In [ ]:
```
#objectives:
#finding which states in India had the highest number of road accidents during each ye
#finding whether there is a trend or pattern in the number of road accidents over the
#finding how the accidents per lakh population change from 2017 to 2020
#finding whether there are any states with consistently high or low rates
#finding whether there are  any correlations between the number of road accidents and
#finding whether there is any relationship between the number of persons killed and th
#finding whether the states are consistently high or low rates?
#finding the rate of persons killed per lakh population in each state from 2017 to 202
#finding the correlation between the number of persons killed and the number of vehicl
#finding how many persons were injured in road accidents in each year from 2017 to 202
#finding whether the states with consistently high numbers of injuries
#finding the correlation between the number of injuries and the number of vehicles or
#finding how the road accidents per 10,000 vehicles change from 2016 to 2019? Are ther
#finding how the road accidents per 10,000 vehicles change from 2016 to 2019? Are ther
```

In [16]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Setting up Jupyter Notebook for inline plotting
%matplotlib inline

# Setting up default plot styles
sns.set_style('darkgrid')
```

In [17]:
```
import pandas as pd

# Read the CSV file
data = pd.read_csv('C:/Users/BTM/Downloads/road accidents from 2017 to 2020.csv')

# Display the first few rows of the DataFrame
data.head()
```

Out[17]:

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	Acc 1 Veh
0	Andhra Pradesh	24475	21992	19509.0	28.7	27.1	24.2	21.5	
1	Arunachal Pradesh	277	237	134.0	18.0	20.5	17.3	9.8	
2	Assam	8248	8350	6595.0	21.6	24.6	24.7	19.5	
3	Bihar	9600	10007	8639.0	8.3	8.9	9.2	8.0	
4	Chhattisgarh	13864	13899	11656.0	51.2	51.8	51.4	43.1	

In [18]:
```
# Checking for duplicate rows
duplicate_rows = data.duplicated()

# Counting the number of duplicate rows
num_duplicates = duplicate_rows.sum()

# Displaying the number of duplicate rows
print("Number of duplicate rows:", num_duplicates)
```

```
Number of duplicate rows: 0
```

In [19]:
```
# Checking for missing values
missing_values = data.isnull().sum()

# Displaying the columns with missing values
columns_with_missing_values = missing_values[missing_values > 0]
print("Columns with missing values:")
print(columns_with_missing_values)
```

```
Columns with missing values:
Road Accidents  during 2020                              1
Accidents  Per Lakh Population - 2017                    1
Total Number of Accidents  Per Lakh Population - 2018    1
Total Number of Accidents  Per Lakh Population - 2019    1
Total Number of Accidents  Per Lakh Population - 2020    1
Accidents per 10,000 Vehicles - 2018                     1
Accidents per 10,000 Vehicles - 2019                     1
Persons Killed 2020                                      1
Persons Killed Per Lakh Population - 2017                1
Persons Killed Per Lakh Population - 2018                1
Persons Killed Per Lakh Population - 2019                1
Persons Killed Per Lakh Population - 2020                1
Persons Injured - 2020                                   1
Injury Per Lakh Population - 2017                        1
Road Accidents per 10,000 Vehicles- 2017                 1
Road Accidents per 10,000 Vehicles- 2018                 1
Road Accidents per 10,000 Vehicles- 2019                 1
dtype: int64
```

In [20]:
```
# Calculate the mean of each column
column_means = data.mean()

# Replace missing values with column means
data_filled = data.fillna(column_means)

# Update the original dataset with the filled values
data.update(data_filled)

# Display the updated dataset
data.head()
```

```
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\4107981681.py:2: FutureWarning: The d
efault value of numeric_only in DataFrame.mean is deprecated. In a future version, it
will default to False. In addition, specifying 'numeric_only=None' is deprecated. Sel
ect only valid columns or specify the value of numeric_only to silence this warning.
  column_means = data.mean()
```

Out[20]:

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	Acc 1 Veh
0	Andhra Pradesh	24475	21992	19509.0	28.7	27.1	24.2	21.5	
1	Arunachal Pradesh	277	237	134.0	18.0	20.5	17.3	9.8	
2	Assam	8248	8350	6595.0	21.6	24.6	24.7	19.5	
3	Bihar	9600	10007	8639.0	8.3	8.9	9.2	8.0	
4	Chhattisgarh	13864	13899	11656.0	51.2	51.8	51.4	43.1	

In [21]:
```
# Remove the last row
data = data.drop(data.index[-1])

# Display the modified DataFrame
data.head()
```

Out[21]:

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	Acc 1 Veh
0	Andhra Pradesh	24475	21992	19509.0	28.7	27.1	24.2	21.5	
1	Arunachal Pradesh	277	237	134.0	18.0	20.5	17.3	9.8	
2	Assam	8248	8350	6595.0	21.6	24.6	24.7	19.5	
3	Bihar	9600	10007	8639.0	8.3	8.9	9.2	8.0	
4	Chhattisgarh	13864	13899	11656.0	51.2	51.8	51.4	43.1	

In [22]:
```
# Adjust the display options
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Display the full DataFrame
data
```

Out[22]:

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	A V
0	Andhra Pradesh	24475	21992	19509.0	28.700000	27.100000	24.200000	21.500000	
1	Arunachal Pradesh	277	237	134.0	18.000000	20.500000	17.300000	9.800000	
2	Assam	8248	8350	6595.0	21.600000	24.600000	24.700000	19.500000	
3	Bihar	9600	10007	8639.0	8.300000	8.900000	9.200000	8.000000	
4	Chhattisgarh	13864	13899	11656.0	51.200000	51.800000	51.400000	43.100000	
5	Goa	3709	3440	2375.0	189.400000	175.200000	158.500000	109.400000	
6	Gujarat	18769	17046	13398.0	29.700000	28.900000	26.000000	20.400000	
7	Haryana	11238	10944	9431.0	39.800000	39.300000	37.700000	32.500000	
8	Himachal Pradesh	3110	2873	2239.0	43.200000	42.800000	39.300000	30.600000	
9	Jammu and Kashmir	5978	5796	4860.0	44.400000	46.800000	45.000000	37.700000	
10	Jharkhand	5394	5217	4405.0	15.100000	15.500000	14.800000	12.500000	
11	Karnataka	41707	40658	34178.0	67.100000	65.200000	63.100000	53.100000	
12	Kerala	40181	41111	27877.0	106.700000	110.900000	112.900000	76.600000	
13	Madhya Pradesh	51397	50669	45266.0	66.700000	63.400000	61.700000	55.100000	
14	Maharashtra	35717	32925	24971.0	29.200000	28.700000	26.200000	19.900000	
15	Manipur	601	672	432.0	21.800000	22.500000	24.900000	16.000000	
16	Meghalaya	399	482	214.0	23.800000	14.000000	16.700000	7.400000	
17	Mizoram	53	62	53.0	6.300000	4.800000	5.600000	4.800000	
18	Nagaland	430	358	500.0	21.900000	17.500000	14.500000	20.200000	
19	Odisha	11262	11064	9817.0	25.200000	25.900000	25.300000	22.400000	
20	Punjab	6428	6348	5203.0	21.200000	21.500000	21.100000	17.300000	
21	Rajasthan	21743	23480	19114.0	29.500000	28.700000	30.600000	24.900000	
22	Sikkim	180	162	138.0	29.700000	27.000000	24.100000	20.500000	
23	Tamil Nadu	63920	57228	45484.0	93.600000	90.900000	81.000000	64.400000	
24	Telangana	22230	21570	19172.0	37.144444	36.408333	34.102778	26.130556	
25	Tripura	552	655	466.0	12.900000	14.000000	16.400000	11.700000	
26	Uttarakhand	1468	1352	1041.0	14.700000	13.300000	12.100000	9.400000	

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	A V
27	Uttar Pradesh	42568	42572	34243.0	17.200000	18.700000	18.400000	14.800000	
28	West Bengal	12705	10158	9180.0	12.200000	13.300000	10.500000	9.500000	
29	Andaman and Nicobar Islands	254	230	141.0	32.900000	43.200000	38.300000	23.500000	
30	Chandigarh	316	305	159.0	17.600000	15.600000	14.400000	7.500000	
31	Dadra and Nagar Haveli	80	68	100.0	14.800000	17.100000	14.000000	20.600000	
32	Daman and Diu	76	69	20341.0	22.000000	20.300000	17.600000	0.000000	
33	Delhi	6515	5610	4178.0	29.600000	28.100000	23.600000	17.500000	
34	Lakshadweep	3	1	1.0	1.200000	3.600000	1.200000	1.200000	
35	Puducherry	1597	1392	969.0	94.200000	85.500000	71.600000	49.800000	

In [23]:
```
# Display the names of the columns
column_names = data.columns
print(column_names)
```

