Name: Tayyaba Fatima

College: Mt.San Antonio College

Professor: Dr. Sohair Zaki

Topic: Adult Prevalence Mental Health of USA in 2023 with comparison To Mass Shootings occured in USA in 2023.

Datframes:

- dfmental1: Adult prevalence Mental Health data of USA in 2023

```
Attributes:
Ranks:- Ranks of state for How many people are affected from AMI(A
dult Mental Illness)
States:- States of USA
Percentage :- Percentage of adults affected by Mental health illne
ss
Numbers:- Number of Adults affected by AMI
```

  - dfmass = Datframe for Mass Shootings Occured in USA in 2023 Attributes:

```
 State:- States of USA
 Incident Date: - Date of incidents occured
 Incident Id:- Id number of an incident
 Victims Killed:- Number of people killed in an incident
 Victims Injured:- Number of people injured in an incident
 Suspect Arresed:- Number of Suspect arrested for an incident
 Suspect Injured:- Number of Suspect Injured  for an incident
 City or county:- Location in the states where Incident occure
d
 Address:- Address of an incident.
```

  - Finaldf: Combination of dfmental1 and dfmass

Data Manipulation and Cleaning

Data Visualization:-It is done from the Tableau and all the visualizations is in pdf

Data Prediction

Derive Conclusion

In [2]:

```
#Importing required Libraries.
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
%matplotlib inline

from urllib.request import urlopen
from bs4 import BeautifulSoup
from sklearn.model_selection import cross_validate
from sklearn import preprocessing ,svm
from sklearn.linear_model import LinearRegression

style.use('ggplot')
```

In [3]:

```
# Input the URL of the dataset we are going to use
url = 'https://www.mhanational.org/issues/2023/mental-health-america-p
html = urlopen(url)
```

In [4]:

```
soup = BeautifulSoup(html,'lxml')
type(soup)
```

Out[4]: bs4.BeautifulSoup

In [5]:

```
#Get the Tittle
title = soup.title
print(title)
```

```
<title>Prevalence Data 2023 | Mental Health America</title>
```

In [6]:

```
#Print out the text
text = soup.get_text()
print(soup.text)
```

```
Prevalence Data 2023 | Mental Health America
```

In [7]:

```
#Let's see what is in the website
table = soup.find_all('table')[2]
table
```

Out[7]:

```
<table class="cols-4">
<thead>
<tr>
<th class="views-field views-field-field-rank is-active" id="view-field
-rank-table-column--3" scope="col"><a href="?order=field_rank&amp;sort=
desc" rel="nofollow" title="sort by Rank">Rank<span class="tablesort ta
blesort--desc">
<span class="visually-hidden">
    Sort descending
      </span>
</span>
</a></th>
<th class="views-field views-field-field-state" id="view-field-state-ta
ble-column--3" scope="col"><a href="?order=field_state&amp;sort=asc" re
l="nofollow" title="sort by State">State</a></th>
<th class="views-field views-field-field-percentage" id="view-field-per
centage-table-column--2" scope="col">Percentage</th>
<th class="views-field views-field-field-number" id="view-field-number-
table-column--2" scope="col">Number</th>
```

In [8]:

```
# Let's check the website for Hyperlinks
all_links = soup.find_all('a')
for link in all_links:
    print(link.get('href'))
```

```
#main-content
tel:800-969-6642
/get-involved/contact-us
/programs
https://arc.mhanational.org/find-affiliate (https://arc.mhanational.or
g/find-affiliate)
https://store.mhanational.org/ (https://store.mhanational.org/)
https://mhanational.org (https://mhanational.org)
/about-us
/2022/annual-conference
/career-center
/youngleaders
/issues/advocacy-network
/about-mental-health
/mentalhealthfacts
/MentalHealthInfo
/programs
/news
/policy-issues
```

In [9]:

```
# Print the first 10 rows
rows = soup.find_all('tr')
print(rows[:10])
```

```
[<tr>
<th class="views-field views-field-field-rank is-active" id="view-field-r
ank-table-column" scope="col"><a href="?order=field_rank&amp;sort=desc" r
el="nofollow" title="sort by Rank">Rank<span class="tablesort tablesort--
desc">
<span class="visually-hidden">
    Sort descending
      </span>
</span>
</a></th>
<th class="views-field views-field-field-state" id="view-field-state-tabl
e-column" scope="col"><a href="?order=field_state&amp;sort=asc" rel="nofo
llow" title="sort by State">State</a></th>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">01          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">Georgia          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">02          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">South Carolina          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">03          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">Texas          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">04          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">New Jersey          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">05          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">North Carolina          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">06          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">Delaware          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">07          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">Florida          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">08          </td>
<td class="views-field views-field-field-state" headers="view-field-state
-table-column">Maryland          </td>
</tr>, <tr class="rankings">
<td class="views-field views-field-edit-node views-field-field-rank is-ac
tive" headers="view-field-rank-table-column">09          </td>
<td class="views-field views-field-field-state" headers="view-field-state
```

```
-table-column">Wisconsin          </td>
</tr>]
```

In [10]:

```
#Let's Create the rows from webpage
for row in rows:
    row_td = row.find_all('td')
    print(row_td)
    type(row_td)
```

```
[]
[<td class="views-field views-field-edit-node views-field-field-rank is
-active" headers="view-field-rank-table-column">01          </td>, <td
class="views-field views-field-field-state" headers="view-field-state-t
able-column">Georgia          </td>]
[<td class="views-field views-field-edit-node views-field-field-rank is
-active" headers="view-field-rank-table-column">02          </td>, <td
class="views-field views-field-field-state" headers="view-field-state-t
able-column">South Carolina          </td>]
[<td class="views-field views-field-edit-node views-field-field-rank is
-active" headers="view-field-rank-table-column">03          </td>, <td
class="views-field views-field-field-state" headers="view-field-state-t
able-column">Texas          </td>]
[<td class="views-field views-field-edit-node views-field-field-rank is
-active" headers="view-field-rank-table-column">04          </td>, <td
class="views-field views-field-field-state" headers="view-field-state-t
able-column">New Jersey          </td>]
[<td class="views-field views-field-edit-node views-field-field-rank is
-active" headers="view-field-rank-table-column">05          </td>, <td
```

In [11]:

```
# Removing the html tags
str_cells = str(row_td)
clean_text = BeautifulSoup(str_cells,'lxml').get_text()
print(clean_text)
```

```
[52          , National          , 6.34          , 1,584,000          ]
```

In [12]:

```python
# Alternative to regular text in order to find usable text
import re

list_rows = []
for row in rows:
    cells = row.find_all('td')
    str_cells = str(cells)
    clean = re.compile('<.*?>')
    clean2 = (re.sub(clean,'',str_cells))
    list_rows.append(clean2)
    print(clean2)
    type(clean2)

```

```
[]
[01          , Georgia          ]
[02          , South Carolina          ]
[03          , Texas          ]
[04          , New Jersey          ]
[05          , North Carolina          ]
[06          , Delaware          ]
[07          , Florida          ]
[08          , Maryland          ]
[09          , Wisconsin          ]
[10          , Kentucky          ]
[11          , New York          ]
[12          , Hawaii          ]
[13          , Mississippi          ]
[14          , Pennsylvania          ]
[15          , Tennessee          ]
[16          , California          ]
[17          , Connecticut          ]
[18          , Nevada          ]
```

In [13]:

```
# Let's see how the tables came out in our dataframe
df = pd.DataFrame(list_rows)
df.head(50)
```

Out[13]:

	0
0	[]
1	[01 , Georgia]
2	[02 , South Carolina]
3	[03 , Texas]
4	[04 , New Jersey]
5	[05 , North Carolina]
6	[06 , Delaware]
7	[07 , Florida]
8	[08 , Maryland]
9	[09 , Wisconsin]
10	[10 , Kentucky]
11	[11 , New York]
12	[12 , Hawaii]
13	[13 , Mississippi]
14	[14 , Pennsylvania]
15	[15 , Tennessee]
16	[16 , California]
17	[17 , Connecticut]
18	[18 , Nevada]
19	[19 , Arkansas]
20	[20 , District of Columbia]
21	[21 , Indiana]
22	[22 , Alabama]
23	[23 , Massachusetts]
24	[24 , Louisiana]
25	[25 , Michigan]
26	[26 , Oklahoma]
27	[27 , Rhode Island]
28	[28 , Iowa]
29	[29 , Virginia]
30	[30 , Missouri]
31	[31 , Illinois]
32	[32 , North Dakota]
33	[33 , Colorado]
34	[34 , Alaska]

	0
35	[35 , New Hampshire]
36	[36 , New Mexico]
37	[37 , Wyoming]
38	[38 , Arizona]
39	[39 , West Virginia]
40	[40 , Ohio]
41	[41 , Minnesota]
42	[42 , Maine]
43	[43 , Washington]
44	[44 , Nebraska]
45	[45 , Vermont]
46	[46 , Utah]
47	[47 , South Dakota]
48	[48 , Idaho]
49	[49 , Montana]

Data Manipulation and cleaning.

In [14]:

```
# splitting the column 0 of our datframe df with [[,]]  saving it in d
df1 = df[0].str.split('[|[|,|]|]',expand = True)
df1.head()
```

Out[14]:

	0	1	2	3	4	5	6	7
0				None	None	None	None	None
1		01	Georgia		None	None	None	None
2		02	South Carolina		None	None	None	None
3		03	Texas		None	None	None	None
4		04	New Jersey		None	None	None	None

In [15]:

```
# printing the tail of the dataframe
df1.tail()
```

Out[15]:

	0	1	2	3	4	5	6	7
365		48	Vermont	7.91	3	000		None
366		49	Oregon	7.97	24	000		None
367		50	Montana	8.60	7	000		None
368		51	Kansas	9.05	22	000		None
369		52	National	6.34	1	584	000	

In [16]:

```
#getting the info of df1
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 370 entries, 0 to 369
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       370 non-null    object
 1   1       370 non-null    object
 2   2       370 non-null    object
 3   3       363 non-null    object
 4   4       312 non-null    object
 5   5       312 non-null    object
 6   6       312 non-null    object
 7   7       36 non-null     object
dtypes: object(8)
memory usage: 23.2+ KB
```

In [17]:

```
# let's rename the columns and save in new dataframe dfnew
dfnew =df1.set_axis(['ex1','Ranks', "State","Percentage","num1","num2"
```

In [18]:

```
# printing the head of dfnew with 10 rows
dfnew.head(10)
```

Out[18]:

	ex1	Ranks	State	Percentage	num1	num2	num5	ex2
0				None	None	None	None	None
1		01	Georgia		None	None	None	None
2		02	South Carolina		None	None	None	None
3		03	Texas		None	None	None	None
4		04	New Jersey		None	None	None	None
5		05	North Carolina		None	None	None	None
6		06	Delaware		None	None	None	None
7		07	Florida		None	None	None	None
8		08	Maryland		None	None	None	None
9		09	Wisconsin		None	None	None	None

In [19]:

```
#printing the tail of dfnew
dfnew.tail()
```

Out[19]:

	ex1	Ranks	State	Percentage	num1	num2	num5	ex2
365		48	Vermont	7.91	3	000		None
366		49	Oregon	7.97	24	000		None
367		50	Montana	8.60	7	000		None
368		51	Kansas	9.05	22	000		None
369		52	National	6.34	1	584	000	

