In [1]:
```
###### Question 1
import os
import pandas as pd
import numpy as np

# Part 1 - Load in data
path = "C:\\Users\\micha\\Documents\\DAAN862"
os.chdir(path)
file = "Assignment4_data.csv"
fp = pd.read_csv(file)

# Part 2 - Find and handle nulls
fp_isnulls = fp[fp.isnull().any(axis=1)]
# Zeroing out the nulls was chosen as a method
fp_fillna = fp.fillna(0)

# Part 3 - Variable column and dummy conversion
fp_dummies = pd.get_dummies(fp['variable'])
fp_with_dummies = fp.join(fp_dummies)

# Part 4 - Convert 'one' into bins
fp_qcuts = pd.qcut(fp_fillna['one'], 3)
```

In [2]:
```
# Part 1 Cont. - Explore
print(fp)
print(fp.iloc[7])

# From the show datasets, we can see that the imported data is 200 rows of
# positive/negative numerical columns titled 'one' to 'five' and then a
# variable column with an aplha-numeric value that appears to be
# either A1, A2, B1, or B2

# For this assignment, we are interested in applying the null-determining
# functions within pandas, as well as the dummy variable and splitting
# functions. As shown in the second print statement, we see a row where
# a null value is located in the 'one' column and we can see that other null
# values exist in the data set.
```

```
      one   two  three  four  five variable
0   -92.0 -76.0  -33.0   3.0 -13.0       B2
1   -21.0  76.0   38.0  -6.0  80.0       B1
2    -2.0 -47.0  -34.0 -86.0 -66.0       A1
3   -76.0  43.0    7.0 -40.0 -42.0       A1
4    44.0  37.0   -7.0 -14.0  30.0       A1
..    ...   ...    ...   ...   ...      ...
195  63.0   3.0  -30.0 -24.0 -59.0       A1
196  97.0 -48.0  -61.0 -25.0 -21.0       B1
197 -93.0 -75.0  -18.0 -67.0 -58.0       B1
198  54.0 -66.0  -80.0  92.0  62.0       A1
199  82.0  53.0  -77.0  79.0  97.0       B2

[200 rows x 6 columns]
one          NaN
two         35.0
three      -51.0
four        75.0
five        93.0
variable      A2
Name: 7, dtype: object
```

In [3]:
```
# Part 2 Cont. - Results
print("Before: ")
print(fp.iloc[7])
print("After: ")
print(fp_fillna.iloc[7])
print("\nBefore: ")
print(fp.iloc[9])
print("After: ")
print(fp_fillna.iloc[9])
```

```
Before:
one          NaN
two         35.0
three      -51.0
four        75.0
five        93.0
variable      A2
Name: 7, dtype: object
After:
one          0.0
two         35.0
three      -51.0
four        75.0
five        93.0
variable      A2
Name: 7, dtype: object

Before:
one        -98.0
two          NaN
three        5.0
four        47.0
five       -37.0
variable      A1
Name: 9, dtype: object
After:
one        -98.0
two          0.0
three        5.0
four        47.0
five       -37.0
variable      A1
Name: 9, dtype: object
```

In [4]:
```
# Part 3 Cont. - Results
fp_with_dummies
```

Out[4]:

	one	two	three	four	five	variable	A1	A2	B1	B2
0	-92.0	-76.0	-33.0	3.0	-13.0	B2	0	0	0	1
1	-21.0	76.0	38.0	-6.0	80.0	B1	0	0	1	0
2	-2.0	-47.0	-34.0	-86.0	-66.0	A1	1	0	0	0
3	-76.0	43.0	7.0	-40.0	-42.0	A1	1	0	0	0
4	44.0	37.0	-7.0	-14.0	30.0	A1	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...
195	63.0	3.0	-30.0	-24.0	-59.0	A1	1	0	0	0
196	97.0	-48.0	-61.0	-25.0	-21.0	B1	0	0	1	0
197	-93.0	-75.0	-18.0	-67.0	-58.0	B1	0	0	1	0
198	54.0	-66.0	-80.0	92.0	62.0	A1	1	0	0	0
199	82.0	53.0	-77.0	79.0	97.0	B2	0	0	0	1

200 rows × 10 columns

In [5]:
```
# Part 4 Cont. - Results
pd.value_counts(fp_qcuts)
```

Out[5]:
```
(-363.001, -32.667]    67
(29.667, 97.0]         67
(-32.667, 29.667]      66
Name: one, dtype: int64
```

In [7]:
```
###### Question 1
s = "I am happy to join with you today in what will go down in history as the great

# We will parse out any punctuation for further processing
# so we are left with just a list of words
punc = ['.', ',', '\'']
s_parsed = s
for p in punc:
    s_parsed = s_parsed.replace(p, '')

# Also convert everything to lowercase for comparing
s_parsed = s_parsed.lower()

words = s_parsed.split(' ')

# Part 1 - Determine unique number of words
# We can use a set to load the split words into, which
# creates a set of only the unqiue words due to set logic

uniques = set(words)
print("Total words: " + str(len(words)))
print("Unique words: " + str(len(uniques)))

# Part 2 - Determine which word appears the most
# We will use th epandas function 'value_counts' to
# achieve this
wordf = pd.DataFrame(words, columns=['word'])
word_counts = wordf['word'].value_counts()
print("\nMost common words: ")
print(word_counts)

# Part 3 - Determine how many words start with 't'
# We will use a for loop and the 'startswith' function
# Also noting, the question does not specify Unique
# words, as in we assume that the goal is to count
# the total times a word that starts with 't' is found
# and count any further duplicates
t_count = 0
for w in wordf.word:
    if w.startswith('t'):
        t_count += 1

print('\nTotal times a word starts with \'t\' : ' + str(t_count))
```

```
Total words: 177
Unique words: 107

Most common words:
the          14
of           12
in            8
a             6
years         5
             ..
beacon        1
decree        1
momentous     1
this          1
condition     1
Name: word, Length: 107, dtype: int64

Total times a word starts with 't' : 23
```

In [ ]: