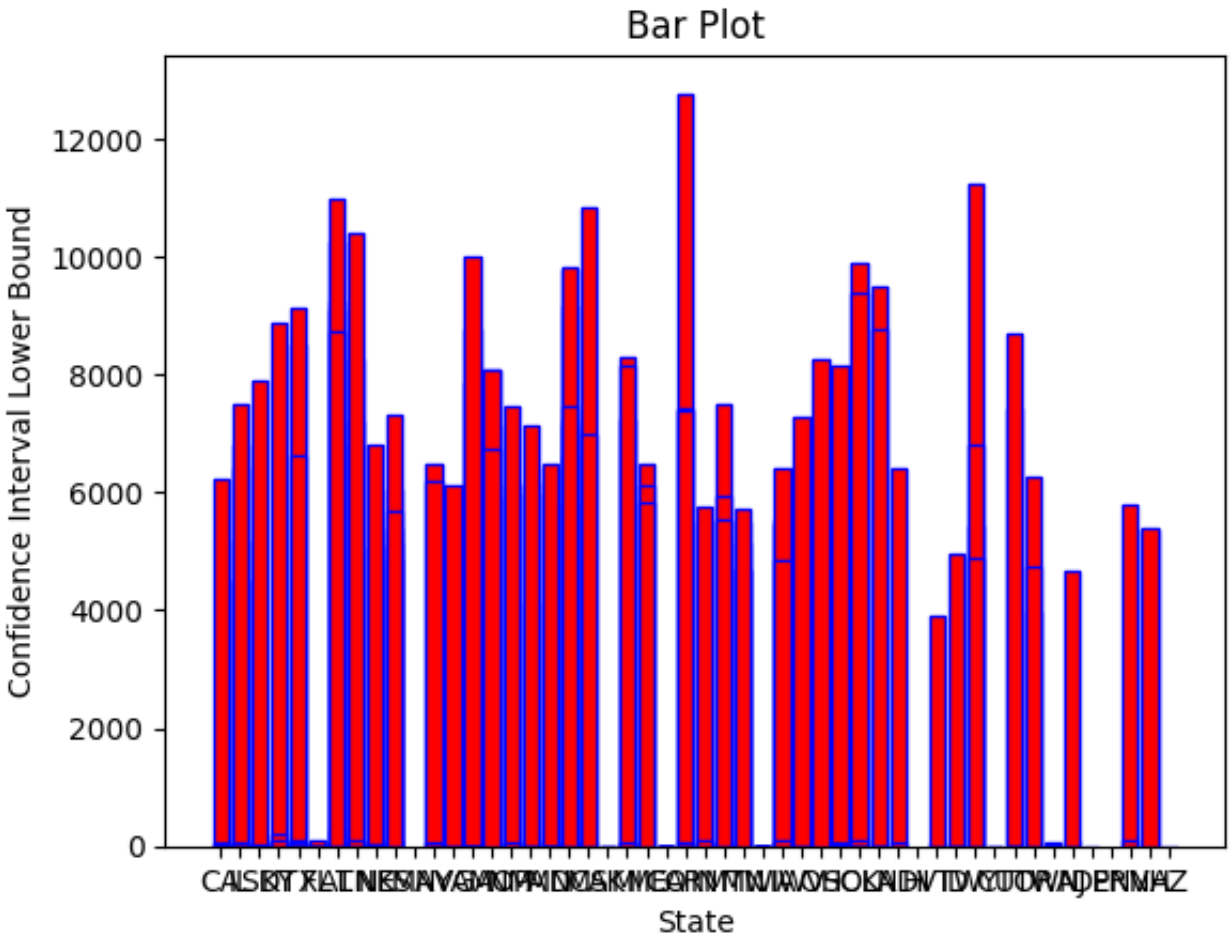
```
sns.distplot(df['County code'], kde=True, color='green').set_title('Confidence I
```

```
<ipython-input-29-1db0bb0945e1>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wit
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['County code'], kde=True, color='green').set_title('Confi
Text(0.5, 1.0, 'Confidence Interval Lower Bound')
```

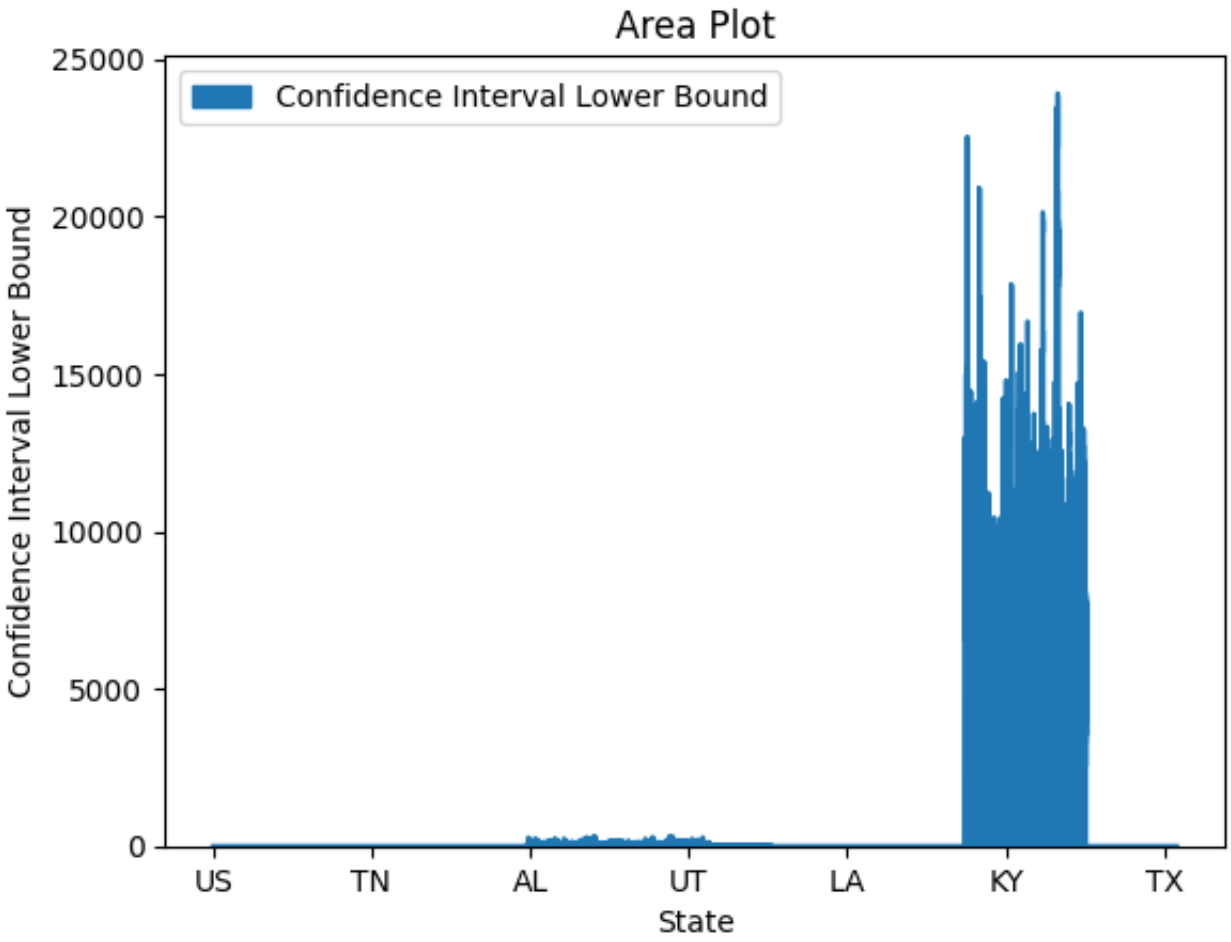


BAR PLOT

```
subset = df.sample(1000)  # Random sample of 1000 rows
plt.bar(subset['State'], subset['Confidence Interval Lower Bound'], color='red'
plt.title('Bar Plot')
plt.xlabel('State')
plt.ylabel('Confidence Interval Lower Bound')
plt.show()
```



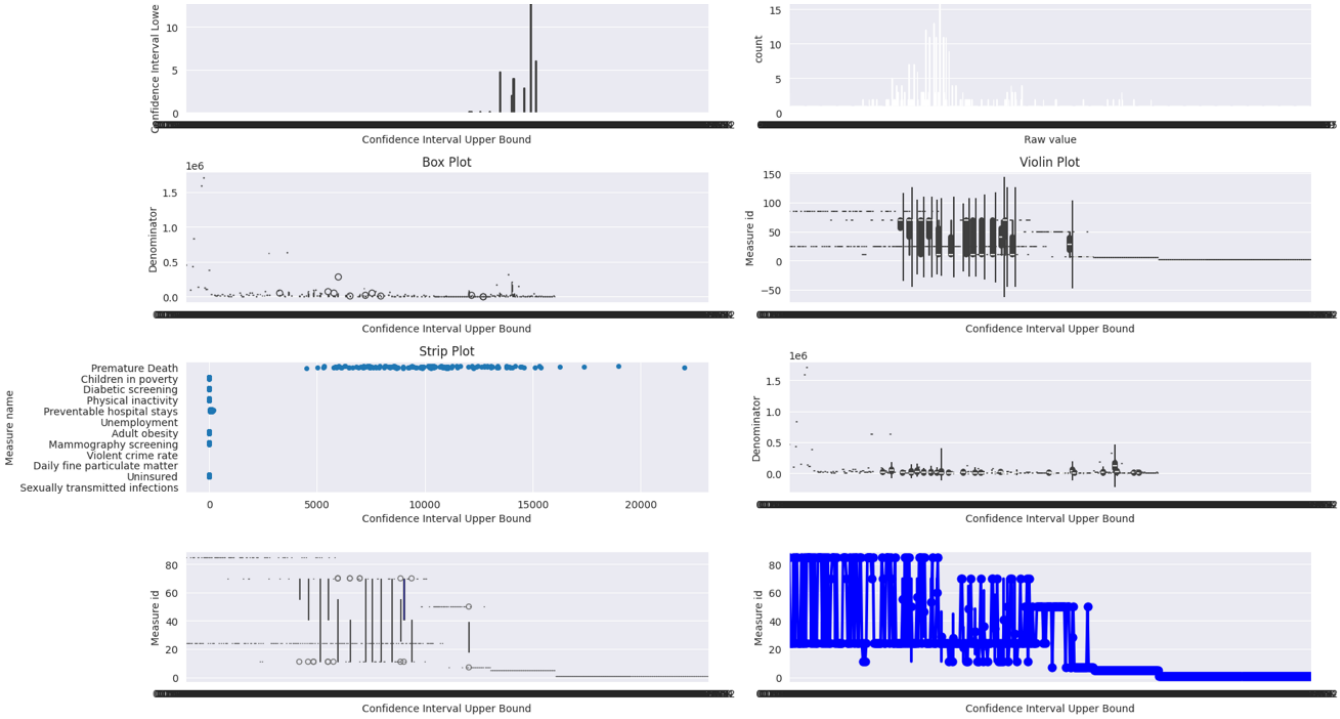
```
df.plot.area(x='State', y='Confidence Interval Lower Bound')
plt.title('Area Plot')
plt.xlabel('State')
plt.ylabel('Confidence Interval Lower Bound')
plt.show()
```




```
subset = df.sample(1000)  # Randomly sample 1000 rows from the dataset

import time
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

sns.set_style('darkgrid')
fig, ax = plt.subplots(nrows=4, ncols=2)
fig.set_size_inches(18.5, 10.5)

start_time = time.time()

sns.barplot(x='Confidence Interval Upper Bound', y='Confidence Interval Lower B
print("Bar Plot Time: ", time.time() - start_time)
```

```
start_time = time.time()
sns.countplot(x='Raw value', data=subset, ax=ax[0,1]).set_title('Count Plot')
print("Count Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.boxplot(x='Confidence Interval Upper Bound', y='Denominator', data=subset,
print("Box Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.violinplot(x='Confidence Interval Upper Bound', y='Measure id', data=subset
print("Violin Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.stripplot(x='Confidence Interval Upper Bound', y='Measure name', data=subse
