First Name:

Last Name:

Import Libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
```

Read in data

In [2]:

```
nesarc = pd.read_csv('nesarc - large.csv', low_memory=False)
pd.set_option('display.float_format', lambda x:'%f'%x)
```

In [3]:

```
nesarc['S2AQ5B'] = pd.to_numeric(nesarc['S2AQ5B'], errors='coerce') #convert variable to nu
nesarc['S2AQ5D'] = pd.to_numeric(nesarc['S2AQ5D'], errors='coerce') #convert variable to nu
nesarc['S2AQ5A'] = pd.to_numeric(nesarc['S2AQ5A'], errors='coerce') #convert variable to nu
nesarc['S2BQ1B1'] = pd.to_numeric(nesarc['S2BQ1B1'], errors='coerce') #convert variable to
nesarc['AGE'] = pd.to_numeric(nesarc['AGE'], errors='coerce') #convert variable to numeric
```

In [4]:

```
sub1=nesarc[(nesarc['AGE']>=26) & (nesarc['AGE']<=50) & (nesarc['S2AQ5A']==1)]
sub2=sub1.copy()
```

In [5]:

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

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)

sub2['S2BQ1B1']=sub2['S2BQ1B1'].replace(9, np.nan)
```

In [6]:

```
recode2 = {1:30, 2:26, 3:14, 4:8, 5:4, 6:2.5, 7:1}
sub2['BEER_FEQMO']= sub2['S2AQ5B'].map(recode2)

recode3 = {2:0, 1:1}
sub2['S2BQ1B1']= sub2['S2BQ1B1'].map(recode3)
```

Remap beer consumption for future analysis.

In [ ]:

```
#secondary variable
sub2['NUMBEERMO_EST']=sub2['BEER_FEQMO'] * sub2['S2AQ5D']
```

Regression

Categorical Explanatory variable

Quantitative Response variable

In [7]:

```
reg1 = smf.ols('NUMBEERMO_EST ~ S2BQ1B1', data=sub2).fit()
print (reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:              BEER_FEQMO   R-squared:                       0.0
16
Model:                             OLS   Adj. R-squared:                  0.0
16
Method:                  Least Squares   F-statistic:                     11
7.2
Date:                 Fri, 27 Apr 2018   Prob (F-statistic):           4.14e-
27
Time:                         14:01:36   Log-Likelihood:                -2537
1.
No. Observations:                7236   AIC:                          5.075e+
04
Df Residuals:                    7234   BIC:                          5.076e+
04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept      7.4565      0.098     75.831      0.000       7.264       7.6
49
S2BQ1B1        4.0102      0.370     10.827      0.000       3.284       4.7
36
==============================================================================
==
Omnibus:                     1893.858   Durbin-Watson:                   1.9
98
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3852.4
37
Skew:                           1.592   Prob(JB):                         0.
00
Kurtosis:                       4.626   Cond. No.                         3.
93
==============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

R-squared value shows 0% of variances are explained by the model, a small Prob(f-statistic) indicates we should reject null hypothesis.

In [8]:

```
sub3 = sub2[['NUMBEERMO_EST', 'S2BQ1B1']].dropna()
```

In [9]:

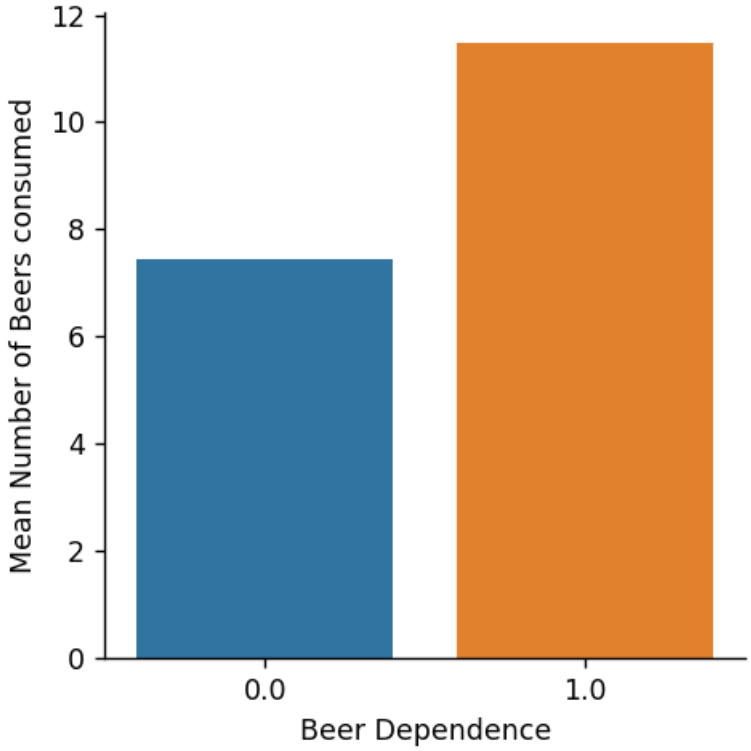
```
# group means & sd
print ("Mean")
ds1 = sub3.groupby('S2BQ1B1').mean()
print (ds1)
print ("Standard deviation")
ds2 = sub3.groupby('S2BQ1B1').std()
print (ds2)
```

```
Mean
          BEER_FEQMO
S2BQ1B1
0.000000    7.456512
1.000000   11.466667
Standard deviation
          BEER_FEQMO
S2BQ1B1
0.000000    7.903663
1.000000    9.946313
```

Calculating mean and std, and creating a pivot table to display result

In [10]:

```
%matplotlib notebook
plt.xlabel('Beer Dependence')
plt.ylabel('Mean Number of Beers consumed')
```



The bar chart shows people with beer dependence consume 5 more bottles of beer per month.

Out[10]:

```
Text(13.8194,0.5,'Mean Number of Beers consumed')
```

Logistical Regression - Scenario 1

In [11]:

```
# logistic regression with GENAXLIFE
lreg1 = smf.logit(formula = 'S2BQ1B1 ~ GENAXLIFE', data = sub2).fit()
print (lreg1.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.212712
         Iterations 7
                           Logit Regression Results
==============================================================================
==
Dep. Variable:                S2BQ1B1   No. Observations:                104
06
Model:                          Logit   Df Residuals:                    104
04
Method:                           MLE   Df Model:
1
Date:                Fri, 27 Apr 2018   Pseudo R-squ.:                0.0072
08
Time:                        14:02:29   Log-Likelihood:                -221
3.5
converged:                       True   LL-Null:                       -222
9.6
                                        LLR p-value:                1.434e-
08
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept     -2.8998      0.045    -64.130      0.000      -2.988      -2.8
11
GENAXLIFE      0.8948      0.144      6.227      0.000       0.613       1.1
76
==============================================================================
==
```

A low p-value indicates we should reject null hypothesis.

In [12]:

```
params = lreg1.params
conf = lreg1.conf_int()
conf['OR'] = params
conf.columns = ['Lower CI', 'Upper CI', 'OR']
print (np.exp(conf))
```

```
           Lower CI  Upper CI       OR
Intercept  0.050367  0.060134 0.055034
GENAXLIFE  1.846265  3.242687 2.446806
```

Logistical Regression - Scenario 2

In [13]:

```
sub2['DYSLIFE'] = pd.to_numeric(sub2['DYSLIFE'], errors='coerce')
```

In [14]:

```
# logistic regression with social phobia and depression
lreg2 = smf.logit(formula = 'S2BQ1B1 ~ GENAXLIFE + DYSLIFE', data = sub2).fit()
print (lreg2.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.212607
         Iterations 7
                           Logit Regression Results
==============================================================================
==
Dep. Variable:                 S2BQ1B1   No. Observations:                 104
06
Model:                           Logit   Df Residuals:                     104
03
Method:                            MLE   Df Model:
2
Date:                Fri, 27 Apr 2018   Pseudo R-squ.:                0.0076
97
Time:                        14:04:02   Log-Likelihood:                -221
2.4
converged:                       True   LL-Null:                       -222
9.6
                                        LLR p-value:                3.524e-
08
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept     -2.9099      0.046    -63.488      0.000      -3.000      -2.8
20
GENAXLIFE      0.8070      0.156      5.162      0.000       0.501       1.1
13
DYSLIFE        0.2638      0.174      1.513      0.130      -0.078       0.6
06
==============================================================================
==
```

A low p-value indicates we should reject null hypothesis.

In [15]:

```
# odd ratios with 95% confidence intervals
params = lreg2.params
conf = lreg2.conf_int()
conf['OR'] = params
conf.columns = ['Lower CI', 'Upper CI', 'OR']
print (np.exp(conf))
```

```
           Lower CI  Upper CI       OR
Intercept  0.049798  0.059600 0.054479
GENAXLIFE  1.649620  3.044656 2.241099
DYSLIFE    0.924975  1.832270 1.301846
```

Logistical Regression - Scenario 3

In [16]:

```
def PANIC (x1):
    if ((x1['S6Q1']==1 and x1['S6Q2']==1) or (x1['S6Q2']==1 and x1['S6Q3']==1) or
    (x1['S6Q3']==1 and x1['S6Q61']==1) or (x1['S6Q61']==1 and x1['S6Q62']==1) or
    (x1['S6Q62']==1 and x1['S6Q63']==1) or (x1['S6Q63']==1 and x1['S6Q64']==1) or
    (x1['S6Q64']==1 and x1['S6Q65']==1) or (x1['S6Q65']==1 and x1['S6Q66']==1) or
    (x1['S6Q66']==1 and x1['S6Q67']==1) or (x1['S6Q67']==1 and x1['S6Q68']==1) or
    (x1['S6Q68']==1 and x1['S6Q69']==1) or (x1['S6Q69']==1 and x1['S6Q610']==1) or
    (x1['S6Q610']==1 and x1['S6Q611']==1) or (x1['S6Q611']==1 and x1['S6Q612']==1) or
    (x1['S6Q612']==1 and x1['S6Q613']==1) or (x1['S6Q613']==1 and x1['S6Q7']==1) or
    x1['S6Q7']==1):
        return 1
    else:
        return 0
sub2['PANIC'] = sub1.apply (lambda x1: PANIC (x1), axis=1)
c7 = sub2["PANIC"].value_counts(sort=False, dropna=False)
print(c7)
```

```
0    9596
1     921
Name: PANIC, dtype: int64
```

In [17]:

```
# logistic regression with panic
lreg3 = smf.logit(formula = 'S2BQ1B1 ~ PANIC', data = sub2).fit()
print (lreg3.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.213531
         Iterations 7
                           Logit Regression Results
==============================================================================
==
Dep. Variable:                 S2BQ1B1   No. Observations:                  104
06
Model:                           Logit   Df Residuals:                      104
04
Method:                            MLE   Df Model:
1
Date:                Fri, 27 Apr 2018   Pseudo R-squ.:                0.0033
85
Time:                        14:05:24   Log-Likelihood:                 -222
2.0
converged:                       True   LL-Null:                        -222
9.6
                                        LLR p-value:                 0.00010
23
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept     -2.8917      0.046    -62.875      0.000      -2.982      -2.8
02
PANIC          0.5210      0.127      4.102      0.000       0.272       0.7
70
==============================================================================
==
```

R-squared value shows only 10% variables are explained by regression model, Prob(f-statistic) is 0.6%, means we should reject null hypothesis.

In [18]:

```
# odd ratios with 95% confidence intervals
print ("Odds Ratios")
params = lreg3.params
conf = lreg3.conf_int()
conf['OR'] = params
conf.columns = ['Lower CI', 'Upper CI', 'OR']
print (np.exp(conf))
```

```
Odds Ratios
           Lower CI  Upper CI       OR
Intercept  0.050699  0.060715 0.055481
PANIC      1.312633  2.159621 1.683684
```

Logistical Regression - Scenario 4

In [19]:

```
# logistic regression with panic and depression
lreg4 = smf.logit(formula = 'S2BQ1B1 ~ PANIC + DYSLIFE', data = sub2).fit()
print (lreg4.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.213222
         Iterations 7
                           Logit Regression Results
==============================================================================
==
Dep. Variable:                 S2BQ1B1   No. Observations:                  104
06
Model:                           Logit   Df Residuals:                      104
03
Method:                            MLE   Df Model:
2
Date:                Fri, 27 Apr 2018   Pseudo R-squ.:                  0.0048
28
Time:                        14:06:22   Log-Likelihood:                 -221
8.8
converged:                       True   LL-Null:                        -222
9.6
                                        LLR p-value:                 2.113e-
05
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept     -2.9109      0.047    -62.201      0.000      -3.003      -2.8
19
PANIC          0.4432      0.131      3.373      0.001       0.186       0.7
01
DYSLIFE        0.4380      0.165      2.655      0.008       0.115       0.7
61
==============================================================================
==
```

A low p-value means we can reject null hypothesis

In [20]:

```
# odd ratios with 95% confidence intervals
print ("Odds Ratios")
params = lreg4.params
conf = lreg4.conf_int()
conf['OR'] = params
conf.columns = ['Lower CI', 'Upper CI', 'OR']
print (np.exp(conf))
```

```
Odds Ratios
           Lower CI  Upper CI       OR
Intercept  0.049659  0.059658 0.054429
PANIC      1.203973  2.015228 1.557652
DYSLIFE    1.121488  2.141030 1.549561
```