FARS Analysis

Written by Domas Budrys

In [29]:

```
#Question 1 and general output data

import os.path
import csv
import collections

total_all_days = 0
total_all_days = []

total_fatal=0
total_fatal = []

def outputData1(state_id, pathFile):
    state = []
    name = []
    pop_estimate = []

    pathname = os.path.join("FARS", "nst-est2017-alldata.csv")
    #Opening nst-est2017-alldata.csv to get state IDs and names
    with open(pathname) as dataIn:
        reader = csv.DictReader(dataIn)

        for row in reader:

            state.append(row['STATE'])
            name.append(row['NAME'])
            pop_estimate.append(row['POPESTIMATE2016'])

    # Indexing
    state_index = state.index(state_id)
    state_name = name[state_index]
    state_pop = pop_estimate[state_index]


    #st_case_column = []
    st_day_week_column = []



    pathname = os.path.join('FARS', pathFile)

    with open(pathname) as dataIn:

        reader = csv.DictReader(dataIn)

        for row in reader:


            st_day_week_column.append(row['DAY_WEEK'])

            #To count all days QUESTION 2
            total_all_days.append(row['DAY_WEEK'])

            #To count total fatalities Question 7
            total_fatal.append(row['MAN_COLL'])

    week_day_names = []
    #change each each number of the day to the name of the day
    for day in st_day_week_column:
        if (day == '1'):
            day = "Sunday"
            week_day_names.append(day)
        elif(day == '2'):
            day = "Monday"
            week_day_names.append(day)
        elif(day == '3'):
```

```
            day = "Tuesday"
            week_day_names.append(day)
        elif(day == '4'):
            day = "Wednesday"
            week_day_names.append(day)
        elif(day == '5'):
            day = "Thursday"
            week_day_names.append(day)
        elif(day == '6'):
            day = "Friday"
            week_day_names.append(day)
        elif(day == '7'):
            day = "Saturday"
            week_day_names.append(day)
        else:
            day = None

    #collections.Counter is used to display the most common one
    display_count = []
    for i in collections.Counter(week_day_names).most_common(1):
        display_count += i

    #output string
    output_days = state_name + ": Day(" + display_count[0]+")", "Count(" + str(display_count[1]) +
")"

    return output_days
```

Question 1

In this cell we create a function called *outputData* which will display data such as, which day occurs the most accidents throughout the week. This function will take in 2 variables: **state_id** and **pathFile**. **state_id** is the number which states are identified and **pathFile** is the name of the file. Then, we separately open *nst-est2017-alldata.csv* to get names of states. Then,

```
st_day_week_column.append(row['DAY_WEEK'])
```

allows us to open any file specified in the fuction and assign each of the WEEK_DAY to the list. After assigning data to *st_day_week_column* list, we are able to specify which integers means which day of the week by:

```
for day in st_day_week_column:
    if (day == '1'):
        day = "Sunday"
        week_day_names.append(day)
    elif(day == '2'):
        day = "Monday"
        week_day_names.append(day)
    elif(day == '3'):
        day = "Tuesday"
        .........
```

And finally, by using collections.Counter() function we are able to display top result of *week_day_names* list

