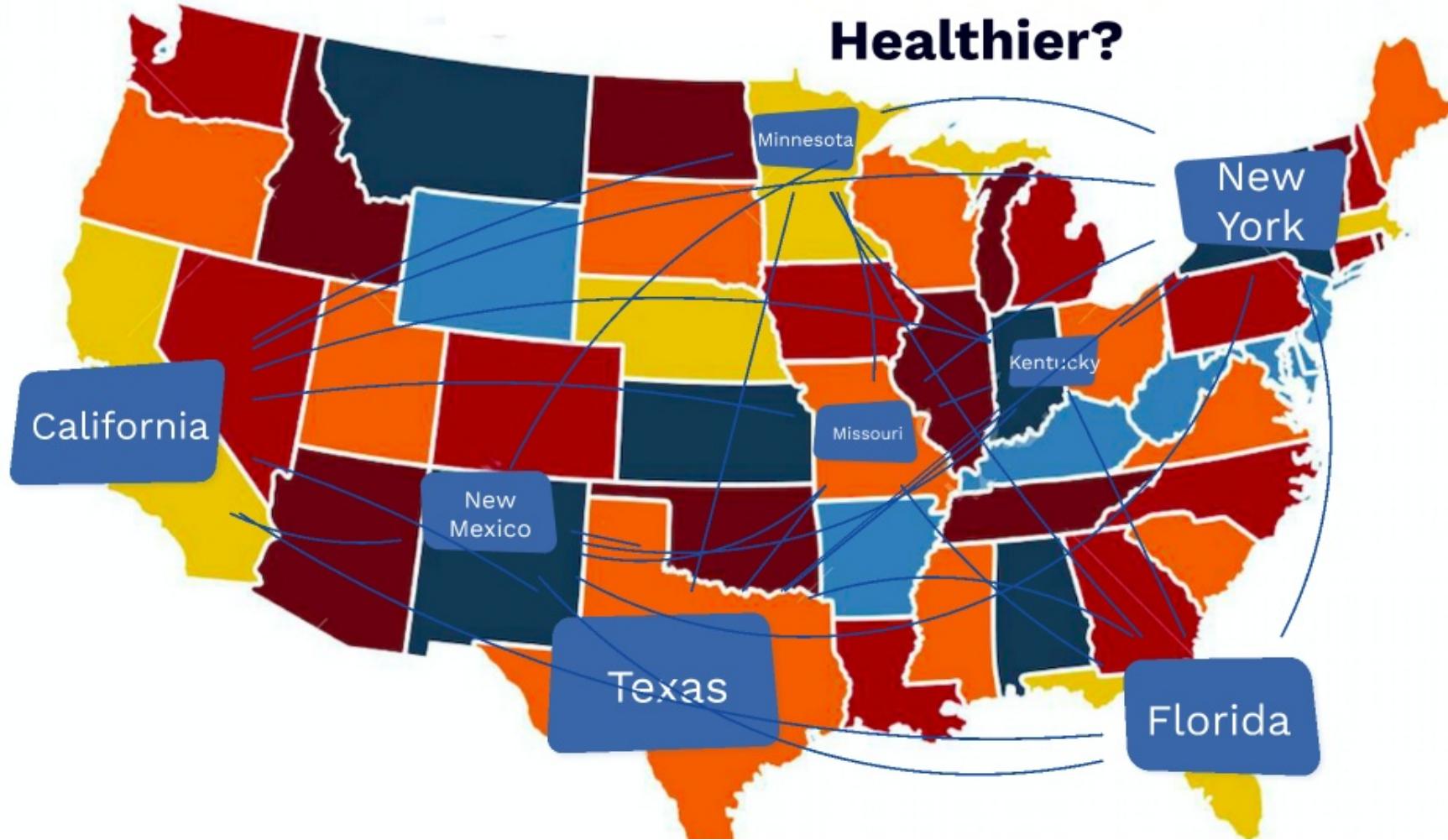


Which State is Healthier?



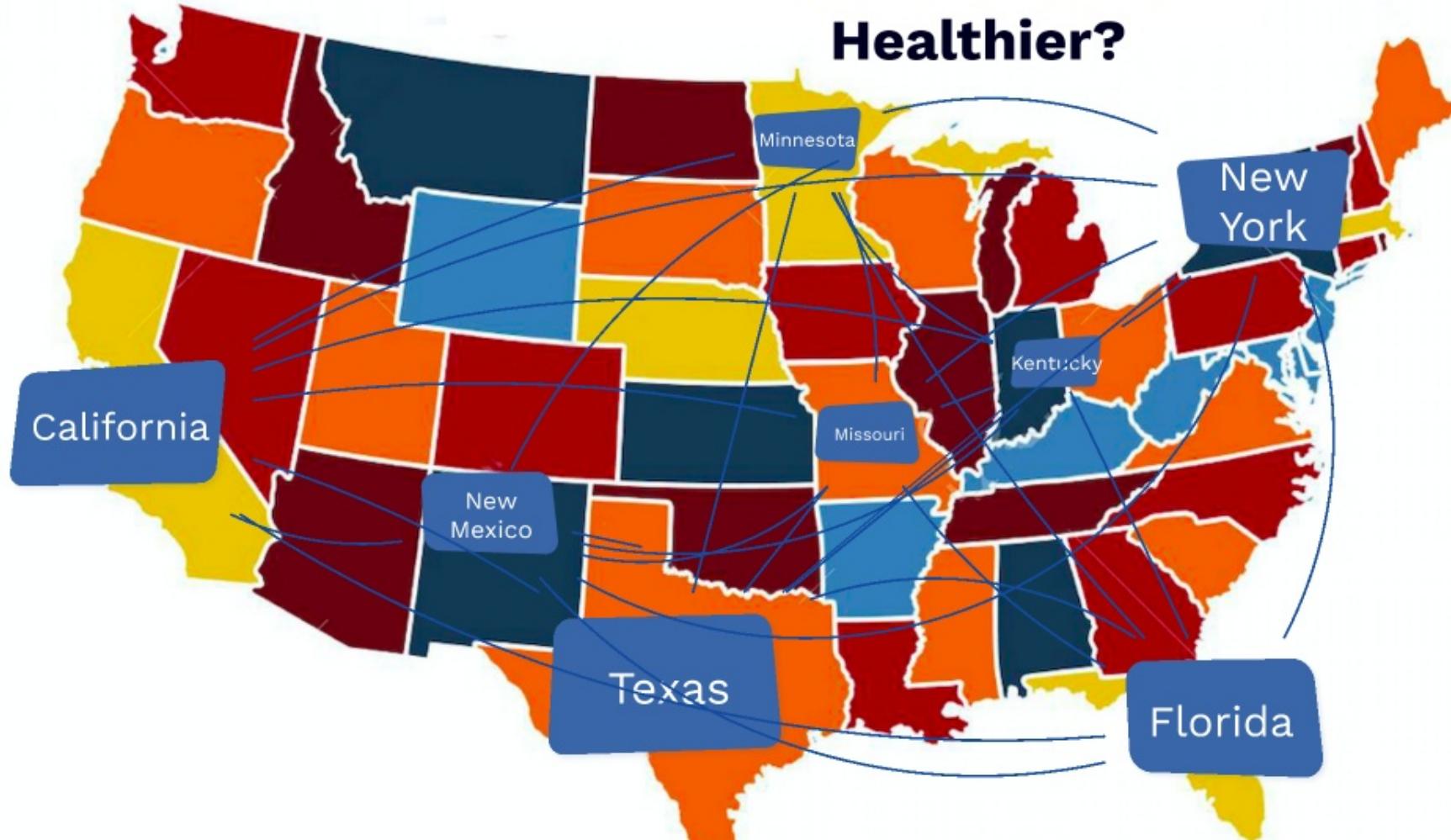
Chronic Diseases in the United States

- Six in ten Adults in the US have at least one chronic disease, and four in ten adults have more than two diseases
- 60% of the worldwide death rate is due to chronic disease



Study the possible relationship between GDP and Chronic Diseases

Which State is Healthier?



Data Cleaning

Prepare the data for chart making

- Using a list comprehension filtered out all of the rows not matching the criteria given in the conditional. So this made it easy to keep only the information we need.
- The challenge with cleaning the data was the use of a regular expression. Since the data had extra parentheses in the latitude and longitude coordinates. So it became necessary to filter these characters out using a simple regular expression, which are more convenient to be used during the G-map visualizing part.



States GDP Data Cleaning

Step 1

- Import the dirty CSV
- Remove the extra column by using the **DEL** function

```
In [1]: 1 import pandas as pd  
2  
In [2]: 1 raw_path = r'C:\Data\Datasets\gdp_by_states.csv'  
2  
3 raw_df = pd.read_csv(raw_path)  
4 raw_df.head(5)  
  
Out[2]:  
    Pts    Ann  BGD  BWE  BWD  BWT  
0  4  United States  10232  10000  10000  10000  
1  2000  China  4570  4500  4500  4500  
2  2000  India  4570  4500  4500  4500  
3  2000  Russia  4570  4500  4500  4500  
4  2000  Germany  4570  4500  4500  4500  
  
In [3]: 1 raw_df.drop('Pts')  
2 raw_df.head()
```

Step 2

- Use **DROPNA** function to remove the incomplete data, and rename the column

```
In [4]: 1 raw_df = raw_df.dropna()  
2  
In [5]: 1 renamed_df = raw_df.rename(columns={  
2     'Ann': 'State'  
3 })  
4  
In [6]: 1 renamed_df.head()  
  
Out[6]:  
    State  BGD  BWE  BWD  BWT  
0  United States  10232  10000  10000  10000  
1  Australia  4570  4500  4500  4500  
2  India  4570  4500  4500  4500  
3  Russia  4570  4500  4500  4500  
4  Germany  4570  4500  4500  4500
```

