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

In [2]:
```
# initializing and cleaning individual datasets starting with smokers
smoker_data = pd.read_excel('Smoker_Data.xlsx')

#rename column to make it more logical when merged
smoker_data = smoker_data.rename(columns={"Proportion": "Proportion_smokers"})
smoker_data = smoker_data.rename(columns={"Current": "Current_smokers"})

smoker_data.head

#filter rest of columns out
smoker_data = smoker_data.loc[:, ["State", "Proportion_smokers", "Current_smokers"]]
```

In [3]:
```
#now with veterans dataset

veterans_data = pd.read_excel('Veterans_by_State.xlsx')

veterans_data = veterans_data.rename(columns={"Grand Total": "number_of_veterans"})

veterans_data = veterans_data.loc[:, ["State", "number_of_veterans"]]
```

In [4]:
```
#now with diabetes data set

diabetes_data = pd.read_csv('Diabetes_By_State.csv', skiprows=[0, 1])

diabetes_data = diabetes_data.rename(columns={"Percentage": "Percentage_diabetic"})

diabetes_data = diabetes_data.loc[:, ["State", "Percentage_diabetic"]]
```

In [5]:
```
#merging above 3

merged_SmokDiabVet = smoker_data.merge(veterans_data, on='State').merge(diabetes_data, on = 'St
```

In [6]:
```
#condensing accidents data

accident_2018 = pd.read_csv('accident2018.csv')

accident_2018_condensed = accident_2018.groupby('STATENAME').size().reset_index(name='Accidents
accident_2018_condensed = accident_2018_condensed.rename(columns={"STATENAME": "State"})
```

In [7]:
```
accident_2019 = pd.read_csv('accident2019.csv', encoding='ISO-8859-1')

accident_2019_condensed = accident_2019.groupby('STATENAME').size().reset_index(name='Accidents
accident_2019_condensed = accident_2019_condensed.rename(columns={"STATENAME": "State"})

accident_2019_condensed.head()
```

```
/var/folders/wk/mh3nlc5d5mz44xkrmh91ptl00000gn/T/ipykernel_7434/241494596.py:1: DtypeWarnin
g: Columns (40,42) have mixed types. Specify dtype option on import or set low_memory=False.
  accident_2019 = pd.read_csv('accident2019.csv', encoding='ISO-8859-1')
```

Out[7]:

	State	Accidents_2019
0	Alabama	856
1	Alaska	62
2	Arizona	908
3	Arkansas	473
4	California	3427

In [8]:

```
#condensing accident 2020 data
accident_2020 = pd.read_csv('accident2020.csv', encoding='ISO-8859-1')

accident_2020_condensed = accident_2020.groupby('STATENAME').size().reset_index(name='Accidents
accident_2020_condensed = accident_2020_condensed.rename(columns={"STATENAME": "State"})

accident_2020_condensed.head()
```

Out[8]:

	State	Accidents_2020
0	Alabama	852
1	Alaska	53
2	Arizona	967
3	Arkansas	585
4	California	3558

In [9]:

```
#merging accident data sets
merged_accidents = accident_2018_condensed.merge(accident_2019_condensed, on='State').merge(acc

merged_accidents.head()
```

Out[9]:

	State	Accidents_2018	Accidents_2019	Accidents_2020
0	Alabama	876	856	852
1	Alaska	69	62	53
2	Arizona	918	908	967
3	Arkansas	476	473	585
4	California	3485	3427	3558

In [10]:

```
full_dataframe = merged_accidents.merge(merged_SmokDiabVet, on = 'State')

full_dataframe.head()
```

Out[10]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Perc
0	Alabama	876	856	852	0.184967	761140	3.595064e+05	
1	Alaska	69	62	53	0.187964	111288	7.145358e+04	
2	Arizona	918	908	967	0.142557	796004	5.081568e+05	
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05	
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06	

In [11]:

```
#Data analysis

#this is the raw data size
tuples_list = [accident_2019.shape, accident_2020.shape, accident_2018.shape, smoker_data.shape
raw_data_size = tuple(sum(values) for values in zip(*tuples_list))
raw_data_size
```

Out[11]:

```
(103334, 270)
```

In [12]:

```
#save file
file_path = 'Data_Viz_FullData.csv'
full_dataframe.to_csv(file_path, index=False)
```

In [13]:
```
#this is the cleaned data size
df = full_dataframe

df.shape
```

Out[13]:
```
(51, 8)
```

In [14]:
```
# column names
df.columns
```

Out[14]:
```
Index(['State', 'Accidents_2018', 'Accidents_2019', 'Accidents_2020',
       'Proportion_smokers', 'Current_smokers', 'number_of_veterans',
       'Percentage_diabetic'],
      dtype='object')
```

In [15]:
```
# classificiation of each column
df.head()
```

Out[15]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Perc
0	Alabama	876	856	852	0.184967	761140	3.595064e+05	
1	Alaska	69	62	53	0.187964	111288	7.145358e+04	
2	Arizona	918	908	967	0.142557	796004	5.081568e+05	
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05	
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06	

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51 entries, 0 to 50
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   State                51 non-null     object 
 1   Accidents_2018       51 non-null     int64  
 2   Accidents_2019       51 non-null     int64  
 3   Accidents_2020       51 non-null     int64  
 4   Proportion_smokers   51 non-null     float64
 5   Current_smokers      51 non-null     int64  
 6   number_of_veterans   51 non-null     float64
 7   Percentage_diabetic  51 non-null     object 
dtypes: float64(2), int64(4), object(2)
memory usage: 3.6+ KB
```

In [17]:
```
# fix datatype of percentage_diabetic
df['Percentage_diabetic'] = df['Percentage_diabetic'].astype('float64')
df.dtypes
```

Out[17]:
```
State                   object
Accidents_2018           int64
Accidents_2019           int64
Accidents_2020           int64
Proportion_smokers     float64
Current_smokers          int64
number_of_veterans     float64
Percentage_diabetic    float64
dtype: object
```

In [18]:

```
# quantitative data information
df.describe()
```

Out[18]:

	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Percentage
count	51.000000	51.000000	51.000000	51.000000	5.100000e+01	5.100000e+01	
mean	665.078431	656.607843	701.294118	0.156580	7.705217e+05	3.770146e+05	
std	742.112224	739.580478	781.515330	0.028843	7.499637e+05	3.689939e+05	
min	30.000000	22.000000	34.000000	0.084496	7.634400e+04	2.979827e+04	
25%	208.000000	206.500000	203.500000	0.135808	1.839015e+05	1.267465e+05	
50%	485.000000	496.000000	540.000000	0.156148	5.871370e+05	2.841411e+05	
75%	891.500000	867.500000	938.000000	0.173454	9.980935e+05	4.808888e+05	
max	3485.000000	3427.000000	3558.000000	0.227227	3.203562e+06	1.642998e+06	

In [19]:

```
# get info to easily display in writeup
def get_range(column):
  return column.max() - column.min()

for column in df.columns:
    # check if if the datatype is numeric! (documentation from online, need to exclude state)
    if pd.api.types.is_numeric_dtype(df[column]):
        # Calculate range, median, mean, and standard deviation
        col_range = get_range(df[column])
        col_median = df[column].median()
        col_mean = df[column].mean()
        col_std = df[column].std()

        # Print results
        print(f"Column: {column}")
        print(f"Range: {col_range}")
        print(f"Median: {col_median}")
        print(f"Mean: {col_mean}")
        print(f"Standard Deviation: {col_std}")
        print()
```

```
Column: Accidents_2018
Range: 3455
Median: 485.0
Mean: 665.0784313725491
Standard Deviation: 742.112224481911

Column: Accidents_2019
Range: 3405
Median: 496.0
Mean: 656.6078431372549
Standard Deviation: 739.5804777961996

Column: Accidents_2020
Range: 3524
Median: 540.0
Mean: 701.2941176470588
Standard Deviation: 781.5153304732453

Column: Proportion_smokers
Range: 0.14273108734
Median: 0.1561476651
Mean: 0.15657997354039216
Standard Deviation: 0.028843442410443156

Column: Current_smokers
Range: 3127218
Median: 587137.0
Mean: 770521.7058823529
Standard Deviation: 749963.703309708

Column: number_of_veterans
Range: 1613199.6216898195
Median: 284141.1059845544
Mean: 377014.6011790353
Standard Deviation: 368993.8703542532

Column: Percentage_diabetic
Range: 6.3999999999999995
Median: 9.3
Mean: 9.588235294117647
Standard Deviation: 1.7529001179557868
```

In [20]:
```
# check for duplicates
df.head(100)
```

Out[20]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans
0	Alabama	876	856	852	0.184967	761140	3.595064e+05
1	Alaska	69	62	53	0.187964	111288	7.145358e+04
2	Arizona	918	908	967	0.142557	796004	5.081568e+05
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06
5	Colorado	588	545	574	0.134601	587137	3.858066e+05
6	Connecticut	275	233	279	0.117712	338849	1.719690e+05
7	Delaware	104	122	104	0.153268	124011	7.095816e+04
8	District of Columbia	30	22	34	0.131734	78869	2.979827e+04
9	Florida	2917	2952	3098	0.146478	2577420	1.494804e+06
10	Georgia	1408	1378	1522	0.158327	1307100	6.903892e+05
11	Hawaii	110	102	81	0.119173	136408	1.126771e+05
12	Idaho	215	201	188	0.133111	176982	1.276412e+05
13	Illinois	951	938	1087	0.142025	1466080	6.080349e+05
14	Indiana	776	752	815	0.194177	1059118	4.067570e+05
15	Iowa	291	313	304	0.156718	400049	1.938606e+05
16	Kansas	367	361	382	0.158237	360137	1.954340e+05
17	Kentucky	664	667	709	0.215920	820721	2.827673e+05
18	Louisiana	719	681	762	0.200879	776192	2.792867e+05
19	Maine	127	143	151	0.158033	179036	1.126261e+05
20	Maryland	485	496	540	0.127874	619227	3.725728e+05
21	Massachusetts	338	323	327	0.126826	717716	3.094468e+05
22	Michigan	907	903	1011	0.173070	1441675	5.623287e+05
23	Minnesota	349	333	369	0.134588	594930	3.139123e+05
24	Mississippi	596	580	687	0.197342	472648	1.870722e+05
25	Missouri	848	819	914	0.187582	948726	4.160352e+05
26	Montana	167	166	190	0.157491	138198	8.927546e+04
27	Nebraska	201	212	217	0.141837	214738	1.258518e+05
28	Nevada	299	285	293	0.160599	394177	2.196977e+05
29	New Hampshire	134	90	98	0.144031	161207	1.012795e+05
30	New Jersey	524	524	547	0.126355	914841	3.380115e+05
31	New Mexico	351	369	365	0.158679	264465	1.497559e+05
32	New York	910	879	963	0.130012	2058331	7.471575e+05
33	North Carolina	1321	1358	1412	0.156148	1320956	7.031418e+05
34	North Dakota	95	91	96	0.166259	103343	5.325447e+04
35	Ohio	996	1039	1154	0.189482	1842396	7.296450e+05
36	Oklahoma	603	584	599	0.179992	576977	2.913155e+05
37	Oregon	446	455	461	0.147941	504603	2.841411e+05
38	Pennsylvania	1103	990	1060	0.169300	1837177	7.694230e+05
39	Rhode Island	56	53	66	0.137014	120811	6.222426e+04
40	South Carolina	969	927	962	0.168886	711229	3.936841e+05

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans
41	South Dakota	110	88	132	0.173839	124347	6.542926e+04
42	Tennessee	973	1041	1119	0.201390	1119838	4.536207e+05
43	Texas	3311	3296	3520	0.142604	3139192	1.567233e+06
44	Utah	237	225	256	0.084496	188767	1.329598e+05
45	Vermont	60	44	58	0.144668	76344	4.202870e+04
46	Virginia	778	774	796	0.149199	1047461	7.076678e+05
47	Washington	490	513	525	0.124935	747860	5.515124e+05
48	West Virginia	265	247	249	0.227227	370689	1.351901e+05
49	Wisconsin	531	527	561	0.148191	689747	3.500415e+05
50	Wyoming	100	120	114	0.169191	82278	4.690648e+04

In [21]:

