In [735]:

```
df1 = pd.read_csv('Data_Washington Fatal Crash Survey.csv')
```

```
C:\Users\19199\AppData\Local\Temp\ipykernel_2552\3835772031.py:1: DtypeWarnin
g: Columns (21,22,23,302,304) have mixed types. Specify dtype option on impor
t or set low_memory=False.
  df1 = pd.read_csv('Data_Washington Fatal Crash Survey.csv')
```

In [737]:

```
df1['x']
```

Out[737]:

```
0      -122.175617
1      -122.583225
2      -122.336422
3      -117.533472
4      -122.334175
           ...    
4127   -118.976794
4128   -778.304936
4129   -122.304894
4130   -122.304894
4131   -123.005892
Name: x, Length: 4132, dtype: float64
```

In [738]:

```
import requests
import pandas as pd

# Set your Mapbox API access token
MAPBOX_ACCESS_TOKEN = 'sk.eyJ1IjoiYWtoaWwyMzQiLCJhIjoiY2xla29yanc5MG04NjN5bDN6

# Define the API endpoint URL
MAPBOX_API_ENDPOINT = 'https://api.mapbox.com/geocoding/v5/mapbox.places/'

# Define the column names for the dataframe
columns = ['x', 'y', 'zipcode']

# Define a function to get the zipcode from x, y coordinates using Mapbox API
def get_zipcode(x, y):
    # Construct the API request URL with the x, y coordinates and API access t
    url = f"{MAPBOX_API_ENDPOINT}{x},{y}.json?types=postcode&access_token={MAP

    # Send a GET request to the API endpoint URL
    response = requests.get(url)

    # Extract the zipcode from the API response
    if response.status_code == 200:
        json_response = response.json()
        features = json_response['features']
        if features:
            zipcode = features[0]['text']
            return zipcode
    return None

# Example dataframe with x, y coordinates
df = pd.DataFrame({
    'x': df1['x'],
    'y': df1['y']
})

# Apply the get_zipcode function to the dataframe to get the zipcode for each
df1['zipcode'] = df.apply(lambda row: get_zipcode(row['x'], row['y']), axis=1)
```

```
In [739]: df1[['x','y','zipcode']]
```

```
Out[739]:
                   x          y  zipcode
         0  -122.175617  48.023547    98201
         1  -122.583225  47.431342    98359
         2  -122.336422  47.688700    98103
         3  -117.533472  47.643175    99224
         4  -122.334175  47.601744    98104
       ...          ...        ...      ...
      4127  -118.976794  48.142939    99155
      4128  -778.304936  78.304936     None
      4129  -122.304894  47.587086    98144
      4130  -122.304894  47.587086    98144
      4131  -123.005892  47.110469    98502

4132 rows × 3 columns
```

```
In [740]: df1['distract1']
```

```
Out[740]:
0       96
1       92
2       93
3       93
4       96
        ..
4127    96
4128    96
4129    96
4130    96
4131    96
Name: distract1, Length: 4132, dtype: int64
```

In [741]:

```
distraction_codes = {
    0: 'Not Distracted',
    1: 'Looked But Did Not See',
    16: 'No Driver Present / Unknown if Driver Present',
    96: 'Not Reported',
    3: 'By Other Occupant(s)',
    4: 'By a Moving Object in Vehicle',
    5: 'While Talking or Listening to Cellular Phone',
    6: 'While Manipulating Cellular Phone',
    7: 'Adjusting Audio or Climate Controls',
    9: 'While Using Other Component/Controls Integral to Vehicle',
    10: 'While Using or Reaching For Device/Object Brought Into Vehicle',
    12: 'Distracted by Outside Person, Object or Event',
    13: 'Eating or Drinking',
    14: 'Smoking Related',
    15: 'Other Cellular Phone Related',
    17: 'Distraction/Inattention',
    18: 'Distraction/Careless',
    19: 'Careless/Inattentive',
    92: 'Distraction (Distracted), Details Unknown',
    93: 'Inattention (Inattentive), Details Unknown',
    97: 'Lost in Thought / Day Dreaming',
    98: 'Other Distraction',
    99: 'Unknown if Distract'
}

df1['distract1'] = df1['distract1'].replace(distraction_codes)
```

In [742]:

```
df1['distract1']
```

Out[742]:

```
0                                     Not Reported
1        Distraction (Distracted), Details Unknown
2       Inattention (Inattentive), Details Unknown
3       Inattention (Inattentive), Details Unknown
4                                     Not Reported
                           ...
4127                                  Not Reported
4128                                  Not Reported
4129                                  Not Reported
4130                                  Not Reported
4131                                  Not Reported
Name: distract1, Length: 4132, dtype: object
```

In [743]:

```
mapping = {
    1: 'None/apparently normal',
    2: 'Asleep, fatigued',
    3: 'Using cane, crutches, etc.',
    4: 'Paraplegic, using wheelchair',
    5: 'Impaired by prior injury',
    6: 'Deaf',
    7: 'Blind',
    8: 'Emotional (disturbed, angry, depressed, etc.)',
    9: 'DUI-drugs, alcohol, medication',
    10: 'Physical impairment - undefined',
    96: 'Other physical impairment',
    98: 'Not reported',
    99: 'Unknown'
}
df1['dricond1'] = df1['dricond1'].replace(distraction_codes)
```

In [744]:

```
state_codes = {
    "01": "Alabama",
    "02": "Alaska",
    "03": "American Samoa",
    "04": "Arizona",
    "05": "Arkansas",
    "06": "California",
    "08": "Colorado",
    "09": "Connecticut",
    "10": "Delaware",
    "11": "District of Columbia",
    "12": "Florida",
    "13": "Georgia",
    "14": "Guam",
    "15": "Hawaii",
    "16": "Idaho",
    "17": "Illinois",
    "18": "Indiana",
    "19": "Iowa",
    "20": "Kansas",
    "21": "Kentucky",
    "22": "Louisiana",
    "23": "Maine",
    "24": "Maryland",
    "25": "Massachusetts",
    "26": "Michigan",
    "27": "Minnesota",
    "28": "Mississippi",
    "29": "Missouri",
    "30": "Montana",
    "31": "Nebraska",
    "32": "Nevada",
    "33": "New Hampshire",
    "34": "New Jersey",
    "35": "New Mexico",
    "36": "New York",
    "37": "North Carolina",
    "38": "North Dakota",
    "39": "Ohio",
    "40": "Oklahoma",
    "41": "Oregon",
    "42": "Pennsylvania",
    "43": "Puerto Rico",
    "44": "Rhode Island",
    "45": "South Carolina",
    "46": "South Dakota",
    "47": "Tennessee",
    "48": "Texas",
    "49": "Utah",
    "50": "Vermont",
    "51": "Virginia",
    "52": "Virgin Islands",
    "53": "Washington",
    "54": "West Virginia",
    "55": "Wisconsin",
    "56": "Wyoming",
    "93": "Indian Nation",
    "94": "U.S. Government",
    "95": "Canada",
    "96": "Mexico",
    "97": "Other Foreign Country",
    "98": "Not Reported",
    "99": "Unknown",
    "81": "Lamp violation",
    "82": "Brake violation",
    "83": "Safety restraint nonuse",
    "84": "Motorcycle equipment violation",
    "85": "Hazardous cargo violation",
    "86": "Size, weight, load violation",
    "87": "Ice, Snow, Slush, Water, Sand, Dirt, Oil, Wet Leaves on Road",
    "88": "Trailer Fishtailing or Swaying",
    "89": "Equipment violation, general",
    "91": "Parking",
    "92": "Theft",
    "93": "Driving where prohibited",
    "95": "No/Unk driver present",
    "97": "Not Reported",
    "98": "Other Moving Violation",
    "99": "Unknown Violations"
}

df1['licstate'] = df1['licstate'].astype(str).map(state_codes)
```