Step 3

- Use **DROP** and **.ILOC** functions to remove unwanted rows

```
In [7]: 1 cleaned_df_drops  
2  
In [8]: 1 raw_df = cleaned_df_drops[10:, 11:14, 15:18, 19:21]  
2  
In [9]: 1 raw_df  
2  
Out[9]:  
    State  BGD  BWE  BWD  BWT  
0  Asia  4570  4500  4500  4500  
1  Africa  4570  4500  4500  4500  
2  Americas  4570  4500  4500  4500  
3  Oceania  4570  4500  4500  4500  
4  Europe  4570  4500  4500  4500  
5  Middle East  4570  4500  4500  4500  
6  Latin America  4570  4500  4500  4500  
7  Central America  4570  4500  4500  4500  
8  South America  4570  4500  4500  4500  
9  North America  4570  4500  4500  4500  
10  Australia  4570  4500  4500  4500  
11  Russia  4570  4500  4500  4500  
12  India  4570  4500  4500  4500  
13  China  4570  4500  4500  4500  
14  United States  4570  4500  4500  4500  
15  Germany  4570  4500  4500  4500  
16  France  4570  4500  4500  4500  
17  United Kingdom  4570  4500  4500  4500  
18  Canada  4570  4500  4500  4500  
19  Japan  4570  4500  4500  4500  
20  Brazil  4570  4500  4500  4500  
21  Australia  4570  4500  4500  4500  
22  Mexico  4570  4500  4500  4500  
23  South Korea  4570  4500  4500  4500  
24  Italy  4570  4500  4500  4500  
25  Spain  4570  4500  4500  4500  
26  Switzerland  4570  4500  4500  4500  
27  Sweden  4570  4500  4500  4500  
28  Norway  4570  4500  4500  4500  
29  Netherlands  4570  4500  4500  4500  
30  Austria  4570  4500  4500  4500  
31  Belgium  4570  4500  4500  4500  
32  Portugal  4570  4500  4500  4500  
33  Greece  4570  4500  4500  4500  
34  Turkey  4570  4500  4500  4500  
35  Poland  4570  4500  4500  4500  
36  Hungary  4570  4500  4500  4500  
37  Czech Republic  4570  4500  4500  4500  
38  Slovakia  4570  4500  4500  4500  
39  Montenegro  4570  4500  4500  4500  
40  Malta  4570  4500  4500  4500  
41  Slovenia  4570  4500  4500  4500  
42  Estonia  4570  4500  4500  4500  
43  Lithuania  4570  4500  4500  4500  
44  Cyprus  4570  4500  4500  4500  
45  Romania  4570  4500  4500  4500  
46  Bulgaria  4570  4500  4500  4500  
47  North Macedonia  4570  4500  4500  4500  
48  Serbia  4570  4500  4500  4500  
49  Montenegro  4570  4500  4500  4500  
50  Albania  4570  4500  4500  4500  
51  Kosovo  4570  4500  4500  4500  
52  Moldova  4570  4500  4500  4500  
53  Georgia  4570  4500  4500  4500  
54  Armenia  4570  4500  4500  4500  
55  Azerbaijan  4570  4500  4500  4500  
56  Turkmenistan  4570  4500  4500  4500  
57  Kyrgyzstan  4570  4500  4500  4500  
58  Tajikistan  4570  4500  4500  4500  
59  Uzbekistan  4570  4500  4500  4500  
60  Kazakhstan  4570  4500  4500  4500  
61  Belarus  4570  4500  4500  4500  
62  Russia  4570  4500  4500  4500  
63  Mongolia  4570  4500  4500  4500  
64  Turkmenistan  4570  4500  4500  4500  
65  Kyrgyzstan  4570  4500  4500  4500  
66  Tajikistan  4570  4500  4500  4500  
67  Uzbekistan  4570  4500  4500  4500  
68  Kazakhstan  4570  4500  4500  4500  
69  Belarus  4570  4500  4500  4500  
70  Russia  4570  4500  4500  4500  
71  Mongolia  4570  4500  4500  4500  
72  Turkmenistan  4570  4500  4500  4500  
73  Kyrgyzstan  4570  4500  4500  4500  
74  Tajikistan  4570  4500  4500  4500  
75  Uzbekistan  4570  4500  4500  4500  
76  Kazakhstan  4570  4500  4500  4500  
77  Belarus  4570  4500  4500  4500  
78  Russia  4570  4500  4500  4500  
79  Mongolia  4570  4500  4500  4500  
80  Turkmenistan  4570  4500  4500  4500  
81  Kyrgyzstan  4570  4500  4500  4500  
82  Tajikistan  4570  4500  4500  4500  
83  Uzbekistan  4570  4500  4500  4500  
84  Kazakhstan  4570  4500  4500  4500  
85  Belarus  4570  4500  4500  4500  
86  Russia  4570  4500  4500  4500  
87  Mongolia  4570  4500  4500  4500  
88  Turkmenistan  4570  4500  4500  4500  
89  Kyrgyzstan  4570  4500  4500  4500  
90  Tajikistan  4570  4500  4500  4500  
91  Uzbekistan  4570  4500  4500  4500  
92  Kazakhstan  4570  4500  4500  4500  
93  Belarus  4570  4500  4500  4500  
94  Russia  4570  4500  4500  4500  
95  Mongolia  4570  4500  4500  4500  
96  Turkmenistan  4570  4500  4500  4500  
97  Kyrgyzstan  4570  4500  4500  4500  
98  Tajikistan  4570  4500  4500  4500  
99  Uzbekistan  4570  4500  4500  4500  
100  Kazakhstan  4570  4500  4500  4500  
101  Belarus  4570  4500  4500  4500  
102  Russia  4570  4500  4500  4500  
103  Mongolia  4570  4500  4500  4500  
104  Turkmenistan  4570  4500  4500  4500  
105  Kyrgyzstan  4570  4500  4500  4500  
106  Tajikistan  4570  4500  4500  4500  
107  Uzbekistan  4570  4500  4500  4500  
108  Kazakhstan  4570  4500  4500  4500  
109  Belarus  4570  4500  4500  4500  
110  Russia  4570  4500  4500  4500  
111  Mongolia  4570  4500  4500  4500  
112  Turkmenistan  4570  4500  4500  4500  
113  Kyrgyzstan  4570  4500  4500  4500  
114  Tajikistan  4570  4500  4500  4500  
115  Uzbekistan  4570  4500  4500  4500  
116  Kazakhstan  4570  4500  4500  4500  
117  Belarus  4570  4500  4500  4500  
118  Russia  4570  4500  4500  4500  
119  Mongolia  4570  4500  4500  4500  
120  Turkmenistan  4570  4500  4500  4500  
121  Kyrgyzstan  4570  4500  4500  4500  
122  Tajikistan  4570  4500  4500  4500  
123  Uzbekistan  4570  4500  4500  4500  
124  Kazakhstan  4570  4500  4500  4500  
125  Belarus  4570  4500  4500  4500  
126  Russia  4570  4500  4500  4500  
127  Mongolia  4570  4500  4500  4500  
128  Turkmenistan  4570  4500  4500  4500  
129  Kyrgyzstan  4570  4500  4500  4500  
130  Tajikistan  4570  4500  4500  4500  
131  Uzbekistan  4570  4500  4500  4500  
132  Kazakhstan  4570  4500  4500  4500  
133  Belarus  4570  4500  4500  4500  
134  Russia  4570  4500  4500  4500  
135  Mongolia  4570  4500  4500  4500  
136  Turkmenistan  4570  4500  4500  4500  
137  Kyrgyzstan  4570  4500  4500  4500  
138  Tajikistan  4570  4500  4500  4500  
139  Uzbekistan  4570  4500  4500  4500  
140  Kazakhstan  4570  4500  4500  4500  
141  Belarus  4570  4500  4500  4500  
142  Russia  4570  4500  4500  4500  
143  Mongolia  4570  4500  4500  4500  
144  Turkmenistan  4570  4500  4500  4500  
145  Kyrgyzstan  4570  4500  4500  4500  
146  Tajikistan  4570  4500  4500  4500  
147  Uzbekistan  4570  4500  4500  4500  
148  Kazakhstan  4570  4500  4500  4500  
149  Belarus  4570  4500  4500  4500  
150  Russia  4570  4500  4500  4500  
151  Mongolia  4570  4500  4500  4500  
152  Turkmenistan  4570  4500  4500  4500  
153  Kyrgyzstan  4570  4500  4500  4500  
154  Tajikistan  4570  4500  4500  4500  
155  Uzbekistan  4570  4500  4500  4500  
156  Kazakhstan  4570  4500  4500  4500  
157  Belarus  4570  4500  4500  4500  
158  Russia  4570  4500  4500  4500  
159  Mongolia  4570  4500  4500  4500  
160  Turkmenistan  4570  4500  4500  4500  
161  Kyrgyzstan  4570  4500  4500  4500  
162  Tajikistan  4570  4500  4500  4500  
163  Uzbekistan  4570  4500  4500  4500  
164  Kazakhstan  4570  4500  4500  4500  
165  Belarus  4570  4500  4500  4500  
166  Russia  4570  4500  4500  4500  
167  Mongolia  4570  4500  4500  4500  
168  Turkmenistan  4570  4500  4500  4500  
169  Kyrgyzstan  4570  4500  4500  4500  
170  Tajikistan  4570  4500  4500  4500  
171  Uzbekistan  4570  4500  4500  4500  
172  Kazakhstan  4570  4500  4500  4500  
173  Belarus  4570  4500  4500  4500  
174  Russia  4570  4500  4500  4500  
175  Mongolia  4570  4500  4500  4500  
176  Turkmenistan  4570  4500  4500  4500  
177  Kyrgyzstan  4570  4500  4500  4500  
178  Tajikistan  4570  4500  4500  4500  
179  Uzbekistan  4570  4500  4500  4500  
180  Kazakhstan  4570  4500  4500  4500  
181  Belarus  4570  4500  4500  4500  
182  Russia  4570  4500  4500  4500  
183  Mongolia  4570  4500  4500  4500  
184  Turkmenistan  4570  4500  4500  4500  
185  Kyrgyzstan  4570  4500  4500  4500  
186  Tajikistan  4570  4500  4500  4500  
187  Uzbekistan  4570  4500  4500  4500  
188  Kazakhstan  4570  4500  4500  4500  
189  Belarus  4570  4500  4500  4500  
190  Russia  4570  4500  4500  4500  
191  Mongolia  4570  4500  4500  4500  
192  Turkmenistan  4570  4500  4500  4500  
193  Kyrgyzstan  4570  4500  4500  4500  
194  Tajikistan  4570  4500  4500  4500  
195  Uzbekistan  4570  4500  4500  4500  
196  Kazakhstan  4570  4500  4500  4500  
197  Belarus  4570  4500  4500  4500  
198  Russia  4570  4500  4500  4500  
199  Mongolia  4570  4500  4500  4500  
200  Turkmenistan  4570  4500  4500  4500  
201  Kyrgyzstan  4570  4500  4500  4500  
202  Tajikistan  4570  4500  4500  4500  
203  Uzbekistan  4570  4500  4500  4500  
204  Kazakhstan  4570  4500  4500  4500  
205  Belarus  4570  4500  4500  4500  
206  Russia  4570  4500  4500  4500  
207  Mongolia  4570  4500  4500  4500  
208  Turkmenistan  4570  4500  4500  4500  
209  Kyrgyzstan  4570  4500  4500  4500  
210  Tajikistan  4570  4500  4500  4500  
211  Uzbekistan  4570  4500  4500  4500  
212  Kazakhstan  4570  4500  4500  4500  
213  Belarus  4570  4500  4500  4500  
214  Russia  4570  4500  4500  4500  
215  Mongolia  4570  4500  4500  4500  
216  Turkmenistan  4570  4500  4500  4500  
217  Kyrgyzstan  4570  4500  4500  4500  
218  Tajikistan  4570  4500  4500  4500  
219  Uzbekistan  4570  4500  4500  4500  
220  Kazakhstan  4570  4500  4500  4500  
221  Belarus  4570  4500  4500  4500  
222  Russia  4570  4500  4500  4500  
223  Mongolia  4570  4500  4500  4500  
224  Turkmenistan  4570  4500  4500  4500  
225  Kyrgyzstan  4570  4500  4500  4500  
226  Tajikistan  4570  4500  4500  4500  
227  Uzbekistan  4570  4500  4500  4500  
228  Kazakhstan  4570  4500  4500  4500  
229  Belarus  4570  4500  4500  4500  
230  Russia  4570  4500  4500  4500  
231  Mongolia  4570  4500  4500  4500  
232  Turkmenistan  4570  4500  4500  4500  
233  Kyrgyzstan  4570  4500  4500  4500  
234  Tajikistan  4570  4500  4500  4500  
235  Uzbekistan  4570  4500  4500  4500  
236  Kazakhstan  4570  4500  4500  4500  
237  Belarus  4570  4500  4500  4500  
238  Russia  4570  4500  4500  4500  
239  Mongolia  4570  4500  4500  4500  
240  Turkmenistan  4570  4500  4500  4500  
241  Kyrgyzstan  4570  4500  4500  4500  
242  Tajikistan  4570  4500  4500  4500  
243  Uzbekistan  4570  4500  4500  4500  
244  Kazakhstan  4570  4500  4500  4500  
245  Belarus  4570  4500  4500  4500  
246  Russia  4570  4500  4500  4500  
247  Mongolia  4570  4500  4500  4500  
248  Turkmenistan  4570  4500  4500  4500  
249  Kyrgyzstan  4570  4500  4500  4500  
250  Tajikistan  4570  4500  4500  4500  
251  Uzbekistan  4570  4500  4500  4500  
252  Kazakhstan  4570  4500  4500  4500  
253  Belarus  4570  4500  4500  4500  
254  Russia  4570  4500  4500  4500  
255  Mongolia  4570  4500  4500  4500  
256  Turkmenistan  4570  4500  4500  4500  
257  Kyrgyzstan  4570  4500  4500  4500  
258  Tajikistan  4570  4500  4500  4500  
259  Uzbekistan  4570  4500  4500  4500  
260  Kazakhstan  4570  4500  4500  4500  
261  Belarus  4570  4500  4500  4500  
262  Russia  4570  4500  4500  4500  
263  Mongolia  4570  4500  4500  4500  
264  Turkmenistan  4570  4500  4500  4500  
265  Kyrgyzstan  4570  4500  4500  4500  
266  Tajikistan  4570  4500  4500  4500  
267  Uzbekistan  4570  4500  4500  4500  
268  Kazakhstan  4570  4500  4500  4500  
269  Belarus  4570  4500  4500  4500  
270  Russia  4570  4500  4500  4500  
271  Mongolia  4570  4500  4500  4500  
272  Turkmenistan  4570  4500  4500  4500  
273  Kyrgyzstan  4570  4500  4500  4500  
274  Tajikistan  4570  4500  4500  4500  
275  Uzbekistan  4570  4500  4500  4500  
276  Kazakhstan  4570  4500  4500  4500  
277  Belarus  4570  4500  4500  4500  
278  Russia  4570  4500  4500  4500  
279  Mongolia  4570  4500  4500  4500  
280  Turkmenistan  4570  4500  4500  4500  
281  Kyrgyzstan  4570  4500  4500  4500  
282  Tajikistan  4570  4500  4500  4500  
283  Uzbekistan  4570  4500  4500  4500  
284  Kazakhstan  4570  4500  4500  4500  
285  Belarus  4570  4500  4500  4500  
286  Russia  4570  4500  4500  4500  
287  Mongolia  4570  4500  4500  4500  
288  Turkmenistan  4570  4500  4500  4500  
289  Kyrgyzstan  4570  4500  4500  4500  
290  Tajikistan  4570  4500  4500  4500  
291  Uzbekistan  4570  4500  4500  4500  
292  Kazakhstan  4570  4500  4500  4500  
293  Belarus  4570  4500  4500  4500  
294  Russia  4570  4500  4500  4500  
295  Mongolia  4570  4500  4500  4500  
296  Turkmenistan  4570  4500  4500  4500  
297  Kyrgyzstan  4570  4500  4500  4500  
298  Tajikistan  4570  4500  4500  4500  
299  Uzbekistan  4570  4500  4500  4500  
300  Kazakhstan  4570  4500  4500  4500  
301  Belarus  4570  4500  4500  4500  
302  Russia  4570  4500  4500  4500  
303  Mongolia  4570  4500  4500  4500  
304  Turkmenistan  4570  4500  4500  4500  
305  Kyrgyzstan  4570  4500  4500  4500  
306  Tajikistan  4570  4500  4500  4500  
307  Uzbekistan  4570  4500  4500  4500  
308  Kazakhstan  4570  4500  4500  4500  
309  Belarus  4570  4500  4500  4500  
310  Russia  4570  4500  4500  4500  
311  Mongolia  4570  4500  4500  4500  
312  Turkmenistan  4570  4500  4500  4500  
313  Kyrgyzstan  4570  4500  4500  4500  
314  Tajikistan  4570  4500  4500  4500  
315  Uzbekistan  4570  4500  4500  4500  
316  Kazakhstan  4570  4500  4500  4500  
317  Belarus  4570  4500  4500  4500  
318  Russia  4570  4500  4500  4500  
319  Mongolia  4570  4500  4500  4500  
320  Turkmenistan  4570  4500  4500  4500  
321  Kyrgyzstan  4570  4500  4500  4500  
322  Tajikistan  4570  4500  4500  4500  
323  Uzbekistan  4570  4500  4500  4500  
324  Kazakhstan  4570  4500  4500  4500  
325  Belarus  4570  4500  4500  4500  
326  Russia  4570  4500  4500  4500  
327  Mongolia  4570  4500  4500  4500  
328  Turkmenistan  4570  4
```

```
In [2]: 1 import pandas as pd
```

```
In [3]: 1 csv_path = ("GDP_Data/bea-gdp-by-state.csv")
2
3 csv_df = pd.read_csv(csv_path)
4 csv_df.head()
```

Out[3]:

	Fips	Area	2013	2014	2015	2016	2017
0	0	United States	48534	49329	50301	50660	51337
1	1000	Alabama	36674	36473	36818	37158	37508
2	2000	Alaska	69711	67179	65971	63304	63610
3	4000	Arizona	38352	38534	38787	38940	39583
4	5000	Arkansas	35888	36265	36295	36502	36714

```
In [4]: 1 del csv_df["Fips"]
2 csv_df.head()
```

States GDP Data Cleaning

Step 1

- Import the dirty CSV
 - Remove the extra column by using the **DEL function**

```
ls [1]: |  __import pandas as pd

In [2]: |  new_path = os.path.join('D:\Data\raw\pdp-by-state.csv')

In [3]: |  new_df = pd.read_csv(new_path)

In [4]: |  new_df.head()

Out[4]:   Pds    Ann      Avg      Min      Max      Std
0  1.0  1000000000  800000000  500000000  2000000000  1000000000
1  1000000000  800000000  500000000  2000000000  1000000000
2  2000000000  800000000  500000000  2000000000  1000000000
3  3000000000  800000000  500000000  2000000000  1000000000
4  4000000000  800000000  500000000  2000000000  1000000000

In [5]: |  del new_df[0]
```

Step 2

- Use **DROPNA** function to remove the incomplete data, and rename the column

Step 3

- Use **DROP** and **.ILOC** functions to remove unwanted rows

```
In [5]: 1 csv_df = csv_df.dropna(how="any")
```

```
In [6]: 1 renamed_df = csv_df.rename(columns={  
2     "Area": "State"  
3 })
```

```
In [7]: 1 renamed_df.head()
```

Out[7]:

	State	2013	2014	2015	2016	2017
0	United States	48534	49329	50301	50660	51337
1	Alabama	36674	36473	36818	37158	37508
2	Alaska	69711	67179	65971	63304	63610
3	Arizona	38352	38534	38787	38940	39583
4	Arkansas	35888	36265	36295	36502	36714

States GDP Data Cleaning

Step 1

- Import the dirty CSV
- Remove the extra column by using the **DEL** function

```
In [1]: 1 import pandas as pd  
2  
In [2]: 1 my_path = r'C:\Users\maximiliano-perez\Downloads\us_gdp.csv'  
2  
3 my_df = pd.read_csv(my_path)  
4  
5 my_df.head()
```

Year	Actual	2010	2011	2012	2013	2014
1	4 United States	102400	103200	103600	104200	104700
2	2009	101200	101500	101800	102100	102400
3	2010	101200	101500	101800	102100	102400
4	2011	101200	101500	101800	102100	102400
5	2012	101200	101500	101800	102100	102400

```
In [3]: 1 my_df.drop('Year')
```

```
In [4]: 1 my_df.head()
```

Step 2

- Use **DROPNA** function to remove the incomplete data, and rename the column

```
In [5]: 1 my_df = my_df.dropna()  
2  
In [6]: 1 renamed_df = my_df.rename(columns={  
2     'Year': 'State'  
3 })  
4  
In [7]: 1 renamed_df.head()
```

State	2010	2011	2012	2013	2014
United States	102400	103200	103600	104200	104700
Alaska	101200	101500	101800	102100	102400
Hawaii	101200	101500	101800	102100	102400
Arizona	101200	101500	101800	102100	102400
Illinois	101200	101500	101800	102100	102400

Step 3

- Use **DROP** and **.ILOC** functions to remove unwanted rows

```
In [8]: 1 renamed_df_drops
```

Date	2010	2011	2012	2013	2014
1	United States	102400	103200	103600	104200
2	Alaska	101200	101500	101800	102100
3	Hawaii	101200	101500	101800	102100
4	Arizona	101200	101500	101800	102100
5	Illinois	101200	101500	101800	102100

```
In [9]: 1 new_df = renamed_df_drops[10:-51, :].reset_index(drop=True)
```

```
In [10]: 1 new_df
```

Date	2010	2011	2012	2013	2014
1	United States	102400	103200	103600	104200
2	Alaska	101200	101500	101800	102100
3	Hawaii	101200	101500	101800	102100
4	Arizona	101200	101500	101800	102100
5	Illinois	101200	101500	101800	102100
6	Pennsylvania	101200	101500	101800	102100
7	Michigan	101200	101500	101800	102100
8	New England	101200	101500	101800	102100
9	Midwest	101200	101500	101800	102100
10	South	101200	101500	101800	102100
11	Southwest	101200	101500	101800	102100
12	West	101200	101500	101800	102100

```
In [11]: 1 renamed_df = new_df.drop([10, 11, 12, 13, 14, 15, 16, 17], axis=0)
```

```
In [12]: 1 renamed_df
```

```
In [8]: 1 renamed_df.dtypes
```

```
Out[8]: State    object
2013     int64
2014     int64
2015     int64
2016     int64
2017     int64
dtype: object
```

```
In [17]: 1 new_df = renamed_df.drop([0], axis = 0)
2 new_df.head()
```

```
Out[17]:
```

	State	2013	2014	2015	2016	2017
1	Alabama	36674	36473	36818	37158	37508
2	Alaska	69711	67179	65971	63304	63610
3	Arizona	38352	38534	38787	38940	39583
4	Arkansas	35888	36265	36295	36502	36714
5	California	53838	55571	57637	58974	60359