print("Strip Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.violinplot(x='Confidence Interval Upper Bound', y='Denominator', data=subse
print("Second Violin Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.boxenplot(x='Confidence Interval Upper Bound', y='Measure id', color="b", s
print("Boxen Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.pointplot(x='Confidence Interval Upper Bound', y='Measure id', color="b", c
print("Point Plot Time: ", time.time() - start_time)

plt.tight_layout()
plt.show()
```

```
<ipython-input-5-dd881d3c081a>:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.barplot(x='Confidence Interval Upper Bound', y='Confidence Interval L
Bar Plot Time:  7.729967832565308
Count Plot Time:  2.4416346549987793
Box Plot Time:  1.727332592010498
Violin Plot Time:  10.683933734893799
Strip Plot Time:  0.06928586959838867
Second Violin Plot Time:  3.7013022899627686
<ipython-input-5-dd881d3c081a>:38: FutureWarning:

The `scale` parameter has been renamed to `width_method` and will be remove
  sns.boxenplot(x='Confidence Interval Upper Bound', y='Measure id', color=
Boxen Plot Time:  6.161583423614502
Point Plot Time:  2.501466989517212
```


Confidence Interval Lowe
Confidence Interval Upper Bound
count
Raw value
Box Plot
1e6
Denominator
Violin Plot
Measure id
Strip Plot
Measure name
Premature Death
Children in poverty
Diabetic screening
Physical inactivity
Preventable hospital stays
Unemployment
Adult obesity
Mammography screening
Violent crime rate
Daily fine particulate matter
Uninsured
Sexually transmitted infections

```
# Check data types
print(df.dtypes)

# Convert data types if necessary
df['Confidence Interval Upper Bound'] = pd.to_numeric(df['Confidence Interval U
df['Confidence Interval Lower Bound'] = pd.to_numeric(df['Confidence Interval L
df['Denominator'] = pd.to_numeric(df['Denominator'])