In [20]:

```
# Replacing the None values with .0
dfnew.replace({None:('.0')},inplace = True)
dfnew
```

Out[20]:

	ex1	Ranks	State	Percentage	num1	num2	num5	ex2
0				.0	.0	.0	.0	.0
1		01	Georgia		.0	.0	.0	.0
2		02	South Carolina		.0	.0	.0	.0
3		03	Texas		.0	.0	.0	.0
4		04	New Jersey		.0	.0	.0	.0
...	...	...	...	...	...	...	...	...
365		48	Vermont	7.91	3	000		.0
366		49	Oregon	7.97	24	000		.0
367		50	Montana	8.60	7	000		.0
368		51	Kansas	9.05	22	000		.0
369		52	National	6.34	1	584	000	

370 rows × 8 columns

In [21]:

```
# combining the num1 num2 and num5 and saving them  into new column na
dfnew['Numbers'] =(dfnew['num1'] + dfnew['num2'] + dfnew['num5'])
dfnew.tail()
```

Out[21]:

	ex1	Ranks	State	Percentage	num1	num2	num5	ex2	Numbers
365		48	Vermont	7.91	3	000		.0	3000
366		49	Oregon	7.97	24	000		.0	24000
367		50	Montana	8.60	7	000		.0	7000
368		51	Kansas	9.05	22	000		.0	22000
369		52	National	6.34	1	584	000		1584000

In [22]:

```
# Dropping any extra columns
dfnew.drop(["ex1",'ex2','num1','num2','num5'],axis = 1,inplace = True)
dfnew.head()
```

Out[22]:

	Ranks	State	Percentage	Numbers
0			.0	.0.0.0
1	01	Georgia		.0.0.0
2	02	South Carolina		.0.0.0
3	03	Texas		.0.0.0
4	04	New Jersey		.0.0.0

In [23]:

```
#dropping the rows and saving the rows that we need in new dataframe d
dfnew1=dfnew.drop(dfnew.index[0:53])
dfnew1.head(50)
```

Out[23]:

	Ranks	State	Percentage	Numbers
53	01	Florida	17.49	2985000
54	02	Georgia	17.55	1397000
55	03	Maryland	17.80	822000
56	04	Hawaii	17.86	189000
57	05	Texas	17.96	3825000
58	06	New Jersey	18.27	1251000
59	07	Connecticut	18.77	524000
60	08	New York	18.83	2855000
61	09	Pennsylvania	19.68	1963000
62	10	North Carolina	19.80	1592000
63	11	Tennessee	20.46	1073000
64	12	California	20.49	6169000
65	13	Virginia	20.51	1331000
66	14	Delaware	20.52	156000
67	15	Illinois	20.72	2000000
68	16	North Dakota	20.79	118000
69	17	Iowa	21.00	503000
70	18	Mississippi	21.06	465000
71	19	New Mexico	21.16	337000
72	20	Louisiana	21.18	733000
73	21	Alabama	21.24	797000
74	22	South Dakota	21.25	139000
75	23	Missouri	21.32	996000
76	24	Nevada	21.38	508000
77	25	Massachusetts	21.39	1172000
78	26	Maine	21.61	234000
79	27	South Carolina	21.69	862000
80	28	Indiana	21.83	1109000
81	29	Wisconsin	21.83	982000
82	30	Kentucky	21.91	742000
83	31	Alaska	22.20	117000
84	32	Michigan	22.33	1729000
85	33	Arkansas	22.61	514000
86	34	District of Columbia	22.95	131000
87	35	Colorado	23.16	1028000

	Ranks	State	Percentage	Numbers
88	36	Minnesota	23.23	997000
89	37	Nebraska	23.41	335000
90	38	Montana	23.43	195000
91	39	Wyoming	23.63	103000
92	40	Vermont	23.71	120000
93	41	New Hampshire	23.74	260000
94	42	Arizona	23.89	1339000
95	43	Rhode Island	24.12	202000
96	44	Ohio	24.32	2177000
97	45	Idaho	24.92	333000
98	46	Washington	25.51	1500000
99	47	Oklahoma	25.59	752000
100	48	Kansas	26.02	560000
101	49	West Virginia	26.05	366000
102	50	Oregon	27.33	909000

In [24]:

```
1 dfnew1.head(52)
```

Out[24]:

	Ranks	State	Percentage	Numbers
53	01	Florida	17.49	2985000
54	02	Georgia	17.55	1397000
55	03	Maryland	17.80	822000
56	04	Hawaii	17.86	189000
57	05	Texas	17.96	3825000
58	06	New Jersey	18.27	1251000
59	07	Connecticut	18.77	524000
60	08	New York	18.83	2855000
61	09	Pennsylvania	19.68	1963000
62	10	North Carolina	19.80	1592000
63	11	Tennessee	20.46	1073000
64	12	California	20.49	6169000
65	13	Virginia	20.51	1331000
66	14	Delaware	20.52	156000
67	15	Illinois	20.72	2000000
68	16	North Dakota	20.79	118000
69	17	Iowa	21.00	503000
70	18	Mississippi	21.06	465000
71	19	New Mexico	21.16	337000
72	20	Louisiana	21.18	733000
73	21	Alabama	21.24	797000
74	22	South Dakota	21.25	139000
75	23	Missouri	21.32	996000
76	24	Nevada	21.38	508000
77	25	Massachusetts	21.39	1172000
78	26	Maine	21.61	234000
79	27	South Carolina	21.69	862000
80	28	Indiana	21.83	1109000
81	29	Wisconsin	21.83	982000
82	30	Kentucky	21.91	742000
83	31	Alaska	22.20	117000
84	32	Michigan	22.33	1729000
85	33	Arkansas	22.61	514000
86	34	District of Columbia	22.95	131000
87	35	Colorado	23.16	1028000

	Ranks	State	Percentage	Numbers
88	36	Minnesota	23.23	997000
89	37	Nebraska	23.41	335000
90	38	Montana	23.43	195000
91	39	Wyoming	23.63	103000
92	40	Vermont	23.71	120000
93	41	New Hampshire	23.74	260000
94	42	Arizona	23.89	1339000
95	43	Rhode Island	24.12	202000
96	44	Ohio	24.32	2177000
97	45	Idaho	24.92	333000
98	46	Washington	25.51	1500000
99	47	Oklahoma	25.59	752000
100	48	Kansas	26.02	560000
101	49	West Virginia	26.05	366000
102	50	Oregon	27.33	909000
103	51	Utah	29.68	675000
104	52	National	20.78	52173000

In [25]:

```
# dropping another set of rows from 105 index so that we will get the
dfmental=dfnew1.drop(df.index[105:])
dfmental.tail()
```

Out[25]:

	Ranks	State	Percentage	Numbers
100	48	Kansas	26.02	560000
101	49	West Virginia	26.05	366000
102	50	Oregon	27.33	909000
103	51	Utah	29.68	675000
104	52	National	20.78	52173000

In [26]:

```
# resetting the index
dfmental.reset_index(drop = True)
```

Out[26]:

	Ranks	State	Percentage	Numbers
0	01	Florida	17.49	2985000
1	02	Georgia	17.55	1397000
2	03	Maryland	17.80	822000
3	04	Hawaii	17.86	189000
4	05	Texas	17.96	3825000
5	06	New Jersey	18.27	1251000
6	07	Connecticut	18.77	524000
7	08	New York	18.83	2855000
8	09	Pennsylvania	19.68	1963000
9	10	North Carolina	19.80	1592000
10	11	Tennessee	20.46	1073000
11	12	California	20.49	6169000
12	13	Virginia	20.51	1331000
13	14	Delaware	20.52	156000
14	15	Illinois	20.72	2000000
15	16	North Dakota	20.79	118000
16	17	Iowa	21.00	503000
17	18	Mississippi	21.06	465000
18	19	New Mexico	21.16	337000
19	20	Louisiana	21.18	733000
20	21	Alabama	21.24	797000
21	22	South Dakota	21.25	139000
22	23	Missouri	21.32	996000
23	24	Nevada	21.38	508000
24	25	Massachusetts	21.39	1172000
25	26	Maine	21.61	234000
26	27	South Carolina	21.69	862000
27	28	Indiana	21.83	1109000
28	29	Wisconsin	21.83	982000
29	30	Kentucky	21.91	742000
30	31	Alaska	22.20	117000
31	32	Michigan	22.33	1729000
32	33	Arkansas	22.61	514000
33	34	District of Columbia	22.95	131000
34	35	Colorado	23.16	1028000

	Ranks	State	Percentage	Numbers
35	36	Minnesota	23.23	997000
36	37	Nebraska	23.41	335000
37	38	Montana	23.43	195000
38	39	Wyoming	23.63	103000
39	40	Vermont	23.71	120000
40	41	New Hampshire	23.74	260000
41	42	Arizona	23.89	1339000
42	43	Rhode Island	24.12	202000
43	44	Ohio	24.32	2177000
44	45	Idaho	24.92	333000
45	46	Washington	25.51	1500000
46	47	Oklahoma	25.59	752000
47	48	Kansas	26.02	560000
48	49	West Virginia	26.05	366000
49	50	Oregon	27.33	909000
50	51	Utah	29.68	675000
51	52	National	20.78	52173000

In [27]:
```
# let's see the datatypes of our dataframe
dfmental.dtypes
```

Out[27]:
```
Ranks         object
State         object
Percentage    object
Numbers       object
dtype: object
```

In [28]:
```
# converting the datatypes using function
dfmental1 = dfmental.convert_dtypes()
dfmental1.dtypes
```

Out[28]:
```
Ranks         string
State         string
Percentage    string
Numbers       string
dtype: object
```

In [29]:
```
# converting the datype of Ranks,Percentage,Numbers to numeric datatyp
dfmental1[['Ranks', 'Percentage','Numbers']] = dfmental1[['Ranks', 'Pe
print(dfmental1.dtypes)
```

```
Ranks           int64
State          string
Percentage    float64
Numbers         int64
dtype: object
```

In [30]:
```
# descrbing the dataframe
dfmental1.describe()
```

Out[30]:

	Ranks	Percentage	Numbers
count	52.000000	52.000000	5.200000e+01
mean	26.500000	21.922308	2.006615e+06
std	15.154757	2.578935	7.174289e+06
min	1.000000	17.490000	1.030000e+05
25%	13.750000	20.517500	3.345000e+05
50%	26.500000	21.500000	7.745000e+05
75%	39.250000	23.480000	1.333000e+06
max	52.000000	29.680000	5.217300e+07

In [62]:
```
dfmental1.shape
```

Out[62]: (52, 4)

In [60]:
```
# saving our datframe into csv file named as Mental_Health 2023
dfmental1.to_csv('Csv_files/Mental_Health2023.csv',index = False)
```

In [32]:
```
# Now let's read another csv file of Mass Shootings 2023 and save into
df2 = pd.read_csv('Data/export-14c01ebe-6437-4bd6-8980-5ed47b94e55b.cs
```

In [33]:

```
# printing the head of df2
df2.head()
```

Out[33]:

	Incident ID	Incident Date	State	City Or County	Address	# Victims Injured	# Victims Killed	# Subjects-Suspects Injured	# Subjects-Suspects Killed
0	2594759	May 10, 2023	Colorado	Denver	9600 E Girard Ave	5	0	0	0
1	2590876	May 7, 2023	New Jersey	Newark	300 block of Orange St	3	1	0	0
2	2590669	May 7, 2023	California	Adelanto	10900 block of Bartlett Ave	5	2	0	0
3	2591640	May 7, 2023	California	Thornton	8600 block of W Mokelumne Ave	3	1	0	0
4	2591449	May 7, 2023	Wisconsin	Fond Du Lac	912 Martin Ave	3	1	0	0

In [34]:

```
# printing the tail of df2
df2.tail()
```

Out[34]:

	Incident ID	Incident Date	State	City Or County	Address	# Victims Injured	# Victims Killed	# Subjects-Suspects Injured	Subj Susp I
205	2492601	January 1, 2023	Illinois	Chicago	300 block of E 57th St	3	1	0	
206	2497601	January 1, 2023	Florida	Miami Gardens	NW 171st St and NW 30th Ave	9	0	0	
207	2492611	January 1, 2023	North Carolina	Durham	1000 N Miami Blvd	5	0	0	
208	2493102	January 1, 2023	Pennsylvania	Allentown	1140 E Clair St	4	0	0	
209	2492314	January 1, 2023	Ohio	Columbus	2830 Johnstown Rd	4	1	0	

In [35]:

```
#Let's drop the column Operations because it has all nan values
df2.drop(['Operations'],axis = 1,inplace = True)
df2.head()
```

Out[35]:

	Incident ID	Incident Date	State	City Or County	Address	# Victims Injured	# Victims Killed	# Subjects-Suspects Injured	# Subjects-Suspects Killed
0	2594759	May 10, 2023	Colorado	Denver	9600 E Girard Ave	5	0	0	0
1	2590876	May 7, 2023	New Jersey	Newark	300 block of Orange St	3	1	0	0
2	2590669	May 7, 2023	California	Adelanto	10900 block of Bartlett Ave	5	2	0	0
3	2591640	May 7, 2023	California	Thornton	8600 block of W Mokelumne Ave	3	1	0	0
4	2591449	May 7, 2023	Wisconsin	Fond Du Lac	912 Martin Ave	3	1	0	0

In [36]:

```
# Renaming the columns in df2
newdf2 = df2.rename({'# Victims Injured': 'Victims Injured', '# Victim
newdf2.head()
```

Out[36]:

	Incident ID	Incident Date	State	City Or County	Address	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed
0	2594759	May 10, 2023	Colorado	Denver	9600 E Girard Ave	5	0	0	0
1	2590876	May 7, 2023	New Jersey	Newark	300 block of Orange St	3	1	0	0
2	2590669	May 7, 2023	California	Adelanto	10900 block of Bartlett Ave	5	2	0	0
3	2591640	May 7, 2023	California	Thornton	8600 block of W Mokelumne Ave	3	1	0	0
4	2591449	May 7, 2023	Wisconsin	Fond Du Lac	912 Martin Ave	3	1	0	0

In [37]:
```
# Checking the null values for each column
newdf2.isnull().sum()
```

Out[37]:
```
Incident ID          0
Incident Date        0
State                0
City Or County       0
Address              0
Victims Injured      0
Victims Killed       0
Suspects Injured     0
Suspects Killed      0
Suspects Arrested    0
dtype: int64
```

In [38]:
```
# printing the dattypes for newdf2
newdf2.dtypes
```

Out[38]:
```
Incident ID           int64
Incident Date        object
State                object
City Or County       object
Address              object
Victims Injured       int64
Victims Killed        int64
Suspects Injured      int64
Suspects Killed       int64
Suspects Arrested     int64
dtype: object
```

In [39]:
```
# Converting the datatypes
dfmass = newdf2.convert_dtypes()
dfmass.dtypes
```

Out[39]:
```
Incident ID           Int64
Incident Date        string
State                string
City Or County       string
Address              string
Victims Injured       Int64
Victims Killed        Int64
Suspects Injured      Int64
Suspects Killed       Int64
Suspects Arrested     Int64
dtype: object
```

In [40]:
```
#changing the datatype of Incident ID
dfmass['Incident ID'] = dfmass['Incident ID'].astype(str)

```

In [64]:
```
dfmass.shape
```

Out[64]:
```
(210, 11)
```

In [65]:
```
1 dfmass.describe()
```

Out[65]:

	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	Suspects Arrested	Total Incidents
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.0
mean	3.938095	1.342857	0.047619	0.061905	0.619048	1.0
std	2.907008	1.699021	0.213468	0.241558	1.052506	0.0
min	0.000000	0.000000	0.000000	0.000000	0.000000	1.0
25%	3.000000	0.000000	0.000000	0.000000	0.000000	1.0
50%	4.000000	1.000000	0.000000	0.000000	0.000000	1.0
75%	4.000000	2.000000	0.000000	0.000000	1.000000	1.0
max	32.000000	11.000000	1.000000	1.000000	6.000000	1.0

In [41]:
```
1 # saving our datframe into csv file named as Mental_Health 2023
2 dfmass.to_csv('Csv_files/Mass_Shootings2023.csv',index = False)
```

In [42]:

```
# Grouping our data by state
groupdf = dfmass.groupby('State')
 # Let's print the first entries
# in all the groups formed.
groupdf.first()
```

Out[42]:

State	Incident ID	Incident Date	City Or County	Address	Victims Injured	Victims Killed	Suspects Injured	Su
Alabama	2585253	April 30, 2023	Birmingham	800 block of 48th St N	4	0	0	
Arizona	2573179	April 15, 2023	Phoenix	4617 W Indian School Rd	4	0	0	
Arkansas	2555301	March 26, 2023	Little Rock	3002 Washington St	5	2	0	
California	2590669	May 7, 2023	Adelanto	10900 block of Bartlett Ave	5	2	0	
Colorado	2594759	May 10, 2023	Denver	9600 E Girard Ave	5	0	0	
Connecticut	2577642	April 20, 2023	Hartford	Huntington St and Asylum Ave	3	1	0	
District of Columbia	2578168	April 21, 2023	Washington	500 block of Lebaum St SE	8	0	0	
Florida	2587288	May 2, 2023	Lake Wales	614 Dawnlight Dr	0	4	0	
Georgia	2587386	May 3, 2023	Atlanta	1110 W Peachtree St NW	4	1	0	
Hawaii	2572960	April 15, 2023	Waianae	87131 Kaukamana Rd	3	2	0	
Illinois	2587918	May 3, 2023	Chicago	6300 block of South Calumet Ave	4	0	0	
Indiana	2570985	April 12, 2023	Fort Wayne	3530 Harvester Ave	3	1	0	
Kentucky	2584772	April 30, 2023	Paducah	505 S 8th St	4	0	0	
Louisiana	2592407	May 5, 2023	Bastrop	Welch Ave. and Pruitt St	2	2	0	
Maine	2575621	April 18, 2023	Bowdoin	1459 Augusta Rd	3	4	0	
Maryland	2590940	May 7, 2023	Frostburg	E College Ave	3	1	0	
Massachusetts	2584930	April 30, 2023	Lawrence	5 Royal St	5	1	0	
Michigan	2571648	April 13, 2023	Detroit	13570 block of Penrod St	4	0	1	

State	Incident ID	Incident Date	City Or County	Address	Victims Injured	Victims Killed	Suspects Injured	Su
Minnesota	2555309	March 26, 2023	Minneapolis	5601 Brooklyn Blvd	6	0	0	
Mississippi	2592771	May 6, 2023	Robinsonville (Tunica Resorts)	2669 Kirby Rd	4	0	0	
Missouri	2590617	May 7, 2023	Saint Louis	2100 block of Branch St	2	2	0	
Nevada	2586747	April 30, 2023	Las Vegas (Enterprise)	Bermuda Rd and Pilot Rd	4	0	0	
New Jersey	2590876	May 7, 2023	Newark	300 block of Orange St	3	1	0	
New York	2571130	April 13, 2023	Bronx	1859 Westchester Ave	4	0	0	
North Carolina	2579262	April 22, 2023	Winston Salem (Winston-salem)	1256 Alder St	4	1	0	
Ohio	2589762	May 6, 2023	Columbus	624 N High St	9	0	1	
Oklahoma	2586872	April 30, 2023	Henryetta	14360 Holly Rd	0	6	0	
Oregon	2546495	March 15, 2023	Portland	7900 NE 82nd Ave	2	2	0	
Pennsylvania	2583589	April 28, 2023	Philadelphia	5900 block of Palmetto St	1	3	0	
South Carolina	2584240	April 29, 2023	Columbia	600 Beckman Rd	9	0	0	
Tennessee	2564328	April 3, 2023	Jackson	1600 block of Hollywood Dr	4	0	0	
Texas	2590046	May 6, 2023	Allen	820 W Stacy Rd	7	8	0	
Utah	2495648	January 4, 2023	Cedar City (Enoch)	4923 N Albert Dr	0	7	0	
Virginia	2565404	April 5, 2023	Virginia Beach	700 block of Lake Edward Dr	4	0	0	
Washington	2584711	April 29, 2023	Auburn	2500 block of B St NW	4	0	0	
Wisconsin	2591449	May 7, 2023	Fond Du Lac	912 Martin Ave	3	1	0	

In [43]:

```
#Counting the  number of incidents of mass shooting occur and saving i
dfmass['Total Incidents'] = dfmass['Incident ID'].map(dfmass['Incident
dfmass.head()
```

Out[43]:

	Incident ID	Incident Date	State	City Or County	Address	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed
0	2594759	May 10, 2023	Colorado	Denver	9600 E Girard Ave	5	0	0	0
1	2590876	May 7, 2023	New Jersey	Newark	300 block of Orange St	3	1	0	0
2	2590669	May 7, 2023	California	Adelanto	10900 block of Bartlett Ave	5	2	0	0
3	2591640	May 7, 2023	California	Thornton	8600 block of W Mokelumne Ave	3	1	0	0
4	2591449	May 7, 2023	Wisconsin	Fond Du Lac	912 Martin Ave	3	1	0	0

In [44]:

```
#grouping and doing sum of all the values based on State
massdata = dfmass.groupby(['State'], as_index=False).sum()
massdata.head()
```

Out[44]:

	State	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	Suspects Arrested	Total Incidents
0	Alabama	45	10	0	0	11	4
1	Arizona	11	1	0	0	1	3
2	Arkansas	11	5	0	0	5	3
3	California	64	43	1	1	8	19
4	Colorado	20	3	0	0	0	5

In [45]:

```
#Now stripping the extra space if any in the column State
dfmental1.State = dfmental1.State.str.strip()
massdata.State = massdata.State.str.strip()
```

In [46]:

```
#merging the dataframe dfmental1 and massdata for further evaluation
mergedf = pd.merge(dfmental1.assign(State=dfmental1.State.astype(str)),
         massdata.assign(State=massdata.State.astype(str)),
         how='left', on='State')
#printing the head of dataframe
mergedf.head()
```

Out[46]:

	Ranks	State	Percentage	Numbers	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	Suspects Arrested
0	1	Florida	17.49	2985000	59	25	0	2	9
1	2	Georgia	17.55	1397000	39	8	0	0	9
2	3	Maryland	17.80	822000	25	6	0	0	3
3	4	Hawaii	17.86	189000	3	2	0	0	2
4	5	Texas	17.96	3825000	61	29	3	1	16