In [745]:

```
license_status = {
           0.0: 'Not Licensed',
           1.0: 'Suspended',
           2.0: 'Revoked',
           3.0: 'Expired',
           4.0: 'Canceled or Denied',
           6.0: 'Valid',
           9.0: 'Unknown License Status'
          }
df1['noncdl'] = df1['noncdl'].map(license_status)
```

In [746]:

```
df1['noncdl'].value_counts()
```

Out[746]:

```
Valid                     3373
Suspended                  383
Not Licensed               196
Unknown License Status     136
Expired                     27
Revoked                     12
Canceled or Denied           4
Name: noncdl, dtype: int64
```

In [747]:

```
df1['weather'].value_counts()
```

Out[747]:

```
1     2793
10     615
2      489
5      106
4       64
98      21
99      13
3       11
6        9
8        9
7        2
Name: weather, dtype: int64
```

In [748]:

```
df2=pd.DataFrame()
```

In [749]:

```
df2=df1
```

In [750]:

```

# create a dictionary mapping numeric codes to text values
weather = {
    0: 'No Additional Atmospheric Conditions',
    1: 'Clear',
    2: 'Rain',
    3: 'Sleet or Hail',
    4: 'Snow',
    5: 'Fog, Smog, Smoke',
    6: 'Severe Crosswinds',
    7: 'Blowing Sand, Soil, Dirt',
    8: 'Other',
    10: 'Cloudy',
    11: 'Blowing Snow',
    12: 'Freezing Rain or Drizzle',
    98: 'Not Reported',
    99: 'Unknown'
}



# use the map method to apply the mapping to the 'coord' column
df1['weather'] = df1['weather'].map(weather)
```

In [751]:

```
df1['weather'].value_counts()
```

Out[751]:

```
Clear                        2793
Cloudy                        615
Rain                          489
Fog, Smog, Smoke              106
Snow                           64
Not Reported                   21
Unknown                        13
Sleet or Hail                  11
Severe Crosswinds               9
Other                           9
Blowing Sand, Soil, Dirt        2
Name: weather, dtype: int64
```

In [752]:

```
lightcond = {
    1: 'Daylight',
    2: 'Dark - Not Lighted',
    3: 'Dark - Lighted',
    4: 'Dawn',
    5: 'Dusk',
    6: 'Dark - Lighting Unknown',
    7: 'Other',
    8: 'Not Reported',
    9: 'Unknown'
}
df1['lightcond'] = df1['lightcond'].map(lightcond)
```

In [753]:

```
df1['lightcond'].value_counts()
```

Out[753]:

```
Daylight                   2040
Dark - Not Lighted          937
Dark - Lighted              857
Dusk                        143
Dawn                        100
Dark - Lighting Unknown      22
Unknown                      21
Not Reported                 11
Other                         1
Name: lightcond, dtype: int64
```

In [754]:

```
surface_cond = {
    1: 'Dry',
    2: 'Wet',
    3: 'Snow or Slush',
    4: 'Ice or Frost',
    5: 'Sand, Dirt, Mud, Gravel',
    6: 'Water (standing or moving)',
    7: 'Oil',
    8: 'Other',
    9: 'Unknown'
}
df2['surfcond'] = df2['surfcond'].map(surface_cond)
```

In [755]:

```
df2['surfcond']
```

Out[755]:

```
0         Ice or Frost
1                  Dry
2                  Dry
3        Snow or Slush
4                  Dry
             ...
4127               NaN
4128               NaN
4129               Dry
4130               Dry
4131               Dry
Name: surfcond, Length: 4132, dtype: object
```

In [756]:

```
df1['surfcond'].value_counts()
```

Out[756]:

```
Dry                           3054
Wet                            807
Ice or Frost                    97
Snow or Slush                   25
Other                           16
Water (standing or moving)      12
Name: surfcond, dtype: int64
```

In [757]:

```
surftype = {
    1: 'Concrete',
    2: 'Blacktop - bituminous or asphalt',
    3: 'Brick or Block',
    4: 'Slag, Gravel, Stone',
    5: 'Dirt',
    7: 'Oil',
    8: 'Other',
    9: 'Unknown'
}
```

In [758]:

```
df1['surftype'] = df1['surftype'].map(surftype)
```

In [759]:

```
df1['surftype'].value_counts()
```

Out[759]:

```
Blacktop - bituminous or asphalt    3532
Concrete                             438
Oil                                   50
Slag, Gravel, Stone                   37
Dirt                                  24
Other                                 14
Brick or Block                         1
Name: surftype, dtype: int64
```

In [760]:

```
hitrun = {0: 'No', 1: 'Yes'}
df1['hitrun'] = df1['hitrun'].map(hitrun)
```

In [761]:

```
df1['hitrun']
```

Out[761]:

```
0        No
1        No
2       Yes
3        No
4        No
       ... 
4127     No
4128     No
4129     No
4130     No
4131     No
Name: hitrun, Length: 4132, dtype: object
```

In [762]:

```
manncol = {
    0: 'The event was Not a collision with a motor vehicle in transport',
    1: 'Front-to-Rear',
    2: 'Front-to-Front',
    3: 'Front-to-Side, Same Direction',
    4: 'Front-to-Side, Opposite Direction',
    5: "Front-to-Side, Right Angle (e.g., 'broadside')",
    6: 'Angle',
    7: 'Sideswipe, Same Direction',
    8: 'Sideswipe, Opposite Direction',
    9: 'Rear-to-Side',
    10: 'Rear-to-Rear',
    11: 'Other',
    98: 'Not Reported',
    99: 'Unknown'
}
df1['manncol'] = df1['manncol'].map(manncol)
```

In [763]:

```
df1['manncol'].value_counts()
```

Out[763]:

```
The event was Not a collision with a motor vehicle in transport    1848
Angle                                                               947
Front-to-Front                                                      686
Front-to-Rear                                                       418
Sideswipe, Opposite Direction                                       108
Sideswipe, Same Direction                                           103
Other                                                                12
Not Reported                                                         10
Name: manncol, dtype: int64
```

In [764]:

```
reljuncinter = {
    0: 'No',
    1: 'Yes',
    8: 'Not reported',
    9: 'Unknown'
}

df1['reljuncinter'] = df1['reljuncinter'].map(reljuncinter)
```