49	West Virginia	35772	36017	36233	36155	37353
50	Wisconsin	45895	46456	47268	48063	48666
51	Wyoming	60806	60853	61304	59327	61091
52	New England	56689	57068	58477	58882	59637
53	Mideast	57659	58353	59364	59756	60421
54	Great Lakes	46076	47035	47644	48276	49034
55	Plains	48092	49005	49473	49919	50145
56	Southeast	40543	40910	41547	41867	42300
57	Southwest	48205	49119	50214	49320	49902
58	Rocky Mountain	45634	46727	47610	47653	48398
59	Far West	53029	54362	56166	57319	58589

```
In [35]: 1 cleaned_df = new_df.drop([52, 53, 54, 55, 56, 57, 58, 59], axis = 0)
          2 cleaned_df
```

States GDP Data Cleaning

Step 1

- Import the dirty CSV
- Remove the extra column by using the **DEL** function

```
In [1]: 1 import pandas as pd  
2  
In [2]: 1 my_path = r'C:\Users\maximiliano-perez\Downloads\us_gdp.csv'  
2  
3 my_df = pd.read_csv(my_path)  
4  
5 my_df.head()
```

Year	Actual	2010	2011	2012	2013	2014
1	4 United States	102400	102400	102400	102400	102400
2	2009	102400	102400	102400	102400	102400
3	2010	102400	102400	102400	102400	102400
4	2011	102400	102400	102400	102400	102400
5	2012	102400	102400	102400	102400	102400

```
In [3]: 1 my_df.drop('Year')
```

```
In [4]: 1 my_df.head()
```

Step 2

- Use **DROPNA** function to remove the incomplete data, and rename the column

```
In [5]: 1 my_df = my_df.dropna()  
2  
In [6]: 1 renamed_df = my_df.rename(columns={  
2     'Year': 'State'  
3 })  
4  
In [7]: 1 renamed_df.head()
```

State	2010	2011	2012	2013	2014
United States	102400	102400	102400	102400	102400
Alaska	102400	102400	102400	102400	102400
Hawaii	102400	102400	102400	102400	102400
Arizona	102400	102400	102400	102400	102400
Illinois	102400	102400	102400	102400	102400

Step 3

- Use **DROP** and **.ILOC** functions to remove unwanted rows

```
In [8]: 1 renamed_df_drops
```

Date	2010	2011	2012	2013	2014
1	Alabama	102400	102400	102400	102400
2	Alaska	102400	102400	102400	102400
3	Arizona	102400	102400	102400	102400
4	Arkansas	102400	102400	102400	102400
5	California	102400	102400	102400	102400

```
In [9]: 1 new_df = renamed_df_drops[10:, 11:16]
```

Date	2010	2011	2012	2013	2014
1	Alabama	102400	102400	102400	102400
2	Alaska	102400	102400	102400	102400
3	Arizona	102400	102400	102400	102400
4	Arkansas	102400	102400	102400	102400
5	California	102400	102400	102400	102400

```
In [10]: 1 renamed_df = new_df.drop([10, 11, 12, 13, 14, 15], axis=0)
```

```
In [11]: 1 renamed_df
```

Chronical Diseases Data Cleaning

Step 2

- importing the CSV into a data frame
 - Debugging the low memory error by using the LOW_MEMORY =FALSE Boolean

Debt Type	Debt Holder	Debt Type	Debt Holder	Debt Type	Debt Holder	Debt Type	Debt Holder	Debt Type	Debt Holder
B	2010	2010	US	United States	2007B	Autor	Debt payments and capital expenditure	Net	N
E	2010	2010	US	United States	2007B	Autor	Debt payments and capital expenditure	Net	N
I	2010	2010	US	United States	2007B	Autor	Debt payments and capital expenditure	Net	N

Step 2

- deleted unwanted columns using the DEL function
 - followed that up with a DROPNAN function that cleaned up the empty data and corresponding rows

Step 3

- then used the ASTYPE function to convert data in certain columns to floats

```

35 [101] : L_CPY_01_01YP00
    (var[1]) : NameMark           : string
    (var[2]) : Inserted           : string
    (var[3]) : LocalNameMark     : string
    (var[4]) : LocalNameNmbr     : string
    (var[5]) : Topic              : string
    (var[6]) : ObjType            : string
    (var[7]) : DataCategory      : string
    (var[8]) : DataCategoryNmbr  : string
    (var[9]) : Inserted          : string
    (var[10]) : LocalNameCategory: string
    (var[11]) : LocalNameNmbr    : string
    (var[12]) : Longitude          : string
    (var[13]) : Latitude           : string

36 [102] : L_CPY_01_01YP00
    (var[1]) : var[4][1].DataValue : var[4][1].DataValue : string

```

```
In [31]: 1 import pandas as pd
```

```
In [32]: 1 csv_path = ("../U.S._Chronic_Disease_Indicators__CDI_.csv")
2
3 csv_df = pd.read_csv(csv_path)
4 low_memory=False
5 csv_df
```

```
/Users/Pariah/anaconda3/envs/PythonData3/lib/python3.7/site-packages/IPython/core/interactive
shell.py:3049: DtypeWarning: Columns (7,10,18,19,20,21,23,30,31,32,33) have mixed types. Spec
ify dtype option on import or set low_memory=False.
    interactivity=interactivity, compiler=compiler, result=result)
```

Out[32]:

	YearStart	YearEnd	LocationAbbr	LocationDesc	DataSource	Topic	Question	Response	DataValueUnit	DataValue
0	2016	2016	US	United States	BRFSS	Alcohol	Binge drinking prevalence among adults aged >=...	NaN	%	Pr
1	2016	2016	AL	Alabama	BRFSS	Alcohol	Binge drinking prevalence among	NaN	%	Pr

Chronical Diseases Data Cleaning

Step 2

- importing the CSV into a data frame
 - Debugging the low memory error by using the LOW_MEMORY =FALSE Boolean

Step 2

- deleted unwanted columns using the DEL function
 - followed that up with a DROPNA function that cleaned up the empty data and corresponding rows

Step 3

- then used the ASTYPE function to convert data in certain columns to floats

```

33 (E11) : - OFP-41_OFPB40

  var [D1] {
    Transaction          : 0x0001
    Identifier           : 0x0004
    LocalTableId        : 0x0005
    LocalTableNumber     : 0x0006
    Topology             : 0x0007
    BfdState             : 0x0008
    DataPathId          : 0x0009
    DataPathFlags        : 0x000A
    DataPathPriority     : 0x000B
    DataPathController   : 0x000C
    DataPathName         : 0x000D
    Longitude            : 0x000E
    Latitude             : 0x000F
  }

34 (E12) : - var [D1][DataValue] = new_d1[DataValue].asType(Ecast)

35 (E13) : - OFP-41_OFPB40

  var [D1] {
    Transaction          : 10000
    Identifier           : 10004
    LocalTableId        : 0x0005
    LocalTableNumber     : 0x0006
    Topology             : 0x0007
    BfdState             : 0x0008
  }

```

```
In [41]: 1 del csv_df["StratificationCategoryID1"]
```

```
In [42]: 1 del csv_df["DataValueFootnoteSymbol"]
```

```
In [43]: 1 del csv_df["StratificationCategory3"]
```

```
In [44]: 1 del csv_df["Stratification3"]
```

```
In [45]: 1 del csv_df["DatavalueFootnote"]
```

```
In [46]: 1 del csv_df["LowConfidenceLimit"]
```

```
In [47]: 1 del csv_df["HighConfidenceLimit"]
```

```
In [48]: 1 del csv_df["StratificationCategory2"]
```

```
In [49]: 1 del csv_df["Stratification2"]
```

```
In [50]: 1 csv_df.head()
```

Out[50]:

adults

```
In [50]: 1 csv_df.head()
```

```
Out[50]:
```

adults
aged >=...

```
In [60]: 1 csv_df = csv_df.dropna(how="any")
```

```
In [61]: 1 csv_df.head()
```

```
Out[61]:
```

	YearStart	YearEnd	LocationAbbr	LocationDesc	Topic	Question	DataValueUnit	DataValue	DataValueAlt	StratificationC:
1	2016	2016	AL	Alabama	Alcohol	Binge drinking prevalence among adults aged >=...	%	13	13.0	
2	2016	2016	AK	Alaska	Alcohol	Binge drinking prevalence among adults aged >=...	%	18.2	18.2	

Chronical Diseases Data Cleaning

Step 2

- importing the CSV into a data frame
- Debugging the low memory error by using the LOW_MEMORY =FALSE Boolean

```
In [10]: %matplotlib inline  
In [10]: import pandas as pd  
In [10]:  
...: raw_gist = "https://gist.githubusercontent.com/.../raw/.../chronic_diseases.csv"  
...: raw_df = pd.read_csv(raw_gist)  
...: raw_df.info()  
...: raw_df  
...:  
...: /usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:1221: UserWarning: All columns are of type object. You should consider setting the dtype argument on select columns to something more specific.  
...: warnings.warn("All columns are of type object. You should consider setting the dtype argument on select columns to something more specific.", UserWarning)  
...:  
...: Out[10]: <class 'pandas.core.frame.DataFrame'>  
...: RangeIndex: 10000 entries, 0 to 9999  
...: Data columns (total 13 columns):  
...: #   Column  Non-Null Count  Dtype     
...: 0   ID      10000 non-null  int64    
...: 1   Year    10000 non-null  int64    
...: 2   Sex     10000 non-null  object    
...: 3   Age     10000 non-null  int64    
...: 4   Region  10000 non-null  object    
...: 5   Income  10000 non-null  object    
...: 6   Education 10000 non-null  object    
...: 7   Health  10000 non-null  object    
...: 8   Disease 10000 non-null  object    
...: 9   DiseaseID 10000 non-null  int64    
...: 10  DiseaseCategory 10000 non-null  object    
...: 11  DiseaseType 10000 non-null  object    
...: 12  DiseaseSubType 10000 non-null  object    
...: 13  DiseaseName 10000 non-null  object  
```

Step 2

- deleted unwanted columns using the DEL function
- followed that up with a DROPNA function that cleaned up the empty data and corresponding rows

```
In [11]: raw_df.drop(['Year', 'Sex', 'Age', 'Region', 'Income', 'Education', 'Health', 'Disease', 'DiseaseID', 'DiseaseCategory', 'DiseaseType', 'DiseaseSubType'], axis=1, inplace=True)  
In [11]: raw_df  
...:  
...: /usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:1221: UserWarning: All columns are of type object. You should consider setting the dtype argument on select columns to something more specific.  
...: warnings.warn("All columns are of type object. You should consider setting the dtype argument on select columns to something more specific.", UserWarning)  
...:  
...: Out[11]: <class 'pandas.core.frame.DataFrame'>  
...: RangeIndex: 10000 entries, 0 to 9999  
...: Data columns (total 3 columns):  
...: #   Column  Non-Null Count  Dtype     
...: 0   ID      10000 non-null  int64    
...: 1   Disease  10000 non-null  object    
...: 2   DiseaseName 10000 non-null  object  
```