In [30]:

```
print("Question 1")
print("The day of the week that has the most accidents for each week and the count of it")
print()
print (outputData1('1','accident_01.csv' ))
print (outputData1('2','accident_02.csv' ))
print (outputData1('4','accident_04.csv' ))
print (outputData1('6','accident_06.csv' ))
print (outputData1('8','accident_08.csv' ))
print (outputData1('9','accident_09.csv' ))
print (outputData1('10','accident_10.csv' ))

print (outputData1('11','accident_11.csv' ))
print (outputData1('12','accident_12.csv' ))
print (outputData1('13','accident_13.csv' ))
print (outputData1('15','accident_15.csv' ))
print (outputData1('16','accident_16.csv' ))
print (outputData1('17','accident_17.csv' ))
```

```
print (outputData1('18','accident_18.csv' ))
print (outputData1('19','accident_19.csv' ))
print (outputData1('20','accident_20.csv' ))

print (outputData1('21','accident_21.csv' ))
print (outputData1('22','accident_22.csv' ))
print (outputData1('23','accident_23.csv' ))
print (outputData1('24','accident_24.csv' ))
print (outputData1('25','accident_25.csv' ))
print (outputData1('26','accident_26.csv' ))
print (outputData1('27','accident_27.csv' ))
print (outputData1('28','accident_28.csv' ))
print (outputData1('29','accident_29.csv' ))
print (outputData1('30','accident_30.csv' ))

print (outputData1('31','accident_31.csv' ))
print (outputData1('32','accident_32.csv' ))
print (outputData1('33','accident_33.csv' ))
print (outputData1('34','accident_34.csv' ))
print (outputData1('35','accident_35.csv' ))
print (outputData1('36','accident_36.csv' ))
print (outputData1('37','accident_37.csv' ))
print (outputData1('38','accident_38.csv' ))
print (outputData1('39','accident_39.csv' ))
print (outputData1('40','accident_40.csv' ))

print (outputData1('41','accident_41.csv' ))
print (outputData1('42','accident_42.csv' ))
print (outputData1('44','accident_44.csv' ))
print (outputData1('45','accident_45.csv' ))
print (outputData1('46','accident_46.csv' ))
print (outputData1('47','accident_47.csv' ))
print (outputData1('48','accident_48.csv' ))
print (outputData1('49','accident_49.csv' ))
print (outputData1('50','accident_50.csv' ))

print (outputData1('51','accident_51.csv' ))
print (outputData1('53','accident_53.csv' ))
print (outputData1('54','accident_54.csv' ))
print (outputData1('55','accident_55.csv' ))
print (outputData1('56','accident_56.csv' ))
```

```
Question 1
The day of the week that has the most accidents for each week and the count of it

('Alabama: Day(Saturday)', 'Count(166)')
('Alaska: Day(Friday)', 'Count(15)')
('Arizona: Day(Saturday)', 'Count(150)')
('California: Day(Saturday)', 'Count(614)')
('Colorado: Day(Friday)', 'Count(95)')
('Connecticut: Day(Thursday)', 'Count(50)')
('Delaware: Day(Saturday)', 'Count(26)')
('District of Columbia: Day(Friday)', 'Count(5)')
('Florida: Day(Saturday)', 'Count(507)')
('Georgia: Day(Saturday)', 'Count(258)')
('Hawaii: Day(Saturday)', 'Count(25)')
('Idaho: Day(Friday)', 'Count(38)')
('Illinois: Day(Saturday)', 'Count(202)')
('Indiana: Day(Friday)', 'Count(127)')
('Iowa: Day(Saturday)', 'Count(65)')
('Kansas: Day(Saturday)', 'Count(63)')
('Kentucky: Day(Saturday)', 'Count(133)')
('Louisiana: Day(Sunday)', 'Count(128)')
('Maine: Day(Saturday)', 'Count(30)')
('Maryland: Day(Saturday)', 'Count(96)')
('Massachusetts: Day(Saturday)', 'Count(61)')
('Michigan: Day(Saturday)', 'Count(175)')
('Minnesota: Day(Saturday)', 'Count(58)')
('Mississippi: Day(Saturday)', 'Count(118)')
('Missouri: Day(Saturday)', 'Count(156)')
('Montana: Day(Saturday)', 'Count(34)')
('Nebraska: Day(Sunday)', 'Count(38)')
('Nevada: Day(Saturday)', 'Count(56)')
('New Hampshire: Day(Friday)', 'Count(26)')
('New Jersey: Day(Saturday)', 'Count(105)')
('New Mexico: Day(Saturday)', 'Count(69)')
('New York: Day(Saturday)', 'Count(159)')
```

```
('North Carolina: Day(Saturday)', 'Count(219)')
('North Dakota: Day(Saturday)', 'Count(22)')
('Ohio: Day(Saturday)', 'Count(194)')
('Oklahoma: Day(Saturday)', 'Count(109)')
('Oregon: Day(Saturday)', 'Count(76)')
('Pennsylvania: Day(Friday)', 'Count(186)')
('Rhode Island: Day(Thursday)', 'Count(14)')
('South Carolina: Day(Saturday)', 'Count(182)')
('South Dakota: Day(Saturday)', 'Count(18)')
('Tennessee: Day(Saturday)', 'Count(187)')
('Texas: Day(Saturday)', 'Count(604)')
('Utah: Day(Saturday)', 'Count(44)')
('Vermont: Day(Wednesday)', 'Count(11)')
('Virginia: Day(Saturday)', 'Count(139)')
('Washington: Day(Saturday)', 'Count(87)')
('West Virginia: Day(Wednesday)', 'Count(41)')
('Wisconsin: Day(Saturday)', 'Count(112)')
('Wyoming: Day(Thursday)', 'Count(19)')
```

In [31]:

```
#Question 2
total_day_count = []

for day in total_all_days:
        if (day == '1'):
            day = "Sunday"
            total_day_count.append(day)
        elif(day == '2'):
            day = "Monday"
            total_day_count.append(day)
        elif(day == '3'):
            day = "Tuesday"
            total_day_count.append(day)
        elif(day == '4'):
            day = "Wednesday"
            total_day_count.append(day)
        elif(day == '5'):
            day = "Thursday"
            total_day_count.append(day)
        elif(day == '6'):

            day = "Friday"
            total_day_count.append(day)
        elif(day == '7'):
            day = "Saturday"
            total_day_count.append(day)
        else:
            day = None

total_display_count = []
for i in collections.Counter(total_day_count).most_common(1):
    total_display_count += i

day_display = "Day: "+ total_display_count[0] +", Count (" +  str(total_display_count[1])+")"

print("Question 2")
print("The day of the week for the USA that has the most accidents and the count of it")
print()
print(day_display)
```

```
Question 2
The day of the week for the USA that has the most accidents and the count of it

Day: Saturday, Count (6030)
```

Question 2

In order to calculate the the total number of all days and display the count of the week day when the most accidents occurred, we must assign this data to the list. We are able to do this is *outputData* function with `total_all_days.append(row['DAY_WEEK'])`. Then, with the code

```
    for day in total_all_days:
```

```
      if (day == '1'):
          day = "Sunday"
          total_day_count.append(day)
      elif(day == '2'):
          day = "Monday"
          total_day_count.append(day)
      elif(day == '3'):
      .......
```

we are able to we to specify which integers means which day of the week. And finally we display the top number by using collections.Counter() function

In [32]:

```
#Question 3
total_all_hours = 0
total_all_hours = []

def outputData2(state_id, pathFile):
    state = []
    name = []

    pathname = os.path.join("FARS", "nst-est2017-alldata.csv")
    #Opening nst-est2017-alldata.csv to get state IDs and names
    with open(pathname) as dataIn:
        reader = csv.DictReader(dataIn)

        for row in reader:

            state.append(row['STATE'])
            name.append(row['NAME'])

    # Indexing
    state_index = state.index(state_id)
    state_name = name[state_index]


    #st_case_column = []
    st_hour_column = []


    pathname = os.path.join('FARS', pathFile)

    with open(pathname) as dataIn:

        reader = csv.DictReader(dataIn)

        for row in reader:

           # st_case_column.append(row['ST_CASE'])
            st_hour_column.append(row['HOUR'])

            #To count all days QUESTION 4
            total_all_hours.append(row['HOUR'])


    total_hour_count = []

    for hour in st_hour_column:
            if (hour == '99'):
                hour = "Unknown"
                total_hour_count.append(hour)
            else:
                total_hour_count.append(hour)

    #collections.Counter is used to display the most common one
    display_hour_count = []
    for i in collections.Counter(total_hour_count).most_common(1):
        display_hour_count += i


    output_hours = state_name + ": Hour(" + str(display_hour_count[0])+")", "Count(" +str(display_h
our_count[1]) + ")"
```

```
    return output_hours
```

Question 3

outputData2() function is almost the same as outputData1() function, but it is copied to the above cell in order to provide a detailed output for the user. Several changes include, `st_hour_column.append(row['HOUR'])` which is used to append all of the HOUR data to the list **st_hour_column**. Then we create a forLoop to check if any value from HOUR column in csv file is '99' and assign value of it to 'Unknown', else we are assigning rest of the numbers as they appear:

```
for hour in st_hour_column:
     if (hour == '99'):
         hour = "Unknown"
         total_hour_count.append(hour)
     else:
         total_hour_count.append(hour)
```

Finally, we are able to count the top 1 number by using collections.Counter()