```
# determine pearson correlation coefficients
df.corr()
```

```
/var/folders/wk/mh3nlc5d5mz44xkrmh91ptl00000gn/T/ipykernel_7434/162719484.py:2: FutureWarnin
g: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_only to s
ilence this warning.
  df.corr()
```

Out[21]:

	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_vetera
Accidents_2018	1.000000	0.999275	0.998531	-0.074976	0.934505	0.9654
Accidents_2019	0.999275	1.000000	0.999095	-0.071356	0.932544	0.9649
Accidents_2020	0.998531	0.999095	1.000000	-0.061168	0.937068	0.9647
Proportion_smokers	-0.074976	-0.071356	-0.061168	1.000000	-0.062484	-0.1558
Current_smokers	0.934505	0.932544	0.937068	-0.062484	1.000000	0.9708
number_of_veterans	0.965402	0.964993	0.964715	-0.155825	0.970834	1.0000
Percentage_diabetic	0.329473	0.333297	0.339788	0.608510	0.294272	0.2307

In [22]:
```
# going to factor for population, then run correlation
pops = pd.read_excel('nst-est2019-01.xlsx', header=0)
pops.head(100)
```

Out[22]:

	State	Population
0	Alabama	4903185
1	Alaska	731545
2	Arizona	7278717
3	Arkansas	3017804
4	California	39512223
5	Colorado	5758736
6	Connecticut	3565287
7	Delaware	973764
8	District of Columbia	705749
9	Florida	21477737
10	Georgia	10617423
11	Hawaii	1415872
12	Idaho	1787065
13	Illinois	12671821
14	Indiana	6732219
15	Iowa	3155070
16	Kansas	2913314
17	Kentucky	4467673
18	Louisiana	4648794
19	Maine	1344212
20	Maryland	6045680
21	Massachusetts	6892503
22	Michigan	9986857
23	Minnesota	5639632
24	Mississippi	2976149
25	Missouri	6137428
26	Montana	1068778
27	Nebraska	1934408
28	Nevada	3080156
29	New Hampshire	1359711
30	New Jersey	8882190
31	New Mexico	2096829
32	New York	19453561
33	North Carolina	10488084
34	North Dakota	762062
35	Ohio	11689100
36	Oklahoma	3956971
37	Oregon	4217737
38	Pennsylvania	12801989
39	Rhode Island	1059361
40	South Carolina	5148714
41	South Dakota	884659

	State	Population
42	Tennessee	6829174
43	Texas	28995881
44	Utah	3205958
45	Vermont	623989
46	Virginia	8535519
47	Washington	7614893
48	West Virginia	1792147
49	Wisconsin	5822434
50	Wyoming	578759

In [23]:
```
# merge df and pops
df_pops = pd.merge(df, pops, on='State')
df_pops.head(10)
```

Out[23]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Pe
0	Alabama	876	856	852	0.184967	761140	3.595064e+05	
1	Alaska	69	62	53	0.187964	111288	7.145358e+04	
2	Arizona	918	908	967	0.142557	796004	5.081568e+05	
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05	
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06	
5	Colorado	588	545	574	0.134601	587137	3.858066e+05	
6	Connecticut	275	233	279	0.117712	338849	1.719690e+05	
7	Delaware	104	122	104	0.153268	124011	7.095816e+04	
8	District of Columbia	30	22	34	0.131734	78869	2.979827e+04	
9	Florida	2917	2952	3098	0.146478	2577420	1.494804e+06	

In [24]:
```
# add rows to account for population
df_pops['Accidents_2018_proportion'] = df_pops['Accidents_2018'] / df_pops['Population']
df_pops['Accidents_2019_proportion'] = df_pops['Accidents_2019'] / df_pops['Population']
df_pops['Accidents_2020_proportion'] = df_pops['Accidents_2020'] / df_pops['Population']
df_pops['Current_smokers_proportion'] = df_pops['Current_smokers'] / df_pops['Population']
df_pops['number_of_veterans_proportion'] = df_pops['number_of_veterans'] / df_pops['Population
```

In [25]:
```
df_pops.head()
```

Out[25]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Perc
0	Alabama	876	856	852	0.184967	761140	3.595064e+05	
1	Alaska	69	62	53	0.187964	111288	7.145358e+04	
2	Arizona	918	908	967	0.142557	796004	5.081568e+05	
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05	
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06	

In [26]:
```
# remove non-proportion columns
df_proportion = df_pops.drop(columns=['Accidents_2018', 'Accidents_2019', 'Accidents_2020', 'Cu
```

In [27]:
```
# now run pearson correlation coeff
df.corr()
```

```
/var/folders/wk/mh3nlc5d5mz44xkrmh91ptl00000gn/T/ipykernel_7434/14569665.py:2: FutureWarnin
g: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_only to s
ilence this warning.
  df.corr()
```

Out[27]:

	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_vetera
Accidents_2018	1.000000	0.999275	0.998531	-0.074976	0.934505	0.9654
Accidents_2019	0.999275	1.000000	0.999095	-0.071356	0.932544	0.9649
Accidents_2020	0.998531	0.999095	1.000000	-0.061168	0.937068	0.9647
Proportion_smokers	-0.074976	-0.071356	-0.061168	1.000000	-0.062484	-0.1558
Current_smokers	0.934505	0.932544	0.937068	-0.062484	1.000000	0.9708
number_of_veterans	0.965402	0.964993	0.964715	-0.155825	0.970834	1.0000
Percentage_diabetic	0.329473	0.333297	0.339788	0.608510	0.294272	0.2307

In [28]:
```
# import packages
import numpy as np
from scipy.optimize import curve_fit
```

In [29]:
```
# Define linear, quadratic, and cubic functions
def linear(x, a, b):
    return a * x + b

def quadratic(x, a, b, c):
    return a * x**2 + b * x + c

def cubic(x, a, b, c, d):
    return a * x**3 + b * x**2 + c * x + d

# Fit the functions to the data
x_data = df['Accidents_2018'].values
y_data = df['Current_smokers'].values

linear_params, linear_covariance = curve_fit(linear, x_data, y_data)
quadratic_params, quadratic_covariance = curve_fit(quadratic, x_data, y_data)
cubic_params, cubic_covariance = curve_fit(cubic, x_data, y_data)

# Calculate the R-squared values
y_linear = linear(x_data, *linear_params)
y_quadratic = quadratic(x_data, *quadratic_params)
y_cubic = cubic(x_data, *cubic_params)

linear_r_squared = np.corrcoef(y_data, y_linear)[0, 1]**2
quadratic_r_squared = np.corrcoef(y_data, y_quadratic)[0, 1]**2
cubic_r_squared = np.corrcoef(y_data, y_cubic)[0, 1]**2

# Print the R-squared values
print(f"Accidents_2018 & current_smokers")
print(f"Linear R-squared: {linear_r_squared}")
print(f"Quadratic R-squared: {quadratic_r_squared}")
print(f"Cubic R-squared: {cubic_r_squared}")
```

```
Accidents_2018 & current_smokers
Linear R-squared: 0.8732990243352582
Quadratic R-squared: 0.8975980400988212
Cubic R-squared: 0.8998985744593831
```

In [30]:

```
# Fit the functions to the data
x_data = df['Accidents_2018'].values
y_data = df['number_of_veterans'].values

linear_params, linear_covariance = curve_fit(linear, x_data, y_data)
quadratic_params, quadratic_covariance = curve_fit(quadratic, x_data, y_data)
cubic_params, cubic_covariance = curve_fit(cubic, x_data, y_data)

# Calculate the R-squared values
y_linear = linear(x_data, *linear_params)
y_quadratic = quadratic(x_data, *quadratic_params)
y_cubic = cubic(x_data, *cubic_params)

linear_r_squared = np.corrcoef(y_data, y_linear)[0, 1]**2
quadratic_r_squared = np.corrcoef(y_data, y_quadratic)[0, 1]**2
cubic_r_squared = np.corrcoef(y_data, y_cubic)[0, 1]**2

# Print the R-squared values
print(f"Accidents_2018 & number_of_veterans")
print(f"Linear R-squared: {linear_r_squared}")
print(f"Quadratic R-squared: {quadratic_r_squared}")
print(f"Cubic R-squared: {cubic_r_squared}")
```

```
Accidents_2018 & number_of_veterans
Linear R-squared: 0.9320010539985575
Quadratic R-squared: 0.9381065807568301
Cubic R-squared: 0.9381170336895743
```

In [31]:

```
# Visuals
full_dataframe = df_pops

full_dataframe.head()
```

Out[31]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Perc
0	Alabama	876	856	852	0.184967	761140	3.595064e+05	
1	Alaska	69	62	53	0.187964	111288	7.145358e+04	
2	Arizona	918	908	967	0.142557	796004	5.081568e+05	
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05	
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06	

In [32]:

```
#preparing data for analysis

full_dataframe['Proportion_diabetic'] = full_dataframe['Percentage_diabetic'] / 100

full_dataframe.head()
```

Out[32]:

	State	Accidents_2018	Accidents_2019	Accidents_2020	Proportion_smokers	Current_smokers	number_of_veterans	Perc
0	Alabama	876	856	852	0.184967	761140	3.595064e+05	
1	Alaska	69	62	53	0.187964	111288	7.145358e+04	
2	Arizona	918	908	967	0.142557	796004	5.081568e+05	
3	Arkansas	476	473	585	0.196323	491610	2.110032e+05	
4	California	3485	3427	3558	0.106327	3203562	1.642998e+06	

In [33]:

```
proportion_columns = ['State', 'Accidents_2018_proportion', 'Accidents_2019_proportion', 'Accic
                      'Current_smokers_proportion', 'number_of_veterans_proportion', 'Proporti

df_proportions = full_dataframe[proportion_columns]

# Display the new DataFrame
df_proportions.head()
```

Out[33]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	nur
0	Alabama	0.000179	0.000175	0.000174	0.155234	
1	Alaska	0.000094	0.000085	0.000072	0.152127	
2	Arizona	0.000126	0.000125	0.000133	0.109360	
3	Arkansas	0.000158	0.000157	0.000194	0.162903	
4	California	0.000088	0.000087	0.000090	0.081078	

In [34]:

```
#employing min-max scaling to give all the proportions scores
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Select the columns to be scaled excluding 'State'
columns_to_scale = [col for col in df_proportions.columns if col != 'State']

# Apply Min-Max scaling to the selected columns
scaled_data = scaler.fit_transform(df_proportions[columns_to_scale])

# Create a new df with the scaled data and the same column names excluding 'State'
df_scaled_values = pd.DataFrame(scaled_data, columns=columns_to_scale)

# Add the 'State' column back to the new df
df_scaled = pd.concat([df['State'], df_scaled_values], axis=1)

# Display
df_scaled.head()
```

Out[34]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	nur
0	Alabama	0.863079	0.814042	0.688806	0.651212	
1	Alaska	0.328448	0.304140	0.136355	0.630217	
2	Arizona	0.530033	0.531169	0.465724	0.341175	
3	Arkansas	0.730409	0.712753	0.798325	0.703046	
4	California	0.289650	0.315382	0.232317	0.150024	

In [35]:

```
#Here I want to assign weights based on predictive capabilities: according to my research
#Car accidents account for 39% of amputations, Diabetic rates 25%, Smokers 12% and Veterans 7%

# Create a list of weights
# one weight for each column in df
weights = [0.13, 0.13, 0.13, 0.12, 0.07, 0.25]

weights_dict = {col: weight for col, weight in zip(df_scaled.columns[1:], weights)}

# Multiply each column except 'State' in the df by its corresponding weight
df_weighted = df_scaled.copy()
for col, weight in weights_dict.items():
    df_weighted[col] = df_scaled[col] * weight

# Display the weighted df
df_weighted.head()
```

Out[35]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	nur
0	Alabama	0.112200	0.105825	0.089545	0.078145	
1	Alaska	0.042698	0.039538	0.017726	0.075626	
2	Arizona	0.068904	0.069052	0.060544	0.040941	
3	Arkansas	0.094953	0.092658	0.103782	0.084366	
4	California	0.037655	0.041000	0.030201	0.018003	

In [36]:

```
# create a total risk score for amputations and demand score for prosthetics
# Compute the total score for each row by summing the values across columns excluding the 'Sta
df_weighted['total_score'] = df_weighted.drop('State', axis=1).sum(axis=1)

# Display the updated df with the 'total_score' column
df_weighted.head()
```

Out[36]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	nur
0	Alabama	0.112200	0.105825	0.089545	0.078145	
1	Alaska	0.042698	0.039538	0.017726	0.075626	
2	Arizona	0.068904	0.069052	0.060544	0.040941	
3	Arkansas	0.094953	0.092658	0.103782	0.084366	
4	California	0.037655	0.041000	0.030201	0.018003	

In [37]:

```
#Scale the risk score
scaler = MinMaxScaler(feature_range=(0, 10))

# Reshape the 'total_score' column to a 2D array and apply Min-Max scaling
scaled_total_score = scaler.fit_transform(df_weighted[['total_score']])

# Replace the 'total_score' column with the scaled data
df_weighted['total_score'] = scaled_total_score

# Display the updated df with the scaled 'total_score' column
df_weighted.head()
```

Out[37]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	nur
0	Alabama	0.112200	0.105825	0.089545	0.078145	
1	Alaska	0.042698	0.039538	0.017726	0.075626	
2	Arizona	0.068904	0.069052	0.060544	0.040941	
3	Arkansas	0.094953	0.092658	0.103782	0.084366	
4	California	0.037655	0.041000	0.030201	0.018003	

In [38]:
```
# plotly map
import plotly.express as px
import plotly.graph_objects as go
```

In [39]:
```
!pip install plotly-orca
!pip install "notebook>=5.3" "ipywidgets>=7.2"
```

```
ERROR: Could not find a version that satisfies the requirement plotly-orca (from versions: n
one)
ERROR: No matching distribution found for plotly-orca

[notice] A new release of pip is available: 23.0.1 -> 23.1.2
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: notebook>=5.3 in /Users/mihirgupta/Documents/College Other/Co
llege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (6.5.3)
Requirement already satisfied: ipywidgets>=7.2 in /Users/mihirgupta/Documents/College Other/
College/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (8.0.4)
Requirement already satisfied: jinja2 in /Users/mihirgupta/Documents/College Other/College/N
otes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from notebook>=5.3) (3.1.2)
Requirement already satisfied: tornado>=6.1 in /Users/mihirgupta/Documents/College Other/Col
lege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from notebook>=5.3) (6.2)
Requirement already satisfied: pyzmq>=17 in /Users/mihirgupta/Documents/College Other/Colleg
e/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from notebook>=5.3) (25.0.0)
Requirement already satisfied: argon2-cffi in /Users/mihirgupta/Documents/College Other/Coll
ege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from notebook>=5.3) (21.3.
0)
```

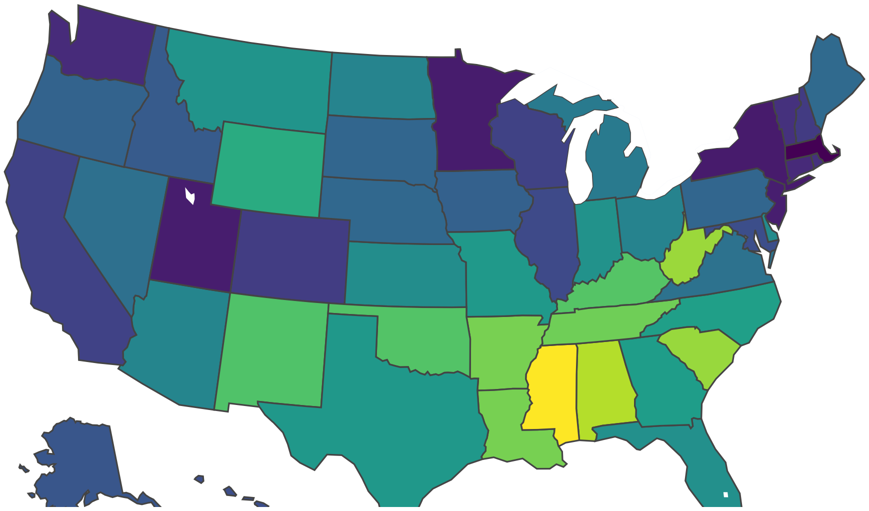
In [40]:
```
!pip install kaleido
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-macosx_10_11_x86_64.whl (85.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.2/85.2 MB 11.0 MB/s eta 0:00:0000:0100:01
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1

[notice] A new release of pip is available: 23.0.1 -> 23.1.2
[notice] To update, run: pip install --upgrade pip
```

In [41]:
```
import plotly.io as pio
```

In [42]:
```
# Dictionary to map full state names to abbreviations
state_abbr = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'California': 'CA', 'Co
    'Connecticut': 'CT', 'Delaware': 'DE', 'Florida': 'FL', 'Georgia': 'GA', 'Hawaii': 'HI', 'I
    'Illinois': 'IL', 'Indiana': 'IN', 'Iowa': 'IA', 'Kansas': 'KS', 'Kentucky': 'KY', 'Louisia
    'Maine': 'ME', 'Maryland': 'MD', 'Massachusetts': 'MA', 'Michigan': 'MI', 'Minnesota': 'MN
    'Missouri': 'MO', 'Montana': 'MT', 'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH'
    'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND', 'Ohio'
    'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI', 'South Carolina': 'SC', 'South
    'Tennessee': 'TN', 'Texas': 'TX', 'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA', 'Washing
    'West Virginia': 'WV', 'Wisconsin': 'WI', 'Wyoming': 'WY'
}

# Convert full state names to abbreviations
df_weighted['State'] = df_weighted['State'].map(state_abbr)

# Create a choropleth map using Plotly Express
fig = px.choropleth(df_weighted,
                    locations='State',
                    locationmode='USA-states',
                    color='total_score',
                    scope='usa',
                    color_continuous_scale='Viridis',
                    labels={'total_score': 'Demand Score'},
                    title='Demand by State')

fig.show()
```

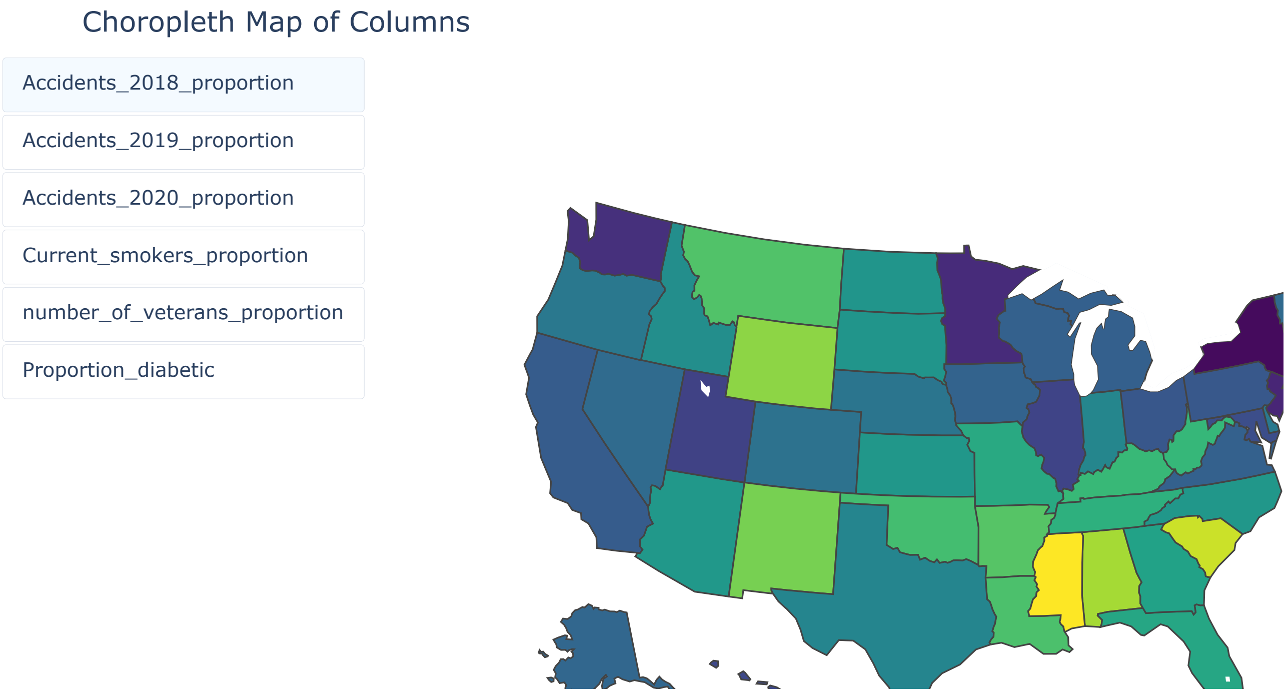
Demand by State



In [43]:
```
#fig.write_html("visual_two.html")
#pio.write_image(fig, "visual_two.png", format='png', engine='kaleido')
```

In [44]:

```
# List of columns to plot excluding 'State' and 'total_score'
columns_to_plot = [col for col in df_weighted.columns if col not in ['State', 'total_score']]

# Create a Choropleth trace for each column
traces = [go.Choropleth(
    locations=df_weighted['State'],
    z=df_weighted[col],
    locationmode='USA-states',
    colorscale='Viridis',
    visible=False,
    name=col,
    showscale=True,
    hovertemplate=f"{col}: %{col}<extra></extra>"
) for col in columns_to_plot]

# Create a Figure with the Choropleth traces
fig = go.Figure(data=traces)

# Set the layout to display the map of the USA and add a title
fig.update_layout(
    title='Choropleth Map of Columns',
    geo=dict(scope='usa', projection=dict(type='albers usa')),
    margin=dict(l=0, r=0, b=0, t=30)
)

# Add a checkbox updatemenu to control the visibility of each trace
fig.update_layout(
    updatemenus=[
        dict(
            type='buttons',
            showactive=True,
            buttons=[dict(label=col,
                          method='update',
                          args=[{'visible': [col == trace.name for trace in traces],
                                 'showscale': [True for _ in traces]
                                 }])
                     for col in columns_to_plot]
        )
    ]
)

fig.show()
```


Choropleth Map of Columns
Accidents_2018_proportion
Accidents_2019_proportion
Accidents_2020_proportion
Current_smokers_proportion
number_of_veterans_proportion
Proportion_diabetic

In [45]:
```
df_pops = df_pops.reset_index()
```

In [46]:
```
# get data by year
df_pops.head()
accidents = df_pops[["State", "Accidents_2018_proportion", "Accidents_2019_proportion", "Accide
accidents.head()

# Select the columns to be scaled excluding the 'State' column
columns_to_scale = [col for col in accidents.columns if col != 'State']

# Apply Min-Max scaling to the selected columns
scaled_data = scaler.fit_transform(accidents[columns_to_scale])

# Create a new df with the scaled data and the same column names excluding 'State'
df_scaled_values = pd.DataFrame(scaled_data, columns=columns_to_scale)

# Add the 'State' column back to the new DataFrame
df_scaled = pd.concat([df['State'], df_scaled_values], axis=1)

# Display the scaled df
df_scaled.head()

accidents_scaled = df_scaled
accidents_scaled

newcolnames = {'Accidents_2018_proportion': '2018', 'Accidents_2019_proportion': '2019', 'Accid
accidents_scaled = accidents_scaled.rename(columns=newcolnames)
accidents_scaled.head()

accidents_scaled["change"] = accidents_scaled["2020"] - accidents_scaled["2018"]
acc = accidents_scaled
acc.head()
```

Out[46]:

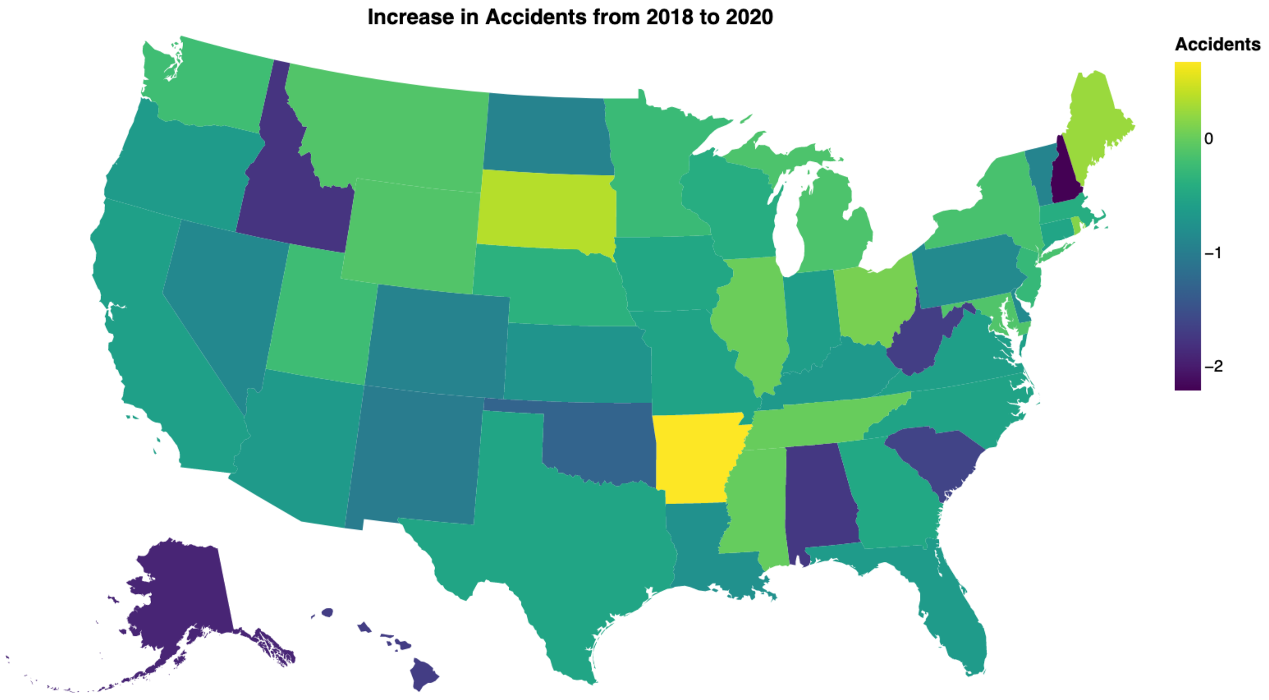
	State	2018	2019	2020	change
0	Alabama	8.630789	8.140420	6.888059	-1.742729
1	Alaska	3.284478	3.041397	1.363555	-1.920923
2	Arizona	5.300329	5.311685	4.657240	-0.643089
3	Arkansas	7.304089	7.127526	7.983250	0.679161
4	California	2.896501	3.153820	2.323174	-0.573327

In [47]:
```
import altair as alt
from vega_datasets import data

# Load US States GeoJSON data
us_states = alt.topo_feature(data.us_10m.url, 'states')
```

In [48]:

```python
# Dictionary to map state abbreviations to FIPS codes
state_name_to_fips = {
    'Alabama': '1',
    'Alaska': '2',
    'Arizona': '4',
    'Arkansas': '5',
    'California': '6',
    'Colorado': '8',
    'Connecticut': '9',
    'Delaware': '10',
    'Florida': '12',
    'Georgia': '13',
    'Hawaii': '15',
    'Idaho': '16',
    'Illinois': '17',
    'Indiana': '18',
    'Iowa': '19',
    'Kansas': '20',
    'Kentucky': '21',
    'Louisiana': '22',
    'Maine': '23',
    'Maryland': '24',
    'Massachusetts': '25',
    'Michigan': '26',
    'Minnesota': '27',
    'Mississippi': '28',
    'Missouri': '29',
    'Montana': '30',
    'Nebraska': '31',
    'Nevada': '32',
    'New Hampshire': '33',
    'New Jersey': '34',
    'New Mexico': '35',
    'New York': '36',
    'North Carolina': '37',
    'North Dakota': '38',
    'Ohio': '39',
    'Oklahoma': '40',
    'Oregon': '41',
    'Pennsylvania': '42',
    'Rhode Island': '44',
    'South Carolina': '45',
    'South Dakota': '46',
    'Tennessee': '47',
    'Texas': '48',
    'Utah': '49',
    'Vermont': '50',
    'Virginia': '51',
    'Washington': '53',
    'West Virginia': '54',
    'Wisconsin': '55',
    'Wyoming': '56'
}

acc['id'] = acc['State'].map(state_name_to_fips)
acc['Accidents'] = acc['change']

state_data = acc
```

In [49]:
```
# Define the color scale for the choropleth map
color_scale = alt.Scale(scheme='viridis', domain=[state_data['Accidents'].min(), state_data['Ac

# Create the choropleth map
choropleth_map = alt.Chart(us_states).mark_geoshape().encode(
    alt.Color('Accidents:Q', scale=color_scale, legend=alt.Legend(title='Accidents')),
    tooltip=['Accidents:Q']
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(state_data, 'id', ['Accidents'])
).project(
    type='albersUsa'
).properties(
    width=700,
    height=400,
    title='Increase in Accidents from 2018 to 2020'
)

choropleth_map.display()
```



In [50]:
```
choropleth_map.save("visual_three.html")

# Install altair_saver
#!pip install altair_saver

# Import altair_saver
#import altair_saver

# Save the chart as a PNG
#altair_saver.save(choropleth_map, "visual_three.png")
```

In [51]:

```
# finding supply side data

# Read the Excel file
file_path = 'ProsthOrth_Data.xlsx'
supply_df = pd.read_excel(file_path, skiprows=5, nrows=48)

supply_df.head()
```

Out[51]:

	Area Name	Employment(1)	Employment percent relative standard error(3)
0	Alabama	170	29.1
1	Alaska	-	-
2	Arizona	230	24.7
3	Arkansas	160	38.4
4	California	1060	18.8

In [52]:

```
# wrangling

supply_df = supply_df.rename(columns={'Area Name': 'State'})

supply_df.head()
```

Out[52]:

	State	Employment(1)	Employment percent relative standard error(3)
0	Alabama	170	29.1
1	Alaska	-	-
2	Arizona	230	24.7
3	Arkansas	160	38.4
4	California	1060	18.8

In [53]:

```
# clean the data of any state values that are blank or list them as 0(?)

# List of states to drop
states_to_drop = ['Alaska', 'Colorado', 'Delaware', 'Vermont', 'Wisconsin', 'District of Columb

# Remove rows containing the specified states
cleaned_supply_df = supply_df[~supply_df['State'].isin(states_to_drop)]

# Reset the index after removing the rows
cleaned_supply_df.reset_index(drop=True, inplace=True)

# Display the cleaned dataframe
# print(cleaned_supply_df)

# rename columns

# Rename the 'Employment(1)' column
cleaned_supply_df = cleaned_supply_df.rename(columns={'Employment(1)': 'Number of Prosthetists

# Display the updated dataframe
cleaned_supply_df.head()
```

Out[53]:

	State	Number of Prosthetists	Employment percent relative standard error(3)
0	Alabama	170	29.1
1	Arizona	230	24.7
2	Arkansas	160	38.4
3	California	1060	18.8
4	Connecticut	70	25.0

In [54]:
```
# scale these values by population

cleaned_supply_df = cleaned_supply_df.merge(pops, on='State')
```

In [55]:
```
cleaned_supply_df.head()
```

Out[55]:

	State	Number of Prosthetists	Employment percent relative standard error(3)	Population
0	Alabama	170	29.1	4903185
1	Arizona	230	24.7	7278717
2	Arkansas	160	38.4	3017804
3	California	1060	18.8	39512223
4	Connecticut	70	25.0	3565287

In [56]:
```
#now add a column to find the prosthetists by capita

cleaned_supply_df['Number of Prostheticists'] = pd.to_numeric(cleaned_supply_df['Number of Prost
cleaned_supply_df['Population'] = pd.to_numeric(cleaned_supply_df['Population'], errors='coerce

# Calculate the 'prosthetists per capita' column
cleaned_supply_df['prosthetists per capita'] = cleaned_supply_df['Number of Prosthetists'] / c

# Display the updated df
cleaned_supply_df.head()
```

Out[56]:

	State	Number of Prosthetists	Employment percent relative standard error(3)	Population	prosthetists per capita
0	Alabama	170	29.1	4903185	0.000035
1	Arizona	230	24.7	7278717	0.000032
2	Arkansas	160	38.4	3017804	0.000053
3	California	1060	18.8	39512223	0.000027
4	Connecticut	70	25.0	3565287	0.000020

In [57]:
```
#min max scale

scaler = MinMaxScaler(feature_range=(0, 10))

# Scale the 'prosthetists per capita' column using the scaler and create the 'Supply Score' co
cleaned_supply_df['Supply Score'] = scaler.fit_transform(cleaned_supply_df[['prosthetists per c
```

In [58]:
```
cleaned_supply_df.head()
```

Out[58]:

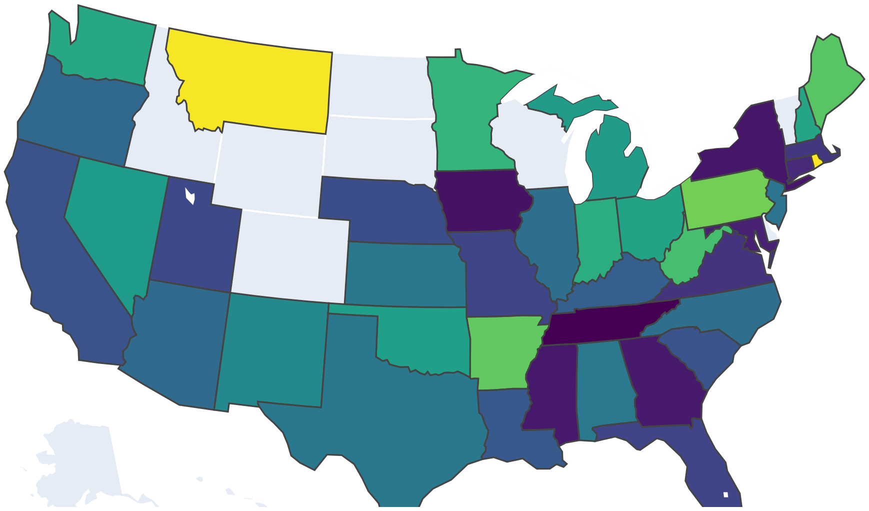
	State	Number of Prosthetists	Employment percent relative standard error(3)	Population	prosthetists per capita	Supply Score
0	Alabama	170	29.1	4903185	0.000035	4.062962
1	Arizona	230	24.7	7278717	0.000032	3.482162
2	Arkansas	160	38.4	3017804	0.000053	7.531347
3	California	1060	18.8	39512223	0.000027	2.580093
4	Connecticut	70	25.0	3565287	0.000020	1.220256

```
In [59]: !pip install geopandas matplotlib
```

```
Requirement already satisfied: geopandas in /Users/mihirgupta/Documents/College Other/Colleg
e/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (0.12.2)
Requirement already satisfied: matplotlib in /Library/Frameworks/Python.framework/Versions/
3.11/lib/python3.11/site-packages (3.6.3)
Requirement already satisfied: pandas>=1.0.0 in /Users/mihirgupta/Documents/College Other/Co
llege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from geopandas) (1.5.2)
Requirement already satisfied: shapely>=1.7 in /Users/mihirgupta/Documents/College Other/Col
lege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from geopandas) (2.0.1)
Requirement already satisfied: fiona>=1.8 in /Users/mihirgupta/Documents/College Other/Colle
ge/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from geopandas) (1.9.2)
Requirement already satisfied: pyproj>=2.6.1.post1 in /Users/mihirgupta/Documents/College Ot
her/College/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from geopandas) (3.
5.0)
Requirement already satisfied: packaging in /Users/mihirgupta/Documents/College Other/Colleg
e/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from geopandas) (23.0)
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Python.framework/Vers
ions/3.11/lib/python3.11/site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.framework/Version
s/3.11/lib/python3.11/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Python.framework/Ver
sions/3.11/lib/python3.11/site-packages (from matplotlib) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Library/Frameworks/Python.framework/Ver
sions/3.11/lib/python3.11/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.19 in /Users/mihirgupta/Documents/College Other/Coll
ege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from matplotlib) (1.24.1)
Requirement already satisfied: pillow>=6.2.0 in /Library/Frameworks/Python.framework/Version
s/3.11/lib/python3.11/site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in /Library/Frameworks/Python.framework/Vers
ions/3.11/lib/python3.11/site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /Users/mihirgupta/Documents/College O
ther/College/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from matplotlib)
(2.8.2)
Requirement already satisfied: attrs>=19.2.0 in /Users/mihirgupta/Documents/College Other/Co
llege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from fiona>=1.8->geopanda
s) (22.2.0)
Requirement already satisfied: certifi in /Users/mihirgupta/Documents/College Other/College/
Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from fiona>=1.8->geopandas) (20
22.12.7)
Requirement already satisfied: click~=8.0 in /Users/mihirgupta/Documents/College Other/Colle
ge/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from fiona>=1.8->geopandas)
(8.1.3)
Requirement already satisfied: click-plugins>=1.0 in /Users/mihirgupta/Documents/College Oth
er/College/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from fiona>=1.8->geo
pandas) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /Users/mihirgupta/Documents/College Other/Colle
ge/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from fiona>=1.8->geopandas)
(0.7.2)
Requirement already satisfied: munch>=2.3.2 in /Users/mihirgupta/Documents/College Other/Col
lege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from fiona>=1.8->geopanda
s) (2.5.0)
Requirement already satisfied: pytz>=2020.1 in /Users/mihirgupta/Documents/College Other/Col
lege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from pandas>=1.0.0->geopan
das) (2022.7)
Requirement already satisfied: six>=1.5 in /Users/mihirgupta/Documents/College Other/Colleg
e/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from python-dateutil>=2.7->ma
tplotlib) (1.16.0)

[notice] A new release of pip is available: 23.0.1 -> 23.1.2
[notice] To update, run: pip install --upgrade pip
```

In [60]:
```
# Convert full state names to abbreviations
cleaned_supply_df['State'] = cleaned_supply_df['State'].map(state_abbr)

# Create a choropleth map using Plotly Express
fig = px.choropleth(cleaned_supply_df,
                    locations='State',
                    locationmode='USA-states',
                    color='Supply Score',
                    scope='usa',
                    color_continuous_scale='Viridis',
                    labels={'Supply Score': 'Supply Score'},
                    title='Supply Scores by State')

# Show the plot
fig.show()
# 10 = lots of prostethists
```

Supply Scores by State



In [62]:
```
# fig.write_html("visual_four.html")
# pio.write_image(fig, "visual_four.png", format='png', engine='kaleido')
```

In [63]:
```
cleaned_supply_df.head()
```

Out[63]:

	State	Number of Prosthetists	Employment percent relative standard error(3)	Population	prosthetists per capita	Supply Score
0	AL	170	29.1	4903185	0.000035	4.062962
1	AZ	230	24.7	7278717	0.000032	3.482162
2	AR	160	38.4	3017804	0.000053	7.531347
3	CA	1060	18.8	39512223	0.000027	2.580093
4	CT	70	25.0	3565287	0.000020	1.220256

In [64]:
```
df_weighted.head()
```

Out[64]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	numbe
0	AL	0.112200	0.105825	0.089545	0.078145	
1	AK	0.042698	0.039538	0.017726	0.075626	
2	AZ	0.068904	0.069052	0.060544	0.040941	
3	AR	0.094953	0.092658	0.103782	0.084366	
4	CA	0.037655	0.041000	0.030201	0.018003	

In [65]:
```
# create a df that has supply and demand scores so we can compare/contrast them

# Merge the df on the "State" column
SupplyVSDemand_df = pd.merge(df_weighted, cleaned_supply_df, on='State')

# Display the new df
SupplyVSDemand_df.head()
```

Out[65]:

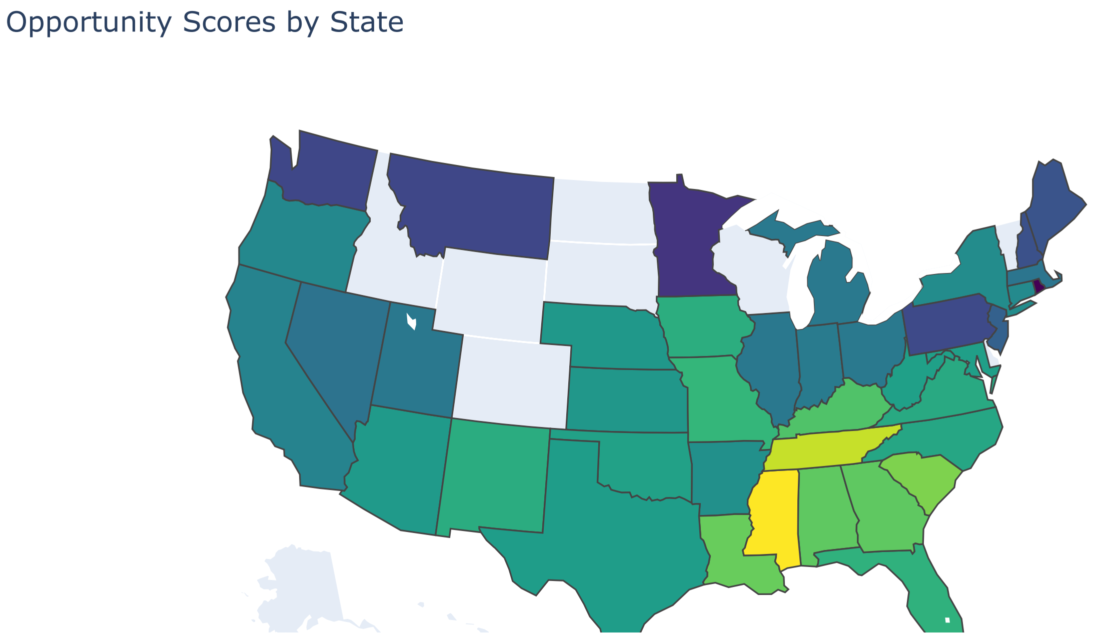
	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	numbe
0	AL	0.112200	0.105825	0.089545	0.078145	
1	AZ	0.068904	0.069052	0.060544	0.040941	
2	AR	0.094953	0.092658	0.103782	0.084366	
3	CA	0.037655	0.041000	0.030201	0.018003	
4	CT	0.028534	0.025222	0.021841	0.029328	

In [66]:
```
#make new column for opportunity zones
SupplyVSDemand_df['Opportunity Score'] = SupplyVSDemand_df['total_score'] - SupplyVSDemand_df[

# Display the updated df
SupplyVSDemand_df.head()
```

Out[66]:

	State	Accidents_2018_proportion	Accidents_2019_proportion	Accidents_2020_proportion	Current_smokers_proportion	numbe
0	AL	0.112200	0.105825	0.089545	0.078145	
1	AZ	0.068904	0.069052	0.060544	0.040941	
2	AR	0.094953	0.092658	0.103782	0.084366	
3	CA	0.037655	0.041000	0.030201	0.018003	
4	CT	0.028534	0.025222	0.021841	0.029328	

```
In [67]: # Create a choropleth map using Plotly Express
         fig = px.choropleth(SupplyVSDemand_df,
                             locations='State',
                             locationmode='USA-states',
                             color='Opportunity Score',
                             scope='usa',
                             color_continuous_scale='Viridis',
                             labels={'Opportunity Score': 'Opportunity Score'},
                             title='Opportunity Scores by State')

         # Show the plot
         fig.show()
```



```
In [70]: # fig.write_html("visual_six.html")
         # pio.write_image(fig, "visual_six.png", format='png', engine='kaleido')
```

```
In [71]: # Make a chart of accidents
         df.head()
         df_accidents = df[['State','Accidents_2018', 'Accidents_2019', 'Accidents_2020']]
         df_accidents.head()
         df_accidents = df_accidents.rename(columns={'Accidents_2018':2018, 'Accidents_2019':2019, 'Acc
         df_accidents.head()
```

Out[71]:

	State	2018	2019	2020
0	Alabama	876	856	852
1	Alaska	69	62	53
2	Arizona	918	908	967
3	Arkansas	476	473	585
4	California	3485	3427	3558

In [72]:

```
df_melted = df_accidents.melt(id_vars=['State'], value_vars=[2018, 2019, 2020], var_name='Year

df_sorted = df_melted.sort_values("State")
df_sorted = df_sorted.rename(columns={"Value":"Accidents"})
df_sorted
```

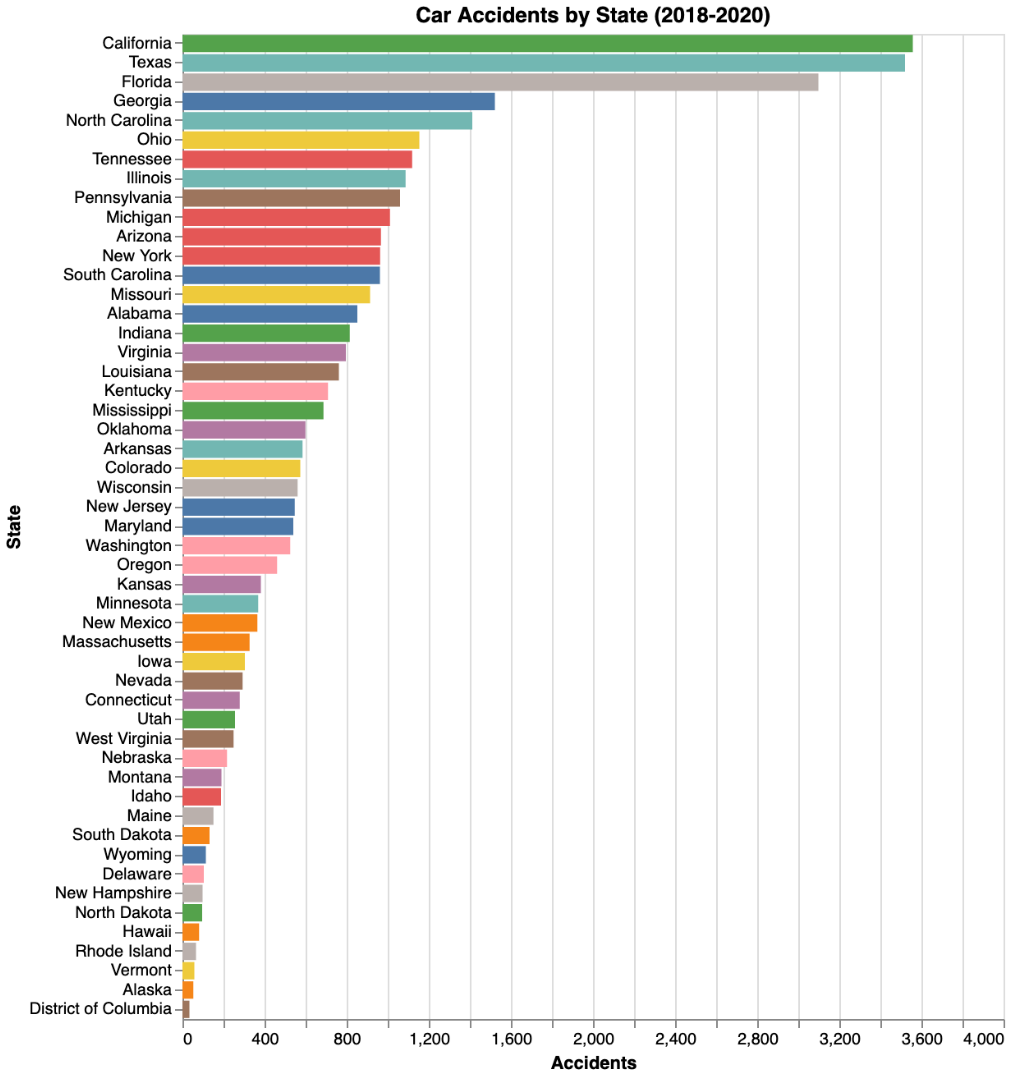
Out[72]:

	State	Year	Accidents
0	Alabama	2018	876
51	Alabama	2019	856
102	Alabama	2020	852
1	Alaska	2018	69
52	Alaska	2019	62
...	...	...	...
100	Wisconsin	2019	527
49	Wisconsin	2018	531
50	Wyoming	2018	100
101	Wyoming	2019	120
152	Wyoming	2020	114

153 rows × 3 columns

In [75]:
```
# create a slider input
slider = alt.binding_range(min=2018, max=2020, step=1)

slider_selection = alt.selection_single(bind=slider, fields=['Year'], name="Select", value=201

# create a chart
car_accidents = alt.Chart(df_sorted).mark_bar().encode(
    y=alt.Y('State:N', sort='-x'),
    x='Accidents:Q',
    color=alt.Color('State:N', legend=None)
).properties(
    width=500,
    height=600,
    title='Car Accidents by State (2018-2020)'
).add_selection(
    slider_selection
).transform_filter(
    slider_selection
)

car_accidents
```

Out[75]:



Select_Year

2020

In [78]:

```
# Read in the CSV file
df_income = pd.read_csv('incomedata.csv')

# Display the first few rows of the dataset
df_income.head()
```

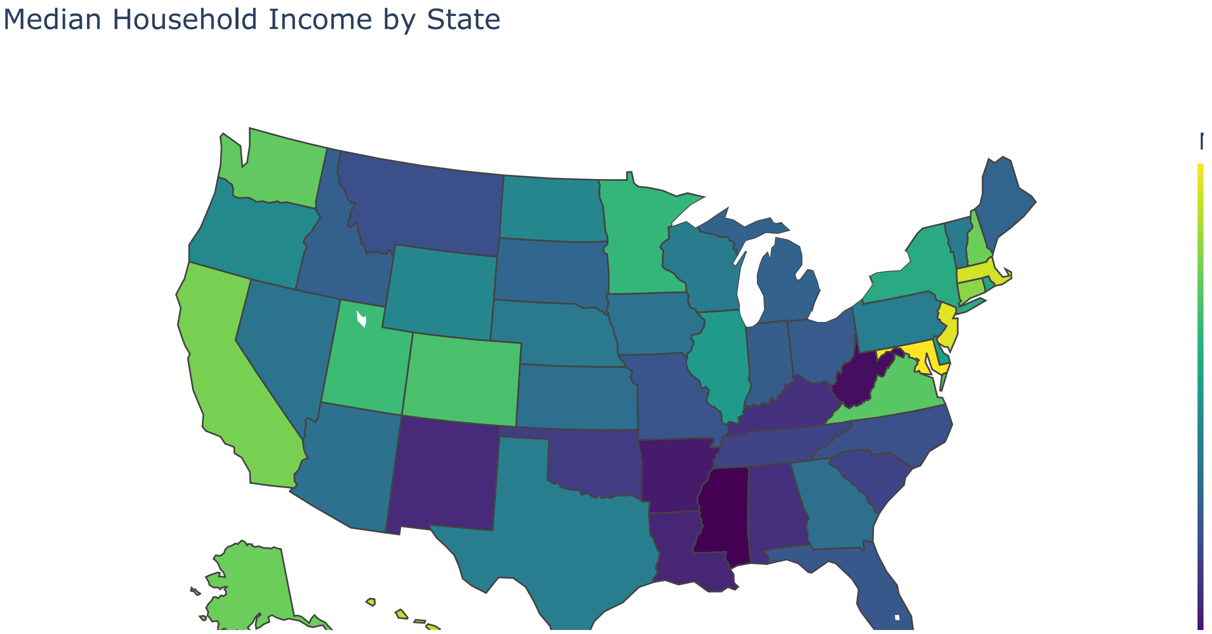
Out[78]:

	fips	state	densityMi	pop2023	pop2022	pop2020	pop2019	pop2010	growthRate	growth	growthSince2010
0	11	District of Columbia	11062.54098	674815	671803	670868	663953.5	601723	0.00448	3012	0.12147
1	24	Maryland	634.04862	6154710	6164660	6173205	6133239.7	5773552	-0.00161	-9950	0.06602
2	34	New Jersey	1258.55820	9255437	9261699	9271689	9223709.5	8791894	-0.00068	-6262	0.05272
3	25	Massachusetts	894.13564	6974258	6981974	6995729	6950919.0	6547629	-0.00111	-7716	0.06516
4	15	Hawaii	223.14152	1433238	1440196	1451043	1441968.8	1360301	-0.00483	-6958	0.05362

In [79]:

```
import plotly.io as pio
import plotly.express as px
```

In [80]:
```
df_income = pd.read_csv('incomedata.csv')

# Remove the row with "state" as "District of Columbia"
df_income = df_income[df_income['state'] != 'District of Columbia']

# Convert full state names to abbreviations
df_income['state'] = df_income['state'].map(state_abbr)

# Create a color map with the income distribution of each state
fig = px.choropleth(
    df_income,
    locations='state',
    locationmode='USA-states',
    color='HouseholdIncome',
    scope='usa',
    title='Median Household Income by State',
    hover_name='state',
    color_continuous_scale= 'Viridis',
    labels={'HouseholdIncome': 'Median Household Income'}
)

# Show the plot
fig.show()
```



In [81]:

```
pip install pandas openpyxl
```

```
Requirement already satisfied: pandas in /Users/mihirgupta/Documents/College Other/College/N
otes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (1.5.2)
Requirement already satisfied: openpyxl in /Users/mihirgupta/Documents/College Other/Colleg
e/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (3.0.10)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/mihirgupta/Documents/College
Other/College/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from pandas) (2.
8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/mihirgupta/Documents/College Other/Col
lege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from pandas) (2022.7)
Requirement already satisfied: numpy>=1.21.0 in /Users/mihirgupta/Documents/College Other/Co
llege/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from pandas) (1.24.1)
Requirement already satisfied: et-xmlfile in /Users/mihirgupta/Documents/College Other/Colle
ge/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from openpyxl) (1.1.0)
Requirement already satisfied: six>=1.5 in /Users/mihirgupta/Documents/College Other/Colleg
e/Notes/3.1/C S 313E/CS313E/venv/lib/python3.11/site-packages (from python-dateutil>=2.8.1->
pandas) (1.16.0)

[notice] A new release of pip is available: 23.0.1 -> 23.1.2
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [82]:

```
# Read the Excel file into a pandas df
df = pd.read_excel('MKTPROSDATA.xlsx', engine='openpyxl')

# Print the DataFrame to see the data
df.head()
```

Out[82]:

	Region	2021	2022	2023	2024	2025	2026	2027	2028	2029	...
0	U.S	31000	32705	34503.775	36401.48263	38403.56417	40515.76020	42744.12701	45095.05400	47575.28196	...
1	Rest of World	57500	59225	61001.750	62831.80250	64716.75658	66658.25927	70657.75483	74897.22012	79391.05333	...
2	Global	88500	91930	95505.525	99233.28513	103120.32070	107174.01950	113401.88180	119992.27410	126966.33530	...

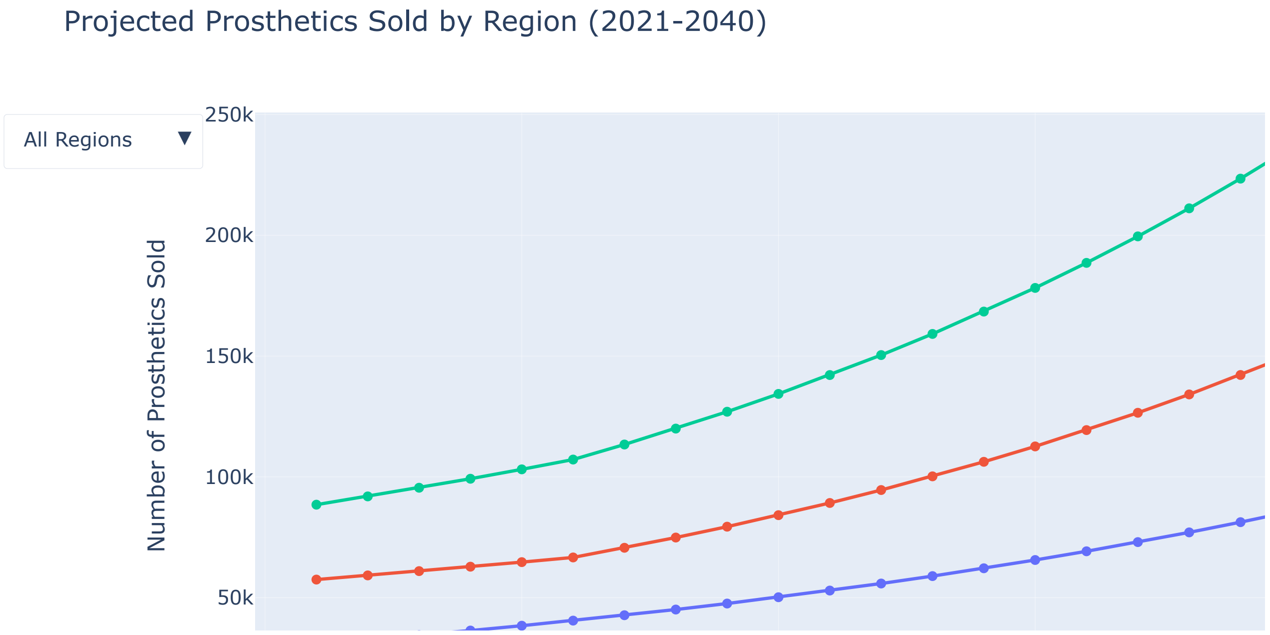
3 rows × 21 columns

In [83]:

```
import plotly.graph_objs as go
```

In [84]:

```
# Melt the df to make it suitable for plotting
df_melted = df.melt(id_vars='Region', var_name='Year', value_name='Prosthetics Sold')

# Create traces for each region
traces = []
for region in df_melted['Region'].unique():
    trace = go.Scatter(
        x=df_melted[df_melted['Region'] == region]['Year'],
        y=df_melted[df_melted['Region'] == region]['Prosthetics Sold'],
        mode='lines+markers',
        name=region,
        visible=(region == 'U.S')
    )
    traces.append(trace)

# Create a layout with a dropdown menu
layout = go.Layout(
    title='Projected Prosthetics Sold by Region (2021-2040)',
    xaxis={'title': 'Year'},
    yaxis={'title': 'Number of Prosthetics Sold'},
    updatemenus=[
        {
            'buttons': [
                {
                    'label': 'All Regions',
                    'method': 'update',
                    'args': [
                        {'visible': [True, True, True]},
                        {'title': 'Projected Prosthetics Sold by Region (2021-2040)'}
                    ]
                }
            ] + [
                {
                    'label': region,
                    'method': 'update',
                    'args': [
                        {'visible': [region == r for r in df_melted['Region'].unique()]},
                        {'title': f'Projected Prosthetics Sold in {region} (2021-2040)'}
                    ]
                }
                for region in df_melted['Region'].unique()
            ],
            'direction': 'down',
            'showactive': True
        }
    ]
)

# Create a Figure and show the plot
fig = go.Figure(data=traces, layout=layout)
fig.show()
```