Step 3

- then used the ASTYPE function to convert data in certain columns to floats

```
In [12]: raw_df['Disease'] = raw_df['Disease'].astype('float')  
In [12]: raw_df  
...:  
...: /usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:1221: UserWarning: All columns are of type object. You should consider setting the dtype argument on select columns to something more specific.  
...: warnings.warn("All columns are of type object. You should consider setting the dtype argument on select columns to something more specific.", UserWarning)  
...:  
...: Out[12]: <class 'pandas.core.frame.DataFrame'>  
...: RangeIndex: 10000 entries, 0 to 9999  
...: Data columns (total 3 columns):  
...: #   Column  Non-Null Count  Dtype     
...: 0   ID      10000 non-null  int64    
...: 1   Disease  10000 non-null  float64  
...: 2   DiseaseName 10000 non-null  object  
```

```
In [81]: 1 csv_df.dtypes
```

```
Out[81]: YearStart          int64
YearEnd            int64
LocationAbbr       object
LocationDesc       object
Topic              object
Question           object
DataValueUnit      object
DataValue          object
StratificationCategory1 object
Latitude           object
Longitude          object
dtype: object
```

```
In [82]: 1 csv_df["DataValue"] = csv_df["DataValue"].astype(float)
```

```
In [83]: 1 csv_df.dtypes
```

```
Out[83]: YearStart          int64
YearEnd            int64
LocationAbbr       object
LocationDesc       object
Topic              object
Question           object
DataValue          float64
dtype: object
```

Chronical Diseases Data Cleaning

Step 2

- importing the CSV into a data frame
 - Debugging the low memory error by using the LOW_MEMORY =FALSE Boolean

Step 2

- deleted unwanted columns using the DEL function
 - followed that up with a DROPNAN function that cleaned up the empty data and corresponding rows

Step 3

- then used the ASTYPE function to convert data in certain columns to floats

```

35 [101]: i = CSV_01_01.read()

36 [101]: print(i)

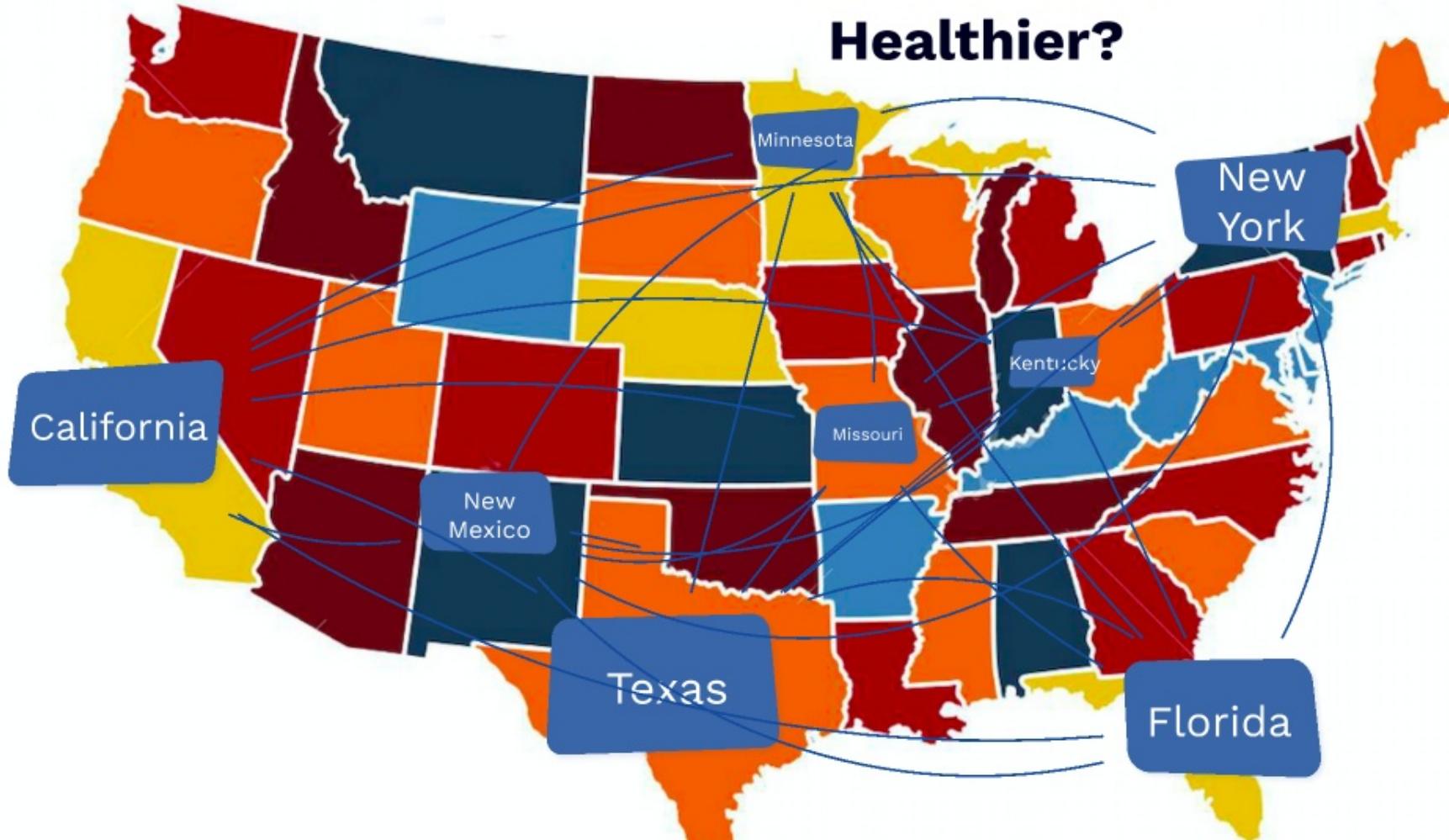
37 [101]: 
38 [101]:    Name          object
39 [101]:    Identifier      object
40 [101]:    Location        object
41 [101]:    LocationInMGR   object
42 [101]:    LocationInSearc object
43 [101]:    Topic          object
44 [101]:    Subject         object
45 [101]:    DateCreated     object
46 [101]:    DateModified    object
47 [101]:    DatePublished   object
48 [101]:    DateLastUpdated object
49 [101]:    Description     object
50 [101]:    DescriptionInCompany object
51 [101]:    DescriptionInSearc object
52 [101]:    Longitude       object
53 [101]:    Latitude        object
54 [101]:    Zipcode         object

55 [102]: i = vars_all["DataValues"] = vars_all["DataSeries"]._asdict()
56 [102]: print(i)

57 [102]: 
58 [102]:    Name          object
59 [102]:    Identifier      object
60 [102]:    Location        object
61 [102]:    LocationInMGR   object
62 [102]:    LocationInSearc object
63 [102]:    Topic          object
64 [102]:    Subject         object

```

Which State is Healthier?



Visualization

Make the data easier to interpret

Line Chart

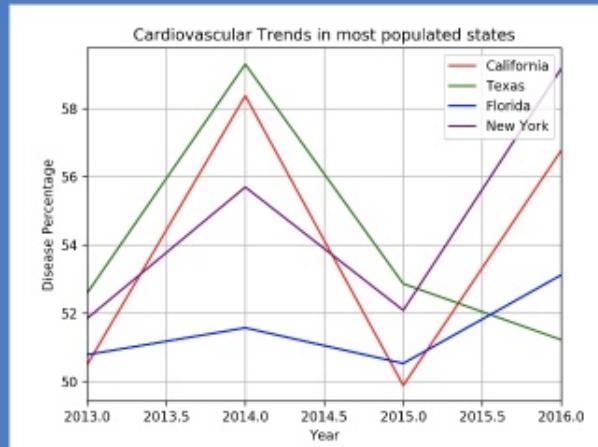
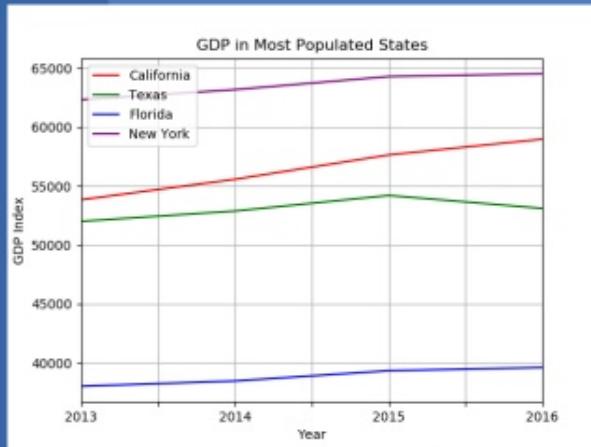


G-map



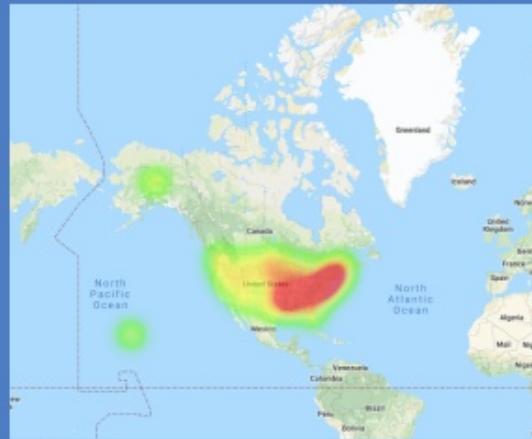
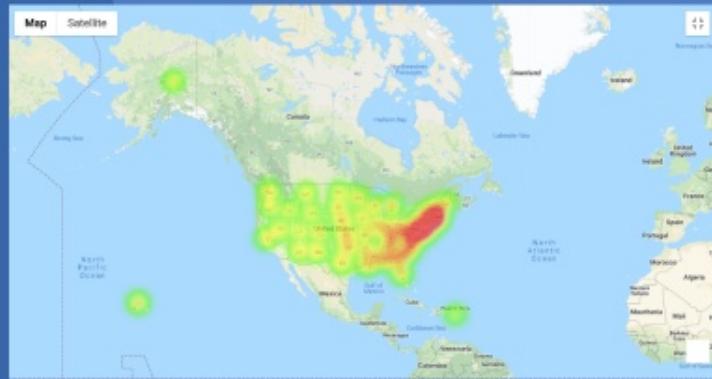
Bar Chart