```
Index(['State', 'Road Accidents  during 2018', ' Road Accidents  during 2019',
       'Road Accidents  during 2020', 'Accidents  Per Lakh Population - 2017',
       'Total Number of Accidents  Per Lakh Population - 2018',
       'Total Number of Accidents  Per Lakh Population - 2019',
       'Total Number of Accidents  Per Lakh Population - 2020',
       'Accidents per 10,000 Vehicles - 2017',
       'Accidents per 10,000 Vehicles - 2018',
       'Accidents per 10,000 Vehicles - 2019',
       'Accidents per 10,000 Km of Roads - 2017',
       'Accidents per 10,000 Km of Roads - 2018', 'Persons Killed 2017',
       'Persons Killed 2018', 'Persons Killed 2019', 'Persons Killed 2020',
       'Share in Death- 2017', 'Share in Death- 2018', 'Share in Death- 2019',
       'Share in Death- 2020', 'Persons Killed Per Lakh Population - 2017',
       'Persons Killed Per Lakh Population - 2018',
       'Persons Killed Per Lakh Population - 2019',
       'Persons Killed Per Lakh Population - 2020',
       'Persons Killed per 10,000 Vehicles - 2017',
       'Persons Killed per 10,000 Vehicles - 2018',
       'Persons Killed per 10,000 Vehicles - 2019',
       'Persons Killed per 10,000 Km of Roads - 2017',
       'Persons Killed per 10,000 Km of Roads - 2018',
       'Persons Injured - 2017', 'Persons Injured - 2018',
       'Persons Injured - 2019', 'Persons Injured - 2020',
       'Injury Per Lakh Population - 2017',
       'Injury Per Lakh Population - 2018',
       'Injury Per Lakh Population - 2019',
       'Injury Per Lakh Population - 2020',
       'Road Accidents per 10,000 Vehicles- 2016',
       'Road Accidents per 10,000 Vehicles- 2017',
       'Road Accidents per 10,000 Vehicles- 2018',
       'Road Accidents per 10,000 Vehicles- 2019',
       'Injury per 10,000 Km of Roads - 2016',
       'Injury per 10,000 Km of Roads - 2017',
       'Injury per 10,000 Km of Roads - 2018'],
      dtype='object')
```

In [24]:

```
# Define the column names to be updated
columns_to_rename = ['Road Accidents  during 2018', 'Road Accidents  during 2019', 'Ro

# Define the corresponding new column names
new_column_names = ['Accidents_2018', 'Accidents_2019', 'Accidents_2020']

# Strip leading and trailing whitespaces from column names
data.columns = data.columns.str.strip()

# Rename the selected columns
data.rename(columns=dict(zip(columns_to_rename, new_column_names)), inplace=True)

# Display the updated column names
print(data.columns)
```

```
Index(['State', 'Accidents_2018', 'Accidents_2019', 'Accidents_2020',
       'Accidents  Per Lakh Population - 2017',
       'Total Number of Accidents  Per Lakh Population - 2018',
       'Total Number of Accidents  Per Lakh Population - 2019',
       'Total Number of Accidents  Per Lakh Population - 2020',
       'Accidents per 10,000 Vehicles - 2017',
       'Accidents per 10,000 Vehicles - 2018',
       'Accidents per 10,000 Vehicles - 2019',
       'Accidents per 10,000 Km of Roads - 2017',
       'Accidents per 10,000 Km of Roads - 2018', 'Persons Killed 2017',
       'Persons Killed 2018', 'Persons Killed 2019', 'Persons Killed 2020',
       'Share in Death- 2017', 'Share in Death- 2018', 'Share in Death- 2019',
       'Share in Death- 2020', 'Persons Killed Per Lakh Population - 2017',
       'Persons Killed Per Lakh Population - 2018',
       'Persons Killed Per Lakh Population - 2019',
       'Persons Killed Per Lakh Population - 2020',
       'Persons Killed per 10,000 Vehicles - 2017',
       'Persons Killed per 10,000 Vehicles - 2018',
       'Persons Killed per 10,000 Vehicles - 2019',
       'Persons Killed per 10,000 Km of Roads - 2017',
       'Persons Killed per 10,000 Km of Roads - 2018',
       'Persons Injured - 2017', 'Persons Injured - 2018',
       'Persons Injured - 2019', 'Persons Injured - 2020',
       'Injury Per Lakh Population - 2017',
       'Injury Per Lakh Population - 2018',
       'Injury Per Lakh Population - 2019',
       'Injury Per Lakh Population - 2020',
       'Road Accidents per 10,000 Vehicles- 2016',
       'Road Accidents per 10,000 Vehicles- 2017',
       'Road Accidents per 10,000 Vehicles- 2018',
       'Road Accidents per 10,000 Vehicles- 2019',
       'Injury per 10,000 Km of Roads - 2016',
       'Injury per 10,000 Km of Roads - 2017',
       'Injury per 10,000 Km of Roads - 2018'],
      dtype='object')
```

In [7]:

```
# Adjust the display options
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Display the full DataFrame
data
```

Out[7]:

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	A V
0	Andhra Pradesh	24475	21992	19509.0	28.700000	27.100000	24.200000	21.500000	
1	Arunachal Pradesh	277	237	134.0	18.000000	20.500000	17.300000	9.800000	
2	Assam	8248	8350	6595.0	21.600000	24.600000	24.700000	19.500000	
3	Bihar	9600	10007	8639.0	8.300000	8.900000	9.200000	8.000000	
4	Chhattisgarh	13864	13899	11656.0	51.200000	51.800000	51.400000	43.100000	
5	Goa	3709	3440	2375.0	189.400000	175.200000	158.500000	109.400000	
6	Gujarat	18769	17046	13398.0	29.700000	28.900000	26.000000	20.400000	
7	Haryana	11238	10944	9431.0	39.800000	39.300000	37.700000	32.500000	
8	Himachal Pradesh	3110	2873	2239.0	43.200000	42.800000	39.300000	30.600000	
9	Jammu and Kashmir	5978	5796	4860.0	44.400000	46.800000	45.000000	37.700000	
10	Jharkhand	5394	5217	4405.0	15.100000	15.500000	14.800000	12.500000	
11	Karnataka	41707	40658	34178.0	67.100000	65.200000	63.100000	53.100000	
12	Kerala	40181	41111	27877.0	106.700000	110.900000	112.900000	76.600000	
13	Madhya Pradesh	51397	50669	45266.0	66.700000	63.400000	61.700000	55.100000	
14	Maharashtra	35717	32925	24971.0	29.200000	28.700000	26.200000	19.900000	
15	Manipur	601	672	432.0	21.800000	22.500000	24.900000	16.000000	
16	Meghalaya	399	482	214.0	23.800000	14.000000	16.700000	7.400000	
17	Mizoram	53	62	53.0	6.300000	4.800000	5.600000	4.800000	
18	Nagaland	430	358	500.0	21.900000	17.500000	14.500000	20.200000	
19	Odisha	11262	11064	9817.0	25.200000	25.900000	25.300000	22.400000	
20	Punjab	6428	6348	5203.0	21.200000	21.500000	21.100000	17.300000	
21	Rajasthan	21743	23480	19114.0	29.500000	28.700000	30.600000	24.900000	
22	Sikkim	180	162	138.0	29.700000	27.000000	24.100000	20.500000	
23	Tamil Nadu	63920	57228	45484.0	93.600000	90.900000	81.000000	64.400000	
24	Telangana	22230	21570	19172.0	37.144444	36.408333	34.102778	26.130556	
25	Tripura	552	655	466.0	12.900000	14.000000	16.400000	11.700000	
26	Uttarakhand	1468	1352	1041.0	14.700000	13.300000	12.100000	9.400000	

	State	Road Accidents during 2018	Road Accidents during 2019	Road Accidents during 2020	Accidents Per Lakh Population - 2017	Total Number of Accidents Per Lakh Population - 2018	Total Number of Accidents Per Lakh Population - 2019	Total Number of Accidents Per Lakh Population - 2020	A V
27	Uttar Pradesh	42568	42572	34243.0	17.200000	18.700000	18.400000	14.800000	
28	West Bengal	12705	10158	9180.0	12.200000	13.300000	10.500000	9.500000	
29	Andaman and Nicobar Islands	254	230	141.0	32.900000	43.200000	38.300000	23.500000	
30	Chandigarh	316	305	159.0	17.600000	15.600000	14.400000	7.500000	
31	Dadra and Nagar Haveli	80	68	100.0	14.800000	17.100000	14.000000	20.600000	
32	Daman and Diu	76	69	20341.0	22.000000	20.300000	17.600000	0.000000	
33	Delhi	6515	5610	4178.0	29.600000	28.100000	23.600000	17.500000	
34	Lakshadweep	3	1	1.0	1.200000	3.600000	1.200000	1.200000	
35	Puducherry	1597	1392	969.0	94.200000	85.500000	71.600000	49.800000	
36	Total	467044	449002	366138.0	35.800000	35.600000	33.800000	27.600000	

In [25]:
```
#finding which states in India had the highest number of road accidents during each ye
# Create a list to store the results
results = []

# Iterate over the years 2018 to 2020
for year in range(2018, 2021):
    # Get the column name for road accidents in the current year
    column_name = f"Accidents_{year}"

    # Find the state with the highest number of road accidents in the current year
    state_with_highest_accidents = data[column_name].idxmax()

    # Get the actual number of road accidents for the state and year
    highest_accidents = data.loc[state_with_highest_accidents, column_name]

    # Get the state name corresponding to the index
    state_name = data.loc[state_with_highest_accidents, 'State']

    # Append the result to the list
    results.append((year, state_name, highest_accidents))

# Display the results
for result in results:
    year, state, accidents = result
    print(f"In {year}, {state} had the highest number of road accidents: {accidents}")
```

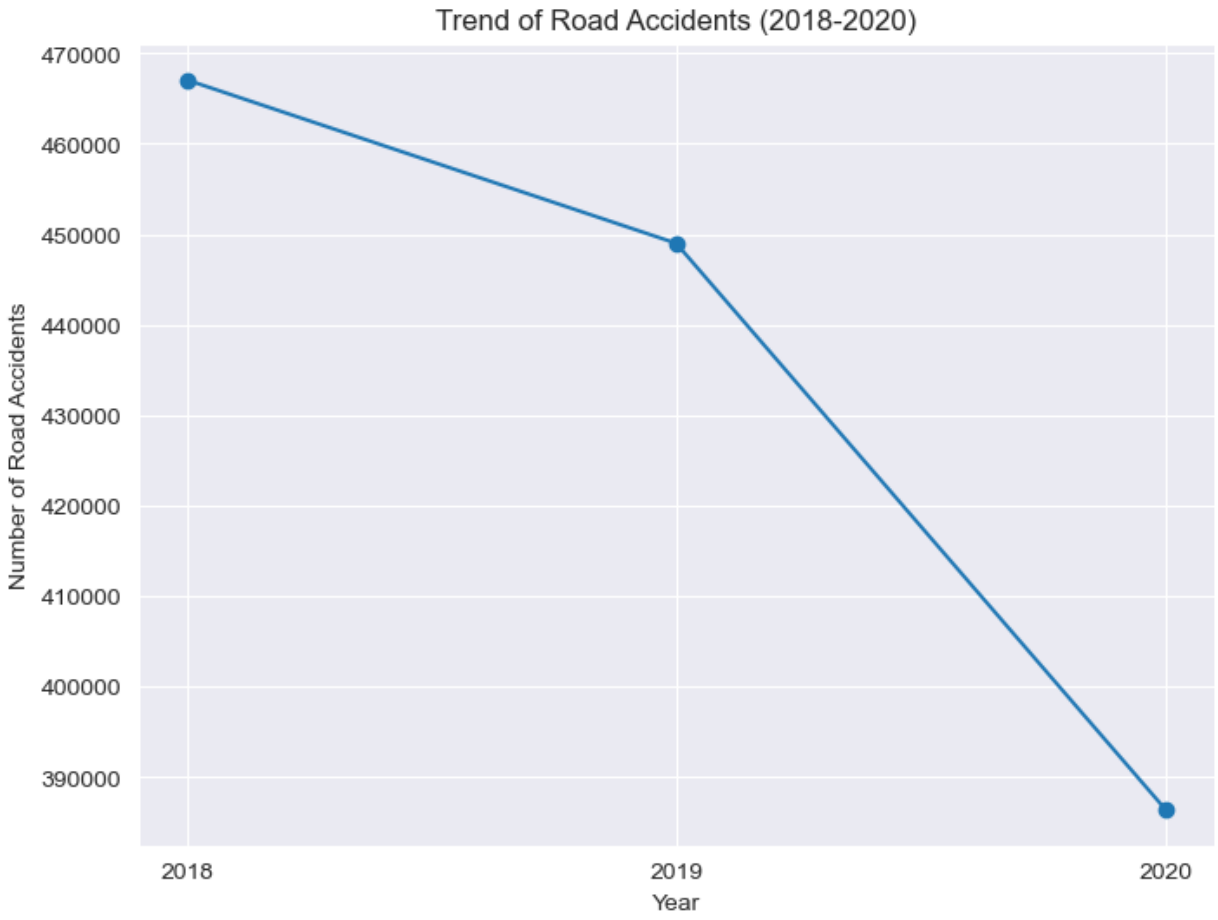
```
In 2018, Tamil Nadu had the highest number of road accidents: 63920
In 2019, Tamil Nadu had the highest number of road accidents: 57228
In 2020, Tamil Nadu had the highest number of road accidents: 45484.0
```

In [26]:
```
#finding whether there is a trend or pattern in the number of road accidents over the
import matplotlib.pyplot as plt

# Define the range of years
years = range(2018, 2021)

# Get the corresponding number of road accidents for each year
accidents = [data[f'Accidents_{year}'].sum() for year in years]

# Plot the line graph
plt.figure(figsize=(8, 6))
plt.plot(years, accidents, marker='o')
plt.xlabel('Year')
plt.ylabel('Number of Road Accidents')
plt.title('Trend of Road Accidents (2018-2020)')
plt.xticks(years)
plt.grid(True)
plt.show()
```

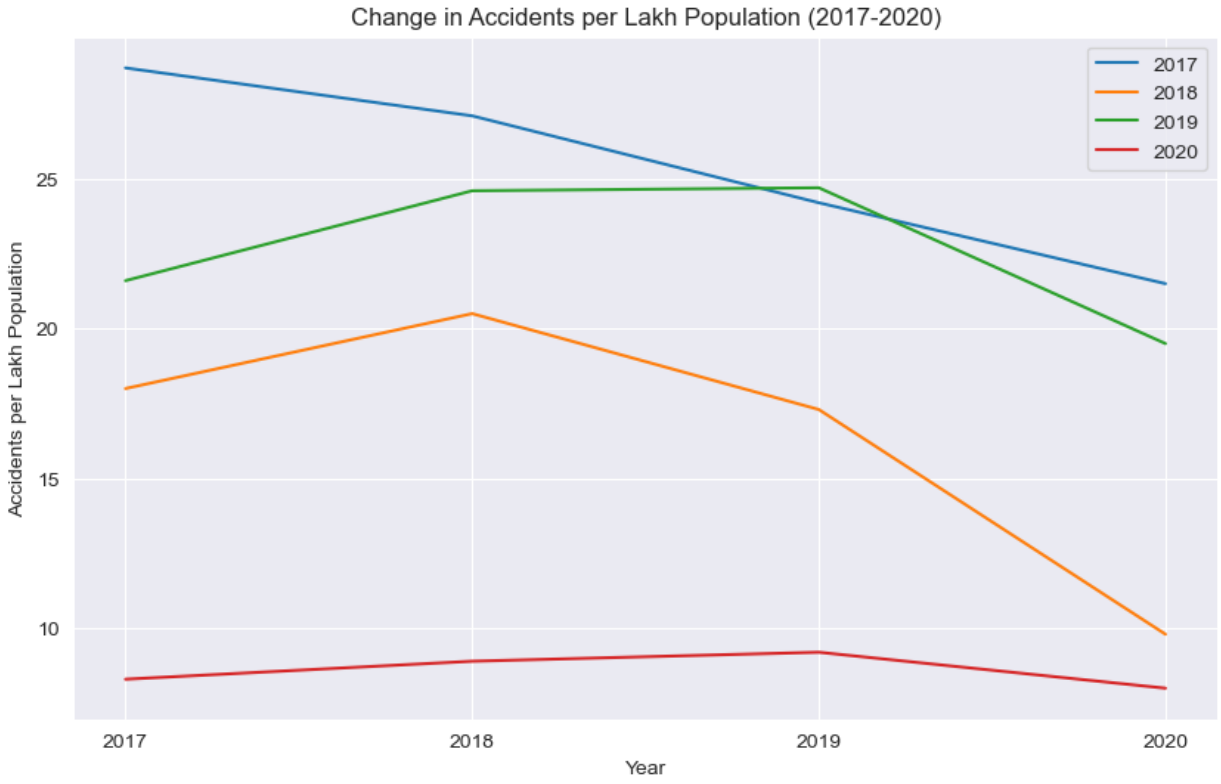


In [27]:
```
#a) finding how the accidents per lakh population change from 2017 to 2020
import matplotlib.pyplot as plt

# Extract the relevant columns for analysis
years = ['2017', '2018', '2019', '2020']
accidents_per_lakh = data[['Accidents  Per Lakh Population - 2017',
                           'Total Number of Accidents  Per Lakh Population - 2018',
                           'Total Number of Accidents  Per Lakh Population - 2019',
                           'Total Number of Accidents  Per Lakh Population - 2020']]
```

```
# Plot the line chart
plt.figure(figsize=(10, 6))
for i in range(len(years)):
    plt.plot(years, accidents_per_lakh.iloc[i], label=years[i])

plt.xlabel('Year')
plt.ylabel('Accidents per Lakh Population')
plt.title('Change in Accidents per Lakh Population (2017-2020)')
plt.legend()
plt.grid(True)
plt.show()
```



In [28]:

```
# Calculate the accidents per lakh population for each year
accidents_per_lakh_2017 = data['Accidents  Per Lakh Population - 2017']
accidents_per_lakh_2018 = data['Total Number of Accidents  Per Lakh Population - 2018'
accidents_per_lakh_2019 = data['Total Number of Accidents  Per Lakh Population - 2019'
accidents_per_lakh_2020 = data['Total Number of Accidents  Per Lakh Population - 2020'

# Calculate the change in accidents per lakh population from 2017 to 2020
change_in_accidents = accidents_per_lakh_2020 - accidents_per_lakh_2017

# Display the change in accidents per lakh population from 2017 to 2020
print("Change in accidents per lakh population from 2017 to 2020:")
print(change_in_accidents)

# Calculate the average rate for each state
average_rates = data[['Accidents  Per Lakh Population - 2017',
                      'Total Number of Accidents  Per Lakh Population - 2018',
                      'Total Number of Accidents  Per Lakh Population - 2019',
                      'Total Number of Accidents  Per Lakh Population - 2020']].mean(axi

# Identify states with consistently high or low rates
high_rate_states = average_rates[average_rates > average_rates.mean() + 2 * average_ra
low_rate_states = average_rates[average_rates < average_rates.mean() - 2 * average_rat
```

```
# Get the names of states with consistently high or low rates
high_rate_states_names = data.loc[high_rate_states.index, 'State']
low_rate_states_names = data.loc[low_rate_states.index, 'State']

# Display the states with consistently high rates
print("\nStates with consistently high rates:")
print(high_rate_states_names)

# Display the states with consistently low rates
print("\nStates with consistently low rates:")
print(low_rate_states_names)
```

```
Change in accidents per lakh population from 2017 to 2020:
0     -7.200000
1     -8.200000
2     -2.100000
3     -0.300000
4     -8.100000
5    -80.000000
6     -9.300000
7     -7.300000
8    -12.600000
9     -6.700000
10    -2.600000
11   -14.000000
12   -30.100000
13   -11.600000
14    -9.300000
15    -5.800000
16   -16.400000
17    -1.500000
18    -1.700000
19    -2.800000
20    -3.900000
21    -4.600000
22    -9.200000
23   -29.200000
24   -11.013889
25    -1.200000
26    -5.300000
27    -2.400000
28    -2.700000
29    -9.400000
30   -10.100000
31     5.800000
32   -22.000000
33   -12.100000
34     0.000000
35   -44.400000
dtype: float64

States with consistently high rates:
5        Goa
12    Kerala
Name: State, dtype: object

States with consistently low rates:
Series([], Name: State, dtype: object)
```

In [29]:
```
#It seems that there are no states identified as consistently low rates in the provide
```

In [30]:
```
#finding whether there are any correlations between the number of road accidents and t
print(data.columns)
```

```
Index(['State', 'Accidents_2018', 'Accidents_2019', 'Accidents_2020',
       'Accidents  Per Lakh Population - 2017',
       'Total Number of Accidents  Per Lakh Population - 2018',
       'Total Number of Accidents  Per Lakh Population - 2019',
       'Total Number of Accidents  Per Lakh Population - 2020',
       'Accidents per 10,000 Vehicles - 2017',
       'Accidents per 10,000 Vehicles - 2018',
       'Accidents per 10,000 Vehicles - 2019',
       'Accidents per 10,000 Km of Roads - 2017',
       'Accidents per 10,000 Km of Roads - 2018', 'Persons Killed 2017',
       'Persons Killed 2018', 'Persons Killed 2019', 'Persons Killed 2020',
       'Share in Death- 2017', 'Share in Death- 2018', 'Share in Death- 2019',
       'Share in Death- 2020', 'Persons Killed Per Lakh Population - 2017',
       'Persons Killed Per Lakh Population - 2018',
       'Persons Killed Per Lakh Population - 2019',
       'Persons Killed Per Lakh Population - 2020',
       'Persons Killed per 10,000 Vehicles - 2017',
       'Persons Killed per 10,000 Vehicles - 2018',
       'Persons Killed per 10,000 Vehicles - 2019',
       'Persons Killed per 10,000 Km of Roads - 2017',
       'Persons Killed per 10,000 Km of Roads - 2018',
       'Persons Injured - 2017', 'Persons Injured - 2018',
       'Persons Injured - 2019', 'Persons Injured - 2020',
       'Injury Per Lakh Population - 2017',
       'Injury Per Lakh Population - 2018',
       'Injury Per Lakh Population - 2019',
       'Injury Per Lakh Population - 2020',
       'Road Accidents per 10,000 Vehicles- 2016',
       'Road Accidents per 10,000 Vehicles- 2017',
       'Road Accidents per 10,000 Vehicles- 2018',
       'Road Accidents per 10,000 Vehicles- 2019',
       'Injury per 10,000 Km of Roads - 2016',
       'Injury per 10,000 Km of Roads - 2017',
       'Injury per 10,000 Km of Roads - 2018'],
      dtype='object')
```

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

# Select the relevant columns
accidents_col = "Accidents_2020"
vehicles_col = "Accidents per 10,000 Vehicles - 2019"
roads_col = "Injury per 10,000 Km of Roads - 2018"
relevant_cols = [accidents_col, vehicles_col, roads_col]

# Create a new DataFrame with only the relevant columns
data2 = data[relevant_cols]

# Calculate the correlations
correlations = data2.corr()

# Display the correlations
print(correlations)
```

```
                                       Accidents_2020  \
Accidents_2020                               1.000000
Accidents per 10,000 Vehicles - 2019         0.466246
Injury per 10,000 Km of Roads - 2018         0.362634

                                       Accidents per 10,000 Vehicles - 2019  \
Accidents_2020                                                     0.466246
Accidents per 10,000 Vehicles - 2019                               1.000000
Injury per 10,000 Km of Roads - 2018                               0.209680

                                       Injury per 10,000 Km of Roads - 2018
Accidents_2020                                                     0.362634
Accidents per 10,000 Vehicles - 2019                               0.209680
Injury per 10,000 Km of Roads - 2018                               1.000000
```

In [32]:
```
#The correlation between "Road Accidents during 2020" and itself is 1.000000, which pe
#The correlation between "Road Accidents during 2020" and "Accidents per 10000 Vehicle
#The correlation between "Road Accidents during 2020" and "Injury per 10,000 Km of Roc
#The correlation between "Accidents per 10,000 Vehicles - 2019" and itself is 1.000000
#The correlation between "Accidents per 10,000 Vehicles - 2019" and "Injury per 10,000
#The correlation between "Injury per 10,000 Km of Roads - 2018" and itself is 1.000000
```

In [14]:
```
import pandas as pd

# Create a new DataFrame with relevant columns
persons_killed_data = data[['State', 'Persons Killed 2017', 'Persons Killed 2018', 'Pe

# Calculate the total number of persons killed over the four-year period
persons_killed_data["Total Persons Killed"] = persons_killed_data.iloc[:, 1:].sum(axis

# Calculate the percentage change in the number of persons killed from 2017 to 2020
persons_killed_data["Percentage Change"] = (persons_killed_data["Persons Killed 2020"]

# Identify states with consistently high fatality rates
high_fatality_states = persons_killed_data[persons_killed_data["Percentage Change"] >

# Display the trend in the number of persons killed and states with consistently high
print("Trend in the number of persons killed from 2017 to 2020:")
print(persons_killed_data[["State", "Persons Killed 2017", "Persons Killed 2018", "Per
print("\nStates with consistently high fatality rates:")
print(high_fatality_states)
```

Trend in the number of persons killed from 2017 to 2020:

	State	Persons Killed 2017	Persons Killed 2018	Persons Killed 2019	Persons Killed 2020
0	Andhra Pradesh	8060	7556	7984	7039.000000
1	Arunachal Pradesh	110	175	127	73.000000
2	Assam	2783	2966	3208	2629.000000
3	Bihar	5554	6729	7205	6699.000000
4	Chhattisgarh	4136	4592	5003	4606.000000
5	Goa	328	262	297	223.000000
6	Gujarat	7289	7996	7390	6170.000000
7	Haryana	5120	5118	5057	4507.000000
8	Himachal Pradesh	1203	1208	1146	893.000000
9	Jammu and Kashmir	926	984	996	728.000000
10	Jharkhand	3256	3542	3801	3044.000000
11	Karnataka	10609	10990	10958	9760.000000
12	Kerala	4131	4303	4440	2979.000000
13	Madhya Pradesh	10177	10706	11249	11141.000000
14	Maharashtra	12264	13261	12788	11569.000000
15	Manipur	136	134	156	127.000000
16	Meghalaya	182	182	179	144.000000
17	Mizoram	60	45	48	42.000000
18	Nagaland	41	39	26	53.000000
19	Odisha	4790	5315		
20	Punjab	4463	4740		
21	Rajasthan	10444	10320		
22	Sikkim	78	85		
23	Tamil Nadu	16157	12216		
24	Telangana	6596	6603		
25	Tripura	161	213		
26	Uttarakhand	942	1047		
27	Uttar Pradesh	20124	22256		
28	West Bengal	5769	5711		
29	Andaman and Nicobar Islands	21	19		
30	Chandigarh	107	98		
31	Dadra and Nagar Haveli	43	54		
32	Daman and Diu	36	35		
33	Delhi	1584	1690		
34	Lakshadweep	0	1		
35	Puducherry	233	226		
36	Total	147913	151417		

```
19                 5333          4738.000000
20                 4525          3898.000000
21                10563          9250.000000
22                   73            47.000000
23                10525          8059.000000
24                 6964          6882.000000
25                  239           192.000000
26                  867           674.000000
27                22655         19149.000000
28                 5500          4927.000000
29                   20            14.000000
30                  104            53.000000
31                   49            64.000000
32                   28          7317.444444
33                 1463          1196.000000
34                    0             0.000000
35                  147           145.000000
36               151113        131714.000000

States with consistently high fatality rates:
3                       Bihar
4                Chhattisgarh
13             Madhya Pradesh
18                   Nagaland
24                  Telangana
25                    Tripura
31    Dadra and Nagar Haveli
32               Daman and Diu
Name: State, dtype: object
```

```
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\299783874.py:7: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  persons_killed_data["Total Persons Killed"] = persons_killed_data.iloc[:, 1:].sum(a
xis=1)
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\299783874.py:10: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  persons_killed_data["Percentage Change"] = (persons_killed_data["Persons Killed 202
0"] - persons_killed_data["Persons Killed 2017"]) / persons_killed_data["Persons Kill
ed 2017"] * 100
```

In [33]:
```
#Trend in the number of persons killed from 2017 to 2020:
#The table displays the number of persons killed in road accidents for each state in t
#Each row represents a state, and the columns represent the respective years.
#The numbers in each cell represent the count of persons killed in road accidents for
#States with consistently high fatality rates:
#Bihar,Chattisgarh,Madhya Pradesh,Nagaland,Telangana,Tripura,Dadra and Nagar Haveli ar
```

In [35]:
```
#finding whether there is any relationship between the number of persons killed and th
# Select relevant columns
data2 = data[['State', 'Persons Killed 2020', 'Persons Killed Per Lakh Population - 20
```

```
# Remove rows with missing values
data2= data2.dropna()

# Perform correlation analysis
correlation_population = data2['Persons Killed 2020'].corr(data['Persons Killed Per La

# Print the correlation
print("Correlation between Persons Killed and Population:", correlation_population)
```

```
Correlation between Persons Killed and Population: 0.38376893251227645
```

In [36]:
```
#The correlation coefficient measures the strength and direction of the linear relatic
#In this case, a correlation coefficient of 0.38376893251227645 suggests a weak positi
```

In [37]:
```
#finding how the share of deaths in road accidents change from 2017 to 2020? and Are t
# Extract relevant columns
data2 = data[['State', 'Share in Death- 2017', 'Share in Death- 2018', 'Share in Death

# Calculate change in share of deaths from 2017 to 2020
data2['Change'] = data2['Share in Death- 2020'] - data['Share in Death- 2017']

# Sort by the absolute change in descending order
significant_changes = data2.sort_values('Change', ascending=False)

# Display the DataFrame
significant_changes
```

```
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\2622516805.py:6: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  data2['Change'] = data2['Share in Death- 2020'] - data['Share in Death- 2017']
```

Out[37]:

	State	Share in Death- 2017	Share in Death- 2018	Share in Death- 2019	Share in Death- 2020	Change
3	Bihar	3.8	4.4	4.8	4.4	0.6
13	Madhya Pradesh	6.9	7.1	7.4	7.4	0.5
4	Chhattisgarh	2.8	3.0	3.3	3.0	0.2
24	Telangana	4.5	4.4	4.6	4.6	0.1
18	Nagaland	0.0	0.0	0.0	0.0	0.0
17	Mizoram	0.0	0.0	0.0	0.0	0.0
15	Manipur	0.1	0.1	0.1	0.1	0.0
25	Tripura	0.1	0.1	0.2	0.1	0.0
29	Andaman and Nicobar Islands	0.0	0.0	0.0	0.0	0.0
31	Dadra and Nagar Haveli	0.0	0.0	0.0	0.0	0.0
32	Daman and Diu	0.0	0.0	0.0	0.0	0.0
34	Lakshadweep	0.0	0.0	0.0	0.0	0.0
16	Meghalaya	0.1	0.1	0.1	0.1	0.0
9	Jammu and Kashmir	0.6	0.6	0.7	0.5	-0.1
30	Chandigarh	0.1	0.1	0.1	0.0	-0.1
22	Sikkim	0.1	0.1	0.0	0.0	-0.1
1	Arunachal Pradesh	0.1	0.1	0.1	0.0	-0.1
35	Puducherry	0.2	0.1	0.1	0.1	-0.1
5	Goa	0.2	0.2	0.2	0.1	-0.1
19	Odisha	3.2	3.5	3.5	3.1	-0.1
0	Andhra Pradesh	5.4	5.0	5.3	5.3	-0.1
2	Assam	1.9	2.0	2.1	1.7	-0.2
26	Uttarakhand	0.6	0.7	0.6	0.4	-0.2
8	Himachal Pradesh	0.8	0.8	0.8	0.6	-0.2
10	Jharkhand	2.2	2.3	2.5	2.0	-0.2
33	Delhi	1.1	1.1	1.0	0.8	-0.3
20	Punjab	3.0	3.1	3.0	2.6	-0.4
7	Haryana	3.5	3.4	3.3	3.0	-0.5
28	West Bengal	3.9	3.8	3.6	3.3	-0.6
14	Maharashtra	8.3	8.8	8.5	7.7	-0.6
11	Karnataka	7.2	7.3	7.3	6.5	-0.7
12	Kerala	2.8	2.8	2.9	2.0	-0.8

	State	Share in Death- 2017	Share in Death- 2018	Share in Death- 2019	Share in Death- 2020	Change
6	Gujarat	4.9	5.3	4.9	4.1	-0.8
27	Uttar Pradesh	13.6	14.7	15.0	12.7	-0.9
21	Rajasthan	7.1	6.8	7.0	6.1	-1.0
23	Tamil Nadu	10.9	8.1	7.0	5.3	-5.6

In [38]:
```
#finding whether the states are consistently high or low rates?
# Extract relevant columns
data2= data[['State', 'Persons Killed Per Lakh Population - 2017', 'Persons Killed Per

# Check consistency of rates across years
consistently_high = []
consistently_low = []
for index, row in data2.iterrows():
    rates = row[1:].values
    if all(rate > 10 for rate in rates):
        consistently_high.append(row['State'])
    elif all(rate < 1 for rate in rates):
        consistently_low.append(row['State'])

# Calculate average rates for each year
data2['Average Rate'] = data2.mean(axis=1)

# Sort by average rate in descending order
sorted_data2 = data2.sort_values('Average Rate', ascending=False)

# Display the DataFrame
sorted_data2

# Display states with consistently high rates
print("States with consistently high rates (> 10 persons killed per lakh population):"
print(consistently_high)

# Display states with consistently low rates (< 1 person killed per lakh population):
print("States with consistently low rates (< 1 person killed per lakh population):")
print(consistently_low)
```

```
States with consistently high rates (> 10 persons killed per lakh population):
['Chhattisgarh', 'Goa', 'Haryana', 'Himachal Pradesh', 'Karnataka', 'Madhya Pradesh',
'Odisha', 'Punjab', 'Rajasthan', 'Tamil Nadu']
States with consistently low rates (< 1 person killed per lakh population):
[]
```

```
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\861939230.py:15: FutureWarning: Dropp
ing of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depreca
ted; in a future version this will raise TypeError.  Select only valid columns before
calling the reduction.
  data2['Average Rate'] = data2.mean(axis=1)
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\861939230.py:15: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  data2['Average Rate'] = data2.mean(axis=1)
```

In [39]:
```
#finding the rate of persons killed per lakh population in each state from 2017 to 202
# Select relevant columns
data2 = data[['State', 'Persons Killed Per Lakh Population - 2017', 'Persons Killed Pe

# Display the DataFrame
data2
```

Out[39]:

	State	Persons Killed Per Lakh Population - 2017	Persons Killed Per Lakh Population - 2018	Persons Killed Per Lakh Population - 2019	Persons Killed Per Lakh Population - 2020
0	Andhra Pradesh	9.100000	8.40	8.80	7.700000
1	Arunachal Pradesh	8.300000	13.00	9.40	5.300000
2	Assam	8.500000	8.90	9.60	7.800000
3	Bihar	5.300000	6.30	6.70	6.200000
4	Chhattisgarh	15.800000	17.30	18.70	17.000000
5	Goa	16.200000	12.70	14.00	10.300000
6	Gujarat	11.500000	12.50	11.40	9.400000
7	Haryana	18.400000	18.10	17.70	15.500000
8	Himachal Pradesh	16.800000	16.80	15.80	12.200000
9	Jammu and Kashmir	7.400000	7.80	7.80	5.600000
10	Jharkhand	9.600000	10.30	10.90	8.600000
11	Karnataka	16.900000	17.30	17.10	15.200000
12	Kerala	11.500000	11.90	12.30	8.200000
13	Madhya Pradesh	12.900000	13.40	13.90	13.600000
14	Maharashtra	10.100000	10.80	10.30	9.200000
15	Manipur	5.200000	5.10	5.80	4.700000
16	Meghalaya	6.500000	6.40	6.30	5.000000
17	Mizoram	5.600000	4.10	4.40	3.800000
18	Nagaland	1.700000	1.60	1.10	2.100000
19	Odisha	11.200000	12.30	12.30	10.800000
20	Punjab	15.200000	16.00	15.10	12.900000
21	Rajasthan	14.100000	13.80	13.90	12.100000
22	Sikkim	11.900000	12.90	10.90	7.000000
23	Tamil Nadu	23.200000	17.40	15.00	11.400000
24	Telangana	10.083333	10.25	9.75	7.952778
25	Tripura	4.200000	5.50	6.10	4.800000
26	Uttarakhand	8.800000	9.60	7.90	6.100000
27	Uttar Pradesh	9.100000	9.90	9.90	8.300000
28	West Bengal	6.100000	6.00	5.70	5.100000
29	Andaman and Nicobar Islands	3.700000	3.30	3.40	2.300000

	State	Persons Killed Per Lakh Population - 2017	Persons Killed Per Lakh Population - 2018	Persons Killed Per Lakh Population - 2019	Persons Killed Per Lakh Population - 2020
30	Chandigarh	5.800000	5.00	5.10	2.500000
31	Dadra and Nagar Haveli	9.800000	11.90	10.50	13.200000
32	Daman and Diu	10.500000	9.70	7.50	0.000000
33	Delhi	7.200000	7.50	6.30	5.000000
34	Lakshadweep	0.000000	1.20	0.00	0.000000
35	Puducherry	13.400000	12.60	7.90	7.500000

In [43]:
```
#finding the correlation between the number of persons killed and the number of vehicl
# Select relevant columns
data2 = data[['State', 'Persons Killed 2020', 'Total Number of Accidents  Per Lakh Pop

# Remove rows with missing values
data2 = data2.dropna()

# Calculate correlation
correlation_vehicles = data2['Persons Killed 2020'].astype(float).corr(data2['Accident
correlation_roads = data2['Persons Killed 2020'].astype(float).corr(data2['Accidents p

# Print correlation
print(f"Correlation between Persons Killed and Accidents per 10,000 Vehicles (2017): {
print(f"Correlation between Persons Killed and Accidents per 10,000 Km of Roads (2017)
```

```
Correlation between Persons Killed and Accidents per 10,000 Vehicles (2017): 0.155882
67017822405
Correlation between Persons Killed and Accidents per 10,000 Km of Roads (2017): 0.080
29261980866093
```

In [44]:
```
#For the correlation between persons killed and accidents per 10,000 vehicles (2017),
#This indicates a weak positive correlation.
#It means that there is a slight tendency for an increase in the number of accidents p
#However, the correlation is weak, suggesting that the relationship is not very strong
#For the correlation between persons killed and accidents per 10,000 km of roads (2017
#This indicates a very weak negative correlation, close to zero.
#It means that there is almost no relationship between the number of accidents per 10,
#The negative sign indicates a slight tendency for a decrease in the number of accider
```

In [45]:
```
#finding how many persons were injured in road accidents in each year from 2017 to 202
# Select relevant columns
injured_data2 = data[['State', 'Persons Injured - 2017', 'Persons Injured - 2018', 'Pe

# Remove rows with missing values
injured_data2 = injured_data2.dropna()

# Calculate the total number of injured persons for each year
injured_data2['Total Injured'] = injured_data2['Persons Injured - 2017'] + injured_dat

# Sort the data by total number of injured persons
injured_data_sorted2 = injured_data2.sort_values('Total Injured', ascending=False)
```

```
# Print the result
print(injured_data_sorted2)
```

	State	Persons Injured - 2017	Persons Injured - 2018	Persons Injured - 2019	Persons Injured - 2020
23	Tamil Nadu	74571	74537	67137	50551.000000
13	Madhya Pradesh	57532	54662	52816	46456.000000
11	Karnataka	52961	51562	50447	39492.000000
12	Kerala	42671	45458	46055	30510.000000
14	Maharashtra	32128	31365	28628	19914.000000
27	Uttar Pradesh	27494	29664	28932	22410.000000
0	Andhra Pradesh	27475	23456	24619	19675.000000
24	Telangana	23990	23613	21999	18661.000000
21	Rajasthan	22071	21547	22979	16769.000000
6	Gujarat	16802	17467	16258	12002.000000
4	Chhattisgarh	12550	12715	13090	10505.000000
19	Odisha	11198	11794	11177	8822.000000
28	West Bengal	10091	11997	9757	8314.000000
7	Haryana	10339	10020	9362	7659.000000
9	Jammu and Kashmir	7419	7845	7532	5894.000000
3	Bihar	6014	6679	7206	7016.000000
2	Assam	6163	7375	7473	5269.000000
33	Delhi	6604	6086	5152	3662.000000
32	Daman and Diu	70	94	74	19348.833333
8	Himachal Pradesh	5452	5551	4904	3223.000000
10	Jharkhand	3918	3975	3818	3295.000000
20	Punjab	4218			
35	Puducherry	1741			
5	Goa	1922			
26	Uttarakhand	1631			
15	Manipur	1027			
25	Tripura	718			
22	Sikkim	479			
18	Nagaland	375			
1	Arunachal Pradesh	316			
30	Chandigarh	302			
16	Meghalaya	354			
29	Andaman and Nicobar Islands	263			
31	Dadra and Nagar Haveli	60			
17	Mizoram	55			
34	Lakshadweep	1			

20	3384	3812	2904.000000
35	1727	1619	1019.000000
5	1549	1448	880.000000
26	1571	1457	854.000000
15	1042	1055	663.000000
25	741	816	470.000000
22	370	318	218.000000
18	335	246	286.000000
1	323	309	185.000000
30	300	275	148.000000
16	205	222	220.000000
29	260	207	145.000000
31	66	105	119.000000
17	80	56	68.000000
34	3	1	1.000000

	Total Injured
23	266796.000000
13	211466.000000
11	194462.000000
12	164694.000000
14	112035.000000
27	108500.000000
0	95225.000000
24	88263.000000
21	83366.000000
6	62529.000000
4	48860.000000
19	42991.000000
28	40159.000000
7	37380.000000
9	28690.000000
3	26915.000000
2	26280.000000
33	21504.000000
32	19586.833333
8	19130.000000
10	15006.000000
20	14318.000000
35	6106.000000
5	5799.000000
26	5513.000000
15	3787.000000
25	2745.000000
22	1385.000000
18	1242.000000
1	1133.000000
30	1025.000000
16	1001.000000
29	875.000000
31	350.000000
17	259.000000
34	6.000000

In [46]:
```
#finding whether the states with consistently high numbers of injuries
# Select relevant columns
injured_data = data[['State', 'Persons Injured - 2017', 'Persons Injured - 2018', 'Per

# Remove rows with missing values
injured_data = injured_data.dropna()
```

```
# Calculate the total number of injured persons for each year
injured_data['Total Injured'] = injured_data.sum(axis=1)

# Group the data by state and calculate the average number of injured persons
grouped_data = injured_data.groupby('State')['Total Injured'].mean().reset_index()

# Sort the data by average number of injured persons
grouped_data_sorted = grouped_data.sort_values('Total Injured', ascending=False)

# Print the top states with consistently high numbers of injuries
top_states = grouped_data_sorted.head(5)  # Change the number to display more states i
print(top_states)
```

```
             State  Total Injured
30      Tamil Nadu       266796.0
19  Madhya Pradesh       211466.0
16       Karnataka       194462.0
17          Kerala       164694.0
20     Maharashtra       112035.0
```

```
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\1995225103.py:9: FutureWarning: Dropp
ing of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depreca
ted; in a future version this will raise TypeError.  Select only valid columns before
calling the reduction.
  injured_data['Total Injured'] = injured_data.sum(axis=1)
```

In [47]:

```
#finding the rate of injuries per lakh population in each state from 2017 to 2020? Are
# Select relevant columns
injured_data = data[['State', 'Persons Injured - 2017', 'Persons Injured - 2018', 'Per

# Remove rows with missing values
injured_data = injured_data.dropna()

# Create a new DataFrame to store the rates of injuries per lakh population
injury_rates = injured_data[['State']]

# Calculate the average rate of injuries per lakh population for each state
for year in range(2017, 2021):
    column_name = f'Injury Per Lakh Population - {year}'
    injury_rates[f'Rate of Injuries {year}'] = injured_data[column_name]

# Check for consistently high or low rates
consistently_high = []
consistently_low = []

for state in injury_rates['State']:
    rates = injury_rates.loc[injury_rates['State'] == state].drop('State', axis=1).val
    if all(rate > 0 for rate in rates):
        consistently_high.append(state)
    elif all(rate == 0 for rate in rates):
        consistently_low.append(state)

# Print the rates of injuries per lakh population in each state from 2017 to 2020
print(injury_rates)

# Print states with consistently high or low rates
print('States with consistently high rates of injuries:', consistently_high)
print('States with consistently low rates of injuries:', consistently_low)
```

	State	Rate of Injuries 2017	Rate of Injuries 2018	Rate of Injuries 2019	Rate of Injuries 2020
0	Andhra Pradesh	30.858651	26.152011	27.254511	21.633003
1	Arunachal Pradesh	23.813112	24.086503	22.821270	13.533285
2	Assam	18.783907	22.236628	22.296813	15.562973
3	Bihar	5.724130	6.289551	6.716188	6.473997
4	Chhattisgarh	47.926373	48.002869	48.876111	38.812532
5	Goa	95.007415	74.903288	68.398677	40.552995
6	Gujarat	26.446515	27.197845	25.054322	18.314716
7	Haryana	37.099900	35.465260	32.698823	26.408524
8	Himachal Pradesh	76.241085	77.033028	67.557515	44.084257
9	Jammu and Kashmir	59.139099	61.942361	58.935837	45.732464
10	Jharkhand	11.500191	11.527419	10.943905	9.340099
11	Karnataka	84.182668	81.283203	78.902340	61.313461
12	Kerala	118.950185	126.055127	127.079827	83.795661
13	Madhya Pradesh	72.858518	68.291647	65.123735	56.561229
14	Maharashtra	26.440840	25.515351	23.026559	15.841096
15	Manipur	39.213440	39.380197	39.468762	24.573758
16	Meghalaya	12.629326	7.238701	7.762238	7.620367
17	Mizoram	5.121043	7.373272	5.109489	6.148282
18	Nagaland	15.592516	13.786008	10.024450	11.546225
19	Odisha	26.158662	27.343967	25.723820	20.159042
20	Punjab	14.360616	11.422785	12.762823	9.647520
21	Rajasthan	29.856339	28.773837		
22	Sikkim	73.353752	56.060606		
23	Tamil Nadu	106.942493	106.409982		
24	Telangana	38.350220	0.000000		
25	Tripura	18.567365	18.970814		
26	Uttarakhand	15.156584	14.430054		
27	Uttar Pradesh	12.414379	13.194027		
28	West Bengal	10.697098	12.613948		
29	Andaman and Nicobar Islands	46.714032	45.217391		
30	Chandigarh	16.245293	15.455951		
31	Dadra and Nagar Haveli	13.729977	14.601770		
32	Daman and Diu	20.348837	26.183844		
33	Delhi	30.160760	27.021267		
34	Lakshadweep	1.219512	3.614458		
35	Puducherry	100.461627	96.051168		

```
21           30.304109            21.846298
22           47.676162            32.392273
23           95.440976            71.584746
24            0.000000             0.000000
25           20.679169            11.800151
26           13.233424             7.673645
27           12.681020             9.670946
28           10.176793             8.603686
29           35.204082            24.126456
30           13.560158             6.974552
31           22.435897            24.485597
32           19.786096             0.000000
33           22.241409            15.374927
34            1.190476             1.176471
35           86.670236            52.417695
States with consistently high rates of injuries: ['Andhra Pradesh', 'Arunachal Prades
h', 'Assam', 'Bihar', 'Chhattisgarh', 'Goa', 'Gujarat', 'Haryana', 'Himachal Prades
h', 'Jammu and Kashmir', 'Jharkhand', 'Karnataka', 'Kerala', 'Madhya Pradesh', 'Mahar
ashtra', 'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha', 'Punjab', 'Rajastha
n', 'Sikkim', 'Tamil Nadu', 'Tripura', 'Uttarakhand', 'Uttar Pradesh', 'West Bengal',
'Andaman and Nicobar Islands', 'Chandigarh', 'Dadra and Nagar Haveli', 'Delhi', 'Laks
hadweep', 'Puducherry']
States with consistently low rates of injuries: []
```

```
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\3801919201.py:14: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  injury_rates[f'Rate of Injuries {year}'] = injured_data[column_name]
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\3801919201.py:14: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  injury_rates[f'Rate of Injuries {year}'] = injured_data[column_name]
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\3801919201.py:14: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  injury_rates[f'Rate of Injuries {year}'] = injured_data[column_name]
C:\Users\BTM\AppData\Local\Temp\ipykernel_26472\3801919201.py:14: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  injury_rates[f'Rate of Injuries {year}'] = injured_data[column_name]
```

In [48]:

```
#Is there any correlation between the number of injuries and the number of vehicles or
# Select relevant columns
data2 = data[['State', 'Persons Injured - 2017', 'Persons Injured - 2018', 'Persons In
```

```
# Remove rows with missing values
data2 = data2.dropna()

# Calculate correlations
correlation_vehicles = data2[['Persons Injured - 2017', 'Persons Injured - 2018', 'Per
correlation_roads = data2[['Persons Injured - 2017', 'Persons Injured - 2018', 'Person

correlation_vehicles
correlation_roads
```

Out[48]:

	Persons Injured - 2017	Persons Injured - 2018	Persons Injured - 2019	Persons Injured - 2020	Accidents per 10,000 Km of Roads - 2017	Accidents per 10,000 Km of Roads - 2018
Persons Injured - 2017	1.000000	0.998051	0.996582	0.966530	0.248935	0.274104
Persons Injured - 2018	0.998051	1.000000	0.997944	0.964242	0.242427	0.269104
Persons Injured - 2019	0.996582	0.997944	1.000000	0.965645	0.235986	0.260523
Persons Injured - 2020	0.966530	0.964242	0.965645	1.000000	0.262805	0.303184
Accidents per 10,000 Km of Roads - 2017	0.248935	0.242427	0.235986	0.262805	1.000000	0.960231
Accidents per 10,000 Km of Roads - 2018	0.274104	0.269104	0.260523	0.303184	0.960231	1.000000

In [50]:

```
#finding how the road accidents per 10,000 vehicles change from 2016 to 2019? Are ther
# Select relevant columns
data2 = data[['State', 'Road Accidents per 10,000 Vehicles- 2016', 'Road Accidents per

# Remove rows with missing values
data2 = data2.dropna()

# Calculate change in road accidents per 10,000 vehicles
data2['Change'] = data2['Road Accidents per 10,000 Vehicles- 2019'] - data2['Road Acci

# Sort by change in ascending order
sorted_data = data2.sort_values('Change')

# Print the states with significant changes
significant_changes = sorted_data[(sorted_data['Change'] > 0.5) | (sorted_data['Change
print(significant_changes[['State', 'Change']])
```

	State	Change
22	Sikkim	-67.370320
8	Himachal Pradesh	-26.665245
9	Jammu and Kashmir	-22.532764
12	Kerala	-19.070573
13	Madhya Pradesh	-16.442752
15	Manipur	-15.241379
0	Andhra Pradesh	-15.075073
23	Tamil Nadu	-14.057689
11	Karnataka	-13.765088
29	Andaman and Nicobar Islands	-13.386220
25	Tripura	-12.995379
24	Telangana	-12.159255
35	Puducherry	-10.576442
5	Goa	-10.317609
4	Chhattisgarh	-9.611281
10	Jharkhand	-8.619442
19	Odisha	-8.543033
2	Assam	-8.489927
28	West Bengal	-7.026115
21	Rajasthan	-6.721790
14	Maharashtra	-5.901194
26	Uttarakhand	-5.535339
27	Uttar Pradesh	-4.635603
18	Nagaland	-4.018524
1	Arunachal Pradesh	-3.897627
33	Delhi	-3.590258
6	Gujarat	-3.489437
7	Haryana	-3.070041
3	Bihar	-2.766996
30	Chandigarh	-2.144904
20	Punjab	-1.903183
31	Dadra and Nagar Haveli	3.937096
32	Daman and Diu	4.935799

In [51]:
```
#finding how the injuries per 10,000 km of roads change from 2016 to 2018? Are there c
# Select relevant columns
data2 = data[['State', 'Injury per 10,000 Km of Roads - 2016', 'Injury per 10,000 Km c

# Remove rows with missing values
data2 = data2.dropna()

# Calculate change in injuries per 10,000 km of roads
data2['Change'] = data2['Injury per 10,000 Km of Roads - 2018'] - data2['Injury per 10

# Sort by change in ascending order
sorted_data = data2.sort_values('Change')

# Print the states with significant changes
significant_changes = sorted_data[(sorted_data['Change'] > 0.5) | (sorted_data['Change
print(significant_changes[['State', 'Change']])
```

	State	Change
35	Puducherry	-1714.548147
33	Delhi	-819.093311
9	Jammu and Kashmir	-799.196962
29	Andaman and Nicobar Islands	-608.055672
13	Madhya Pradesh	-533.174594
0	Andhra Pradesh	-447.370597
5	Goa	-418.329814
23	Tamil Nadu	-370.258891
22	Sikkim	-345.719408
12	Kerala	-333.534389
7	Haryana	-260.921441
24	Telangana	-223.650122
8	Himachal Pradesh	-185.524410
21	Rajasthan	-134.744246
6	Gujarat	-132.024927
20	Punjab	-121.936828
16	Meghalaya	-112.288345
11	Karnataka	-109.784370
10	Jharkhand	-103.743263
14	Maharashtra	-66.818259
15	Manipur	-52.974724
3	Bihar	-43.263050
1	Arunachal Pradesh	-41.835598
18	Nagaland	-36.813663
4	Chhattisgarh	-33.791392
17	Mizoram	-25.963261
19	Odisha	-25.789367
28	West Bengal	-22.147216
34	Lakshadweep	-3.254951
25	Tripura	6.427856
26	Uttarakhand	8.400237
27	Uttar Pradesh	12.190752
30	Chandigarh	12.421877
2	Assam	30.455660
31	Dadra and Nagar Haveli	355.332502
32	Daman and Diu	607.911449

In [ ]: