Project 5 -- SEYI OGUNMODEDE

TA Help: Lucas and Jeong Ik Su

- Help with figuring out how to write a function.

Question 1

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

In [2]:
```
cars = pd.read_csv("/anvil/projects/tdm/data/craigslist/vehicles.csv")
```

In [3]:
```
cars.shape
```

Out[3]:
```
(435849, 25)
```

In [4]:
```
pd.options.display.max_columns = None
```

In [5]:
```
cars.head()
```

Out[5]:

	id	url	region	region_url
0	7119256118	https://mohave.craigslist.org/ctd/d/lake-havas...	mohave county	https://mohave.craigslist.org
1	7120880186	https://oregoncoast.craigslist.org/cto/d/warre...	oregon coast	https://oregoncoast.craigslist.org
2	7115048251	https://greenville.craigslist.org/cto/d/sparta...	greenville / upstate	https://greenville.craigslist.org
3	7119250502	https://mohave.craigslist.org/cto/d/lake-havas...	mohave county	https://mohave.craigslist.org
4	7120433904	https://maine.craigslist.org/ctd/d/searsport-t...	maine	https://maine.craigslist.org

In [6]:
```
mycarcount= 'craigslist/vehicles'
```

In [7]:
```
myyear = 2016
```

In [8]:
```
total_count = 0
```

In [9]:
```
for mydf in pd.read_csv(f'/anvil/projects/tdm/data/{mycarcount}.csv', chunksize=10000)
    for index, row in mydf.iterrows ():
        if row ['year'] == myyear:
            total_count +=1
```

In [10]:
```
total_count
```

Out[10]:
```
32096
```

In [11]:
```
mycarcounts= 'vehicles'
```

In [12]:
```
myyears = 2016
```

In [13]:
```
total_count = 0
```

In [14]:
```
for mydf in pd.read_csv(f'/anvil/projects/tdm/data/{mycarcount}.csv', chunksize=10000)
    for index, row in mydf.iterrows ():
        if row ['year'] == myyears:
            total_count +=1
```

In [15]:
```
total_count
```

Out[15]:
```
32096
```

In [38]:
```
import pandas as pd
```

In [39]:
```
cars = pd.read_csv("/anvil/projects/tdm/data/craigslist/vehicles.csv")
```

In [40]:
```
def mycarcount(cars, year):
    """
    mycarcount is a function that accept cars and year as argument
    and returns the number of cars that occur on year for cars.

    Args:
    cars (df): The dataframe from which we are counting the number of cars.
    year (int): The years on which we are counting the number of vehicles.

    Returns:
        The numbers of cars on my dataframe during the year
    """
    # you are telling the python to run a row at atime and compare the values within t

    total_count = 0
    for index, row in cars.iterrows ():
        if row ['year'] ==year:
            total_count +=1
    return total_count
```

In [19]:
```
mycarcount(cars, 2016)
```

Out[19]:
```
32096
```

In [41]:
```
mycarcount(cars, 2011)
```

Out[41]:
```
26532
```

In [42]:
```
mycarcount(cars, 1989)
```

Out[42]:
```
630
```

In [43]:
```
mycarcount(cars, 1997)
```

Out[43]:
```
2062
```

In [20]:
```
mycarcount= 'craigslist/vehicles'
```

In [21]:
```
myyear = 2011
```

In [22]:
```
total_count = 0
```

In [23]:
```
for mydf in pd.read_csv(f'/anvil/projects/tdm/data/{mycarcount}.csv', chunksize=10000)
    for index, row in mydf.iterrows ():
        if row ['year'] == myyear:
            total_count +=1
```

In [24]:
```
total_count
```

Out[24]:
```
26532
```

In [26]:
```
# To write a function just named it anything eg cc

def cc(carsdf, myyear):
```

```
    mycount = len(carsdf[carsdf.year == myyear])
    return mycount
```

In [27]:
```
cc(cars, 2011)
```

Out[27]:
```
26532
```

In [28]:
```
cc(cars, 1989)
```

Out[28]:
```
630
```

In [29]:
```
cc(cars, 1997)
```

Out[29]:
```
2062
```

In [30]:
```
def mycarcount(year):
    """
    mycarcount is a function that accept cars and year as argument
    and returns the number of cars that occur on year for cars.

    Args:
    cars (df): The dataframe from which we are counting the number of cars.
    year (int): The years on which we are counting the number of vehicles.

    Returns:
        The numbers of cars on my dataframe during the year
    """
    # you are telling the python to run a row at atime and compare the values within t

    cars = pd.read_csv("/anvil/projects/tdm/data/craigslist/vehicles.csv")
    total_count = 0
    for index, row in cars.iterrows ():
        if row ['year'] ==year:
            total_count +=1
    return total_count
```

In [31]:
```
 mycarcount(2016)
```

Out[31]:
```
32096
```

In [32]:
```
mycarcount(2011)
```

Out[32]:
```
26532
```

In [33]:
```
mycarcount(1989)
```

Out[33]:
```
630
```

In [34]:
```
mycarcount(1997)
```

Out[34]:
```
2062
```

a. Write a function called mycarcount that takes two parameters: cars as a data frame, and year as an integer, and outputs the number of cars from that year. (Alternatively, you can just use 1

argument, the year, as a parameter, and then read through the cars data frame inside the function. Either way is OK.)

b. Run the function for each of the years from Project 4, Question 4, namely, for the years 2011, 1989, 1997. Make sure that your answers agree with the results from that earlier project.

Question 2

In [35]:
```
# to generate the unique years in the data

cars.year.unique()
```

Out[35]:
```
array([2012., 2014., 2001., 2004., 2021., 2010., 2005., 2017., 2016.,
       2006., 2009., 2011., 2018., 1999., 2019., 1998., 2015., 2007.,
       2000., 1988., 2008., 2013., 1965., 1997., 2002., 1976., 1996.,
       1980., 1944., 2020., 2003., 1978., 1940., 1952., 1989.,   nan,
       1986., 1991., 1995., 1967., 1993., 1966., 1974., 1987., 1957.,
       1994., 1970., 1981., 1979., 1960., 1975., 1990., 1977., 1984.,
       1956., 1982., 1985., 1992., 1983., 1973., 1963., 1972., 1968.,
       1931., 1958., 1961., 1962., 1964., 1938., 1949., 1942., 1955.,
       1969., 1939., 1971., 1948., 1954., 1934., 1930., 1950., 1919.,
       1953., 1951., 1941., 1937., 1947., 1927., 1959., 1946., 1913.,
       1933., 1929., 1936., 1923., 1925., 1932., 1900., 1928., 1935.,
       1916., 1912., 1911., 1926., 1917., 1922., 1915., 1924., 1943.,
       1945., 1920.])
```

In [36]:
```
# To make it a variable
yrlist =cars.year.unique()
```

In [37]:
```
# To sort them out, rearranged
yrlist.sort()
```

In [38]:
```
# To generate the values after sorting
yrlist
```

Out[38]:
```
array([1900., 1911., 1912., 1913., 1915., 1916., 1917., 1919., 1920.,
       1922., 1923., 1924., 1925., 1926., 1927., 1928., 1929., 1930.,
       1931., 1932., 1933., 1934., 1935., 1936., 1937., 1938., 1939.,
       1940., 1941., 1942., 1943., 1944., 1945., 1946., 1947., 1948.,
       1949., 1950., 1951., 1952., 1953., 1954., 1955., 1956., 1957.,
       1958., 1959., 1960., 1961., 1962., 1963., 1964., 1965., 1966.,
       1967., 1968., 1969., 1970., 1971., 1972., 1973., 1974., 1975.,
       1976., 1977., 1978., 1979., 1980., 1981., 1982., 1983., 1984.,
       1985., 1986., 1987., 1988., 1989., 1990., 1991., 1992., 1993.,
       1994., 1995., 1996., 1997., 1998., 1999., 2000., 2001., 2002.,
       2003., 2004., 2005., 2006., 2007., 2008., 2009., 2010., 2011.,
       2012., 2013., 2014., 2015., 2016., 2017., 2018., 2019., 2020.,
       2021.,   nan])
```

In [39]:
```
# To eliminate the nan
ylist = yrlist[:-1]
```

In [40]:
```
# The years after removing nan
ylist
```

Out[40]:
```
array([1900., 1911., 1912., 1913., 1915., 1916., 1917., 1919., 1920.,
       1922., 1923., 1924., 1925., 1926., 1927., 1928., 1929., 1930.,
       1931., 1932., 1933., 1934., 1935., 1936., 1937., 1938., 1939.,
       1940., 1941., 1942., 1943., 1944., 1945., 1946., 1947., 1948.,
       1949., 1950., 1951., 1952., 1953., 1954., 1955., 1956., 1957.,
       1958., 1959., 1960., 1961., 1962., 1963., 1964., 1965., 1966.,
       1967., 1968., 1969., 1970., 1971., 1972., 1973., 1974., 1975.,
       1976., 1977., 1978., 1979., 1980., 1981., 1982., 1983., 1984.,
       1985., 1986., 1987., 1988., 1989., 1990., 1991., 1992., 1993.,
       1994., 1995., 1996., 1997., 1998., 1999., 2000., 2001., 2002.,
       2003., 2004., 2005., 2006., 2007., 2008., 2009., 2010., 2011.,
       2012., 2013., 2014., 2015., 2016., 2017., 2018., 2019., 2020.,
       2021.])
```

In [41]:
```
# i used for loop to generate my output

for i in yrlist:
    a = cc(cars,i)
    print (i, a)
```

```
1900.0 58
1911.0 1
1912.0 5
1913.0 1
1915.0 1
1916.0 3
1917.0 1
1919.0 2
1920.0 1
1922.0 4
1923.0 37
1924.0 8
1925.0 9
1926.0 11
1927.0 34
1928.0 27
1929.0 69
1930.0 53
1931.0 60
1932.0 39
1933.0 17
1934.0 44
1935.0 18
1936.0 50
1937.0 72
1938.0 22
1939.0 60
1940.0 94
1941.0 60
1942.0 22
1943.0 1
1944.0 3
1945.0 1
1946.0 79
1947.0 63
1948.0 101
1949.0 113
1950.0 120
1951.0 113
1952.0 88
1953.0 93
1954.0 94
1955.0 216
1956.0 188
1957.0 218
1958.0 88
1959.0 100
1960.0 103
1961.0 87
1962.0 170
1963.0 214
1964.0 299
1965.0 376
1966.0 490
1967.0 427
1968.0 415
1969.0 464
1970.0 412
1971.0 383
1972.0 427
```

```
1973.0 392
1974.0 305
1975.0 219
1976.0 266
1977.0 313
1978.0 408
1979.0 468
1980.0 263
1981.0 219
1982.0 185
1983.0 243
1984.0 430
1985.0 477
1986.0 499
1987.0 635
1988.0 539
1989.0 630
1990.0 646
1991.0 649
1992.0 762
1993.0 825
1994.0 1113
1995.0 1433
1996.0 1529
1997.0 2062
1998.0 2351
1999.0 3742
2000.0 4778
2001.0 5742
2002.0 7153
2003.0 9235
2004.0 12037
2005.0 14209
2006.0 17191
2007.0 20457
2008.0 22643
2009.0 15899
2010.0 19956
2011.0 26532
2012.0 29108
2013.0 31434
2014.0 31703
2015.0 32918
2016.0 32096
2017.0 34592
2018.0 20147
2019.0 15531
2020.0 2820
2021.0 119
nan 0
```

a. Run the function mycarcount for each year in the data set. (Of course, be sure to only run it once for each year!)

b. Now make sure that the results agree, if you compare with the value_counts() from the year column.

Question 3

In [8]:
```
import pandas as pd
from pathlib import Path
```

In [9]:
```
def getflight(myorigin, yr):

# f' means for the most path we have regular string, instead for the curly bracket eve
# year is an integer. they dont play well together.
    total_count=0
    airport = pd.read_csv(f'/anvil/projects/tdm/data/flights/subset/{yr}.csv')
    for index, row in airport.iterrows ():
        if row ['Origin'] ==myorigin:
            total_count +=1
    return total_count
```

In [3]:
```
# this is specified to csv file of year 2016
f'/anvil/projects/tdm/data/flights/subset/{2016}.csv'
```

Out[3]:
```
'/anvil/projects/tdm/data/flights/subset/2016.csv'
```

In [4]:
```
# To lnow the different Origin available
# You will consider for a particular year (1989)
# Then look for the uniqueness

airport = pd.read_csv(f'/anvil/projects/tdm/data/flights/subset/1989.csv')
```

In [5]:
```
airport.Origin.unique()
```

Out[5]:
```
array(['SFO', 'DEN', 'HNL', 'LIH', 'PHL', 'OGG', 'IAD', 'EWR', 'LAX',
       'KOA', 'ORD', 'MKE', 'IAH', 'MSY', 'RIC', 'SEA', 'FLL', 'MCO',
       'BWI', 'HOU', 'MCI', 'SJC', 'CMH', 'OMA', 'ORF', 'BOS', 'ABQ',
       'SMF', 'OKC', 'SGF', 'SLC', 'ONT', 'SAN', 'BUF', 'LGB', 'MIA',
       'BDL', 'IND', 'TPA', 'SYR', 'PIT', 'STL', 'PDX', 'CLT', 'PHX',
       'CVG', 'MBS', 'ATL', 'DSM', 'LAS', 'AUS', 'JAN', 'MEM', 'DTW',
       'SRQ', 'MSP', 'DFW', 'PSP', 'MDT', 'FAT', 'BGR', 'PWM', 'ROC',
       'MDW', 'FAR', 'CHS', 'SAV', 'LGA', 'ANC', 'FSD', 'OAK', 'CLE',
       'CAE', 'TUL', 'HPN', 'GSO', 'RDU', 'DAY', 'GRR', 'MSN', 'JAX',
       'BNA', 'COS', 'SAT', 'CID', 'PBI', 'GEG', 'LNK', 'PVD', 'BIL',
       'SBA', 'ELP', 'SDF', 'TUS', 'SNA', 'ICT', 'BUR', 'ABE', 'RAP',
       'GTF', 'LIT', 'ALB', 'RNO', 'BHM', 'HSV', 'BOI', 'DCA', 'SUX',
       'TYS', 'MHT', 'EUG', 'MLI', 'BTV', 'PIA', 'FAI', 'MFR', 'MRY',
       'JFK', 'RSW', 'ERI', 'PHF', 'PSC', 'ISP', 'AVP', 'EVV', 'ELM',
       'LEX', 'ORH', 'BGM', 'TRI', 'CRW', 'TOL', 'MYR', 'ITH', 'GSP',
       'HTS', 'ACY', 'BLI', 'SCK', 'RDM', 'CCR', 'YKM', 'DAL', 'LBB',
       'CRP', 'AMA', 'HRL', 'MAF', 'DET', 'VPS', 'AZO', 'FWA', 'DLH',
       'ATW', 'GRB', 'BIS', 'GPT', 'LAN', 'GFK', 'RST', 'BZN', 'EAU',
       'LSE', 'MOT', 'MOB', 'MSO', 'BTR', 'CHA', 'SHV', 'MGM', 'PFN',
       'HDN', 'SBN', 'STT', 'STX', 'FAY', 'AVL', 'CAK', 'OAJ', 'TLH',
       'ROA', 'ILM', 'AGS', 'DAB', 'FNT', 'ISO', 'CHO', 'CMI', 'LYH',
       'UCA', 'PNS', 'EYW', 'APF', 'GNV', 'SJU', 'PUB', 'MLU', 'MLB',
       'CSG', 'CPR', 'IDA', 'JAC', 'HLN', 'FCA', 'JNU', 'BTM', 'DRO',
       'GJT', 'YUM', 'FLG', 'GCN', 'PIE', 'TVL', 'BFL', 'GUC', 'BET',
       'OME', 'OTZ', 'SCC', 'KTN', 'CDV', 'YAK', 'SIT', 'PSG', 'WRG',
       'GUM', 'MFE', 'LFT', 'YAP', 'ROR', 'SPN', 'ROP', 'TVC', 'ABI',
       'GST', 'EGE', 'SUN'], dtype=object)
```

In [6]:
```
getflight("ORD", 1989)
```

Out[6]:
```
261259
```

In [23]:
```
getflight("BTR", 1989)
```

Out[23]:
```
8684
```

In [24]:
```
getflight("PUB", 1999)
```

Out[24]:
```
0
```

In [25]:
```
getflight("ATL", 2004)
```

Out[25]:
```
418072
```

In [26]:
```
getflight("DFW", 1998)
```

Out[26]:
```
244013
```

In [7]:
```
for i in range(1987,2009):
    print(f"{i}:{getflight(f'IND', i)}")
```

```
1987:8817
1988:37399
1989:40567
1990:43826
1991:42890
1992:43620
1993:37684
1994:38612
1995:37092
1996:34177
1997:35318
1998:33810
1999:34471
2000:35261
2001:37871
2002:32599
/tmp/ipykernel_16/91472592.py:6: DtypeWarning: Columns (22) have mixed types. Specify
dtype option on import or set low_memory=False.
  airport = pd.read_csv(f'/anvil/projects/tdm/data/flights/subset/{yr}.csv')
2003:41617
2004:42098
2005:43174
2006:37615
2007:43576
2008:14402
```

In [8]:
```
def getflight(myorigin, yr):
    file = Path(f'/anvil/projects/tdm/data/flights/subset/{yr}.csv')
    count = 0
    with open(file, 'r') as f: # file destination, r decides what you are using it for
        for line in f:
            if line.split(",")[16] == myorigin: # we spiliting into a list, removing t
```

```
                count += 1
    return count
```

In [9]:
```
getflight('IND',2004)
```

Out[9]:
```
42098
```

In [10]:
```
getflight('ATL',2004)
```

Out[10]:
```
418072
```

In [11]:
```
getflight('DFW',1998)
```

Out[11]:
```
244013
```

In [12]:
```
getflight('IND',1997)
```

Out[12]:
```
35318
```

a Write a function that takes two parameters: myorigin as a string with three characters, and year as an integer, and outputs the number of flights that depart during that year, from the Origin airport indicated in myorigin.

b. Test your function for a few years and airports of your choice. You can choose! Do your results look reasonable, i.e., do the airports in the big cities have lots of flights, compared to airports in smaller cities?

c. Run the function for each of the years from 1987 to 2008, checking how many flights depart from IND in each year. Make sure that you use the method from the end of Project 3, Question 5.

Question 4

In [10]:
```
def getflight(myorigin,mydest, yr):

# f' means for the most path we have regular string, instead for the curly bracket eve
# year is an integer. they dont play well together.
    total_count=0
    airport = pd.read_csv(f'/anvil/projects/tdm/data/flights/subset/{yr}.csv')
    for index, row in airport.iterrows ():
        if (row ['Origin'] ==myorigin) and (row ['Dest'] ==mydest):
            total_count +=1
    return total_count
```

In [45]:
```
getflight('IND','ORD', 1987)
```

Out[45]:
```
935
```

In [11]:
```
getflight('IND','MDW', 1997)
```

Out[11]:
```
1786
```

In [12]:
```
getflight('SLC','GNV',2000)
```

Out[12]:
```
0
```

In [13]:
```
getflight('GNV','ORD', 1997)
```

Out[13]:
```
0
```

In [14]:
```
for k in range(1987,2009):
    print(f"{k}:{getflight(f'IND','ORD', k)}")
```

```
1987:935
1988:3859
1989:3359
1990:3401
1991:3884
1992:4207
1993:4151
1994:3484
1995:2715
1996:2376
1997:2473
1998:2583
1999:2802
2000:2657
2001:5741
2002:5719
/tmp/ipykernel_16/882744575.py:6: DtypeWarning: Columns (22) have mixed types. Specif
y dtype option on import or set low_memory=False.
  airport = pd.read_csv(f'/anvil/projects/tdm/data/flights/subset/{yr}.csv')
2003:5825
2004:5083
2005:3990
2006:3281
2007:3871
2008:1324
```

a. Modify your function so that it takes three parameters: myorigin and mydest as strings that each have three characters, and year as an integer, and outputs the number of flights that depart during that year, from the Origin airport indicated in myorigin, and arrive at the Dest airport indicated in mydest.

b. Test your function for a few years and pairs of airports (origin and destination airports) of your choice. Do the results look reasonable, e.g., if you compare popular flight paths, versus unpopular flight paths?

c. Run the function for each of the years from 1987 to 2008, checking how many flights depart from IND and arrive at ORD in each year.

Question 5

In [15]:
```
def records(sex,maritalstatus):

# f' means for the most path we have regular string, instead for the curly bracket eve
# year is an integer. they dont play well together.
    total_count=0
    people = pd.read_csv(f'/anvil/projects/tdm/data/death_records/DeathRecords.csv')
    for index, row in people.iterrows ():
        if (row ['Sex'] ==sex) and (row ['MaritalStatus'] ==maritalstatus):
            total_count +=1
    return total_count
```

In [16]:
```
pd.read_csv(f'/anvil/projects/tdm/data/death_records/DeathRecords.csv')
```

Out[16]:

	Id	ResidentStatus	Education1989Revision	Education2003Revision	EducationReporting
0	1	1	0	2	
1	2	1	0	2	
2	3	1	0	7	
3	4	1	0	6	
4	5	1	0	3	
...	...	...	...	...	...
2631166	2631167	3	0	6	
2631167	2631168	3	0	9	
2631168	2631169	3	0	1	
2631169	2631170	4	0	9	
2631170	2631171	3	0	1	

2631171 rows × 38 columns

In [17]:
```
people = pd.read_csv(f'/anvil/projects/tdm/data/death_records/DeathRecords.csv')
```

In [34]:
```
records("F","D")
```

Out[34]:
```
187807
```

In [35]:
```
records("M","D")
```

Out[35]:
```
213152
```

In [36]:
```
records("F","M")
```

Out[36]:
```
324347
```

In [37]:
```
records("M","S")
```

Out[37]:
```
206175
```

a. Write a function that takes two parameters: Sex (which will be F or M) and MaritalStatus (D or M or S or U or W), and outputs the number of people with the indicated Sex and MaritalStatus in the data set. (If you look at an earlier version of this question, in which we asked about the year of death, well, everyone in the data set died in 2014, so you do not need to worry about the year of death.)

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.