Line Chart for comparing the Trend between Cardiovascular and GDP



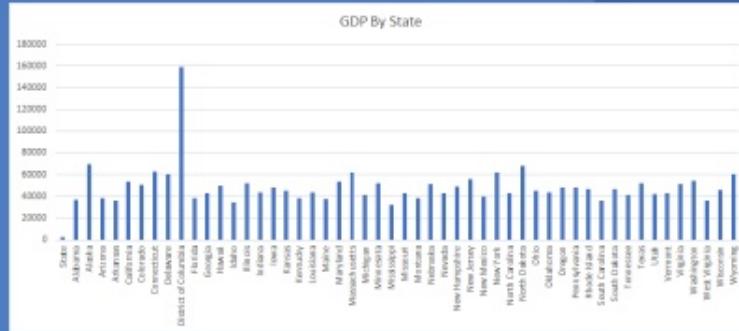
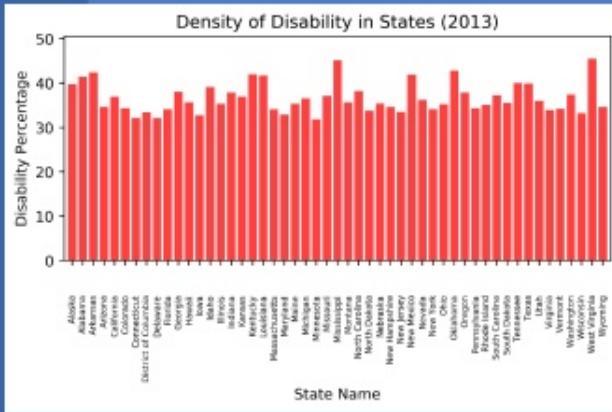
- we have chosen Cardiovascular Disease to further our analyzation. Here we have a trend chart of Cardiovascular Disease with the top 4 most populated states which are California, Texas, Florida and New York.
- I did the line chart by first by using the 'get group' function which has helped me group the dataframe according to 'Cardiovascular Diseases'. From there, I have managed to the loc function for each of the 4 states. Each state has been converted into a data frame.
- From this chart, it can be analyzed that Texas has the highest rate of Cardiovasculardisease and Florida is the lowest

G-Map for Chronic Diseases and GDP Distribution



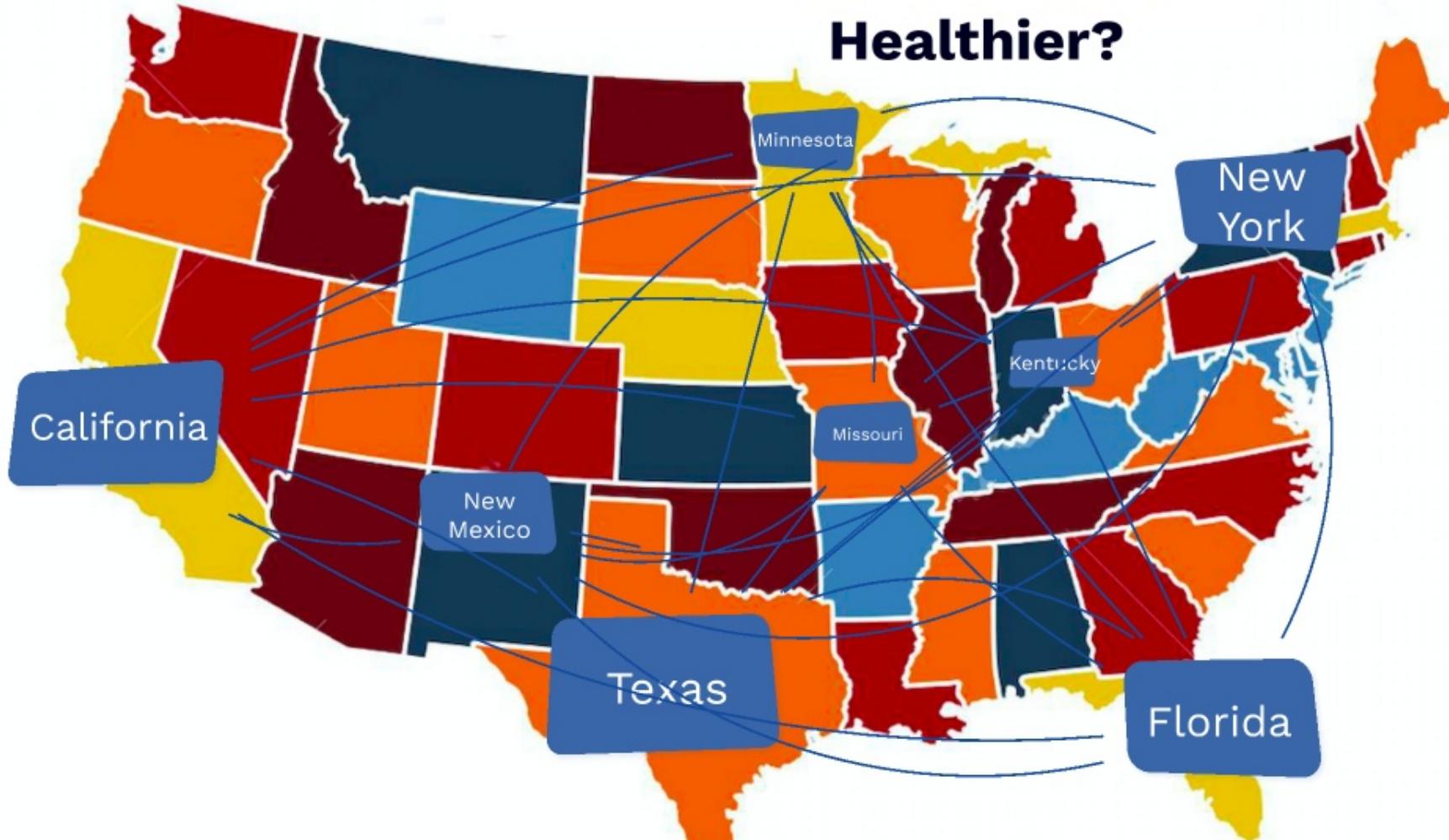
- Imported all my dependencies that I needed to use for the mapping and added my google API key.
- Created my locations variable which included the Latitude and Longitude columns from the data files.
- Next, created the rating variable. For the Disease data rating I used the 2017 values from the DATAVALUE column and for the GDP Data rating I used the 2017 data column to compare the most recent year of general data we had. Then I plotted the figure and added a heat layer to the figure.
- Finally, saved the final figure as a PNG

Bar Chart for Comparing the density between Diseases and GDP of states



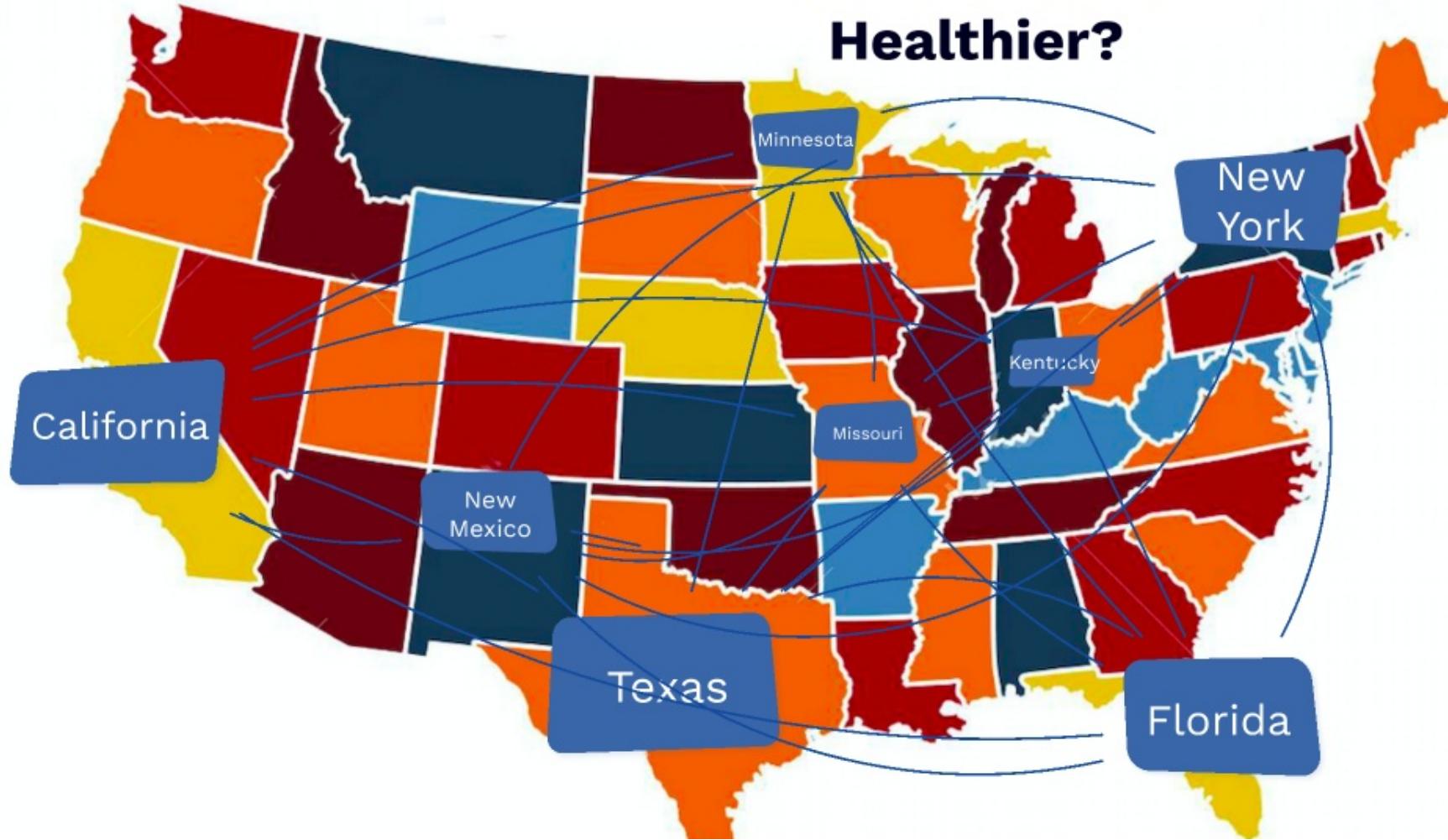
- According to our bar chart, the states with the highest disability rates were Arkansas, Mississippi, Oklahoma, and West Virginia. The states with the lowest disability rates were
- 3 of the four poorest states match up with 4 of the states with highest disability rates.
- There is a relationship between GDP and health of a state, though not for both ends of the spectrum.

Which State is Healthier?



