# First Name: Last Name:

In [1]:

**import** pandas **as** pd **import** numpy **as** np

In [2]:

nesarc = pd.read\_csv('nesarc.csv', low\_memory=**False**) pd.set\_option('display.float\_format', **lambda** x:'%f'**%**x)

In [3]:

nesarc['S2AQ5A'] = pd.to\_numeric(nesarc['S2AQ5A'], errors='coerce') nesarc['S2AQ5B'] = pd.to\_numeric(nesarc['S2AQ5B'], errors='coerce') nesarc['S2AQ5D'] = pd.to\_numeric(nesarc['S2AQ5D'], errors='coerce') nesarc['S2BQ1B1'] = pd.to\_numeric(nesarc['S2BQ1B1'], errors='coerce')

# For Beer drinking status (S2AQ5A) fill in nan value with 11 & print first 10 rows

In [4]:

nesarc['S2AQ5A'].fillna(11, inplace=**True**) nesarc['S2AQ5A'].head(10)

Out[4]:

0 11.000000

1 1.000000

2 11.000000

3 11.000000

4 11.000000

5 2.000000

6 2.000000

7 2.000000

8 1.000000

9 2.000000

Name: S2AQ5A, dtype: float64

# For S2BQ1B1 - Effects of beer drinking (Beer Dependence) in the last 12 months replace 9 (unknown) in S2BQ1B1 (effects of beer consumtion in the last 12 months) to nan

**& print first 10 rows**

In [5]:

nesarc['S2BQ1B1']=nesarc['S2BQ1B1'].replace(9, np.nan) nesarc['S2BQ1B1'].head(10)

Out[5]:

0 nan

1 2.000000

1. nan
2. nan
3. nan

5 2.000000

6 2.000000

7 2.000000

8 2.000000

9 1.000000

Name: S2BQ1B1, dtype: float64

# Recode S2BQ1B1 so that 0 is no

**1 is yes currently 2 is no**

# & print first 5 rows

In [6]:

recode = {2:0, 1:1}

nesarc['S2BQ1B1']= nesarc['S2BQ1B1'].map(recode) nesarc['S2BQ1B1'].head()

Out[6]:

0 nan

1 0.000000

1. nan
2. nan
3. nan

Name: S2BQ1B1, dtype: float64

# Obtain a subset of nesarc data, with the following criteria

**Age from 26 to 50**

# Beer drinking status - S2AQ5A = Y

In [7]:

nesarc['AGE'] = pd.to\_numeric(nesarc['AGE'])

*#subset data to adults age 26 to 50 who have drink beer in the past 12 months* sub1=nesarc[(nesarc['AGE']**>**=26) **&** (nesarc['AGE']**<**=50) **&** (nesarc['S2AQ5A']==1)]

# Copy sub 1 to sub 2

In [8]:

sub2 = sub1.copy() sub2.head() len(sub2)

Out[8]: 10517

# Use sub2 data

**Print the count of HOW OFTEN DRANK BEER IN LAST 12 MONTHS (S2AQ5B)**

In [9]:

c\_beer\_feq = sub2['S2AQ5B'].value\_counts(sort=**False**, dropna=**False**) print ('counts for original S2AQ5B')

print(c\_beer\_feq)

|  |  |
| --- | --- |
| counts for | original S2AQ5B |
| 10.000000 | 1270 |
| 7.000000 | 1229 |
| 6.000000 | 1579 |
| 4.000000 | 1310 |
| 8.000000 | 682 |
| 5.000000 | 1485 |
| 2.000000 | 369 |
| 1.000000 | 417 |
| 3.000000 | 925 |
| 99.000000 | 25 |
| 9.000000 | 1226 |

Name: S2AQ5B, dtype: int64

# Based on my research, I'm assuming that drinking less than once a month is not going to affect a person. So, we are going replace the following in 'HOW OFTEN DRANK BEER IN LAST 12 MONTHS (S2AQ5B)' to

**nan**

# 8

**9**

# 10

**99**

In [10]:

sub2['S2AQ5B']=sub2['S2AQ5B'].replace(8, np.nan) sub2['S2AQ5B']=sub2['S2AQ5B'].replace(9, np.nan) sub2['S2AQ5B']=sub2['S2AQ5B'].replace(10, np.nan) sub2['S2AQ5B']=sub2['S2AQ5B'].replace(99, np.nan)

# Use sub2 data

**Print the count of HOW OFTEN DRANK BEER IN LAST 12 MONTHS (S2AQ5B) with 8, 9, 10 and 99 set nan**

In [11]:

c\_beer\_feq\_nan = sub2['S2AQ5B'].value\_counts(sort=**False**, dropna=**False**) print ('counts for original S2AQ5B with 8, 9, 10 and 99 set to NAN ') print(c\_beer\_feq\_nan)

|  |  |
| --- | --- |
| counts for | original S2AQ5B with 8, 9, 10 and 99 set to NAN |
| nan | 3203 |
| 7.000000 | 1229 |
| 6.000000 | 1579 |
| 4.000000 | 1310 |
| 5.000000 | 1485 |
| 2.000000 | 369 |
| 1.000000 | 417 |
| 3.000000 | 925 |

Name: S2AQ5B, dtype: int64

# Use sub2 data

**Count the NUMBER OF BEERS USUALLY CONSUMED ON DAYS WHEN DRANK BEER IN LAST 12 MONTHS (S2AQ5D)**

In [12]:

c\_beer\_quan = sub2['S2AQ5D'].value\_counts(sort=**False**,dropna=**False**) print ('counts for S2AQ5D')

print(c\_beer\_quan)

|  |  |
| --- | --- |
| counts for | S2AQ5D |
| 1.000000 | 3625 |
| 4.000000 | 749 |
| 3.000000 | 1619 |
| 2.000000 | 3087 |
| 10.000000 | 53 |
| 6.000000 | 702 |
| 8.000000 | 106 |
| 12.000000 | 150 |
| 7.000000 | 57 |
| 5.000000 | 278 |
| 24.000000 | 12 |
| 20.000000 | 3 |
| 14.000000 | 3 |
| 17.000000 | 1 |
| 99.000000 | 31 |
| 9.000000 | 19 |
| 18.000000 | 12 |
| 15.000000 | 7 |
| 13.000000 | 1 |
| 30.000000 | 1 |
| 11.000000 | 1 |

Name: S2AQ5D, dtype: int64

# Replace the 99 in 'NUMBER OF BEERS USUALLY CONSUMED ON DAYS WHEN DRANK BEER IN LAST 12 MONTHS (S2AQ5D)' to nan

In [13]:

sub2['S2AQ5D']=sub2['S2AQ5D'].replace(99, np.nan)

# Print the count of 'NUMBER OF BEERS USUALLY CONSUMED ON DAYS WHEN DRANK BEER IN LAST 12 MONTHS (S2AQ5D)'- with 99 set to NAN

c\_beer\_quan\_nan = sub2['S2AQ5D'].value\_counts(sort=**False**) print ('counts for S2AQ5D with 99 set to NAN') print(c\_beer\_quan\_nan)

counts for S2AQ5D with 99 set to NAN 1.000000 3625

4.000000 749

3.000000 1619

2.000000 3087

10.000000 53

6.000000 702

8.000000 106

12.000000 150

7.000000 57

5.000000 278

24.000000 12

20.000000 3

14.000000 3

17.000000 1

9.000000 19

18.000000 12

15.000000 7

13.000000 1

30.000000 1

11.000000 1

Name: S2AQ5D, dtype: int64

# Recode HOW OFTEN DRANK BEER IN LAST 12 MONTHS (S2AQ5B)

**as following 1 to 7**

# to 6

1. **to 5**

# to 3

1. **to 2**

# to 1

**so that larger categorical numbers indicate more frequently someone drinks beer**

recode1 = {1:7, 2:6, 3:5 , 4:4, 5:3, 6:2, 7:1} *#recoding so that higher numbers mean more s* sub2['BEER\_FEQ'] = sub2['S2AQ5B'].map(recode1)

recode\_beer\_feq = sub2['BEER\_FEQ'].value\_counts(sort=**False**) *#get count in each category* print ('counts for S2AQ5B')

print(recode\_beer\_feq)

|  |  |
| --- | --- |
| counts for | S2AQ5B |
| 1.000000 | 1229 |
| 2.000000 | 1579 |
| 4.000000 | 1310 |
| 3.000000 | 1485 |
| 6.000000 | 369 |
| 7.000000 | 417 |
| 5.000000 | 925 |

Name: BEER\_FEQ, dtype: int64

# Recode HOW OFTEN DRANK BEER IN LAST 12 MONTHS (S2AQ5B)

**as following 1 to 30**

# to 26

1. **to 14**

# to 8

1. **to 4**

# 6 to 2.5

**7 to 1**

# so that larger categorical numbers indicate more frequently someone drinks beer

*#recoding values for S2AQ5B into a new variable, BEER\_FEQMO*

recode2 = {1:30, 2:26, 3:14, 4:8, 5:4, 6:2.5, 7:1} *#recode to quantitative variable* sub2['BEER\_FEQMO']= sub2['S2AQ5B'].map(recode2)

recode\_beer\_feq\_m = sub2['BEER\_FEQMO'].value\_counts(sort=**False**) *#get count in each category* print ('counts for BEER\_FEQMO')

print(recode\_beer\_feq\_m)

|  |  |
| --- | --- |
| counts for | BEER\_FEQMO |
| 1.000000 | 1229 |
| 2.500000 | 1579 |
| 8.000000 | 1310 |
| 4.000000 | 1485 |
| 14.000000 | 925 |
| 26.000000 | 369 |
| 30.000000 | 417 |

Name: BEER\_FEQMO, dtype: int64

# Create secondary variable NUMBEERMO\_EST NUMBEERMO\_EST = BEER\_FEQMO \* S2AQ5D

In [17]:

*#secondary variable multiplying the number of days smoked/month and the approx number of ci* sub2['NUMBEERMO\_EST']=sub2['BEER\_FEQMO'] **\*** sub2['S2AQ5D'] *#get the number of cigarettes smo* sub2['NUMBEERMO\_EST'].head()

Out[17]:

1 nan

8 nan

12 4.000000

16 nan

24 nan

Name: NUMBEERMO\_EST, dtype: float64

# use sub2

**print the count for age**

*#examining frequency distributions for age* c\_age = sub2['AGE'].value\_counts(sort=**False**) print ('counts for AGE')

print(c\_age)

counts for AGE

|  |  |  |  |
| --- | --- | --- | --- |
| 32 | 502 |  | |
| 40 | 497 |  |  |
| 48 | 377 |  |  |
| 33 | 423 |  |  |
| 41 | 445 |  |  |
| 49 | 331 |  |  |
| 26 | 325 |  |  |
| 34 | 462 |  |  |
| 42 | 463 |  |  |
| 50 | 325 |  |  |
| 27 | 397 |  |  |
| 35 | 416 |  |  |
| 43 | 398 |  |  |
| 28 | 347 |  |  |
| 36 | 464 |  |  |
| 44 | 381 |  |  |
| 29 | 407 |  |  |
| 37 | 498 |  |  |
| 45 | 434 |  |  |
| 30 | 443 |  |  |
| 38 | 504 |  |  |
| 46 | 396 |  |  |
| 31 | 453 |  |  |
| 39 | 464 |  |  |
| 47 | 365 |  |  |
| Name: | AGE, | dtype: | int64 |

# use sub2

**print percentag for age**

p\_age = sub2['AGE'].value\_counts(sort=**False**, normalize=**True**) print ('percentages for AGE')

print (p\_age)

percentages for AGE

|  |  |
| --- | --- |
| 32 | 0.047732 |
| 40 | 0.047257 |
| 48 | 0.035847 |
| 33 | 0.040221 |
| 41 | 0.042312 |
| 49 | 0.031473 |
| 26 | 0.030902 |
| 34 | 0.043929 |
| 42 | 0.044024 |
| 50 | 0.030902 |
| 27 | 0.037748 |
| 35 | 0.039555 |
| 43 | 0.037843 |
| 28 | 0.032994 |
| 36 | 0.044119 |
| 44 | 0.036227 |
| 29 | 0.038699 |
| 37 | 0.047352 |
| 45 | 0.041267 |
| 30 | 0.042122 |
| 38 | 0.047922 |
| 46 | 0.037653 |
| 31 | 0.043073 |
| 39 | 0.044119 |
| 47 | 0.034706 |

Name: AGE, dtype: float64

# Group age into 3 groups 26 - 33

**34 - 41**

# 42 - 50

In [20]:

*# categorize quantitative variable based on customized splits using cut function*

*# splits into 3 groups (26-50) - remember that Python starts counting from 0, not 1* sub2['AGEGROUP3'] = pd.cut(sub2.AGE, [25, 33, 41, 51])

# print the count of this new group

In [21]:

c\_age\_group = sub2['AGEGROUP3'].value\_counts(sort=**False**, dropna=**True**) print('counts for AGEGROUP3')

print(c\_age\_group)

counts for AGEGROUP3

|  |  |  |
| --- | --- | --- |
| (25, | 33] | 3297 |
| (33, | 41] | 3750 |
| (41, | 51] | 3470 |

Name: AGEGROUP3, dtype: int64

# print the percentage of this new group

In [22]:

print('percentages for AGEGROUP3')

p\_age\_group = sub2['AGEGROUP3'].value\_counts(sort=**False**, normalize=**True**) print(p\_age\_group)

percentages for AGEGROUP3

|  |  |  |
| --- | --- | --- |
| (25, | 33] | 0.313492 |
| (33, | 41] | 0.356566 |
| (41, | 51] | 0.329942 |

Name: AGEGROUP3, dtype: float64

# Print the crosstab between AGEGROUP3 and AGE

In [23]:

*#crosstabs evaluating which ages were put into which AGEGROUP3* print (pd.crosstab(sub2['AGEGROUP3'], sub2['AGE']))

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AGE | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 ... 41 | 42 |
| \ |  |  |  |  |  |  |  |  |  |  |  |
| AGEGROUP3 |  |  |  |  |  |  |  |  |  | ... |  |
| (25, 33] | 325 | 397 | 347 | 407 | 443 | 453 | 502 | 423 | 0 | 0 ... 0 | 0 |
| (33, 41] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 462 | 416 ... 445 | 0 |
| (41, 51] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 ... 0 | 463 |

AGE 43 44 45 46 47 48 49 50 AGEGROUP3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (25, | 33] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (33, | 41] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (41, | 51] | 398 | 381 | 434 | 396 | 365 | 377 | 331 | 325 |

[3 rows x 25 columns]

# Group age into 4 groups automatically - use cut

**- not in practical**

In [24]:

*# quartile split (use qcut function & ask for 4 groups - gives you quartile split)* sub2['AGEGROUP4']=pd.qcut(sub2.AGE, 4, labels=["1=0%tile","2=25%tile","3=50%tile","4=75%til c8 = sub2['AGEGROUP4'].value\_counts(sort=**False**, dropna=**True**)

print('AGE - 4 categories - quartiles') print(c8)

AGE - 4 categories - quartiles

|  |  |
| --- | --- |
| 1=0%tile | 2874 |
| 2=25%tile | 2767 |
| 3=50%tile | 2267 |
| 4=75%tile | 2609 |

Name: AGEGROUP4, dtype: int64

# Print the crosstab between AGEGROUP4 and AGE

**- not in practical**

In [25]:

print (pd.crosstab(sub2['AGEGROUP4'], sub2['AGE']))

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AGE  \ AGEGROUP4 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 ... 41  ... | 42 |
| 1=0%tile | 325 | 397 | 347 | 407 | 443 | 453 | 502 | 0 | 0 | 0 ... 0 | 0 |
| 2=25%tile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 423 | 462 | 416 ... 0 | 0 |
| 3=50%tile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 ... 445 | 463 |
| 4=75%tile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 ... 0 | 0 |
| AGE AGEGROUP4  1=0%tile | 43  0 | 44  0 | 45  0 | 46  0 | 47  0 | 48  0 | 49  0 | 50  0 |  |  |  |
| 2=25%tile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |
| 3=50%tile | 398 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |
| 4=75%tile | 0 | 381 | 434 | 396 | 365 | 377 | 331 | 325 |  |  |  |

[4 rows x 25 columns]