In [33]:

```
print("Question 3")
print("The hour(military time) of the day in each state that has the most accidents and the count
of it")

print (outputData2('1','accident_01.csv' ))
print (outputData2('2','accident_02.csv' ))
print (outputData2('4','accident_04.csv' ))
print (outputData2('6','accident_06.csv' ))
print (outputData2('8','accident_08.csv' ))
print (outputData2('9','accident_09.csv' ))
print (outputData2('10','accident_10.csv' ))

print (outputData2('11','accident_11.csv' ))
print (outputData2('12','accident_12.csv' ))
print (outputData2('13','accident_13.csv' ))
print (outputData2('15','accident_15.csv' ))
print (outputData2('16','accident_16.csv' ))
print (outputData2('17','accident_17.csv' ))
print (outputData2('18','accident_18.csv' ))
print (outputData2('19','accident_19.csv' ))
print (outputData2('20','accident_20.csv' ))

print (outputData2('21','accident_21.csv' ))
print (outputData2('22','accident_22.csv' ))
print (outputData2('23','accident_23.csv' ))
print (outputData2('24','accident_24.csv' ))
print (outputData2('25','accident_25.csv' ))
print (outputData2('26','accident_26.csv' ))
print (outputData2('27','accident_27.csv' ))
print (outputData2('28','accident_28.csv' ))
print (outputData2('29','accident_29.csv' ))
print (outputData2('30','accident_30.csv' ))

print (outputData2('31','accident_31.csv' ))
print (outputData2('32','accident_32.csv' ))
print (outputData2('33','accident_33.csv' ))
print (outputData2('34','accident_34.csv' ))
print (outputData2('35','accident_35.csv' ))
print (outputData2('36','accident_36.csv' ))
print (outputData2('37','accident_37.csv' ))
print (outputData2('38','accident_38.csv' ))
print (outputData2('39','accident_39.csv' ))
print (outputData2('40','accident_40.csv' ))

print (outputData2('41','accident_41.csv' ))
print (outputData2('42','accident_42.csv' ))
print (outputData2('44','accident_44.csv' ))
print (outputData2('45','accident_45.csv' ))
print (outputData2('46','accident_46.csv' ))
print (outputData2('47','accident_47.csv' ))
print (outputData2('48','accident_48.csv' ))
```

```
print (outputData2('49','accident_49.csv' ))
print (outputData2('50','accident_50.csv' ))

print (outputData2('51','accident_51.csv' ))
print (outputData2('53','accident_53.csv' ))
print (outputData2('54','accident_54.csv' ))
print (outputData2('55','accident_55.csv' ))
print (outputData2('56','accident_56.csv' ))
```

```
Question 3
The hour(military time) of the day in each state that has the most accidents and the count of it
('Alabama: Hour(16)', 'Count(55)')
('Alaska: Hour(5)', 'Count(7)')
('Arizona: Hour(19)', 'Count(65)')
('California: Hour(20)', 'Count(210)')
('Colorado: Hour(13)', 'Count(38)')
('Connecticut: Hour(17)', 'Count(22)')
('Delaware: Hour(21)', 'Count(10)')
('District of Columbia: Hour(2)', 'Count(4)')
('Florida: Hour(21)', 'Count(208)')
('Georgia: Hour(18)', 'Count(84)')
('Hawaii: Hour(20)', 'Count(10)')
('Idaho: Hour(16)', 'Count(17)')
('Illinois: Hour(16)', 'Count(64)')
('Indiana: Hour(17)', 'Count(50)')
('Iowa: Hour(17)', 'Count(26)')
('Kansas: Hour(16)', 'Count(28)')
('Kentucky: Hour(16)', 'Count(57)')
('Louisiana: Hour(17)', 'Count(48)')
('Maine: Hour(18)', 'Count(12)')
('Maryland: Hour(22)', 'Count(32)')
('Massachusetts: Hour(22)', 'Count(25)')
('Michigan: Hour(18)', 'Count(59)')
('Minnesota: Hour(15)', 'Count(28)')
('Mississippi: Hour(17)', 'Count(41)')
('Missouri: Hour(18)', 'Count(57)')
('Montana: Hour(21)', 'Count(13)')
('Nebraska: Hour(Unknown)', 'Count(29)')
('Nevada: Hour(23)', 'Count(22)')
('New Hampshire: Hour(19)', 'Count(11)')
('New Jersey: Hour(18)', 'Count(42)')
('New Mexico: Hour(20)', 'Count(31)')
('New York: Hour(17)', 'Count(59)')
('North Carolina: Hour(18)', 'Count(92)')
('North Dakota: Hour(16)', 'Count(9)')
('Ohio: Hour(19)', 'Count(77)')
('Oklahoma: Hour(16)', 'Count(41)')
('Oregon: Hour(17)', 'Count(31)')
('Pennsylvania: Hour(17)', 'Count(64)')
('Rhode Island: Hour(1)', 'Count(6)')
('South Carolina: Hour(17)', 'Count(64)')
('South Dakota: Hour(14)', 'Count(10)')
('Tennessee: Hour(16)', 'Count(61)')
('Texas: Hour(20)', 'Count(200)')
('Utah: Hour(13)', 'Count(26)')
('Vermont: Hour(15)', 'Count(6)')
('Virginia: Hour(17)', 'Count(60)')
('Washington: Hour(15)', 'Count(34)')
('West Virginia: Hour(15)', 'Count(24)')
('Wisconsin: Hour(17)', 'Count(45)')
('Wyoming: Hour(12)', 'Count(11)')
```

In [34]:

```
#Question 4
total_hour_count = []

total_display_hour_count = []
for i in collections.Counter(total_all_hours).most_common(1):
    total_display_hour_count += i


hour_display = "Hour: "+ total_display_hour_count[0] +", Count (" +  str(total_display_hour_count[1
])+")"
print("Question 4")
```

```
print("The hour(military time) of the day which has the most accidents in the USA, and the count o
f it")
print()
print(hour_display)
```

```
Question 4
The hour(military time) of the day which has the most accidents in the USA, and the count of it

Hour: 18, Count (1954)
```

Question 4

To calculate value for Question 4, we use `total_all_hours.append(row['HOUR'])` in our helper function outputData2(). This statement allows to append data each time outputData2() is executed and store every value is this list. Then, we are able to retrieve the most common value by using collections.Counter()