In [765]:

```
df1['reljuncinter'].value_counts()
```

Out[765]:

```
No     3997
Yes     135
Name: reljuncinter, dtype: int64
```

In [766]:

```
df1['reljunc']
```

Out[766]:

```
0       1
1       1
2       3
3       1
4       2
       ..
4127    8
4128    1
4129    3
4130    3
4131    1
Name: reljunc, Length: 4132, dtype: int64
```

In [767]:

```
reljunc = {
    1: "Non-Junction",
    2: "Intersection",
    3: "Intersection Related",
    4: "Driveway Access",
    5: "Entrance/Exit Ramp Related",
    6: "Railway Grade Crossing",
    7: "Crossover Related",
    8: "Driveway Access Related",
    16: "Shared-Use Path or Trail",
    17: "Acceleration/Deceleration Lane",
    18: "Through Roadway",
    19: "Other Location Within Interchange Area",
    98: "Not Reported",
    99: "Unknown"
}

df1["reljunc"] = df1["reljunc"].replace(reljunc)
```

In [768]:

```
df1['reljunc'].value_counts()
```

Out[768]:

```
Non-Junction                              2651
Intersection                               794
Intersection Related                       394
Driveway Access Related                    134
Entrance/Exit Ramp Related                  75
Driveway Access                             31
20                                          25
Railway Grade Crossing                       8
Other Location Within Interchange Area       8
Acceleration/Deceleration Lane               6
Not Reported                                 2
Through Roadway                              2
Shared-Use Path or Trail                     1
Unknown                                      1
Name: reljunc, dtype: int64
```

In [769]:

```
df1['funcsystem']
```

Out[769]:

```
0        1
1        5
2        4
3        3
4        4
        ..
4127     4
4128    96
4129     3
4130     3
4131     5
Name: funcsystem, Length: 4132, dtype: int64
```

In [770]:

```
# define the mapping dictionary
funcsystem = {
                1: 'Interstate',
                2: 'Principal Arterial-Other Freeway / Expressway',
                3: 'Principal Arterial-Other',
                4: 'Minor Arterial',
                5: 'Major Collector',
                6: 'Minor Collector',
                7: 'Local',
                96: 'Trafficway not in State Inventory',
                98: 'Not Reported',
                99: 'Unknown'
}

# apply the mapping using replace
df1['funcsystem'] = df1['funcsystem'].replace(funcsystem)
```

In [771]:

```
df1['funcsystem'].value_counts()
```

Out[771]:

```
Principal Arterial-Other                         1448
Minor Arterial                                    717
Major Collector                                   708
Interstate                                        591
Local                                             381
Minor Collector                                   129
Principal Arterial-Other Freeway / Expressway     125
Trafficway not in State Inventory                  20
Not Reported                                        9
Unknown                                             4
Name: funcsystem, dtype: int64
```

In [772]:

```
df1['landuse']
```

Out[772]:

```
0       2
1       1
2       2
3       2
4       2
       ..
4127    1
4128    6
4129    2
4130    2
4131    1
Name: landuse, Length: 4132, dtype: int64
```

In [773]:

```
landuse = {1: 'Rural', 2: 'Urban', 6: 'Trafficway not in State Inventory', 8:

df1['landuse'] = df1['landuse'].replace(landuse)
```

In [774]:

```
df1['landuse'].value_counts()
```

Out[774]:

```
Urban                                    2364
Rural                                    1745
Trafficway not in State Inventory          20
Unknown                                     3
Name: landuse, dtype: int64
```

In [775]:

```
# define a dictionary mapping of values to replacements
ownership = {
    1: "State Highway Agency",
    2: "County Highway Agency",
    3: "Town or Township Highway Agency",
    4: "City or Municipal Highway Agency",
    11: "State Park, Forest or Reservation Agency",
    12: "Local Park, Forest or Reservation Agency",
    21: "Other State Agency",
    25: "Other Local Agency",
    26: "Private (other than Railroad)",
    27: "Railroad",
    31: "State Toll Road",
    32: "Local Toll Authority",
    40: "Other Public Instrumentality (i.e., Airport)",
    50: "Indian Tribe Nation",
    60: "Other Federal Agency",
    62: "Bureau of Indian Affairs",
    63: "Bureau of Fish and Wildlife",
    64: "U.S. Forest Service",
    66: "National Park Service",
    67: "Tennessee Valley Authority",
    68: "Bureau of Land Management",
    69: "Bureau of Reclamation",
    70: "Corps of Engineers",
    72: "Air Force",
    74: "Navy/Marines",
    80: "Army",
    96: "Trafficway Not in State Inventory",
    98: "Not Reported",
    99: "Unknown"
}

# replace the values in the "coord" column with the corresponding replacements
df1['ownership'] = df1['ownership'].replace(ownership)
```

In [776]:

```
df1['ownership']
```

Out[776]:

```
0                    State Highway Agency
1                   County Highway Agency
2        City or Municipal Highway Agency
3                    State Highway Agency
4        City or Municipal Highway Agency
                      ...
4127                 State Highway Agency
4128    Trafficway Not in State Inventory
4129     City or Municipal Highway Agency
4130     City or Municipal Highway Agency
4131                County Highway Agency
Name: ownership, Length: 4132, dtype: object
```

In [777]:

```

# define the replacement dictionary
intersectiontype = {
    1: 'No Intersection',
    2: '4-Way Intersection',
    3: 'T-Intersection',
    4: 'Y-Intersection',
    5: 'Traffic Circle',
    6: 'Roundabout',
    7: 'Five-point or more',
    10: 'L-Intersection',
    11: 'Other Intersection Type',
    98: 'Not Reported',
    99: 'Unknown'
}

# replace the values in the "coord" column using the replacement dictionary
df1['intersectiontype'] = df1['intersectiontype'].replace(intersectiontype)
```

In [778]:

```
df1['intersectiontype'].value_counts()
```

Out[778]:

```
No Intersection       2943
4-Way Intersection     758
T-Intersection         355
Y-Intersection          49
Roundabout              11
L-Intersection           8
Traffic Circle           4
Five-point or more       3
Not Reported             1
Name: intersectiontype, dtype: int64
```

In [779]:

```
df1['schlbus'].value_counts()
```

Out[779]:

```
0    4122
1      10
Name: schlbus, dtype: int64
```

In [780]:

```
df1['schlbus'] = df1['schlbus'].replace({0: 'No', 1: 'Yes', 8: 'Not Reported'}
```