df['Measure id'] = pd.to_numeric(df['Measure id'])
df['Raw value'] = pd.to_numeric(df['Raw value'], errors='coerce')
df['Measure name'] = df['Measure name'].astype('category')
df['State'] = df['State'].astype('category')
```

```
State                                object
County                               object
State code                          float64
County code                         float64
Year span                            object
Measure name                         object
Measure id                          float64
Numerator                           float64
Denominator                         float64
Raw value                           float64
Confidence Interval Lower Bound     float64
Confidence Interval Upper Bound     float64
Data Release Year                   float64
fipscode                            float64
dtype: object
```

```
print(df.isnull().sum())
df = df.dropna()  # or use df.fillna() to fill missing values
```

```
State                                6
County                               6
State code                           4
County code                          4
Year span                          474
Measure name                       474
Measure id                         474
Numerator                        89788
Denominator                     119085
Raw value                        13908
Confidence Interval Lower Bound  114452
Confidence Interval Upper Bound  114452
Data Release Year                153735
fipscode                           9581
dtype: int64
```

```
import time
```

```
import time
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Sample a subset of the data
subset = df.sample(1000)  # Randomly sample 1000 rows from the dataset

sns.set_style('darkgrid')
fig, ax = plt.subplots(nrows=4, ncols=2)
fig.set_size_inches(18.5, 10.5)

start_time = time.time()

sns.barplot(x='Confidence Interval Upper Bound', y='Confidence Interval Lower B
print("Bar Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.countplot(x='Raw value', data=subset, ax=ax[0,1]).set_title('Count Plot')
print("Count Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.boxplot(x='Confidence Interval Upper Bound', y='Denominator', data=subset,
print("Box Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.violinplot(x='Confidence Interval Upper Bound', y='Measure id', data=subset
print("Violin Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.stripplot(x='Confidence Interval Upper Bound', y='Measure name', data=subse
print("Strip Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.violinplot(x='Confidence Interval Upper Bound', y='Denominator', data=subse
print("Second Violin Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.boxenplot(x='Confidence Interval Upper Bound', y='Measure id', color="b", s
print("Boxen Plot Time: ", time.time() - start_time)

start_time = time.time()
sns.pointplot(x='Confidence Interval Upper Bound', y='Measure id', color="b", c
print("Point Plot Time: ", time.time() - start_time)

plt.tight_layout()
plt.show()
```

```
<ipython-input-10-97b05876bdc2>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.barplot(x='Confidence Interval Upper Bound', y='Confidence Interval L
Bar Plot Time:  10.5068998336792
Count Plot Time:  0.7199633121490479
Box Plot Time:  3.302156686782837
Violin Plot Time:  9.474660158157349
Strip Plot Time:  0.1015787124633789
Second Violin Plot Time:  10.605979681015015
<ipython-input-10-97b05876bdc2>:39: FutureWarning:

The `scale` parameter has been renamed to `width_method` and will be remove
  sns.boxenplot(x='Confidence Interval Upper Bound', y='Measure id', color=
Boxen Plot Time:  5.638623237609863
Point Plot Time:  3.6228911876678467
```