In [35]:

```
#Question 5

total_all_drunk = 0
total_all_drunk =[]


def outputData3(state_id, pathFile):
    state = []
    name = []

    pathname = os.path.join("FARS", "nst-est2017-alldata.csv")
    #Opening nst-est2017-alldata.csv to get state IDs and names
    with open(pathname) as dataIn:
        reader = csv.DictReader(dataIn)

        for row in reader:

            state.append(row['STATE'])
            name.append(row['NAME'])

    # Indexing
    state_index = state.index(state_id)
    state_name = name[state_index]

    drunk_column = []


    pathname = os.path.join('FARS', pathFile)

    with open(pathname) as dataIn:

        reader = csv.DictReader(dataIn)

        for row in reader:

            drunk_column.append(row['DRUNK_DR'])


    drunk_column = [int(i) for i in drunk_column]
    total_drunk_count = []

    for drunk in drunk_column:
        if (drunk > 0):
            total_drunk_count.append(drunk)


    drunk_percent = (sum(total_drunk_count) / len(drunk_column))*100


    #Question 6
    total_all_drunk.append(str(drunk_percent))

    output_percent = state_name + ": " + str(round(drunk_percent, 2))+"%"


    return output_percent
```

Question 5

outputData3() function is almost the same as outputData1() function, but it is copied to the above cell in order to provide a detailed output for the user. Several changes include, `drunk_column.append(row['DRUNK_DR'])` which is used to append all of the DRUNK_DR data to the list drunk_column. Then we create a forLoop to is value of **drunk_column** is more than 0:

```
for drunk in drunk_column:
    if (drunk > 0):
        total_drunk_count.append(drunk)
```

and append all of the eligible number to new list **total_drunk_count** Then, to get the percentage we use `drunk_percent = (sum(total_drunk_count) / len(drunk_column))*100`

In [36]:

```
print("Question 5")
print("Percentages of fatal accidents that involved at least one drunk driver")
print()
print (outputData3('1','accident_01.csv' ))
print (outputData3('2','accident_02.csv' ))
print (outputData3('4','accident_04.csv' ))
print (outputData3('6','accident_06.csv' ))
print (outputData3('8','accident_08.csv' ))
print (outputData3('9','accident_09.csv' ))
print (outputData3('10','accident_10.csv' ))

print (outputData3('11','accident_11.csv' ))
print (outputData3('12','accident_12.csv' ))
print (outputData3('13','accident_13.csv' ))
print (outputData3('15','accident_15.csv' ))
print (outputData3('16','accident_16.csv' ))
print (outputData3('17','accident_17.csv' ))
print (outputData3('18','accident_18.csv' ))
print (outputData3('19','accident_19.csv' ))
print (outputData3('20','accident_20.csv' ))

print (outputData3('21','accident_21.csv' ))
print (outputData3('22','accident_22.csv' ))
print (outputData3('23','accident_23.csv' ))
print (outputData3('24','accident_24.csv' ))
print (outputData3('25','accident_25.csv' ))
print (outputData3('26','accident_26.csv' ))
print (outputData3('27','accident_27.csv' ))
print (outputData3('28','accident_28.csv' ))
print (outputData3('29','accident_29.csv' ))
print (outputData3('30','accident_30.csv' ))

print (outputData3('31','accident_31.csv' ))
print (outputData3('32','accident_32.csv' ))
print (outputData3('33','accident_33.csv' ))
print (outputData3('34','accident_34.csv' ))
print (outputData3('35','accident_35.csv' ))
print (outputData3('36','accident_36.csv' ))
print (outputData3('37','accident_37.csv' ))
print (outputData3('38','accident_38.csv' ))
print (outputData3('39','accident_39.csv' ))
print (outputData3('40','accident_40.csv' ))

print (outputData3('41','accident_41.csv' ))
print (outputData3('42','accident_42.csv' ))
print (outputData3('44','accident_44.csv' ))
print (outputData3('45','accident_45.csv' ))
print (outputData3('46','accident_46.csv' ))
print (outputData3('47','accident_47.csv' ))
print (outputData3('48','accident_48.csv' ))
print (outputData3('49','accident_49.csv' ))
print (outputData3('50','accident_50.csv' ))

print (outputData3('51','accident_51.csv' ))
print (outputData3('53','accident_53.csv' ))
```

```
print (outputData3('54','accident_54.csv' ))
print (outputData3('55','accident_55.csv' ))
print (outputData3('56','accident_56.csv' ))
```

```
Question 5
Percentages of fatal accidents that involved at least one drunk driver

Alabama: 14.94%
Alaska: 46.15%
Arizona: 23.93%
California: 24.19%
Colorado: 34.77%
Connecticut: 31.67%
Delaware: 33.62%
District of Columbia: 38.46%
Florida: 21.89%
Georgia: 22.15%
Hawaii: 25.69%
Idaho: 28.88%
Illinois: 27.02%
Indiana: 17.97%
Iowa: 26.12%
Kansas: 20.73%
Kentucky: 25.29%
Louisiana: 31.25%
Maine: 31.13%
Maryland: 25.21%
Massachusetts: 29.25%
Michigan: 26.12%
Minnesota: 27.73%
Mississippi: 16.56%
Missouri: 28.57%
Montana: 48.54%
Nebraska: 40.21%
Nevada: 32.01%
New Hampshire: 30.0%
New Jersey: 22.32%
New Mexico: 30.17%
New York: 17.51%
North Carolina: 28.93%
North Dakota: 49.02%
Ohio: 34.76%
Oklahoma: 28.37%
Oregon: 27.35%
Pennsylvania: 24.17%
Rhode Island: 37.5%
South Carolina: 35.9%
South Dakota: 42.72%
Tennessee: 22.98%
Texas: 25.62%
Utah: 20.08%
Vermont: 47.37%
Virginia: 29.92%
Washington: 32.14%
West Virginia: 27.6%
Wisconsin: 33.27%
Wyoming: 29.0%
```

In [37]:

```
#Question 6
total_all_drunk = [float(i) for i in total_all_drunk]
all_drunk_percent = sum(total_all_drunk) / len(total_all_drunk)
output_drunk_percent = round(all_drunk_percent, 2)

print("Question 6")
print()
print ("The percentage of fatal accidents in USA that involved at "+
       "least one drunk driver is : ", str(output_drunk_percent)+"%")
```

```
Question 6

The percentage of fatal accidents in USA that involved at least one drunk driver is :  29.54%
```

Question 6

To get the total percentage of fatal accidents in the USA, we append all of the final percentages of each state to **total_all_drunk** in helper function outputData3(). Then, we convert each value to float number in order to output the percentage. Then we can get total percentage using:

```
    all_drunk_percent = sum(total_all_drunk) / len(total_all_drunk)
```

In [38]:

```
#Question 7
named_fatal = []

for coll in total_fatal:

    if (coll == '0'):
        coll = "Not Collision with Motor Vehicle in Transport"
        named_fatal.append(coll)

    elif (coll == '1'):
        coll = "Front-to-Rear"
        named_fatal.append(coll)

    elif (coll == '2'):
        coll = "Front-to-Front"
        named_fatal.append(coll)

    elif (coll == '6'):
        coll = "Angle"
        named_fatal.append(coll)

    elif (coll == '7'):
        coll = "Sideswipe – Same Direction"
        named_fatal.append(coll)

    elif (coll == '8'):
        coll = "Sideswipe – Opposite Direction"
        named_fatal.append(coll)

    elif (coll == '9'):
        coll = "Rear-to-Side"
        named_fatal.append(coll)

    elif (coll == '10'):
        coll = "Rear-to-Rear"
        named_fatal.append(coll)

    elif (coll == '11'):
        coll = "Other (End-Swipes and Others)"
        named_fatal.append(coll)

    elif (coll == '98'):
        coll = "Not Reported"
        named_fatal.append(coll)

    elif (coll == '99'):
        coll = "Unknown"
        named_fatal.append(coll)

sorted_fatal = []

print("Question 7")
print()

for i in collections.Counter(named_fatal).most_common(14):

    print(i)
```

```
Question 7

('Not Collision with Motor Vehicle in Transport', 21010)
('Angle', 6044)
('Front-to-Front', 3444)
('Front-to-Rear', 2316)
```

```
('Sideswipe – Same Direction', 514)
('Sideswipe – Opposite Direction', 404)
('Other (End-Swipes and Others)', 86)
('Unknown', 76)
('Rear-to-Side', 32)
('Not Reported', 23)
('Rear-to-Rear', 2)
```

Question 7

To store all of the data we create the list **total_fatal** in outputData1() function. Then using **total_fatal** list, we check the value of each data entry and assign the proper description of it and append to the new list called **named_fatal**. Finally, we are able print sorted list based on the count of collisions

In [ ]:

```
```

In [39]:

```
#Question 8 AND Question 9
accidents = 0
accidents = []

drunk_pop = 0
drunk_pop =[]

def outputData4(state_id, pathFile):
    state = []
    name = []
    pop_estimate = []

    pathname = os.path.join("FARS", "nst-est2017-alldata.csv")
    #Opening nst-est2017-alldata.csv to get state IDs and names
    with open(pathname) as dataIn:
        reader = csv.DictReader(dataIn)

        for row in reader:

            state.append(row['STATE'])
            name.append(row['NAME'])
            pop_estimate.append(row['POPESTIMATE2016'])

    # Indexing
    state_index = state.index(state_id)
    state_name = name[state_index]
    state_pop = pop_estimate[state_index]

    state_column = []
    drunk_column = []

    pathname = os.path.join('FARS', pathFile)

    with open(pathname) as dataIn:

        reader = csv.DictReader(dataIn)

        for row in reader:

            #to count total accidents Question 8
            state_column.append(row['STATE'])

            #to count total drunks accidents Question 9
            drunk_column.append(row['DRUNK_DR'])


    calc_each_fatal = round((len(state_column) / int(state_pop)) * 10000, 4)

    #output string
    output_fatal = state_name + ": " , str(calc_each_fatal)

    accidents.append(output_fatal)
```

```
    drunk_column = [int(i) for i in drunk_column]

    total_drunk_pop_count = []

    for drunk in drunk_column:
        if (drunk > 0):
            total_drunk_pop_count.append(drunk)


    calc_each_drunk = round((sum(total_drunk_pop_count) / int(state_pop)) * 10000, 4)

    output_drunk = state_name + ": " , str(calc_each_drunk)

    drunk_pop.append(output_drunk)

    return ()
```

Question 8 and Question 9

outputData4() function is used for Questions 8 & 9. To get the number of all accidents in the state we simply append the value of STATE column from the .csv file to **state_column** variable and then calculate the length of it. Then we have to divide it by population estimate which we have appended to **pop_estimate** and then, multiply by 10000. This will give is a percentage of specified state. The same steps are applied to solve Question 9 except the calculate the sum of DRUNK_DR because there can be more than one drunk driver involved in accident.

In [40]:

```
outputData4('1','accident_01.csv' )
outputData4('2','accident_02.csv' )
outputData4('4','accident_04.csv' )
outputData4('6','accident_06.csv' )
outputData4('8','accident_08.csv' )
outputData4('9','accident_09.csv' )
outputData4('10','accident_10.csv' )

outputData4('11','accident_11.csv' )
outputData4('12','accident_12.csv' )
outputData4('13','accident_13.csv' )
outputData4('15','accident_15.csv' )
outputData4('16','accident_16.csv' )
outputData4('17','accident_17.csv' )
outputData4('18','accident_18.csv' )
outputData4('19','accident_19.csv' )
outputData4('20','accident_20.csv' )

outputData4('21','accident_21.csv' )
outputData4('22','accident_22.csv' )
outputData4('23','accident_23.csv' )
outputData4('24','accident_24.csv' )
outputData4('25','accident_25.csv' )
outputData4('26','accident_26.csv' )
outputData4('27','accident_27.csv' )
outputData4('28','accident_28.csv' )
outputData4('29','accident_29.csv' )
outputData4('30','accident_30.csv' )

outputData4('31','accident_31.csv' )
outputData4('32','accident_32.csv' )
outputData4('33','accident_33.csv' )
outputData4('34','accident_34.csv' )
outputData4('35','accident_35.csv' )
outputData4('36','accident_36.csv' )
outputData4('37','accident_37.csv' )
outputData4('38','accident_38.csv' )
outputData4('39','accident_39.csv' )
outputData4('40','accident_40.csv' )

outputData4('41','accident_41.csv' )
outputData4('42','accident_42.csv' )
outputData4('44','accident_44.csv' )
outputData4('45','accident_45.csv' )
outputData4('46','accident_46.csv' )
```

```
outputData4('47','accident_47.csv' )
outputData4('48','accident_48.csv' )
outputData4('49','accident_49.csv' )
outputData4('50','accident_50.csv' )

outputData4('51','accident_51.csv' )
outputData4('53','accident_53.csv' )
outputData4('54','accident_54.csv' )
outputData4('55','accident_55.csv' )
outputData4('56','accident_56.csv' )
```

Out[40]:

```
()
```

In [41]:

```
#Question 8
accidents_progress = 0
accidents_progress = []

for a in accidents:
    aa = float(a[1])
    forming= aa, a[0]
    accidents_progress.append(forming)

sorted_states = sorted(accidents_progress, reverse=True)

output_states= []
for s in sorted_states:
    formatted = s[1], str(s[0])+'%'
    output_states.append(formatted)


print("Question 8")
print()
print("Sorted list of fatal accident rate per 10000 people:")

for i in output_states:
    print(i )
```

```
Question 8

Sorted list of fatal accident rate per 10000 people:
('Mississippi: ', '2.1036%')
('Alabama: ', '1.9278%')
('South Carolina: ', '1.8872%')
('Kentucky: ', '1.72%')
('New Mexico: ', '1.7167%')
('Wyoming: ', '1.7097%')
('Montana: ', '1.6464%')
('Oklahoma: ', '1.5913%')
('Louisiana: ', '1.5023%')
('Tennessee: ', '1.4528%')
('Missouri: ', '1.425%')
('Florida: ', '1.4199%')
('Idaho: ', '1.3809%')
('Georgia: ', '1.3788%')
('West Virginia: ', '1.3671%')
('North Dakota: ', '1.35%')
('North Carolina: ', '1.3272%')
('Kansas: ', '1.3103%')
('Arizona: ', '1.2521%')
('Texas: ', '1.2209%')
('Delaware: ', '1.2176%')
('South Dakota: ', '1.1955%')
('Indiana: ', '1.1577%')
('Iowa: ', '1.1371%')
('Maine: ', '1.1351%')
('Oregon: ', '1.0915%')
('Alaska: ', '1.0519%')
('Nevada: ', '1.0309%')
('Nebraska: ', '1.017%')
('Colorado: ', '1.009%')
```

```
('Michigan: ', '0.9866%')
('New Hampshire: ', '0.9738%')
('Wisconsin: ', '0.9423%')
('Vermont: ', '0.9144%')
('Ohio: ', '0.906%')
('Virginia: ', '0.8581%')
('California: ', '0.8543%')
('Pennsylvania: ', '0.8509%')
('Utah: ', '0.8508%')
('Maryland: ', '0.7834%')
('Connecticut: ', '0.7832%')
('Illinois: ', '0.7814%')
('Hawaii: ', '0.7629%')
('Washington: ', '0.6922%')
('Minnesota: ', '0.6461%')
('New Jersey: ', '0.6337%')
('Massachusetts: ', '0.5261%')
('New York: ', '0.4865%')
('Rhode Island: ', '0.4539%')
('District of Columbia: ', '0.3799%')
```

Question 8 Continuing

After getting a percentage for each state, we can take each value and split it into separate elements. Then convert numbers into float values and place those values as the first element, pushing state name to be the second. This way we are able to apply sort function to each value and assign to new list:

```
    for a in accidents:
    aa = float(a[1])
    forming= aa, a[0]
    accidents_progress.append(forming)

    sorted_states = sorted(accidents_progress, reverse=True)
```

Then, we create another forLoop to format values to a proper output format.

In [42]:

```
#Question 9
drunk_progress = 0
drunk_progress = []
for d in drunk_pop:
    dd = float(d[1])
    forming= dd, d[0]
    drunk_progress.append(forming)

sorted_drunk = sorted(drunk_progress, reverse=True)

output_drunk= []
for dr in sorted_drunk:
    formatted = dr[1], str(dr[0])+'%'
    output_drunk.append(formatted)


print("Question 9")
print()
print("Sorted list of accidents that involded one or more drunk drivers rate per 10000 people:")

for i in output_drunk:
    print(i )
```

```
Question 9

Sorted list of accidents that involded one or more drunk drivers rate per 10000 people:
('Montana: ', '0.7991%')
('South Carolina: ', '0.6774%')
('North Dakota: ', '0.6618%')
('New Mexico: ', '0.5179%')
```

```
('South Dakota: ', '0.5107%')
('Wyoming: ', '0.4958%')
('Alaska: ', '0.4855%')
('Louisiana: ', '0.4695%')
('Oklahoma: ', '0.4514%')
('Kentucky: ', '0.4351%')
('Vermont: ', '0.4331%')
('Delaware: ', '0.4094%')
('Nebraska: ', '0.4089%')
('Missouri: ', '0.4071%')
('Idaho: ', '0.3988%')
('North Carolina: ', '0.384%')
('West Virginia: ', '0.3773%')
('Maine: ', '0.3533%')
('Colorado: ', '0.3508%')
('Mississippi: ', '0.3484%')
('Tennessee: ', '0.3339%')
('Nevada: ', '0.33%')
('Ohio: ', '0.3149%')
('Wisconsin: ', '0.3135%')
('Texas: ', '0.3128%')
('Florida: ', '0.3108%')
('Georgia: ', '0.3054%')
('Arizona: ', '0.2996%')
('Oregon: ', '0.2986%')
('Iowa: ', '0.297%')
('New Hampshire: ', '0.2921%')
('Alabama: ', '0.288%')
('Kansas: ', '0.2717%')
('Michigan: ', '0.2577%')
('Virginia: ', '0.2567%')
('Connecticut: ', '0.2481%')
('Washington: ', '0.2225%')
('Illinois: ', '0.2111%')
('Indiana: ', '0.208%')
('California: ', '0.2066%')
('Pennsylvania: ', '0.2057%')
('Maryland: ', '0.1975%')
('Hawaii: ', '0.196%')
('Minnesota: ', '0.1792%')
('Utah: ', '0.1708%')
('Rhode Island: ', '0.1702%')
('Massachusetts: ', '0.1539%')
('District of Columbia: ', '0.1461%')
('New Jersey: ', '0.1415%')
('New York: ', '0.0852%')
```

Question 9 Continuing

After getting a percentage for each state, we can take each value and split it into separate elements. Then convert numbers into float values place those values as the first element, pushing state name to be the second. This way we are able to apply sort function to each value and assign to new list:

```
for d in drunk_pop:
    dd = float(d[1])
    forming= dd, d[0]
    drunk_progress.append(forming)

sorted_drunk = sorted(drunk_progress, reverse=True)
```

Then, we create another forLoop to format values to a proper output format.