In [781]:

```
# Create a dictionary to map values to conditions
contdev = {
    0: 'No Controls',
    1: 'Traffic Control Signal (on colors), without Pedestrian Signal',
    2: 'Traffic Control Signal (on colors), with Pedestrian Signal',
    3: 'Traffic Control Signal (on colors), Pedestrian Signal unknown',
    4: 'Flashing Traffic Control Signal',
    5: 'Flashing Beacon',
    6: 'Flashing Highway Traffic Signal, type unknown',
    7: 'Lane Use Control Signal',
    8: 'Other Highway Traffic Signal',
    9: 'Unknown Highway Traffic Signal',
    20: 'Stop Sign',
    21: 'Yield Sign',
    23: 'School Zone Sign/Device',
    28: 'Other Regulatory Sign',
    29: 'Unknown Regulatory Sign',
    30: 'School Speed Limit',
    31: 'School Advance/Crossing',
    38: 'Other School Related',
    39: 'Unknown School Zone Sign',
    40: 'Warning Sign',
    41: 'Electronic Warning Sign',
    50: 'Officer, Cross Guard, Flagger',
    60: 'RR-Gate',
    61: 'RR-Flashing Lights',
    62: 'RR-Traffic Control Signal',
    63: 'RR-Wigwags',
    64: 'RR-Bells',
    68: 'RR-Other Train-Activated Device',
    69: 'RR-Active Device, Type Unknown',
    70: 'RR-Cross Bucks',
    71: 'RR-Stop Sign',
    72: 'RR-Other Crossing',
    73: 'RR-Special, e.g., watchman flagged by crew',
    78: 'RR-Other Passive Device',
    79: 'RR-Passive Device, Type Unknown',
    80: 'RR-Grade Crossing Control, Type Unknown',
    98: 'Other',
    99: 'Unknown'
}

# Assume that the DataFrame is named "df" and the "coord" column contains the
df1['contdev'] = df1['contdev'].map(contdev)
```

In [782]:

```
df1['contdev'].value_counts()
```

Out[782]:

```
No Controls                                                      3444
Traffic Control Signal (on colors), Pedestrian Signal unknown     363
Stop Sign                                                         214
Traffic Control Signal (on colors), with Pedestrian Signal         22
Flashing Traffic Control Signal                                    16
Officer, Cross Guard, Flagger                                      14
Yield Sign                                                         10
Other                                                               9
Other Regulatory Sign                                               7
Traffic Control Signal (on colors), without Pedestrian Signal       4
Warning Sign                                                        3
Unknown Regulatory Sign                                             2
Unknown Highway Traffic Signal                                      1
Name: contdev, dtype: int64
```

In [783]:

```
df1['devfunc'].value_counts()
```

Out[783]:

```
0    3444
3     673
8      15
Name: devfunc, dtype: int64
```

In [784]:

```
# define the mapping dictionary
devfunc = {
    0: 'No Device',
    1: 'Device Not Functioning',
    2: 'Device Functioning Improperly',
    3: 'Device Functioning Properly',
    4: 'Device Not Functioning or Device Functioning Improperly, specifics unk
    9: 'Unknown'
}

# replace the values in the coord column
df1['devfunc'] = df1['devfunc'].replace(devfunc)
```

In [785]:

```
df1['spdlim'].value_counts()
```

Out[785]:

```
35    797
60    746
50    515
55    416
98    369
40    285
45    256
70    232
25    221
30    218
0      37
65     25
20     11
15      2
10      1
5       1
Name: spdlim, dtype: int64
```

In [786]:

```
#this can be done or not discuss it later
df1['spdlim'] = df1['spdlim'].apply(lambda x:
    'No statutory limit' if x == 0 else
    'Actual Speed Limit' if 5 <= x <= 95 else
    'Not Reported' if x == 98 else
    'Unknown' if x == 99 else None
)
```

In [787]:

```
df1['spdlim'].value_counts()
```

Out[787]:

```
Actual Speed Limit    3726
Not Reported           369
No statutory limit      37
Name: spdlim, dtype: int64
```

In [788]:

```
df1['criticaleventcat']
```

Out[788]:

```
0       1
1       2
2       5
3       2
4       2
       ..
4127    3
4128    2
4129    3
4130    3
4131    1
Name: criticaleventcat, Length: 4132, dtype: int64
```

In [789]:

```
# create a dictionary with the mappings
criticaleventcat = {
            1: 'Loss of Control Due To',
           2: 'This Vehicle Traveling',
           3: 'Other Motor Vehicle in Lane...',
           4: 'Other Motor Vehicle Encroaching Into Lane',
           5: 'Pedestrian, Pedalcyclist, Other NonMotorist',
           6: 'Object or Animal',
           7: 'Other',
           9: 'Unknown'}
df1['criticaleventcat'] = df1['criticaleventcat'].replace(criticaleventcat)
```

In [790]:

```
df1['trailer'].value_counts()
```

Out[790]:

```
0    3857
1     241
2      27
9       6
6       1
Name: trailer, dtype: int64
```

In [791]:

```
# create a dictionary to map the values to descriptions
trailer = {
    0: 'No trailing unit',
    1: 'One trailing unit',
    2: 'Two trailing units',
    3: 'Three or more trailing units',
    4: 'Trailer present, number of units unknown',
    5: 'Vehicle towing another vehicle, Fixed linkage',
    6: 'Vehicle towing another vehicle, Unfixed linkage',
    9: 'Unknown'
}
df1['trailer'] = df1['trailer'].replace(trailer)
```

In [792]:

```
df1['emerg'].value_counts()
```

Out[792]:

```
0    4125
5       3
2       2
6       1
8       1
Name: emerg, dtype: int64
```

In [793]:

```
# create mapping dictionary
emerg = {
    0: 'Not Applicable',
    2: 'Non-Emergency, Non-Transport',
    3: 'Non-Emergency Transport',
    4: 'Emergency Operation, emergency warning not used',
    5: 'Emergency Operation, emergency warning used',
    6: 'Emergency Operation, emergency warning use unknown',
    8: 'Not Reported',
    9: 'Unknown'
}

df1['emerg'] = df1['emerg'].replace(emerg)
```

In [794]:

```
df1['roadalgn'].value_counts()
```

Out[794]:

```
1    2888
3     573
2     453
8     113
4      68
0      36
9       1
Name: roadalgn, dtype: int64
```

In [795]:

```
roadalgn = {
    0: 'Non-trafficway or Driveway Access',
    1: 'Straight',
    2: 'Curve Right',
    3: 'Curve Left',
    4: 'Curve - Unknown Direction',
    8: 'Not Reported',
    9: 'Unknown'
}
df1['roadalgn'] = df1['roadalgn'].replace(roadalgn)
```

In [796]:

```
df1['intersectiontype']
```

Out[796]:

```
0          No Intersection
1          No Intersection
2        4-Way Intersection
3          No Intersection
4        4-Way Intersection
                ...
4127       No Intersection
4128       No Intersection
4129     4-Way Intersection
4130     4-Way Intersection
4131       No Intersection
Name: intersectiontype, Length: 4132, dtype: object
```

```
In [ ]:
```

```
In [798]: df1_subset = pd.DataFrame()
```

```
In [799]: df1_subset['vehtype']=df1['vehtype']
```

```
In [800]: df1['vehtype'].value_counts()
```

```
Out[800]: PV          3198
          MC           441
          MHTRUCK      367
          NR-U          56
          OTHVT         31
          BUS           22
          OTHMC         13
          MHOME          4
          Name: vehtype, dtype: int64
```

```
In [825]: df1_subset
```

Out[825]:

	vehtype
0	PV
1	PV
2	PV
3	PV
4	PV
...	...
4127	PV
4128	PV
4129	PV
4130	PV
4131	PV

4132 rows × 1 columns

```
In [824]: df1.to_csv('sample.csv', index=False)
```

```
In [803]: df1['dr_dist'].value_counts()
```

```
Out[803]: 0    3554
          1     578
          Name: dr_dist, dtype: int64
```

In [804]:
```
df1['dzip']=df1['dzip'].fillna(0)
```

In [805]:
```
df1['dzip']=df1['dzip'].astype(int)
```

In [806]:
```
df1['zipcode']=df1['zipcode'].fillna(0)
```

In [807]:
```
df1['zipcode']=df1['zipcode'].astype(int)
```

In [808]:
```
matched_zipcode=pd.DataFrame()
```

In [809]:
```
matched_zipcode=df1[df1['dzip']==df1['zipcode']]
```

In [810]:
```
matched_zipcode
```

Out[810]:

	year	case	par	repjur	crash_dt	crash_tm	accday	accmon	holiday	county	...
10	2017	8	E629981	6.0	1/7/2017	14:12	7	1	0.0	37	...
19	2017	14	E631092	83.0	1/6/2017	6:01	6	1	0.0	41	...
29	2017	20	E633291	184.0	1/17/2017	21:05	17	1	0.0	61	...
39	2017	26	E635844	1.0	1/23/2017	17:19	23	1	0.0	21	...
46	2017	31	E638169	8.0	1/27/2017	10:11	27	1	0.0	35	...
...	...	...	...	...	...	...	...	...	...	...	...
4111	2021	630	EB61178	289.0	8/21/2021	22:19	21	8	NaN	77	...
4123	2021	633	TECH031	32.0	3/11/2021	3:15	11	3	NaN	47	...
4127	2021	636	Incid08	334.0	8/8/2021	9:38	8	8	NaN	47	...
4130	2021	639	EB48605	263.0	4/21/2021	17:32	21	4	NaN	33	...
4131	2021	640	Inciden	42.0	10/15/2021	14:55	15	10	NaN	67	...

970 rows × 307 columns

In [811]:
```
matched_zip_count = len(matched_zipcode)
```

1.Among drivers involved in fatal crashes, what proportion are involved in crashes in communities where they live?

In [812]:
```
total_count = len(df1)
```

In [813]:
```
proportion = matched_zip_count / total_count
```

In [904]:
```
print(f"The proportion of drivers involved in fatal crashes in communities whe
```

```
The proportion of drivers involved in fatal crashes in communities where they
live is: 23.48%
```

This code reads the accident dataset into a pandas dataframe. It then calculates the number of fatal crashes where the driver origin zipcode matches the accident location zipcode by using the len() function to count the number of rows where these two columns have the same value. It calculates the total number of fatal crashes using the same method. It then calculates the proportion of fatal crashes where the driver origin zipcode matches the accident location zipcode by dividing the number of same-zipcode crashes by the total number of crashes. Finally, it prints the proportion as a percentage using f-string formatting.

In [815]:
```
a=df1['dzip'] == df1['zipcode']
```

In [816]:
```
#determining the residency
df1['is_resident'] = df1['dzip'] == df1['zipcode']
```

In [817]:
```
df1['is_resident'].value_counts()
```

Out[817]:
```
False    3162
True      970
Name: is_resident, dtype: int64
```

In [821]:
```
df1[['licstate','dzip','d_state','zipcode']]
```

Out[821]:

	licstate	dzip	d_state	zipcode
0	Washington	98204	WA	98201
1	Washington	98465	WA	98359
2	Washington	98125	WA	98103
3	Washington	99021	WA	99224
4	Washington	98065	WA	98104
...	...	...	...	...
4127	Washington	99155	WA	99155
4128	Washington	99138	WA	0
4129	Washington	98503	WA	98144
4130	Washington	98144	WA	98144
4131	Washington	98502	WA	98502

4132 rows × 4 columns

In [827]:

```
df1['d_state'].value_counts()
```

Out[827]:

```
WA    3729
OR      80
ID      45
CA      38
AZ      16
TX       9
FL       8
MT       8
UT       7
CO       4
OK       4
AK       4
GA       4
MI       3
PA       3
LA       3
NY       3
IN       3
VA       3
NV       3
IL       3
HI       2
OH       2
WI       1
MS       1
IA       1
KS       1
TN       1
NJ       1
NC       1
NH       1
NM       1
MN       1
AR       1
MA       1
MO       1
NE       1
RI       1
Name: d_state, dtype: int64
```

In [ ]:

```
 'state_abbr': ['AK', 'AR', 'AZ', 'CA', 'CO', 'FL', 'GA', 'HI', 'IA', 'ID', 'I
    'state_name': ['Alaska', 'Arkansas', 'Arizona', 'California', 'Colorado',
```

In [828]:

```
df1[['dzip','zipcode','d_state']]
```

Out[828]:

	dzip	zipcode	d_state
0	98204	98201	WA
1	98465	98359	WA
2	98125	98103	WA
3	99021	99224	WA
4	98065	98104	WA
...	...	...	...
4127	99155	99155	WA
4128	99138	0	WA
4129	98503	98144	WA
4130	98144	98144	WA
4131	98502	98502	WA

4132 rows × 3 columns

In [830]:

```
df1['race'].value_counts()
```

Out[830]:

```
0.0     890
1.0     535
98.0     44
2.0      31
3.0      27
7.0       9
99.0      6
78.0      5
58.0      4
68.0      4
4.0       3
28.0      2
38.0      2
48.0      2
18.0      2
97.0      1
6.0       1
Name: race, dtype: int64
```

In [832]:
```
df1['race1'].value_counts()
```

Out[832]:
```
0.0      1466
1.0       819
98.0       54
298.0      49
2.0        42
3.0        38
201.0      29
202.0      13
48.0        7
68.0        7
203.0       6
18.0        6
7.0         5
99.0        3
268.0       3
278.0       3
28.0        3
38.0        2
4.0         2
204.0       2
207.0       1
58.0        1
5.0         1
6.0         1
Name: race1, dtype: int64
```

In [834]:
```
df1['race3']
```

Out[834]:
```
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
         ..
4127    NaN
4128    NaN
4129    NaN
4130    NaN
4131    NaN
Name: race3, Length: 4132, dtype: float64
```

In [845]:
```
df1['race4']=df1['race4'].astype(int)
```

In [837]:
```
df1['race1']=df1['race1'].fillna(0)
df1['race2']=df1['race2'].fillna(0)
df1['race3']=df1['race3'].fillna(0)
df1['race4']=df1['race4'].fillna(0)
```

In [847]:

```
df1['race'] = df1['race1'] + df1['race2'] + df1['race3'].astype(int)+df1['race
```

In [851]:

```
df1['race'].value_counts()
```

Out[851]:

```
Not a Fatality                            3035
White                                      819
Other Race                                  54
Multiple Race - Other Races                 49
Black                                       42
North American Indian or Alaska Native      38
403                                         13
499                                         11
Other Asian or Pacific Islander              8
404                                          8
Vietnamese                                   7
Asian Indian                                 6
Filipino                                     5
Unknown                                      3
Korean                                       3
500                                          3
469                                          3
501                                          2
Chinese                                      2
686                                          2
Samoan                                       2
405                                          2
697                                          1
Native Hawaiian                              1
Japanese                                     1
406                                          1
666                                          1
Guamanian or Chamorro                        1
419                                          1
566                                          1
450                                          1
606                                          1
472                                          1
407                                          1
705                                          1
413                                          1
683                                          1
Name: race, dtype: int64
```