Data cleaning

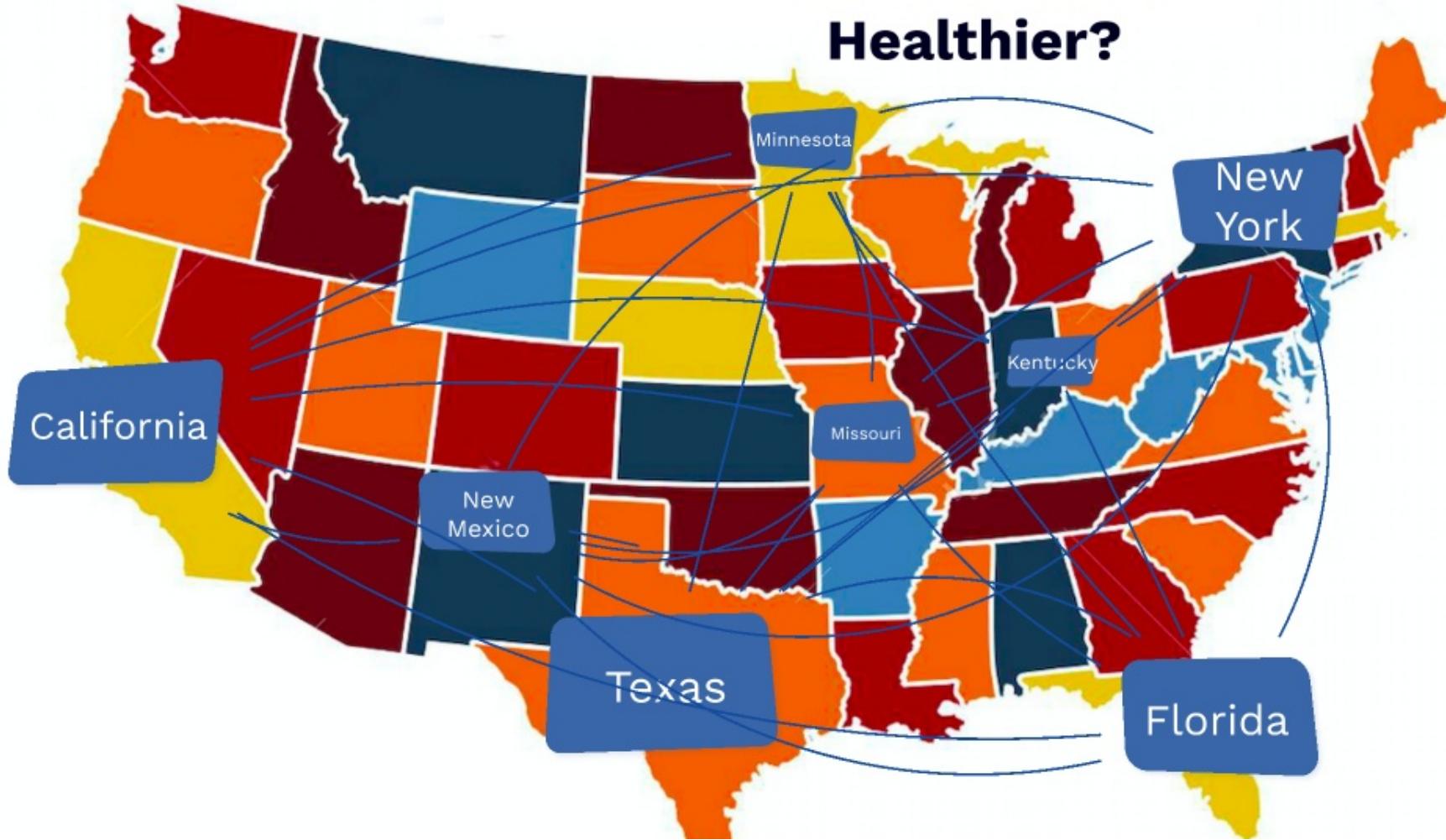
Which State is Healthier?

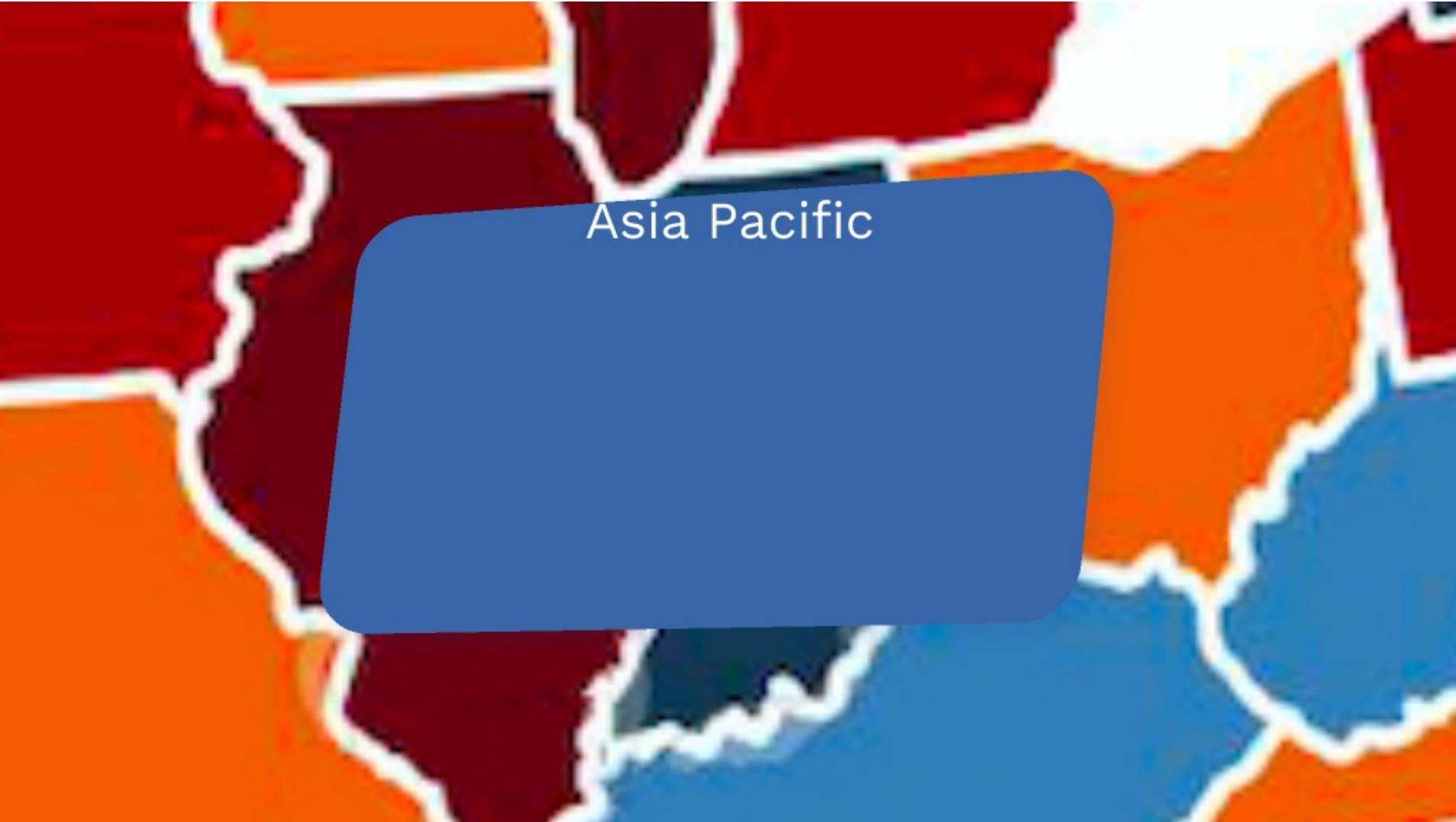




Middle East

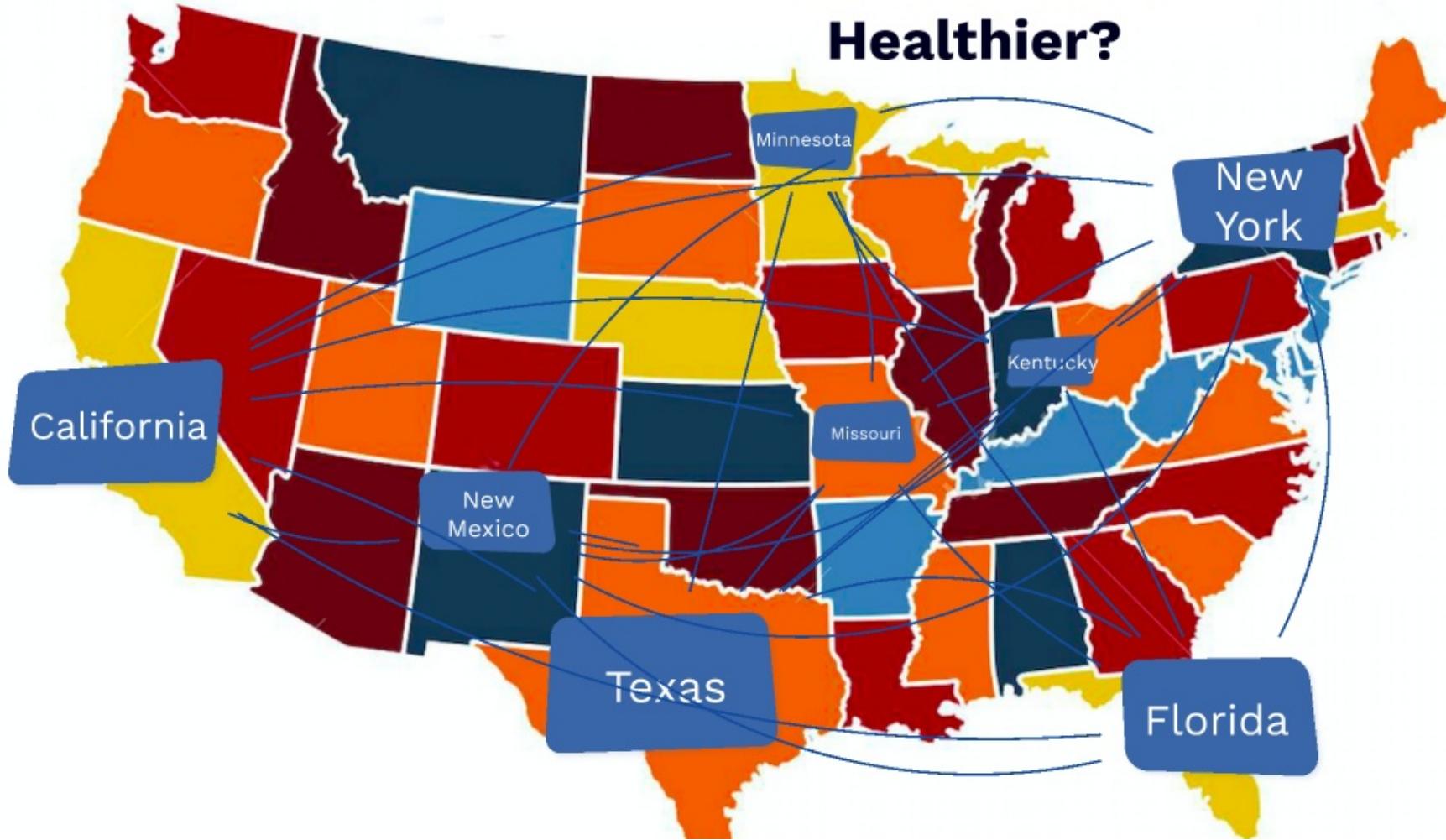
Which State is Healthier?

