Project 11 -- OGUNMODEDE SEYI

INSTRUCTOR Help: Dr. Ward

- Help with figuring out how to write a function.
- Helped figuring out how to load the dataset.
- Helped debug error with my plot.

Question 1

In [1]:
```
# To see what is in the file, use ls and the files
# There is a folder for each year from 1975 - 2017
# # It has a file called 7581.csv and another state.csv
```

In [2]:
```
ls/anvil/projects/tdm/data/fars
```

```
1975/  1980/  1985/  1990/  1995/  2000/  2005/  2010/  2015/
1976/  1981/  1986/  1991/  1996/  2001/  2006/  2011/  2016/
1977/  1982/  1987/  1992/  1997/  2002/  2007/  2012/  2017/
1978/  1983/  1988/  1993/  1998/  2003/  2008/  2013/  7581.csv*
1979/  1984/  1989/  1994/  1999/  2004/  2009/  2014/  states.csv*
```

In [3]:
```
# For me to see what is inside directory for 1985
```

In [4]:
```
ls/anvil/projects/tdm/data/fars/1985
```

```
ACC_AUX.CSV   MIACC.CSV     MIPER.CSV    PERSON.CSV   VEHICLE.CSV
ACCIDENT.CSV  MIDRVACC.CSV  PER_AUX.CSV  VEH_AUX.CSV
```

In [5]:
```
import pandas as pd
```

In [6]:
```
myDF=pd.read_csv("/anvil/projects/tdm/data/fars/1985/ACCIDENT.CSV")
```

In [7]:
```
myDF.head(3)
```

Out[7]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE	...	CF
0	1	53	1	4	85	8	10	1	1	2	...	
1	1	85	1	6	85	5	45	1	2	1	...	
2	1	119	1	6	85	4	5	1	1	2	...	

3 rows × 47 columns

In [8]:
```
# We want to put 19 infront of these 85
# To do this you can make/convert your column as string - myDF["myCol"].astype(str).
```

```
# Then you can add another string. myDF["myCol"].astype(str) + "appending_this_string"
# We can append strings to every value in a column by first converting the column to s
```

In [9]:
```
# looking at the head focusing on the year column

myDF["YEAR"].head()
```

Out[9]:
```
0    85
1    85
2    85
3    85
4    85
Name: YEAR, dtype: int64
```

In [10]:
```
# Fisrt convert it to string and add 19 = 19+85 =1985

"19" + myDF["YEAR"].head().astype(str)
```

Out[10]:
```
0    1985
1    1985
2    1985
3    1985
4    1985
Name: YEAR, dtype: object
```

In [11]:
```
# If you run this nothing has changed, you have to assign it to the column

myDF.head()
```

Out[11]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE	...	CF
0	1	53	1	4	85	8	10	1	1	2	...	
1	1	85	1	6	85	5	45	1	2	1	...	
2	1	119	1	6	85	4	5	1	1	2	...	
3	1	89	1	4	85	20	45	1	1	2	...	
4	1	45	1	5	85	0	45	1	1	2	...	

5 rows × 47 columns

In [12]:
```
# This code changed the column to 1985

myDF["YEAR"]="19" + myDF["YEAR"].astype(str)
```

In [13]:
```
myDF.head()
```

Out[13]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE	...	CF
0	1	53	1	4	1985	8	10	1	1	2	...	
1	1	85	1	6	1985	5	45	1	2	1	...	
2	1	119	1	6	1985	4	5	1	1	2	...	
3	1	89	1	4	1985	20	45	1	1	2	...	
4	1	45	1	5	1985	0	45	1	1	2	...	