In [850]:

```



# define a dictionary to map the codes to their respective values
race_dict = {0: "Not a Fatality", 1: "White", 2: "Black", 3: "North American I
             4: "Chinese", 5: "Japanese", 6: "Native Hawaiian", 7: "Filipino",
             19: "Other Indian (So.& Cent. America)", 28: "Korean", 38: "Samoa
             58: "Guamanian or Chamorro", 68: "Other Asian or Pacific Islander
             78: "Asian/Pacific Islander, No specific race", 97: "Multiple Rac
             98: "Other Race", 99: "Unknown", 201: "White", 202: "Black",
             203: "North American Indian or Alaska Native", 204: "Chinese", 20
             206: "Native Hawaiian", 207: "Filipino", 218: "Asian Indian",
             219: "Other Indian (So.& Cent. America)", 228: "Korean", 238: "Sa
             248: "Vietnamese", 258: "Guamanian or Chamorro", 268: "Other Asia
             278: "Asian/Pacific Islander, No specific race", 298: "Multiple R

# use replace() method to map the codes to their respective values
df1['race'] = df1['race'].replace(race_dict)
```

In [852]:

```
# define a lambda function to map numeric values to 'Other'
def map_numeric(x):
    if isinstance(x, (int, float)):
        return 'Other'
    else:
        return x

# use the apply method to apply the mapping function
df1['race'] = df1['race'].apply(map_numeric)
```

In [853]:

```
df1['race'].value_counts()
```

Out[853]:

```
Not a Fatality                             3035
White                                       819
Other                                        56
Other Race                                   54
Multiple Race - Other Races                  49
Black                                        42
North American Indian or Alaska Native       38
Other Asian or Pacific Islander               8
Vietnamese                                    7
Asian Indian                                  6
Filipino                                      5
Unknown                                       3
Korean                                        3
Chinese                                       2
Samoan                                        2
Guamanian or Chamorro                         1
Japanese                                      1
Native Hawaiian                               1
Name: race, dtype: int64
```

In [857]:

```
df1['dzip']
```

Out[857]:

```
0       98204
1       98465
2       98125
3       99021
4       98065
        ...  
4127    99155
4128    99138
4129    98503
4130    98144
4131    98502
Name: dzip, Length: 4132, dtype: int32
```

In [859]:

```
df1['is_resident'].value_counts()
```

Out[859]:

```
False    3162
True      970
Name: is_resident, dtype: int64
```

In [863]:

```
df1_is_resident = df1[df1['is_resident']]
```

In [ ]:

In [864]:

```
#for q1
import scipy.stats as stats
```

In [877]:

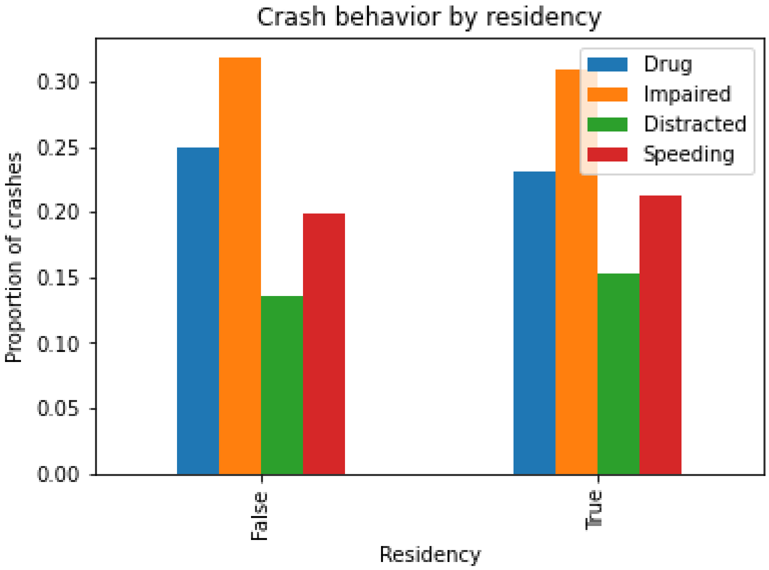
```
import pandas as pd
from scipy.stats import ttest_ind



# separate the data into two groups based on residency status
residents = df1[df1['is_resident'] == 1]
non_residents = df1[df1['is_resident'] == 0]

# perform a two-sample t-test for each behavior factor column
for column in ['dr_drug', 'dr_imp', 'dr_dist', 'dr_spd']:
    t, p = ttest_ind(residents[column], non_residents[column])
    print(f"{column}: t = {t:.2f}, p-value = {p:.3f}\n")
```

```
dr_drug: t = -1.15, p-value = 0.250

dr_imp: t = -0.52, p-value = 0.603

dr_dist: t = 1.30, p-value = 0.193

dr_spd: t = 0.96, p-value = 0.339
```

The code separates the data into two groups based on the is_resident column, performs a two-sample t-test for each behavior factor column, and prints the results (t-value and p-value) to the console. The null hypothesis for each test is that there is no difference in the means of the two groups for that behavior factor. The alternative hypothesis is that there is a difference. The p-value tells you the probability of observing a difference as extreme or more extreme than the one observed, assuming the null hypothesis is true. If the p-value is less than your chosen significance level (e.g. 0.05), you can reject the null hypothesis and conclude that there is evidence of a difference in means between the two groups for that behavior factor.

```
In [879]: residency_grouped = df1.groupby("is_resident").mean()[["dr_drug", "dr_imp", "c
```

```
In [880]: import matplotlib.pyplot as plt

          residency_grouped.plot(kind="bar")
          plt.xlabel("Residency")
          plt.ylabel("Proportion of crashes")
          plt.title("Crash behavior by residency")
          plt.legend(["Drug", "Impaired", "Distracted", "Speeding"])
          plt.show()
```



In [883]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split



# Separate the input features (X) and target variable (y)
X = df1[['dr_drug', 'dr_imp', 'dr_dist', 'dr_spd']]
y = df1['is_resident']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Create a logistic regression model
model = LogisticRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict the target variable for the testing data
y_pred = model.predict(X_test)

# Print the accuracy of the model
print('Accuracy:', model.score(X_test, y_test))

# Print the weights of each input feature
print('Weights:', model.coef_)
```

```
Accuracy: 0.7835550181378477
Weights: [[-0.11747718 -0.05028275  0.08884009  0.13854002]]
```

In [884]:
```
df1['criticaleventcat']
```

Out[884]:
```
0                            Loss of Control Due To
1                            This Vehicle Traveling
2       Pedestrian, Pedalcyclist, Other NonMotorist
3                            This Vehicle Traveling
4                            This Vehicle Traveling
                           ...                     
4127                 Other Motor Vehicle in Lane...
4128                         This Vehicle Traveling
4129                 Other Motor Vehicle in Lane...
4130                 Other Motor Vehicle in Lane...
4131                         Loss of Control Due To
Name: criticaleventcat, Length: 4132, dtype: object
```

In [885]:

```

# Group the data by resident status and behavior factors
grouped = df1.groupby(['is_resident', 'dr_drug', 'dr_imp', 'dr_dist', 'dr_spd'

# Calculate the average number of crashes for each group
averages = grouped.size().reset_index(name='count').groupby(['is_resident', 'c

# Print the results
```

```
is_resident  dr_drug  dr_imp  dr_dist  dr_spd  criticaleventcat
False        0        0       0        0       Loss of Control Due To
57.0

                                               Object or Animal

21.0

                                               Other

138.0

                                               Other Motor Vehicle Encroachin

g Into Lane      596.0

                                               Other Motor Vehicle in Lane...

235.0

...
True         1        1       1        0       Other Motor Vehicle in Lane...
4.0

                                               Pedestrian, Pedalcyclist, Othe

r NonMotorist      1.0

                                               This Vehicle Traveling

24.0

                                       1       Loss of Control Due To

3.0

                                               This Vehicle Traveling

2.0
Name: count, Length: 128, dtype: float64
```

In [886]:
```
averages
```

Out[886]:
```
is_resident  dr_drug  dr_imp  dr_dist  dr_spd  criticaleventcat
False        0        0       0        0       Loss of Control Due To
57.0
                                               Object or Animal

21.0
                                               Other

138.0
                                               Other Motor Vehicle Encroachin

g Into Lane      596.0
                                               Other Motor Vehicle in Lane...

235.0

...
True         1        1       1        0       Other Motor Vehicle in Lane...
4.0
                                               Pedestrian, Pedalcyclist, Othe

r NonMotorist      1.0
                                               This Vehicle Traveling

24.0
                                       1       Loss of Control Due To

3.0
                                               This Vehicle Traveling

2.0
Name: count, Length: 128, dtype: float64
```

In [892]:
```
diedscene = {0: 'Not Applicable', 7: 'Died at Scene', 8: 'Died En Route', 9: '

# map the values in the column using the dictionary
df1['diedscene'] = df1['diedscene'].map(diedscene)
```

In [887]:
```
df1['mhevent']
```

Out[887]:
```
0        3
1       42
2        8
3        9
4        8
        ..
4127    12
4128    42
4129    12
4130    12
4131    42
Name: mhevent, Length: 4132, dtype: int64
```

In [893]:
```
df1_resident=df1[df1['is_resident']]
```

In [895]:

```
df1_resident['diedscene'].value_counts()
```

Out[895]:

```
Not Applicable    667
Died at Scene     301
Died En Route       2
Name: diedscene, dtype: int64
```

In [899]:

```
df1_non_resident = df1[df1['is_resident']==False]
```

In [901]:

```
df1_non_resident['diedscene'].value_counts()
```

Out[901]:

```
Not Applicable    2190
Died at Scene      965
Died En Route        7
Name: diedscene, dtype: int64
```

In [906]:

```
df1['is_resident'].value_counts()
```

Out[906]:

```
False    3162
True      970
Name: is_resident, dtype: int64
```

In [917]:

```
df1.to_csv('sample1.csv', index=False)
```

In [909]:

```
# define mapping
vehicle_mapping = {
    'PV': 'Passenger Vehicle',
    'BUS': 'Bus',
    'MHOME': 'Motorhome',
    'MHTRUCK': 'Medium/Heavy Truck',
    'MC': 'Motorcycle',
    'OTHMC': 'Other Motored Cycle',
    'OTHVT': 'Other Vehicle Type'
}

# map vehicle types
df1['vehtype'] = df1['vehtype'].map(vehicle_mapping)
```

In [912]:

```
df1['crashtype'].value_counts()
```

```
Out[912]: 98    711
          13    601
          1     340
          6     278
          50    259
          51    258
          2     134
          7     123
          66    117
          69     95
          68     95
          89     89
          88     89
          83     86
          82     86
          87     62
          86     62
          24     62
          25     60
          21     57
          20     57
          52     48
          64     36
          65     36
          45     30
          28     29
          29     29
          0      22
          14     21
          11     21
          48     16
          46     16
          47     12
          32     10
          78      9
          79      9
          12      7
          92      7
          15      6
          72      5
          73      5
          80      4
          81      4
          44      4
          77      3
          76      3
          84      2
          3       2
          10      2
          74      2
          99      2
          62      2
          5       2
          26      1
          27      1
          71      1
          93      1
```

```
70      1
Name: crashtype, dtype: int64
```

In [914]:

```
    crashtype = {
        0: 'No Impact',
        1: 'Drive Off Road',
        2: 'Control/Traction Loss',
        3: 'Avoid Collision with Vehicle, Pedestrian, Animal',
        4: 'Specifics Other',
        5: 'Specifics Unknown',
        6: 'Drive Off Road',
        7: 'Control/Traction Loss',
        8: 'Avoid Collision With Vehicle, Pedestrian, Animal',
        9: 'Specifics Other',
        10: 'Specifics Unknown',
        11: 'Parked Vehicle',
        12: 'Stationary Object',
        13: 'Pedestrian/Animal',
        14: 'End Departure',
        15: 'Specifics Other',
        16: 'Specifics Unknown',
        20: 'Stopped',
        21: 'Stopped, Straight',
        22: 'Stopped, Left',
        23: 'Stopped, Right',
        24: 'Slower',
        25: 'Slower, Going Straight',
        26: 'Slower, Going Left',
        27: 'Slower, Going Right',
        28: 'Decelerating (Slowing)',
        29: 'Decelerating (Slowing), Going Straight',
        30: 'Decelerating (Slowing), Going Left',
        31: 'Decelerating (Slowing), Going Right',
        32: 'Specifics Other',
        33: 'Specifics Unknown',
        34: 'This Vehicle''s Frontal Area Impacts Another Vehicle.',
        35: 'This Vehicle Is Impacted by Frontal Area of Another Vehicle',
        36: 'This Vehicle''s Frontal Area Impacts Another Vehicle.',
        37: 'This Vehicle Impacted by Front Area of Another Vehicle',
        38: 'This Vehicles Frontal Area Impacts Another Vehicle.',
        39: 'This Vehicle Impacted by Frontal Area of Another Vehicle',
        40: 'This Vehicle''s Frontal Area Impacts Another Vehicle.',
        41: 'This Vehicle Impacted by Frontal Area of Another Vehicle',
        42: 'Specifics Other',
        43: 'Specifics Unknown',
        44: 'Straight Ahead on Left.',
        45: 'Straight Ahead on Left/Right.',
        46: 'Changing Lanes to the Right',
        47: 'Changing Lanes to the Left',
        48: 'Specifics Other',
        49: 'Specifics Unknown',
        50: 'Lateral Move (Left/Right)',
        51: 'Lateral Move (Going Straight)',
        52: 'Specifics Other',
        53: 'Specifics Unknown',
        54: 'This Vehicles Frontal Area Impacts Another Vehicle.',
        55: 'This Vehicle Impacted by Frontal Area of Another Vehicle',
        56: 'This Vehicles Frontal Area Impacts Another Vehicle.',
        57: 'This Vehicle Is Impacted by Frontal Area of Another Vehicle',
        58: 'This Vehicles Frontal Area Impacts Another Vehicle.',
        59: 'This Vehicle Is Impacted by Frontal Area of Another Vehicle',
        60: 'This Vehicles Frontal Area Impacts Another Vehicle.',
        61: 'This Vehicle Impacted by Frontal Area of Another Vehicle',
        62: 'Specifics Other',
        63: 'Specifics Unknown',
        64: 'Lateral Move (Left/Right)',
        65: 'Lateral Move (Going Straight)',
        66: 'Specifics Other',
        67: 'Specifics Unknown',
        68: 'Initial Opposite Directions (Left/Right)',
        69: 'Initial Opposite Directions (Going Straight)',
        86: 'Striking from the Right',
        87: 'Struck on the Right',
        88: 'Striking from the Left',
        89: 'Struck on the Left',
        90: 'Specifics Other',
        91: 'Specifics Unknown',
        92: 'Backing Vehicle',
        93: 'Other Vehicle or Object',
        98: 'Other Crash Type',
        99: 'Unknown Crash Type',
        70: 'Initial Same Directions (Turning Right)',
        71: 'Initial Same Directions (Going Straight)',
        72: 'Initial Same Directions (Turning Left)',
        73: 'Initial Same Directions (Going Straight)',
        74: 'Specifics Other',
        75: 'Specifics Unknown',
        76: 'Turn Into Same Direction (Turning Left)',
        77: 'Turn Into Same Direction (Going Straight)',
        78: 'Turn Into Same Direction (Turning Right)',
        79: 'Turn Into Same Direction (Going Straight)',
        80: 'Turn Into Opposite Directions (Turning Right)',
        81: 'Turn Into Opposite Directions (Going Straight)',
        82: 'Turn Into Opposite Directions (Turning Left)',
        83: 'Turn Into Opposite Directions (Going Straight)',
        84: 'Specifics Other',
        85: 'Specifics Unknown'
    }