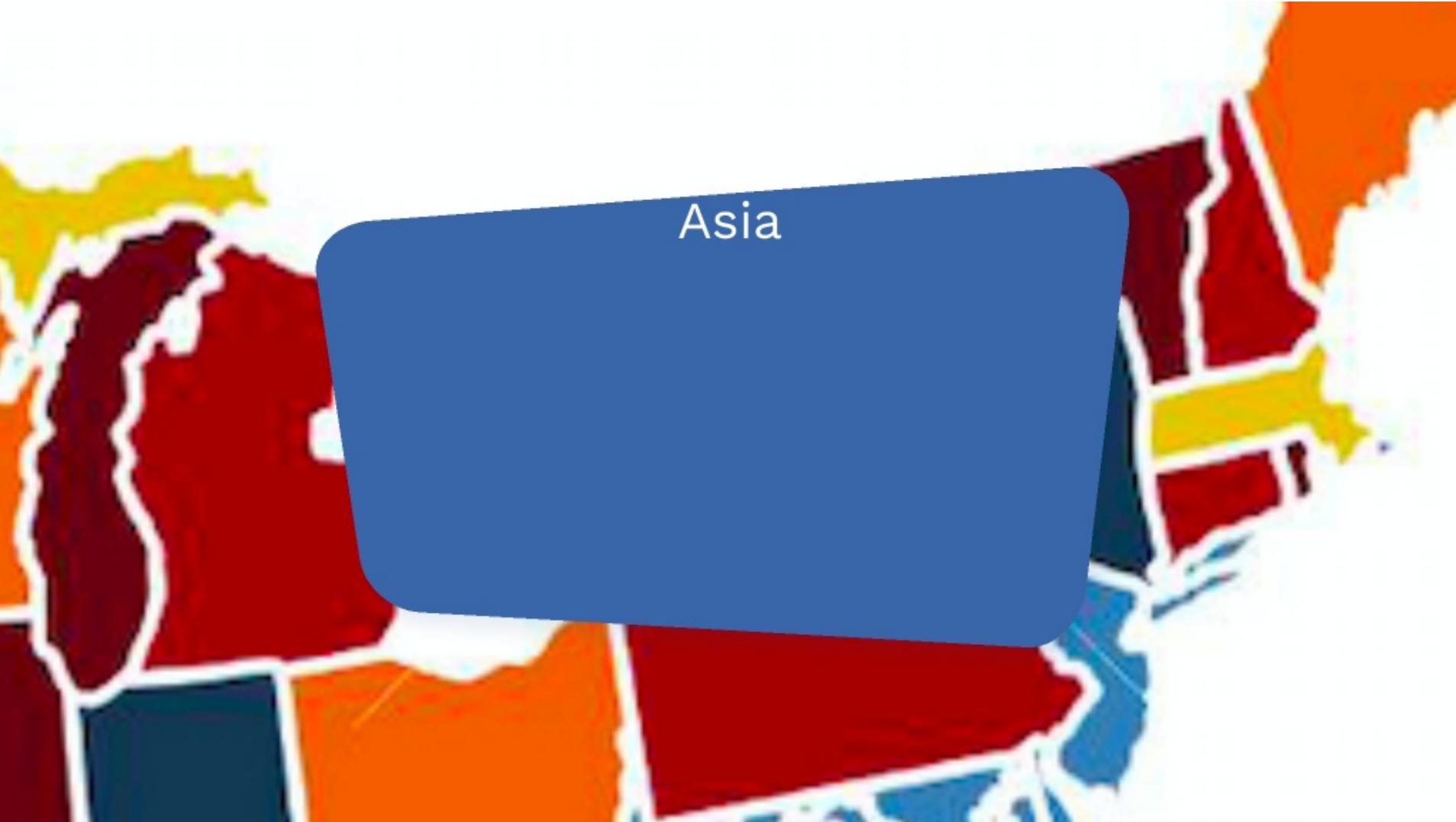




Asia Pacific

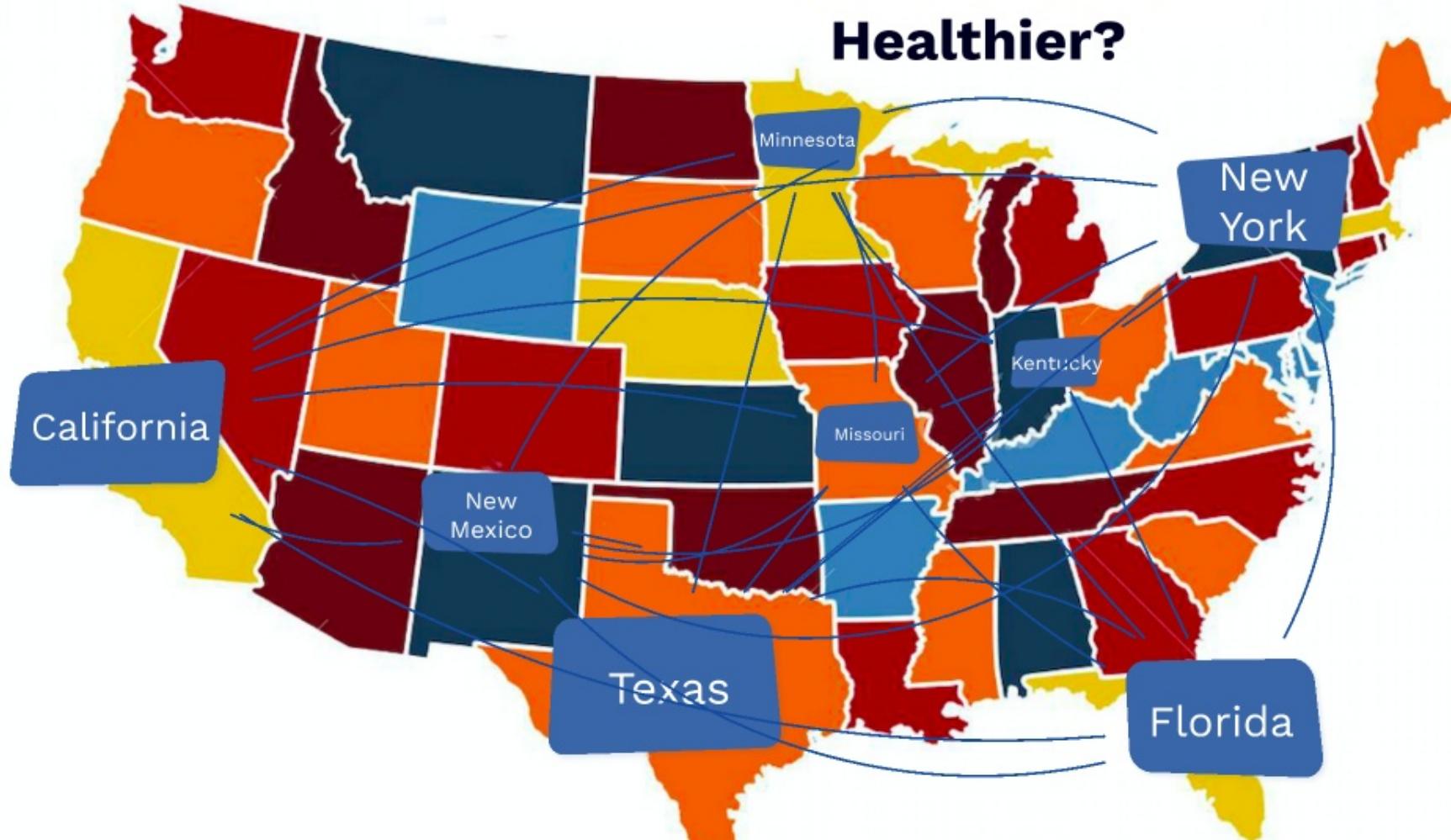
Which State is Healthier?

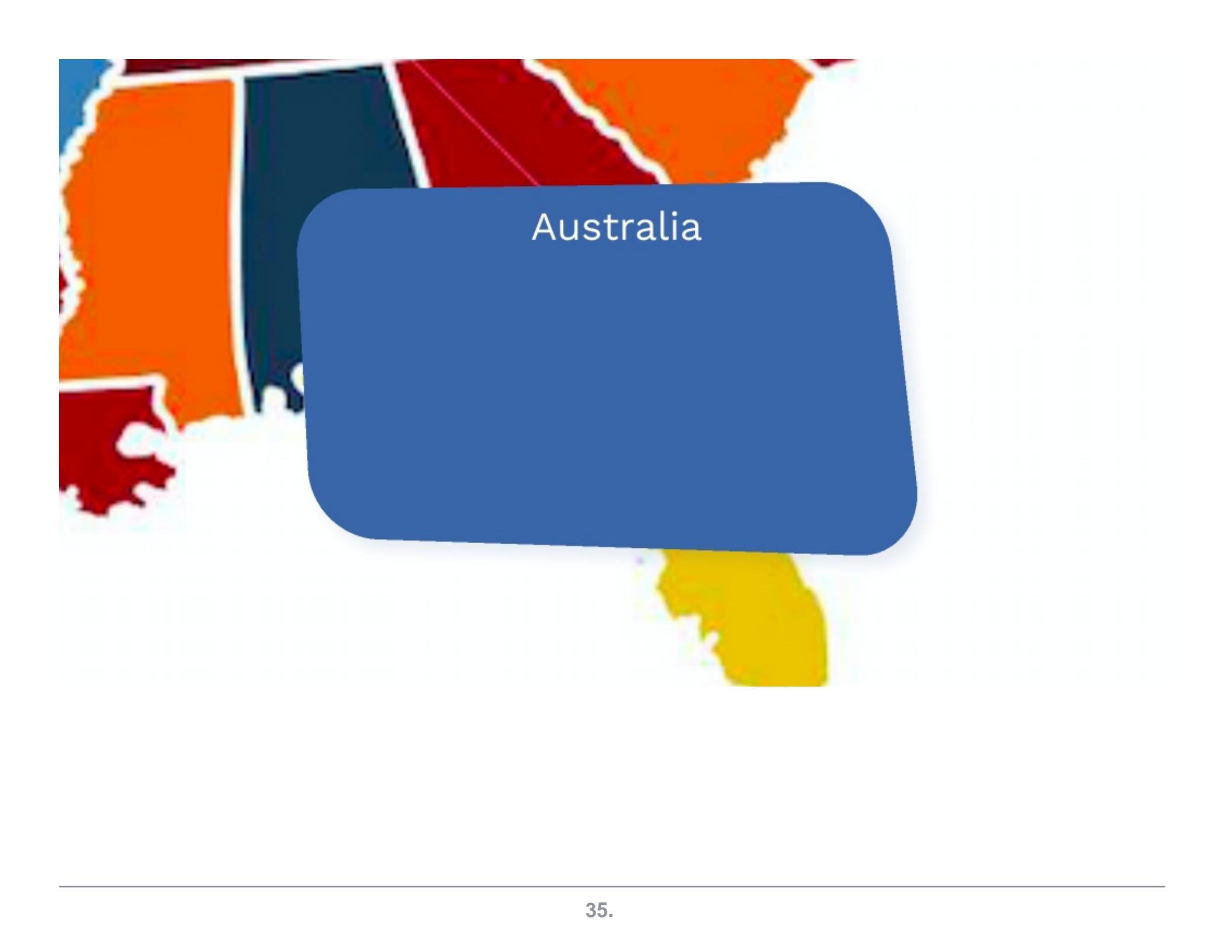




Asia

Which State is Healthier?





Australia

Which State is Healthier?

