Project 5 Solutions

Kevin Choe

Collaborators: (Collaborators listed here. Include names, which part of the project you gave or sought help with, and how you helped or were helped.)

TA help: Summeth Guda

Online resources used: (List of links/resources (if any) here. Include web addresses, which part of the project the resource helped with, and how you were helped.)

Question 1

```python
from pathlib import Path
p = Path("/class/datamine/data/stackoverflow/unprocessed/2018.csv")
size_in_csv = p.stat().st_size
size_in_csv
```

```
195595827
```

```python
print(f'Size in bytes: {size_in_csv}')
```

```
Size in bytes: 195595827
```

```python
from pathlib import Path
p = Path("/class/datamine/data/stackoverflow/unprocessed/2018.parquet")
size_in_parquet = p.stat().st_size
size_in_parquet
```

```
8775374
```

```python
print(f'Size in bytes: {size_in_parquet}')
```

```
Size in bytes: 8775374
```

```python
from pathlib import Path
p = Path("/class/datamine/data/stackoverflow/unprocessed/2018.feather")
size_in_feather = p.stat().st_size
size_in_feather
```

```
54140466
```

```python
print(f'Size in bytes: {size_in_feather}')
```

```
Size in bytes: 54140466
```

```python
print(f'The parquet file is smaller than the csv by {(size_in_csv-size_in_parquet)/size_in_parquet:.2%}
```

```
The parquet file is smaller than the csv by 2128.92%
```

```python
print(f'The feather file is smaller than csv by {(size_in_csv-size_in_feather)/size_in_feather:.2%}')
```

```
The feather file is smaller than csv by 261.27%
```

```python
from block_timer.timer import Timer
import pandas as pd
with Timer(title="csv") as csv:
  myDF = pd.read_csv("/class/datamine/data/stackoverflow/unprocessed/2018.csv")
```

```
sys:1: DtypeWarning: Columns
(8,12,13,14,15,16,50,51,52,53,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,7
have mixed types.Specify dtype option on import or set low_memory=False.
[csv] Total time 2.13789 seconds.
```

```python
with Timer(title="parquet") as parquet:
  myDF1 = pd.read_parquet("/class/datamine/data/stackoverflow/unprocessed/2018.parquet")
```

```
[parquet] Total time 0.86203 seconds.
```

```python
with Timer(title="feather") as feather:
  myDF1 = pd.read_feather("/class/datamine/data/stackoverflow/unprocessed/2018.feather")
```

```
[feather] Total time 0.46717 seconds.
```

```python
print(f'The parquet file is faster than the csv by {(csv.elapsed-parquet.elapsed)/parquet.elapsed:.2%}'
```

```
The parquet file is faster than the csv by 148.01%
```

```python
print(f'The feather file is faster than the csv by {(csv.elapsed-feather.elapsed)/feather.elapsed:.2%}'
```

```
The feather file is faster than the csv by 357.62%
```

```python
("/class/datamine/data/stackoverflow/unprocessed/2018.csv")
```

```
'/class/datamine/data/stackoverflow/unprocessed/2018.csv'
```

```python
with Timer(title="csv") as csv:
  myDF.to_csv("/scratch/scholar/choe29/2018.csv")
```

```
[csv] Total time 7.24383 seconds.
```

```python
with Timer(title="parquet") as parquet:
  myDF.to_parquet("/scratch/scholar/choe29/2018.parquet")
```

```
[parquet] Total time 1.07168 seconds.
```

```python
with Timer(title="feather") as feather:
  myDF.to_feather(("/scratch/scholar/choe29/2018.feather"))
```

```
[feather] Total time 0.79518 seconds.
```

```python
print(f'The parquet file is faster than the csv by {(csv.elapsed-parquet.elapsed)/parquet.elapsed:.2%}'
```

```
The parquet file is faster than the csv by 575.93%
```

```python
print(f'The feather file is faster than the csv by {(csv.elapsed-feather.elapsed)/feather.elapsed:.2%}'
```

```
The feather file is faster than the csv by 810.97%
```

Question 2

```python
import pandas as pd
```

```
myDF = pd.read_csv("/class/datamine/data/stackoverflow/unprocessed/2018.csv")
```

```
sys:1: DtypeWarning: Columns
(8,12,13,14,15,16,50,51,52,53,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,7
have mixed types.Specify dtype option on import or set low_memory=False.
```

```
not_studentsDF = myDF.loc[myDF.loc[:,"Student"] == 'No', :]
percentage = len(not_studentsDF.loc[:,"Respondent"])/len(myDF.loc[:,"Respondent"])

print(f'{percentage*100}%')
```

```
71.2144049365232%
```

Question 3

```
professions = [p.split(";") for p in not_studentsDF.loc[:, "DevType"].dropna().tolist()]

professions = [p for li in professions for p in li]
professions = list(set(professions))
print(professions)
```

```
['Data or business analyst', 'Front-end developer', 'Desktop or enterprise
applications developer', 'DevOps specialist', 'Database administrator',
'Engineering manager', 'Mobile developer', 'Back-end developer', 'System
administrator', 'C-suite executive (CEO, CTO, etc.)', 'Game or graphics
developer', 'Designer', 'Product manager', 'QA or test developer', 'Data
scientist or machine learning specialist', 'Student', 'Embedded applications or
devices developer', 'Marketing or sales professional', 'Full-stack developer',
'Educator or academic researcher']
```

```
print(len(professions))
```

```
20
```

```
studentsDF = myDF.loc[(myDF.loc[:,"Student"] == 'No') & (myDF.loc[:,"DevType"].str.contains("Student"))
len(studentsDF)

#There are 20 professions. There are 3723 number of respondents that replied "No" to Student, yet put "
```

```
3723
```

Question 4

```
import matplotlib.pyplot as plt
import pandas as pd
import random
print(f"A random integer between 1 and 100 is {random.randint(1, 101)}")
```

```
A random integer between 1 and 100 is 87
```

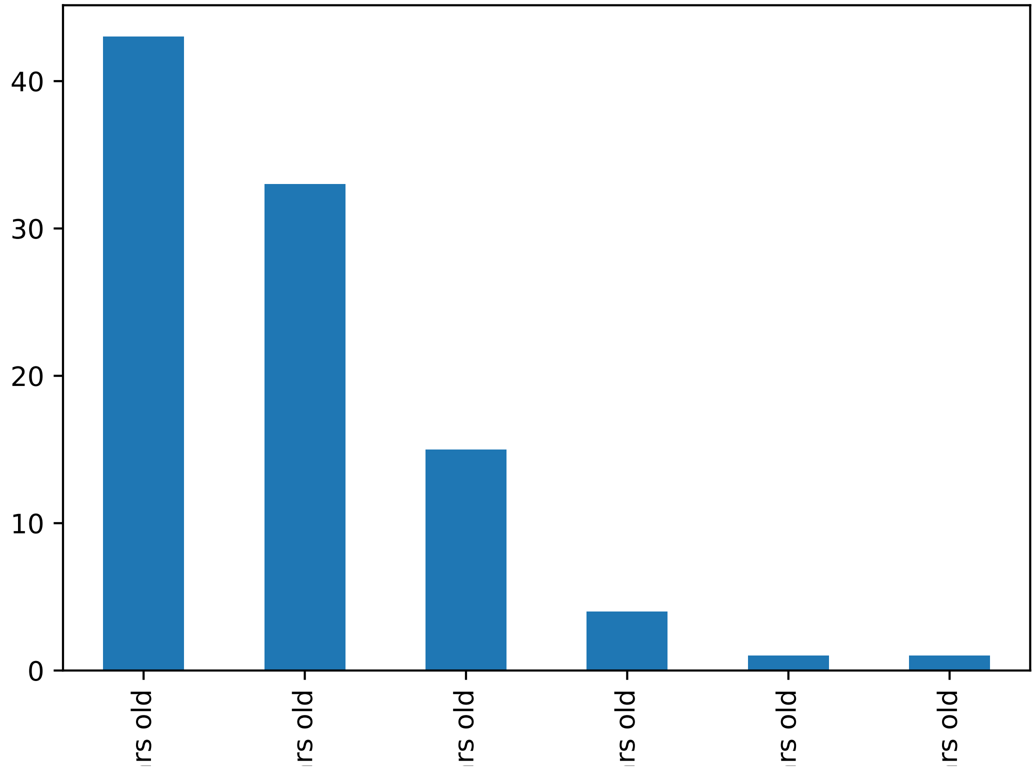
```
females = myDF.loc[(myDF.loc[:, "Gender"]=="Female"), :]
femaleage=[]
femaleage = [random.randint(0, len(females)) for i in range(0,100)]
females = females.iloc[femaleage]
print(femaleage)
```

```
[2419, 2016, 3728, 3203, 3070, 989, 3495, 284, 3414, 1401, 1377, 2838, 3800,
3718, 121, 3064, 2481, 3830, 2881, 1720, 2692, 402, 45, 826, 2621, 3908, 358,
70, 3233, 1469, 2609, 260, 427, 1985, 2111, 3523, 2000, 1670, 1580, 2477, 1151,
507, 3408, 2594, 31, 3728, 1162, 1463, 1570, 1051, 556, 522, 1077, 1076, 3980,
2326, 3174, 1684, 942, 3141, 2212, 2136, 1065, 3518, 968, 776, 621, 3682, 1801,
1414, 1890, 701, 1558, 3262, 3422, 1099, 1749, 1405, 102, 3451, 1496, 1248,
3589, 1178, 3627, 2754, 3922, 2004, 3941, 2190, 1219, 3001, 2141, 352, 1174,
1769, 300, 620, 1005, 1639]
```

```
females.loc[:,"Age"].value_counts().plot.bar()
print(females)
```

```
       Respondent  ...                  SurveyEasy
54412       77161  ...  Neither easy nor difficult
45329       64296  ...                   Very easy
85411       74558  ...  Neither easy nor difficult
72635       12543  ...               Somewhat easy
69483       98643  ...                   Very easy
...           ...  ...                         ...
40019       56837  ...               Somewhat easy
6816         9732  ...  Neither easy nor difficult
14195       20203  ...          Somewhat difficult
22799       32388  ...  Neither easy nor difficult
37034       52625  ...               Somewhat easy

[100 rows x 129 columns]
```

```
plt.show()
```



```
import random
print(f"A random integer between 1 and 100 is {random.randint(1, 101)}")
```

```
A random integer between 1 and 100 is 84
```

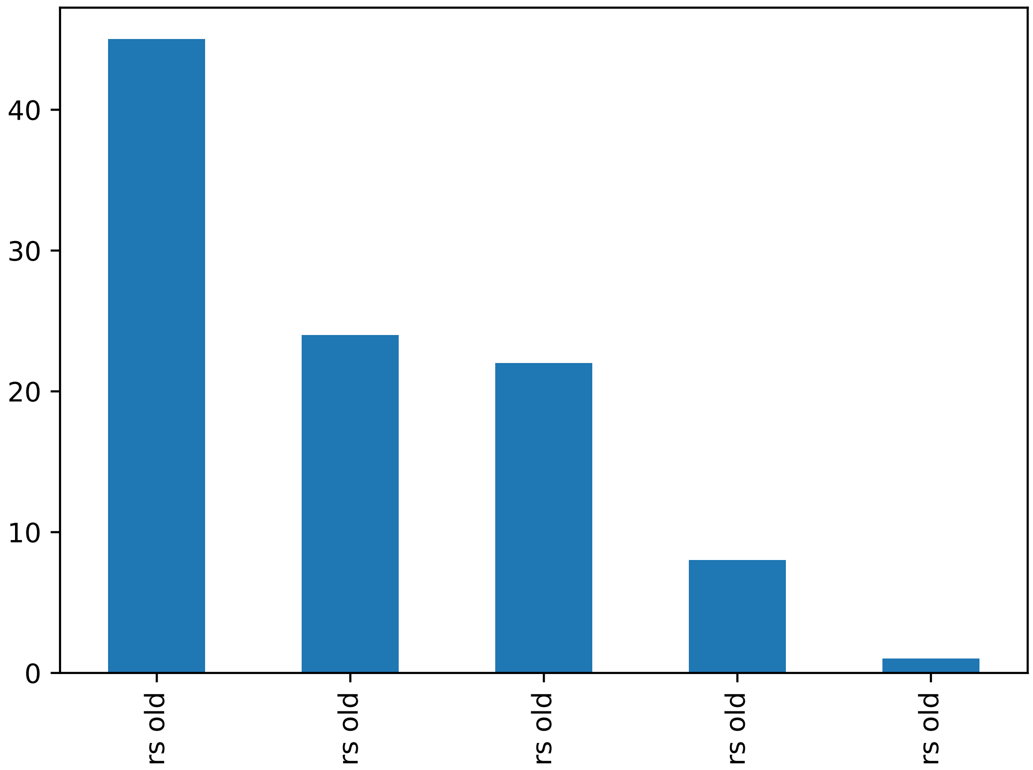
```
males = myDF.loc[(myDF.loc[:, "Gender"]=="Male"), :]
maleage=[]
maleage = [random.randint(0, len(males)) for i in range(0,100)]
males = males.iloc[femaleage]
print(maleage)
```

```
[58605, 43949, 38175, 50937, 19676, 35530, 2530, 22810, 7946, 34629, 21558,
54134, 5784, 9842, 6662, 6274, 8291, 2687, 25074, 456, 33410, 11223, 33962,
4748, 6288, 16048, 5758, 46716, 57219, 45553, 44255, 19786, 5769, 30611, 58396,
21470, 47596, 45050, 42477, 12103, 12712, 1104, 19872, 29428, 39604, 37301,
10434, 36230, 25705, 9839, 28288, 41409, 38534, 40772, 19277, 53558, 28009,
3434, 16230, 4782, 45801, 5286, 53888, 26722, 14578, 26855, 39147, 17471,
38851, 47199, 20253, 30802, 54739, 48948, 39672, 5098, 50435, 28177, 22335,
5486, 6037, 20331, 18082, 4001, 30965, 45796, 48831, 35478, 43834, 41712,
41782, 488, 12507, 32576, 40675, 15159, 34171, 50932, 42306, 30999]
```

```
males.loc[:,"Age"].value_counts().plot.bar()
print(males)
```

```
      Respondent  ...                  SurveyEasy
3754        5388  ...               Somewhat easy
3154        4507  ...                   Very easy
5787        8284  ...  Neither easy nor difficult
4977        7139  ...          Somewhat difficult
4769        6838  ...                   Very easy
...          ...  ...                         ...
2773        3965  ...               Somewhat easy
469          667  ...  Neither easy nor difficult
978         1376  ...  Neither easy nor difficult
1587        2225  ...  Neither easy nor difficult
2556        3637  ...  Neither easy nor difficult

[100 rows x 129 columns]
```

```
plt.show()
```



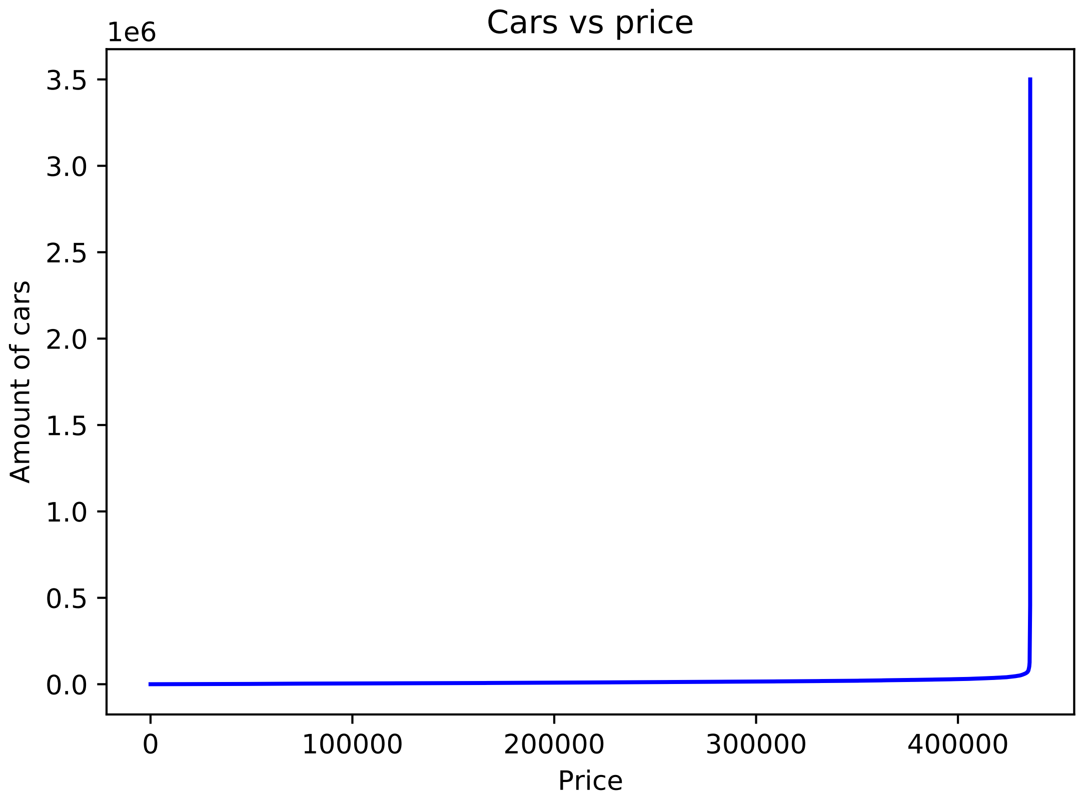
Question 5

```
import pandas as pd
from matplotlib import pyplot as plt
myDF = pd.read_csv("/class/datamine/data/craigslist/vehicles.csv")
pd.set_option('display.max_columns', None)
myDF.head()
```

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

                 region                          region_url  price    year  \
0         mohave county       https://mohave.craigslist.org   3495  2012.0
1          oregon coast  https://oregoncoast.craigslist.org  13750  2014.0
2  greenville / upstate   https://greenville.craigslist.org   2300  2001.0
3         mohave county       https://mohave.craigslist.org   9000  2004.0
4                 maine        https://maine.craigslist.org      0  2021.0

  manufacturer                             model  condition    cylinders  \
0         jeep                           patriot   like new  4 cylinders
1          bmw                      328i m-sport       good          NaN
2        dodge                           caravan  excellent  6 cylinders
```

```
3    chevrolet                       colorado ls  excellent  5 cylinders
4          NaN  Honda-Nissan-Kia-Ford-Hyundai-VW        NaN          NaN

    fuel  odometer title_status transmission                vin drive  \
0    gas       NaN        clean    automatic                NaN   NaN
1    gas   76237.0        clean    automatic                NaN   rwd
2    gas  199000.0        clean    automatic                NaN   NaN
3    gas   54000.0        clean    automatic  1GCCS196448191644   rwd
4  other       NaN        clean        other                NaN   NaN

       size    type paint_color  \
0       NaN     NaN      silver
1       NaN   sedan        grey
2       NaN     NaN         NaN
3  mid-size  pickup         red
4       NaN     NaN         NaN

                                           image_url  \
0  https://images.craigslist.org/00B0B_k2AXIJ21ok...
1  https://images.craigslist.org/00U0U_3cLk0WG0J8...
2  https://images.craigslist.org/00k0k_t4WqYn5nDC...
3  https://images.craigslist.org/00J0J_lJEzfeVLHI...
4  https://images.craigslist.org/01010_j0IW34mCsm...

                                         description  county state      lat  \
0  THIS 2012 JEEP PATRIOT IS A 4CYL. AC, STEREO, ...     NaN    az  34.4554
1  Selling my 2014 BMW 328i with the following be...     NaN    or  46.1837
2  01 DODGE CARAVAN,3.3 ENGINE,AUT TRANS,199000 M...     NaN    sc  34.9352
3  2004 Chevy Colorado LS, ONLY 54000 ORIGINAL MI...     NaN    az  34.4783
4  CALL: 207.548.6500 TEXT: 207.407.5598  **WE FI...     NaN    me  44.4699

       long
0 -114.2690
1 -123.8240
2  -81.9654
3 -114.2710
4  -68.8963
```

```python
my_values = list(tuple(myDF.loc[:, 'price'].dropna().to_list()))
my_values.sort()
plt.plot(my_values[0:-50], color="blue")
plt.title("Cars vs price")
plt.xlabel("Price")
plt.ylabel("Amount of cars")
plt.show()
```



```
plt.close()

#I created a lineplot of the price from all of the vehicles in our dataset.
```

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.