    df1['crashtype'] = df1['crashtype'].map(crashtype)
```

In [919]:
```
df1['crashtype'].value_counts()
```

Out[919]:
```
Other Crash Type                                    711
Drive Off Road                                      618
Pedestrian/Animal                                   601
Lateral Move (Left/Right)                           295
Lateral Move (Going Straight)                       294
Control/Traction Loss                               257
Specifics Other                                     203
Initial Opposite Directions (Left/Right)             95
Initial Opposite Directions (Going Straight)         95
Turn Into Opposite Directions (Going Straight)       90
Striking from the Left                               89
Struck on the Left                                   89
Turn Into Opposite Directions (Turning Left)         86
Striking from the Right                              62
Slower                                               62
Struck on the Right                                  62
Slower, Going Straight                               60
Stopped                                              57
Stopped, Straight                                    57
Straight Ahead on Left/Right.                        30
Decelerating (Slowing)                               29
Decelerating (Slowing), Going Straight               29
No Impact                                            22
End Departure                                        21
Parked Vehicle                                       21
Changing Lanes to the Right                          16
Changing Lanes to the Left                           12
Turn Into Same Direction (Going Straight)            12
Turn Into Same Direction (Turning Right)              9
Stationary Object                                     7
Backing Vehicle                                       7
Initial Same Directions (Going Straight)              6
Initial Same Directions (Turning Left)                5
Specifics Unknown                                     4
Turn Into Opposite Directions (Turning Right)         4
Straight Ahead on Left.                               4
Turn Into Same Direction (Turning Left)               3
Unknown Crash Type                                    2
Avoid Collision with Vehicle, Pedestrian, Animal      2
Slower, Going Right                                   1
Other Vehicle or Object                               1
Initial Same Directions (Turning Right)               1
Slower, Going Left                                    1
Name: crashtype, dtype: int64
```

In [918]:
```
df1.to_csv('sample1.csv', index=False)
```

In [942]:

```
# Create a dataframe of residents
residents_df = df1[df1['is_resident'] == True]

# Create a dataframe of non-residents
non_residents_df = df1[df1['is_resident'] == False]

# Calculate the total number of crashes for each type of crash in the resident
residents_crash_counts = residents_df.groupby('crashtype').size().reset_index(

# Calculate the total number of crashes for each type of crash in the non-resi
non_residents_crash_counts = non_residents_df.groupby('crashtype').size().rese


residents_dr_drug_counts = residents_df.groupby('dr_drug').size().reset_index(
non_residents_dr_drug_counts = non_residents_df.groupby('dr_drug').size().rese

residents_dr_imp_counts = residents_df.groupby('dr_imp').size().reset_index(na
non_residents_dr_imp_counts = non_residents_df.groupby('dr_imp').size().reset_

residents_dr_dist_counts = residents_df.groupby('dr_dist').size().reset_index(
non_residents_dr_dist_counts = non_residents_df.groupby('dr_dist').size().rese

residents_dr_spd_counts = residents_df.groupby('dr_spd').size().reset_index(na
non_residents_dr_spd_counts = non_residents_df.groupby('dr_spd').size().reset_

print(f"The proportion of drivers involved in fatal crashes in communities whe
```

```
The proportion of drivers involved in fatal crashes in communities where they
live and tested drug positive are: 23.195876288659793
```

In [930]:

```
residents_dr_drug_counts['count'][1]
```

Out[930]:

```
225
```

In [939]:

```
(residents_dr_drug_counts['count'][1]/len(residents_df) )*100
```

Out[939]:

```
23.195876288659793
```

In [943]:

```
residents_crash_counts
```

Out[943]:

	crashtype	count
0	Backing Vehicle	3
1	Changing Lanes to the Left	2
2	Changing Lanes to the Right	4
3	Control/Traction Loss	67
4	Decelerating (Slowing)	1
5	Decelerating (Slowing), Going Straight	3
6	Drive Off Road	153
7	End Departure	6
8	Initial Opposite Directions (Going Straight)	28
9	Initial Opposite Directions (Left/Right)	35
10	Initial Same Directions (Going Straight)	2
11	Initial Same Directions (Turning Right)	1
12	Lateral Move (Going Straight)	66
13	Lateral Move (Left/Right)	65
14	No Impact	4
15	Other Crash Type	130
16	Other Vehicle or Object	1
17	Parked Vehicle	5
18	Pedestrian/Animal	137
19	Slower	15
20	Slower, Going Straight	9
21	Specifics Other	42
22	Specifics Unknown	2
23	Stationary Object	3
24	Stopped	12
25	Stopped, Straight	19
26	Straight Ahead on Left.	1
27	Straight Ahead on Left/Right.	6
28	Striking from the Left	17
29	Striking from the Right	22
30	Struck on the Left	25
31	Struck on the Right	16
32	Turn Into Opposite Directions (Going Straight)	21
33	Turn Into Opposite Directions (Turning Left)	33
34	Turn Into Opposite Directions (Turning Right)	1
35	Turn Into Same Direction (Going Straight)	3

	crashtype	count
36	Turn Into Same Direction (Turning Left)	3
37	Turn Into Same Direction (Turning Right)	6
38	Unknown Crash Type	1

```
In [ ]:
```