In [47]:

```
# dropping the rows having NaN values
mergedf = mergedf.dropna()

# To reset the indices
Finaldf = mergedf.reset_index(drop=True)

# Print the dataframe head and tail
display(Finaldf.head())
display(Finaldf.tail())
```

	Ranks	State	Percentage	Numbers	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	Suspects Arrested
0	1	Florida	17.49	2985000	59	25	0	2	9
1	2	Georgia	17.55	1397000	39	8	0	0	9
2	3	Maryland	17.80	822000	25	6	0	0	3
3	4	Hawaii	17.86	189000	3	2	0	0	2
4	5	Texas	17.96	3825000	61	29	3	1	16

	Ranks	State	Percentage	Numbers	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	Suspe Arrest
31	44	Ohio	24.32	2177000	30	13	1	2	
32	46	Washington	25.51	1500000	7	1	0	0	
33	47	Oklahoma	25.59	752000	3	7	0	1	
34	50	Oregon	27.33	909000	2	2	0	0	
35	51	Utah	29.68	675000	0	7	0	1	

In [48]:
```
#describing the Finaldf
Finaldf.describe()
```

Out[48]:

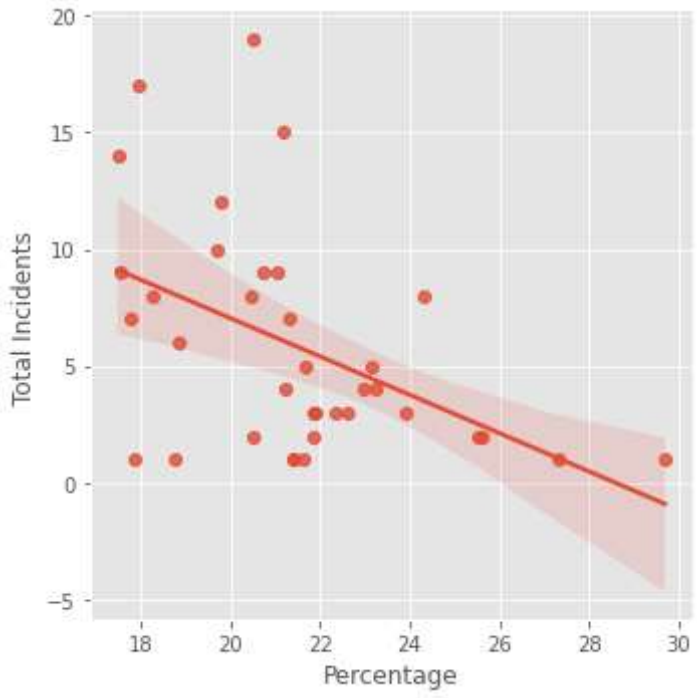
	Ranks	Percentage	Numbers	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	S A
count	36.000000	36.000000	3.600000e+01	36.000000	36.000000	36.000000	36.000000	36
mean	22.972222	21.478611	1.342417e+06	22.972222	7.833333	0.277778	0.361111	:
std	14.915928	2.746672	1.148825e+06	19.498698	8.910668	0.659485	0.592948	3
min	1.000000	17.490000	1.310000e+05	0.000000	0.000000	0.000000	0.000000	0
25%	9.750000	19.770000	7.397500e+05	7.000000	2.000000	0.000000	0.000000	0
50%	23.500000	21.350000	1.012500e+06	19.000000	5.500000	0.000000	0.000000	2
75%	33.250000	22.695000	1.523000e+06	33.500000	10.500000	0.000000	1.000000	5
max	51.000000	29.680000	6.169000e+06	73.000000	43.000000	3.000000	2.000000	16

Data Prediction

In [59]:
```
#converting finaldf into Csv file.
Finaldf.to_csv('Csv_files/Finaldf.csv',index = False)
```

In [50]:
```
#plotting lmplot to see the relationship between Total Incidents of ma
sns.lmplot(x='Percentage',y='Total Incidents',data=Finaldf,fit_reg=Tru
```

Out[50]: <seaborn.axisgrid.FacetGrid at 0x12b58acfca0>



In [51]:

```
#find the correlation for the final dataset
corr = Finaldf.corr()
corr
```

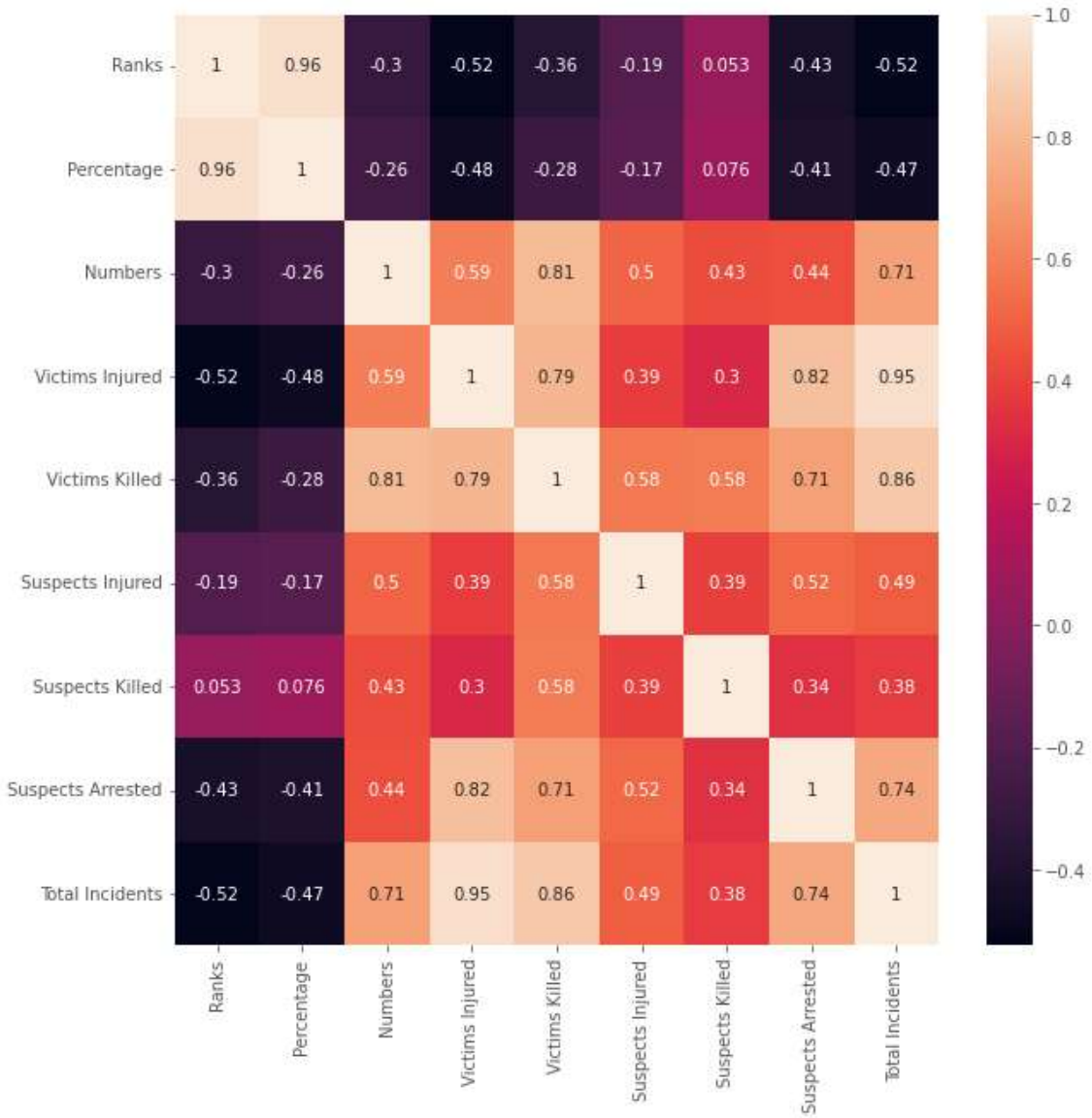
Out[51]:

	Ranks	Percentage	Numbers	Victims Injured	Victims Killed	Suspects Injured	Suspects Killed	Su Ar
Ranks	1.000000	0.957730	-0.302027	-0.518892	-0.364190	-0.193797	0.052854	-0.4
Percentage	0.957730	1.000000	-0.256025	-0.475568	-0.277579	-0.170447	0.075577	-0.4
Numbers	-0.302027	-0.256025	1.000000	0.589884	0.808251	0.500349	0.431662	0.4
Victims Injured	-0.518892	-0.475568	0.589884	1.000000	0.787985	0.385003	0.304851	0.8
Victims Killed	-0.364190	-0.277579	0.808251	0.787985	1.000000	0.576960	0.579515	0.7
Suspects Injured	-0.193797	-0.170447	0.500349	0.385003	0.576960	1.000000	0.393740	0.5
Suspects Killed	0.052854	0.075577	0.431662	0.304851	0.579515	0.393740	1.000000	0.3
Suspects Arrested	-0.427801	-0.409115	0.438010	0.821877	0.705847	0.522165	0.340236	1.0
Total Incidents	-0.523618	-0.465338	0.712711	0.947934	0.862525	0.487817	0.378797	0.7

In [52]:

```
# Plotting the heatmap for corr.
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr,annot = True)
```

Out[52]: <AxesSubplot:>



Conclusion:

- Percentage of adults affected by mental Health is negatively correlated with Total Incidents that means Incidents of Mass Shootings.
- Ranks of state and percentage are positively correlated.

Derive Conclusion

In [53]:
```
#Importing  train_test_split from sklearn.model-selection
from sklearn.model_selection import train_test_split
#Importing LinearRegression from sklearn.linear_model
from sklearn.linear_model import LinearRegression
X = Finaldf[['Percentage','Numbers','Ranks']]
y = Finaldf['Total Incidents']
X_train,X_test,y_train, y_test = train_test_split(X,y,test_size = 0.4,
#Instantiating LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[53]:
```
LinearRegression()
```

In [54]:
```
#printing intercept and coefficient
print(lm.intercept_)
print(lm.coef_)
```

```
-63.495853477704685
[ 3.93527926e+00  1.58874217e-06 -7.50189008e-01]
```

In [55]:
```
#importing metrics from sklearn
from sklearn import metrics
y_pred = lm.predict(X_test)
#printing RMSE
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
5.5886897584993696
```

In [56]:
```
#Importing  train_test_split from sklearn.model-selection
from sklearn.model_selection import train_test_split
#Importing LinearRegression from sklearn.linear_model
from sklearn.linear_model import LinearRegression
X = Finaldf[['Percentage','Numbers']]
y = Finaldf['Total Incidents']
X_train,X_test,y_train, y_test = train_test_split(X,y,test_size = 0.4,
#Instantiating LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[56]:
```
LinearRegression()
```

In [57]:
```
from sklearn import metrics
y_pred = lm.predict(X_test)
#printing RMSE
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
2.4894156512025534
```

Conclusion:

- The RMSE decreased when we removed Ranks From the model
- Error is something we want to minimize
- Thus it is unlikely that this feature is useful for Predicting AMI and should be removed from the model.

In [ ]:

```
1
```