Projected Prosthetics Sold by Region (2021-2040)

In [85]:
```
df2 = pd.read_excel('MKTLIMBLDATA.xlsx', engine='openpyxl')

# Print the DataFrame to see the data
df2.head()
```

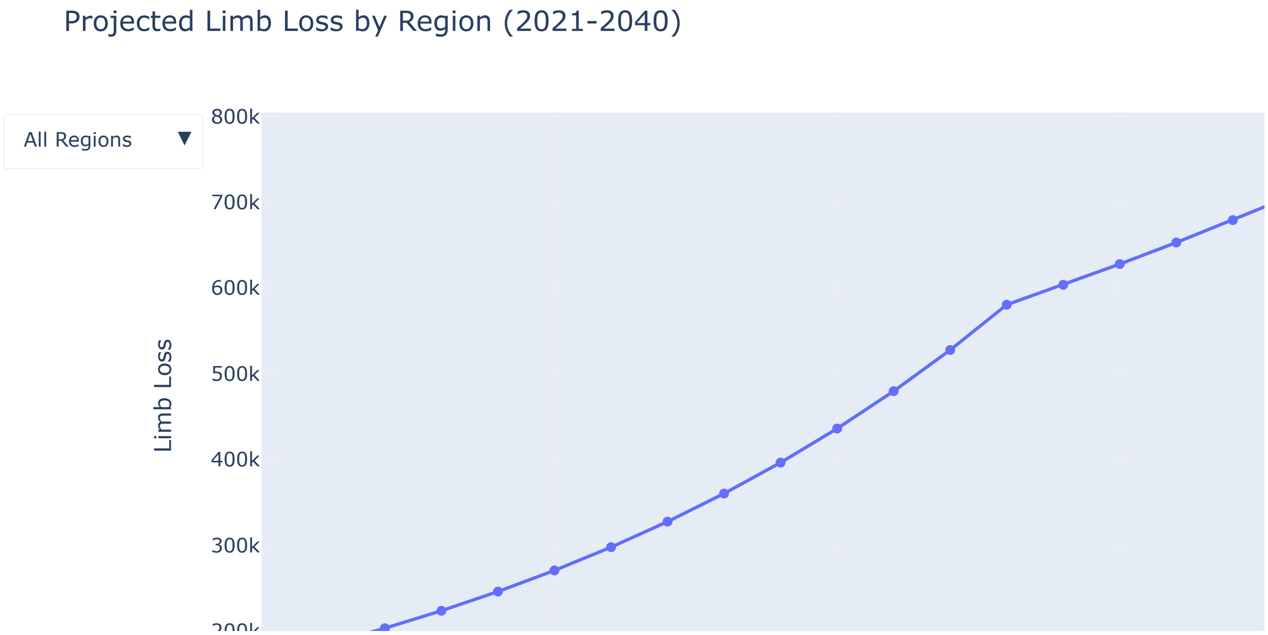
Out[85]:

	Region	2021	2022	2023	2024	2025	2026	2027	2028	2029	...	
0	U.S	185000	203500	223850	246235.0	270858.500	297944.350	327738.785	3.605127e+05	3.965639e+05	...	4.7
1	Rest of World	950000	1007000	1067420	1131465.2	1199353.112	1271314.299	1347593.157	1.428449e+06	1.514156e+06	...	1.7
2	Global	1135000	1210500	1291270	1377700.2	1470211.612	1569258.649	1675331.942	1.788961e+06	1.910720e+06	...	2.1

3 rows × 21 columns

In [86]:

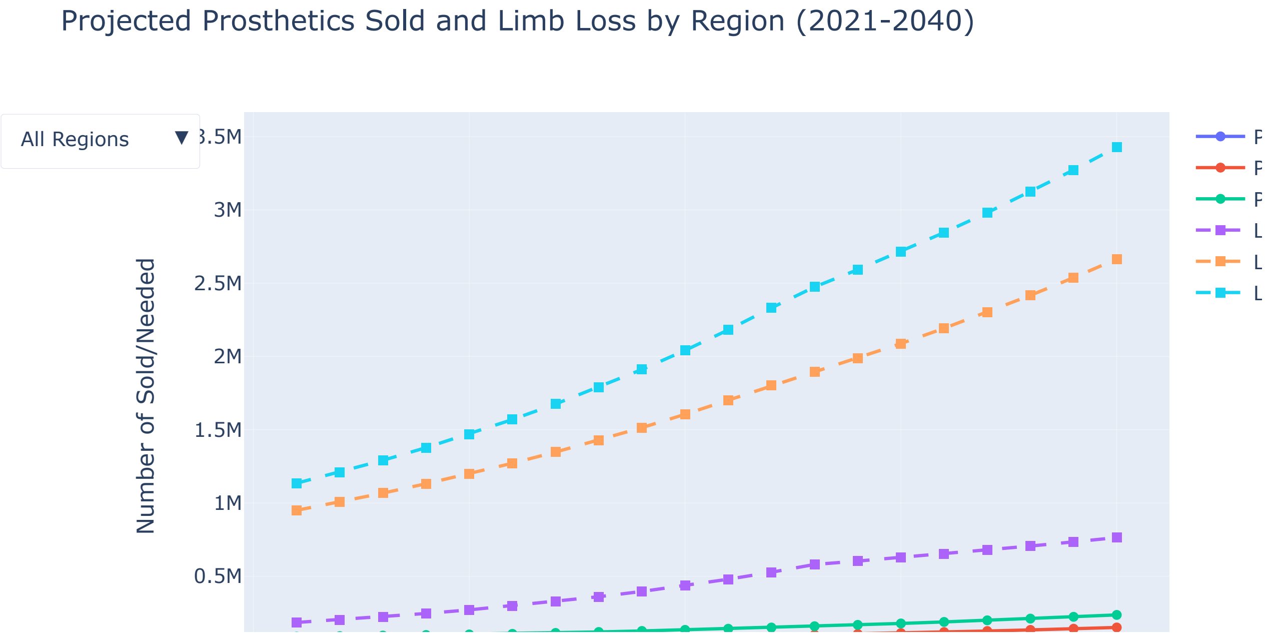
```python
# Melt the DataFrame to make it suitable for plotting
df_melted2 = df2.melt(id_vars='Region', var_name='Year', value_name='Limb Loss')

# Create traces for each region
traces = []
for region in df_melted['Region'].unique():
    trace = go.Scatter(
        x=df_melted2[df_melted2['Region'] == region]['Year'],
        y=df_melted2[df_melted2['Region'] == region]['Limb Loss'],
        mode='lines+markers',
        name=region,
        visible=(region == 'U.S')  # Show only the 'U.S' trace initially
    )
    traces.append(trace)

# Create a layout with a dropdown menu
layout = go.Layout(
    title='Projected Limb Loss by Region (2021-2040)',
    xaxis={'title': 'Year'},
    yaxis={'title': 'Limb Loss'},
    updatemenus=[
        {
            'buttons': [
                {
                    'label': 'All Regions',
                    'method': 'update',
                    'args': [
                        {'visible': [True, True, True]},
                        {'title': 'Projected Limb Loss by Region (2021-2040)'}
                    ]
                }
            ] + [
                {
                    'label': region,
                    'method': 'update',
                    'args': [
                        {'visible': [region == r for r in df_melted2['Region'].unique()]},
                        {'title': f'Projected Limb Loss in {region} (2021-2040)'}
                    ]
                }
                for region in df_melted2['Region'].unique()
            ],
            'direction': 'down',
            'showactive': True
        }
    ]
)

# Create a Figure and show the plot
fig = go.Figure(data=traces, layout=layout)
fig.show()
```


Projected Limb Loss by Region (2021-2040)
All Regions ▼
Limb Loss
800k
700k
600k
500k
400k
300k
200k

In [87]:

```
df_melted = df.melt(id_vars='Region', var_name='Year', value_name='Prosthetics Sold')
df2_melted = df2.melt(id_vars='Region', var_name='Year', value_name='Limb Loss')

# Create traces for prosthetics sold
prosthetics_traces = []
for region in df_melted['Region'].unique():
    trace = go.Scatter(
        x=df_melted[df_melted['Region'] == region]['Year'],
        y=df_melted[df_melted['Region'] == region]['Prosthetics Sold'],
        mode='lines+markers',
        name=f'Prosthetics Sold ({region})',
        marker=dict(symbol='circle'),
        visible=(region == 'U.S')  # Show only the 'U.S' trace initially
    )
    prosthetics_traces.append(trace)

# Create traces for limb loss
limb_loss_traces = []
for region in df2_melted['Region'].unique():
    trace = go.Scatter(
        x=df2_melted[df2_melted['Region'] == region]['Year'],
        y=df2_melted[df2_melted['Region'] == region]['Limb Loss'],
        mode='lines+markers',
        name=f'Limb Loss ({region})',
        marker=dict(symbol='square'),
        line=dict(dash='dash'),
        visible=False
    )
    limb_loss_traces.append(trace)

# Combine the traces
all_traces = prosthetics_traces + limb_loss_traces

# Create a layout with a dropdown menu
layout = go.Layout(
    title='Projected Prosthetics Sold and Limb Loss by Region (2021-2040)',
    xaxis={'title': 'Year'},
    yaxis={'title': 'Number of Sold/Needed'},
    updatemenus=[
        {
            'buttons': [
                {
                    'label': 'All Regions',
                    'method': 'update',
                    'args': [
                        {'visible': [True, True, True] * 2},
                        {'title': 'Projected Prosthetics Sold and Limb Loss by Region (2021-20
                    ]
                }
            ] + [
                {
                    'label': region,
                    'method': 'update',
                    'args': [
                        {'visible': [(region == r) or (region == r2) for r, r2 in zip(df_melte
                        {'title': f'Projected Prosthetics Sold and Limb Loss in {region} (2021-
                    ]
                }
                for region in df_melted['Region'].unique()
            ],
            'direction': 'down',
            'showactive': True
        }
    ]
)

# Create a Figure and show the plot
fig = go.Figure(data=all_traces, layout=layout)
```

```
fig.show()
```


Projected Prosthetics Sold and Limb Loss by Region (2021-2040)
All Regions ▼
Number of Sold/Needed
3.5M
3M
2.5M
2M
1.5M
1M
0.5M
P
P
P
L
L
L

```
In [ ]:
```