5 rows × 47 columns

In [14]:
```
# To combine the MONTH, DAY, YEAR columns into a new column called DATE
```

In [15]:
```
myDF["DATE"] = myDF["MONTH"].astype(str) + "/" + myDF["DAY"].astype(str) + "/" + myDF[
```

In [16]:
```
myDF.head()
```

Out[16]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE	...	CF
0	1	53	1	4	1985	8	10	1	1	2	...	
1	1	85	1	6	1985	5	45	1	2	1	...	
2	1	119	1	6	1985	4	5	1	1	2	...	
3	1	89	1	4	1985	20	45	1	1	2	...	
4	1	45	1	5	1985	0	45	1	1	2	...	

5 rows × 48 columns

List what files are in the year 1985

Read in the ACCIDENTS.CSV and then go ahead and change the values in the YEAR column from two digits to four digits. For example, we should change 89 to 1989. Do this by adding a 19 to each year value.

Now combine the MONTH, DAY, YEAR columns into a new column called DATE

Question 2

In [17]:
```
# I want to create a Dataframe called accidents that joins the ACCIDENT.CSV files
# from the years 1985-1989 (inclusive) into one large Dataframe
# To do this, we can (Merge, Join,0r Concat) but i will concatenate them
# I will concatenate them on top of the next one

# https://pandas.pydata.org/docs/reference/api/pandas.concat.html
```

In [18]:
```
accidents=pd.concat([pd.read_csv("/anvil/projects/tdm/data/fars/1985/ACCIDENT.CSV"),
                     pd.read_csv("/anvil/projects/tdm/data/fars/1986/ACCIDENT.CSV"),
                     pd.read_csv("/anvil/projects/tdm/data/fars/1987/ACCIDENT.CSV"),
                     pd.read_csv("/anvil/projects/tdm/data/fars/1988/ACCIDENT.CSV"),
                     pd.read_csv("/anvil/projects/tdm/data/fars/1989/ACCIDENT.CSV")], i
```

In [19]:
```
accidents.head(3)
```

Out[19]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE	...	DI
0	1	53	1	4	85	8	10	1	1	2.0	...	
1	1	85	1	6	85	5	45	1	2	1.0	...	
2	1	119	1	6	85	4	5	1	1	2.0	...	

3 rows × 51 columns

In [20]:
```
accidents.tail(3)
```

Out[20]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE
204592	56	37	12	22	89	18	5	4	6	NaN
204593	56	25	12	28	89	2	40	1	1	NaN
204594	56	33	12	26	89	14	15	1	2	NaN

3 rows × 51 columns

In [21]:
```
# This gives the values each of the years

accidents["YEAR"].value_counts()
```

Out[21]:
```
88    42130
87    41438
86    41090
89    40741
85    39196
Name: YEAR, dtype: int64
```

What we want to do now is create a Dataframe called accidents that joins the ACCIDENT.CSV files from the years 1985-1989 (inclusive) into one large Dataframe.

Question 3

In [22]:
```
# To convert the YEAR column from a 2 digit year to a 4 digit year,
# like we did in the last question, but using a different method.
# we will use the to_datetime function, then take the old format 2digits represented b
# and once you have coverted to a datetime format, then you can display the year with

#  the syntax code   pd.to_datetime(df[''], format='%y').dt.strftime('%Y')
```

In [23]:
```
pd.to_datetime(accidents["YEAR"], format='%y').dt.strftime('%Y')
```

Out[23]:
```
0         1985
1         1985
2         1985
3         1985
4         1985
          ...
204590    1989
204591    1989
204592    1989
204593    1989
204594    1989
Name: YEAR, Length: 204595, dtype: object
```

In [24]:
```
accidents["YEAR"]= pd.to_datetime(accidents["YEAR"], format='%y').dt.strftime('%Y')
```

In [25]:
```
accidents.head(3)
```

Out[25]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE	...	DI
0	1	53	1	4	1985	8	10	1	1	2.0	...	
1	1	85	1	6	1985	5	45	1	2	1.0	...	
2	1	119	1	6	1985	4	5	1	1	2.0	...	

3 rows × 51 columns

In [26]:
```
accidents.tail(3)
```

Out[26]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE
204592	56	37	12	22	1989	18	5	4	6	NaN
204593	56	25	12	28	1989	2	40	1	1	NaN
204594	56	33	12	26	1989	14	15	1	2	NaN

3 rows × 51 columns

In [27]:
```
# To know how many accidents are there in which one or more
# drunk drivers were involved in an accident with a school bus
```

In [28]:
```
accidents["DRUNK_DR"].value_counts()
```

Out[28]:
```
0    120921
1     78737
2      4854
3        80
4         2
6         1
Name: DRUNK_DR, dtype: int64
```

In [29]:
```
accidents["SCH_BUS"].value_counts()
```

Out[29]:
```
0    203956
1       639
Name: SCH_BUS, dtype: int64
```

In [30]:
```
accidents[(accidents["DRUNK_DR"] >= 1) & (accidents["SCH_BUS"]==1)]
```

Out[30]:

	STATE	COUNTY	MONTH	DAY	YEAR	HOUR	MINUTE	VE_FORMS	PERSONS	LAND_USE
1403	4	25	9	25	1985	15	45	2	2	1.0
1588	4	17	10	17	1985	19	35	2	5	2.0
3587	6	37	5	8	1985	16	50	2	2	2.0
6071	6	37	9	18	1985	16	45	2	2	1.0
7638	11	1	8	15	1985	1	50	1	1	1.0
...	...	...	...	...	...	...	...	...	...	...
177919	17	31	11	28	1989	15	0	2	11	NaN
178698	18	141	9	20	1989	7	10	2	2	NaN
182027	24	33	11	27	1989	8	50	2	2	NaN
186730	31	143	3	25	1989	20	17	1	3	NaN
194122	41	47	2	24	1989	23	42	2	2	NaN

79 rows × 51 columns

In [31]:
```
accidents[(accidents["DRUNK_DR"] >= 1) & (accidents["SCH_BUS"]==1)].shape
```

Out[31]:
```
(79, 51)
```

In [32]:
```
# There are 79 rows, i.e, There are 79 accidents with at least 1 drunk driver and a sc
```

Change the values in the YEAR column from a 2 digit year to a 4 digit year, like we did in the last question, but using a different method.

How many accidents are there in which one or more drunk drivers were involved in an accident with a school bus?Markdown notes and sentences and analysis written here.

Question 4

In [33]:
```
# to find the accidents that happen in total per year between 1 or more drunk drivers
# Instead of taken the shape, we can take the DataFrame and extract the year
```

In [34]:
```
accidents[(accidents["DRUNK_DR"] >= 1) & (accidents["SCH_BUS"]==1)]["YEAR"]
```

Out[34]:
```
1403      1985
1588      1985
3587      1985
6071      1985
7638      1985
          ...
177919    1989
178698    1989
182027    1989
186730    1989
194122    1989
Name: YEAR, Length: 79, dtype: object
```

In [35]:
```
accidents[(accidents["DRUNK_DR"] >= 1) & (accidents["SCH_BUS"]==1)]["YEAR"].value_cour
```

Out[35]:
```
1986    22
1987    17
1985    16
1988    16
1989     8
Name: YEAR, dtype: int64
```

In [36]:
```
# 1989 had the lowest number of accidents
# 1986 had the most number of accidents
```

In [37]:
```
# To know the days of the week that has the most accidents occurrence
# one of the days was recorded as 99 to have 31 but we don't know which day it was.

accidents["DAY"].value_counts()
```

Out[37]:
```
1     7120
17    6978
4     6942
22    6874
18    6844
5     6836
23    6835
26    6808
3     6807
24    6802
27    6780
19    6770
2     6754
25    6750
20    6742
7     6720
14    6703
11    6675
28    6620
15    6609
12    6600
16    6590
10    6588
6     6533
13    6532
9     6520
21    6517
8     6512
30    6181
29    6099
31    3923
99      31
Name: DAY, dtype: int64
```

In [38]:
```
# To avoid that, let see how many accident all together

accidents.shape
```

Out[38]:
```
(204595, 51)
```

In [39]:
```
# let us make newaccidents to be the DataFrame
# That got everything from accidents, except the ones which day was equal to 99

newaccidents = accidents [accidents["DAY"] !=99]
```

In [40]:
```
# Looking at the shape of accidents to newaccidents i only lost few values (31)

newaccidents.shape
```

Out[40]:
```
(204564, 51)
```

In [41]:
```
# This will convert the year, month and day into a datetime value

pd.to_datetime(newaccidents[["YEAR", "MONTH", "DAY"]])
```

Out[41]:
```
0        1985-01-04
1        1985-01-06
2        1985-01-06
3        1985-01-04
4        1985-01-05
            ...
204590   1989-12-16
204591   1989-12-20
204592   1989-12-22
204593   1989-12-28
204594   1989-12-26
Length: 204564, dtype: datetime64[ns]
```

In [42]:
```
pd.to_datetime(newaccidents[["YEAR", "MONTH", "DAY"]]).dt.day_name()
```

Out[42]:
```
0            Friday
1            Sunday
2            Sunday
3            Friday
4          Saturday
             ...
204590     Saturday
204591    Wednesday
204592       Friday
204593     Thursday
204594      Tuesday
Length: 204564, dtype: object
```

In [43]:
```
pd.to_datetime(newaccidents[["YEAR", "MONTH", "DAY"]]).dt.day_name().value_counts()
```

Out[43]:
```
Saturday     40671
Friday       33640
Sunday       32397
Thursday     26331
Wednesday    24167
Monday       24071
Tuesday      23287
dtype: int64
```

In [44]:
```
# looking at the accidents, tons of accidents happened on Saturdays, Fridays and Sunda
# Most accidents occur during the weekend.
```

In [45]:
```
# what time of day do you see more accidents? Using 12am-6am/ 6am-12pm/ 12pm-6pm/ 6pm-
```

In [46]:
```
# First let us take look at the hour
# sometimes the hour is 99

accidents["HOUR"].value_counts()
```

Out[46]:
```
18    11537
23    11285
17    11258
21    11121
19    11115
22    10875
20    10813
1     10758
16    10615
0     10517
15    10423
2     10217
14     8760
13     7545
12     6804
3      6379
11     6376
7      5905
6      5824
10     5608
9      5052
8      5050
4      4489
5      4435
99     1637
24      197
Name: HOUR, dtype: int64
```

In [48]:
```
# Same thing for the minutes if all were displayed
# sometimes the hour is 99
accidents["MINUTE"].value_counts()
```

Out[48]:
```
0     19618
30    19445
15    13998
45    13969
50    11433
      ...
41     1013
11     1007
51      992
21      979
31      931
Name: MINUTE, Length: 61, dtype: int64
```

In [49]:
```
# looking at the minutes values < 0, none of them are negative

accidents[accidents["MINUTE"]<0].shape
```

Out[49]:
```
(0, 51)
```

In [50]:
```
# There are rows that there minutes are bigger than 60

accidents[accidents["MINUTE"]>60].shape
```

Out[50]:
```
(1677, 51)
```

In [52]:
```
# But fortunately, those are just the rows for  which is 99
# Those are the ones that are unknown
```

```
accidents[accidents["MINUTE"]==99].shape
```

Out[52]:
```
(1677, 51)
```

In [53]:
```
# i will make a new DataFrame, takes data from accidents and
# ensure minutes not equal to 99 and ensure the hours also not equal to 99

newDF=accidents[(accidents["MINUTE"]!=99) & (accidents["HOUR"]!=99)]
```

In [55]:
```
# Looking at the shape of accident to newDF i only lost few values (1677)

accidents.shape
```

Out[55]:
```
(204595, 51)
```

In [56]:
```
newDF.shape
```

Out[56]:
```
(202918, 51)
```

In [57]:
```
# go into my newDF, take the hour and multiply by 60 to convert to minutes
# then add on the amount of minutes. Then output will be in minutes

pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"]).between(0,360, inclusive="left")
```

Out[57]:
```
0         False
1          True
2          True
3         False
4          True
          ...
204590    False
204591    False
204592    False
204593     True
204594    False
Length: 202918, dtype: bool
```

In [59]:
```
# all these are between 12 midnight and 6:00 am
# meaning from Zero up to 360 in the day

pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"]).between(0,360, inclusive="left").sum()
```

Out[59]:
```
46772
```

In [60]:
```
# Those are the ones between 6:00 am and 12:00 noon

pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"]).between(360,720, inclusive="left").sum()
```

Out[60]:
```
33815
```

In [61]:
```
# Those are the ones between 12:00 noon and 6:00 pm

pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"]).between(720, 1080, inclusive="left").sum
```

Out[61]:
```
55401
```

In [62]:
```
# Those are the ones between 6:00 pm and 12 midnight

pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"]).between(1080, 1440, inclusive="left").su
```

Out[62]:
```
66733
```

In [63]:
```
# for verification
# We are almost to the value in the newDF

46772+33815+55401+66733
```

Out[63]:
```
202721
```

In [64]:
```
# Those missing if you go all the way back are actually the ones for which the hour wa
# In other words, for which the hour values was not recorded as Zero at midnight,
# but rather was recorded as it is 24 at midnight

202918-202721
```

Out[64]:
```
197
```

In [65]:
```
# This sum to what was in the newDF

46772+33815+55401+66733+197
```

Out[65]:
```
202918
```

In [ ]:
```
# Here is anothe way to solving it
```

In [71]:
```
import numpy as np
```

In [70]:
```
# considering the earlier coversion divide it by 360 and basically rounded down

pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"])/360
```

Out[70]:
```
0         1.361111
1         0.958333
2         0.680556
3         3.458333
4         0.125000
             ...
204590    1.333333
204591    1.291667
204592    3.013889
204593    0.444444
204594    2.375000
Length: 202918, dtype: float64
```

In [72]:
```
# basically rounded down here to 0,1,2,3.......

(pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"])/360).apply(np.floor)
```

Out[72]:
```
0         1.0
1         0.0
2         0.0
3         3.0
4         0.0
         ...
204590    1.0
204591    1.0
204592    3.0
204593    0.0
204594    2.0
Length: 202918, dtype: float64
```

In [73]:
```
(pd.Series(newDF["HOUR"]*60 + newDF["MINUTE"])/360).apply(np.floor).value_counts()
```

Out[73]:
```
3.0    66733
2.0    55401
0.0    46772
1.0    33815
4.0      197
dtype: int64
```

In [ ]:
```
# 0.0    46772  between 12 midnight and 6:00 am
# 1.0    33815  between 6:00 am and 12:00 noon
# 2.0    55401  between 12:00 noon and 6:00 pm
# 3.0    66733  between 6:00 pm and 12 midnight
# 4.0      197  The extra 197 accidents are from those recorded at midnight as 24 inst
```

Find how many accidents happen in total per year between 1 or more drunk drivers and school bus.

what year had the lowest number of accidents

what year had the most number of accidents

Now we want to consider which days of the week had the most accidents occur

Is there a time of day where you see more accidents? Using 12am-6am/ 6am-12pm/ 12pm-6pm/ 6pm-12am as your time frames.

Question 5

In [47]:
```
# code here
```

Markdown notes and sentences and analysis written here.

Pledge

By submitting this work I hereby pledge that this is my own, personal work. I've acknowledged in the designated place at the top of this file all sources that I used to complete said work, including but not limited to: online resources, books, and electronic communications. I've noted all collaboration with fellow students and/or TA's. I did not copy or plagiarize another's work.

> As a Boilermaker pursuing academic excellence, I pledge to be honest and true in all that I do. Accountable together – We are Purdue.