Activity 5: Problem 1

Part 1

1) Read the data into your software system

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

df = pd.read_csv("C:/Users/danma/Downloads/HeartDisease.csv")
x = df.loc[:, df.columns != 'chd']
y = df['chd']

from sklearn.preprocessing import LabelEncoder
# label encoder to transform the Present or Absent column to 0 or 1
le = LabelEncoder()
label = le.fit_transform(x['famhist'])
x = x.loc[:, x.columns != 'famhist']
x['famhist'] = label

print(x.head(10))
print("\n",y.head(10))
```

```
   names  sbp  tobacco   ldl  adiposity  typea  obesity  alcohol  age  famhist
0      1  160    12.00  5.73      23.11     49    25.30    97.20   52        1
1      2  144     0.01  4.41      28.61     55    28.87     2.06   63        0
2      3  118     0.08  3.48      32.28     52    29.14     3.81   46        1
3      4  170     7.50  6.41      38.03     51    31.99    24.26   58        1
4      5  134    13.60  3.50      27.78     60    25.99    57.34   49        1
5      6  132     6.20  6.47      36.21     62    30.77    14.14   45        1
6      7  142     4.05  3.38      16.20     59    20.81     2.62   38        0
7      8  114     4.08  4.59      14.60     62    23.11     6.72   58        1
8      9  114     0.00  3.83      19.40     49    24.86     2.49   29        1
9     10  132     0.00  5.80      30.96     69    30.11     0.00   53        1

 0    1
1    1
2    0
3    1
4    1
5    0
6    0
7    1
8    0
9    1
Name: chd, dtype: int64
```

2) Examine univariate statistics for the following variables: sbp, tobacco, ldl, adiposity, typea, obesity, alcohol, and age. (not including the target variable)

In [2]:
```
print('spb mean', x['sbp'].mean())
print('spb median', x['sbp'].median())
print('spb mode', x['sbp'].mode())
print('sbp skewness', x['sbp'].skew())
print('spb 25% quantile', x['sbp'].quantile(.25))
```

```
print('spb 50% quantile', x['sbp'].quantile(.50))
print('spb 75% quantile', x['sbp'].quantile(.75))
```

```
spb mean 138.32683982683983
spb median 134.0
spb mode 0    134
1    136
Name: sbp, dtype: int64
sbp skewness 1.1805906253694305
spb 25% quantile 124.0
spb 50% quantile 134.0
spb 75% quantile 148.0
```

In [3]:
```
print('tobacco mean', x['tobacco'].mean())
print('tobacco median', x['tobacco'].median())
print('tobacco mode', x['tobacco'].mode())
print('tobacco skewness', x['tobacco'].skew())
print('tobacco 25% quantile', x['tobacco'].quantile(.25))
print('tobacco 50% quantile', x['tobacco'].quantile(.50))
print('tobacco 75% quantile', x['tobacco'].quantile(.75))
```

```
tobacco mean 3.635649350649348
tobacco median 2.0
tobacco mode 0    0.0
Name: tobacco, dtype: float64
tobacco skewness 2.0792096673876146
tobacco 25% quantile 0.052500000000000005
tobacco 50% quantile 2.0
tobacco 75% quantile 5.5
```

In [4]:
```
print('ldl mean', x['ldl'].mean())
print('ldl median', x['ldl'].median())
print('ldl mode', x['ldl'].mode())
print('ldl skewness', x['ldl'].skew())
print('ldl 25% quantile', x['ldl'].quantile(.25))
print('ldl 50% quantile', x['ldl'].quantile(.50))
print('ldl 75% quantile', x['ldl'].quantile(.75))
```

```
ldl mean 4.7403246753246835
ldl median 4.34
ldl mode 0    3.57
1    3.95
2    4.37
Name: ldl, dtype: float64
ldl skewness 1.3131039798013922
ldl 25% quantile 3.2824999999999998
ldl 50% quantile 4.34
ldl 75% quantile 5.79
```

In [5]:
```
print('adiposity mean', x['adiposity'].mean())
print('adiposity median', x['adiposity'].median())
print('adiposity mode', x['adiposity'].mode())
print('adiposity skewness', x['adiposity'].skew())
print('adiposity 25% quantile', x['adiposity'].quantile(.25))
print('adiposity 50% quantile', x['adiposity'].quantile(.50))
print('adiposity 75% quantile', x['adiposity'].quantile(.75))
```

```
adiposity mean 25.40673160173159
adiposity median 26.115000000000002
adiposity mode 0    21.10
1    27.55
2    29.30
3    30.79
Name: adiposity, dtype: float64
adiposity skewness -0.21464592856083986
adiposity 25% quantile 19.775
adiposity 50% quantile 26.115000000000002
adiposity 75% quantile 31.2275
```

In [6]:
```
print('typea mean', x['typea'].mean())
print('typea median', x['typea'].median())
print('typea mode', x['typea'].mode())
print('typea skewness', x['typea'].skew())
print('typea 25% quantile', x['typea'].quantile(.25))
print('typea 50% quantile', x['typea'].quantile(.50))
print('typea 75% quantile', x['typea'].quantile(.75))
```

```
typea mean 53.103896103896105
typea median 53.0
typea mode 0    52
Name: typea, dtype: int64
typea skewness -0.34643775469900984
typea 25% quantile 47.0
typea 50% quantile 53.0
typea 75% quantile 60.0
```

In [7]:
```
print('obesity mean', x['obesity'].mean())
print('obesity median', x['obesity'].median())
print('obesity mode', x['obesity'].mode())
print('obesity skewness', x['obesity'].skew())
print('obesity 25% quantile', x['obesity'].quantile(.25))
print('obesity 50% quantile', x['obesity'].quantile(.50))
print('obesity 75% quantile', x['obesity'].quantile(.75))
```

```
obesity mean 26.044112554112576
obesity median 25.805
obesity mode 0    24.86
1    26.09
Name: obesity, dtype: float64
obesity skewness 0.9052194041401875
obesity 25% quantile 22.985
obesity 50% quantile 25.805
obesity 75% quantile 28.4975
```

In [8]:
```
print('alcohol mean', x['alcohol'].mean())
print('alcohol median', x['alcohol'].median())
print('alcohol mode', x['alcohol'].mode())
print('alcohol skewness', x['alcohol'].skew())
print('alcohol 25% quantile', x['alcohol'].quantile(.25))
print('alcohol 50% quantile', x['alcohol'].quantile(.50))
print('alcohol 75% quantile', x['alcohol'].quantile(.75))
```

```
alcohol mean 17.044393939393952
alcohol median 7.51
alcohol mode 0    0.0
Name: alcohol, dtype: float64
alcohol skewness 2.3126989374804183
alcohol 25% quantile 0.51
alcohol 50% quantile 7.51
alcohol 75% quantile 23.8925
```

In [9]:
```
print('age mean', x['age'].mean())
```

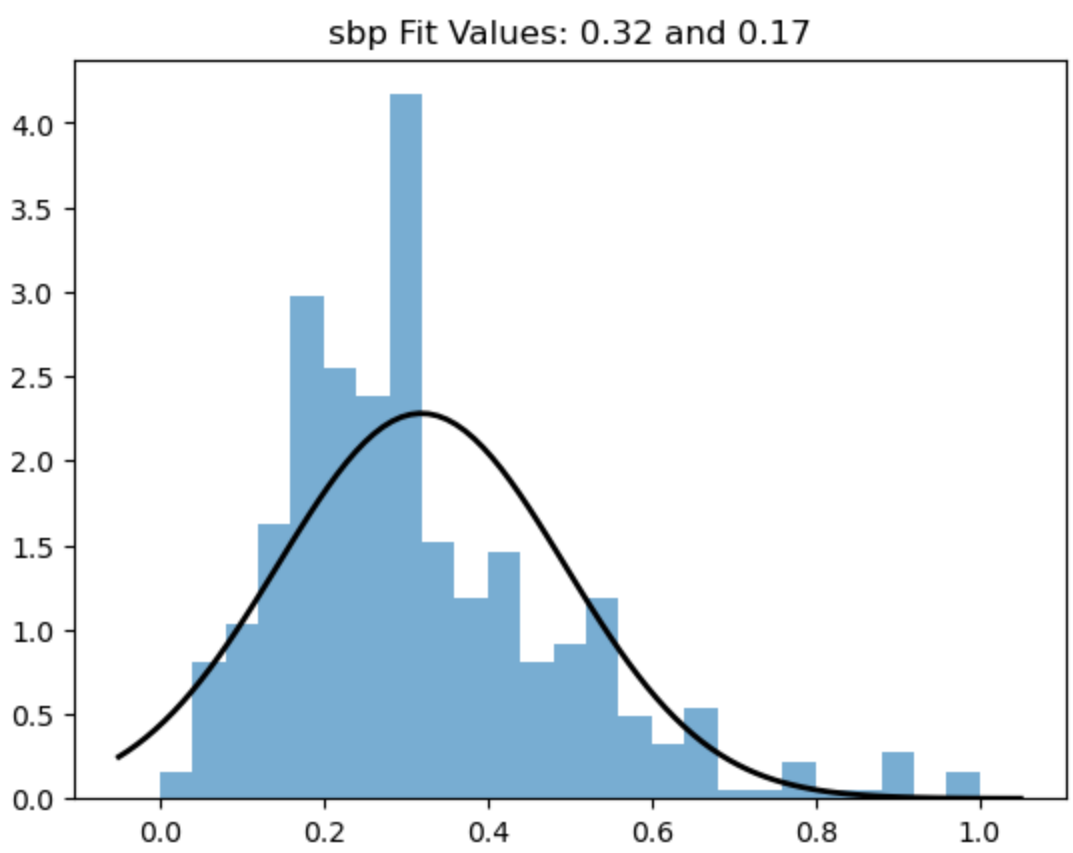
```
print('age median', x['age'].median())
print('age mode', x['age'].mode())
print('age skewness', x['age'].skew())
print('age 25% quantile', x['age'].quantile(.25))
print('age 50% quantile', x['age'].quantile(.50))
print('age 75% quantile', x['age'].quantile(.75))
```

```
age mean 42.816017316017316
age median 45.0
age mode 0    16
Name: age, dtype: int64
age skewness -0.38173425852590315
age 25% quantile 31.0
age 50% quantile 45.0
age 75% quantile 55.0
```

3) Produce histogram of each of the following variables with imposing normal curve: sbp, tobacco, ldl, adiposity, typea, obesity, alcohol, and age.

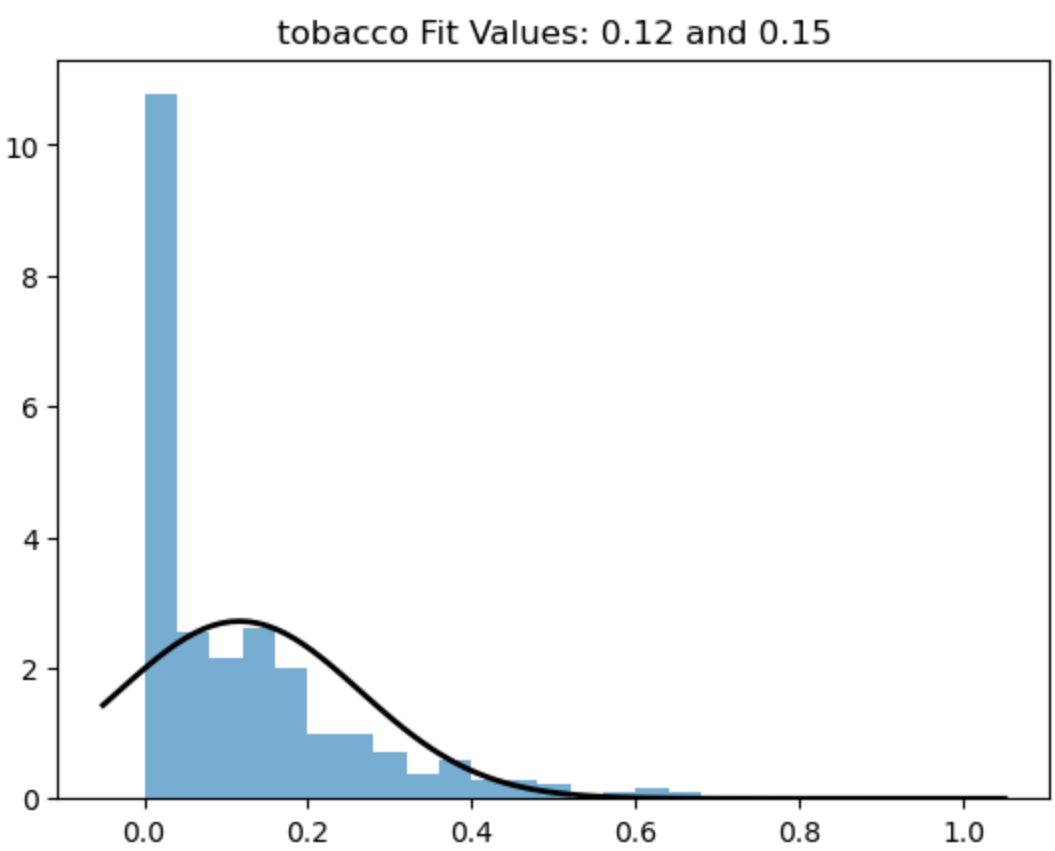
In [10]:

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

data = (x['sbp']-x['sbp'].min())/(x['sbp'].max()-x['sbp'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "sbp Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



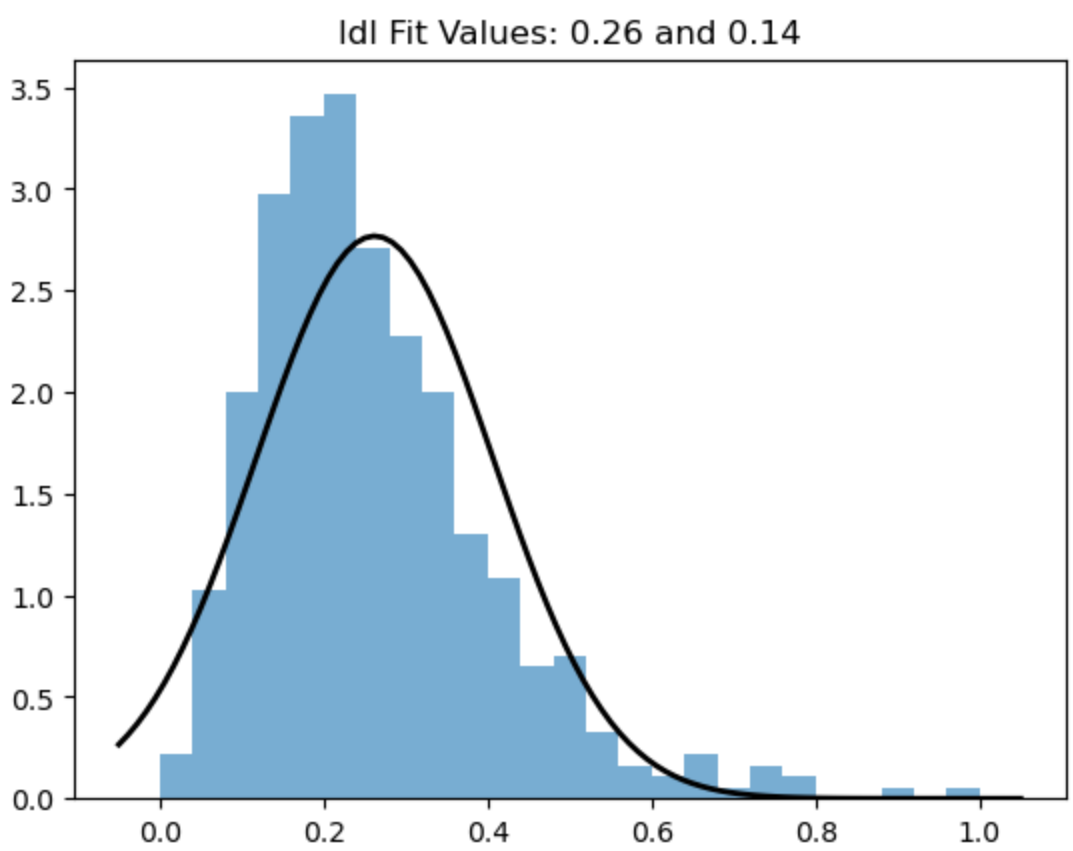
In [11]:

```
data = (x['tobacco']-x['tobacco'].min())/(x['tobacco'].max()-x['tobacco'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "tobacco Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



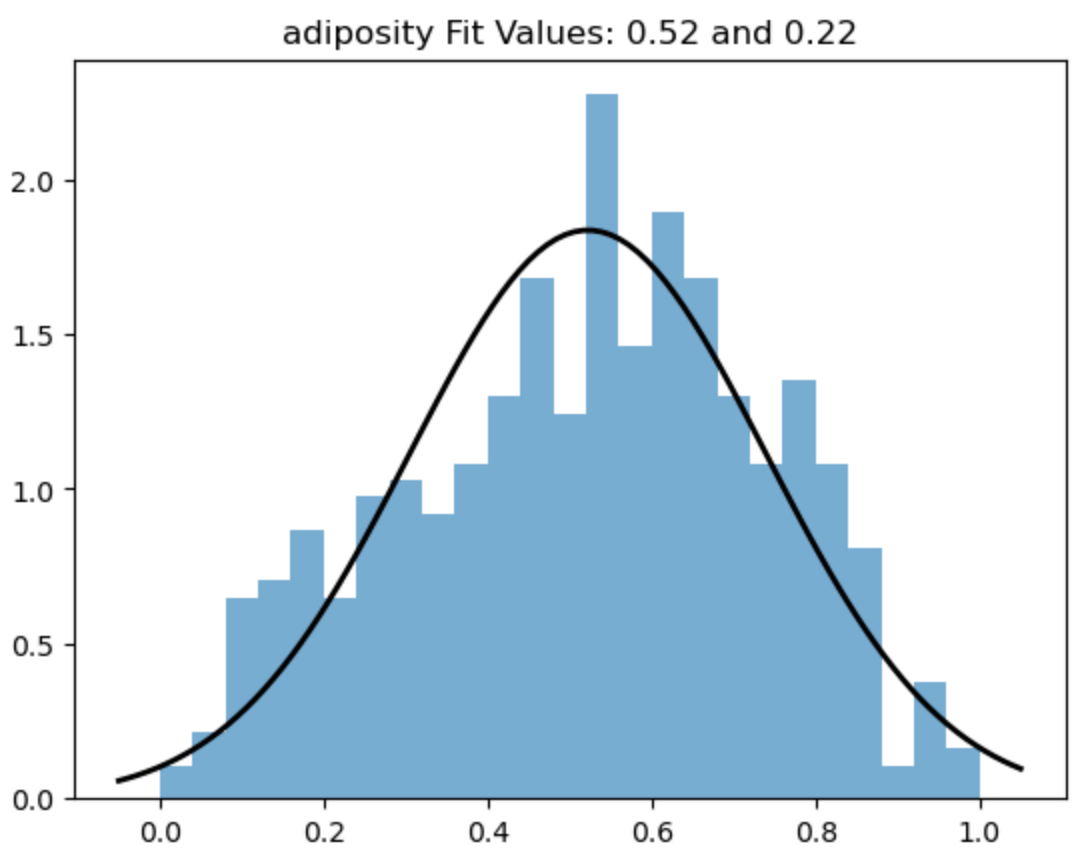
In [12]:

```
data = (x['ldl']-x['ldl'].min())/(x['ldl'].max()-x['ldl'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "ldl Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



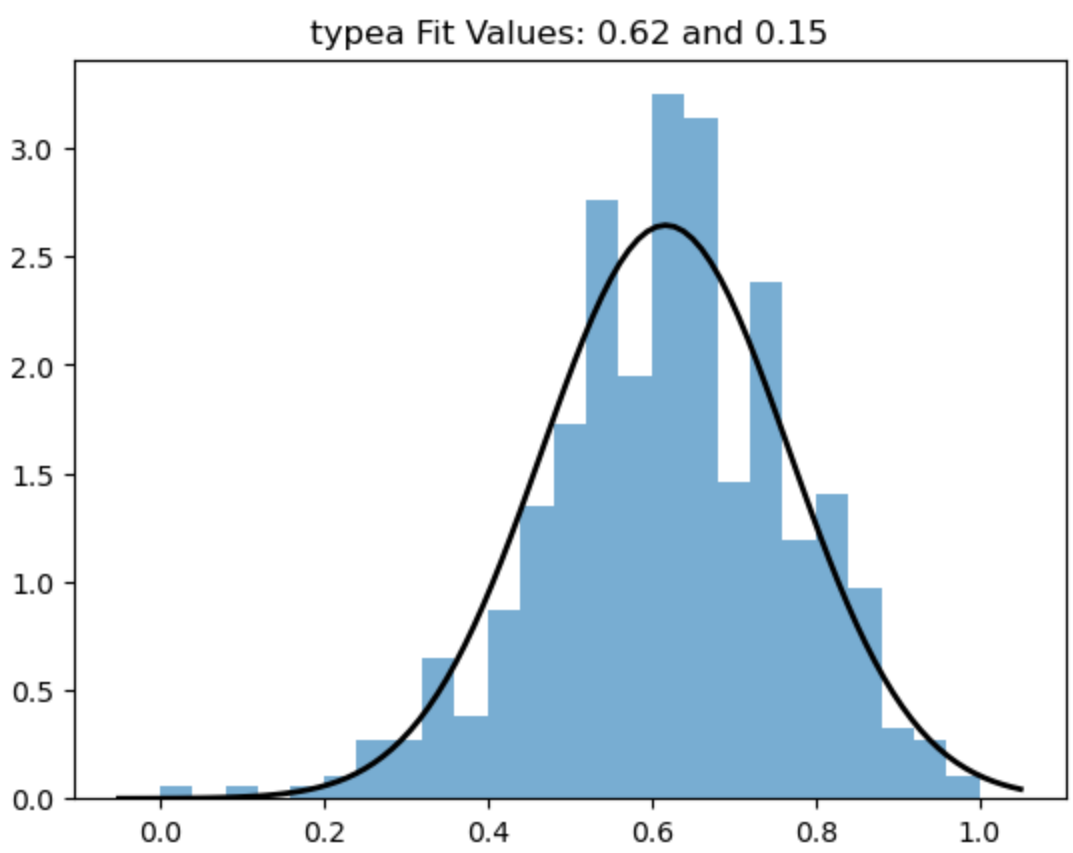
In [13]:

```
data = (x['adiposity']-x['adiposity'].min())/(x['adiposity'].max()-x['adiposity'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "adiposity Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```

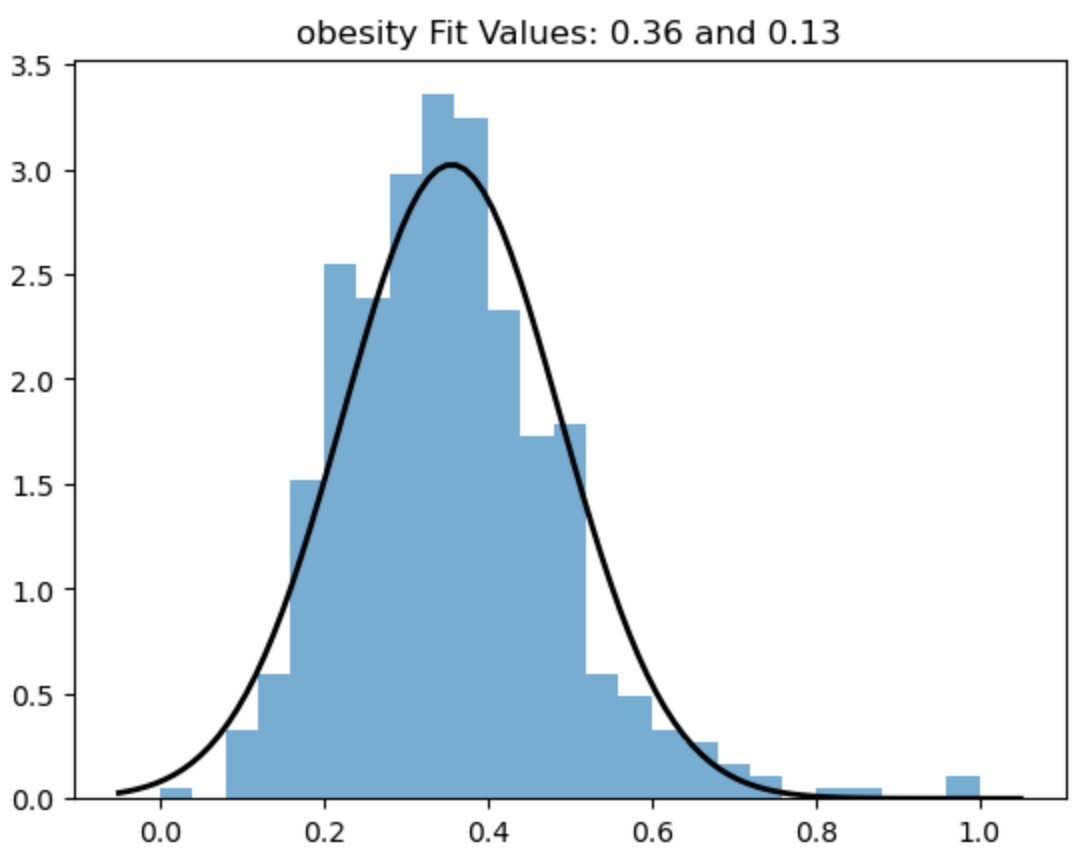


In [14]:

```
data = (x['typea']-x['typea'].min())/(x['typea'].max()-x['typea'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "typea Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```

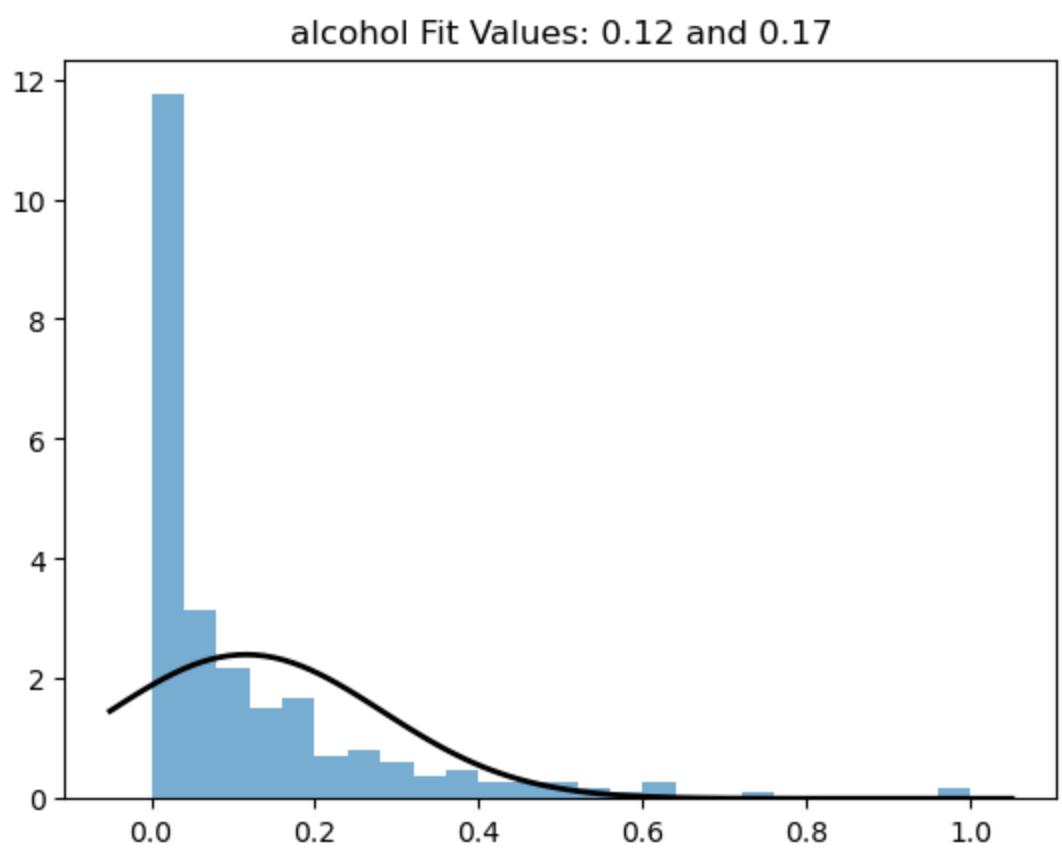


In [15]:
```
data = (x['obesity']-x['obesity'].min())/(x['obesity'].max()-x['obesity'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "obesity Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



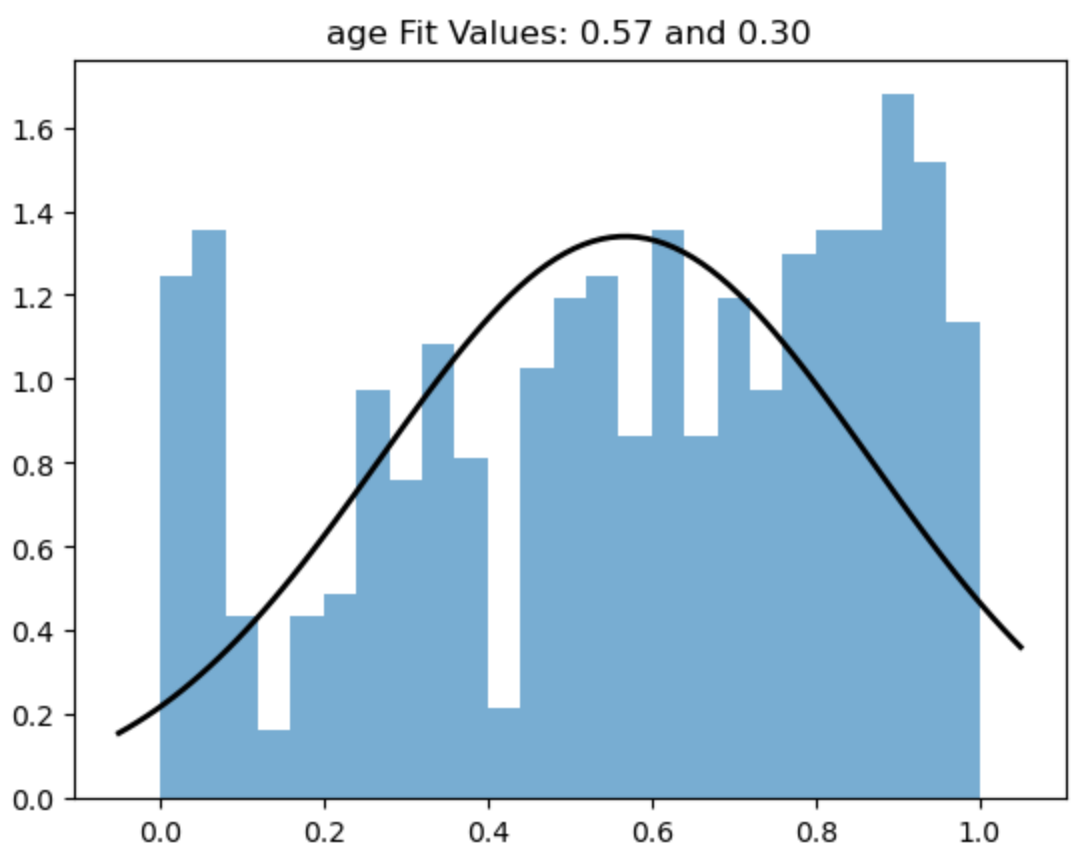
In [16]:

```
data = (x['alcohol']-x['alcohol'].min())/(x['alcohol'].max()-x['alcohol'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "alcohol Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



In [17]:

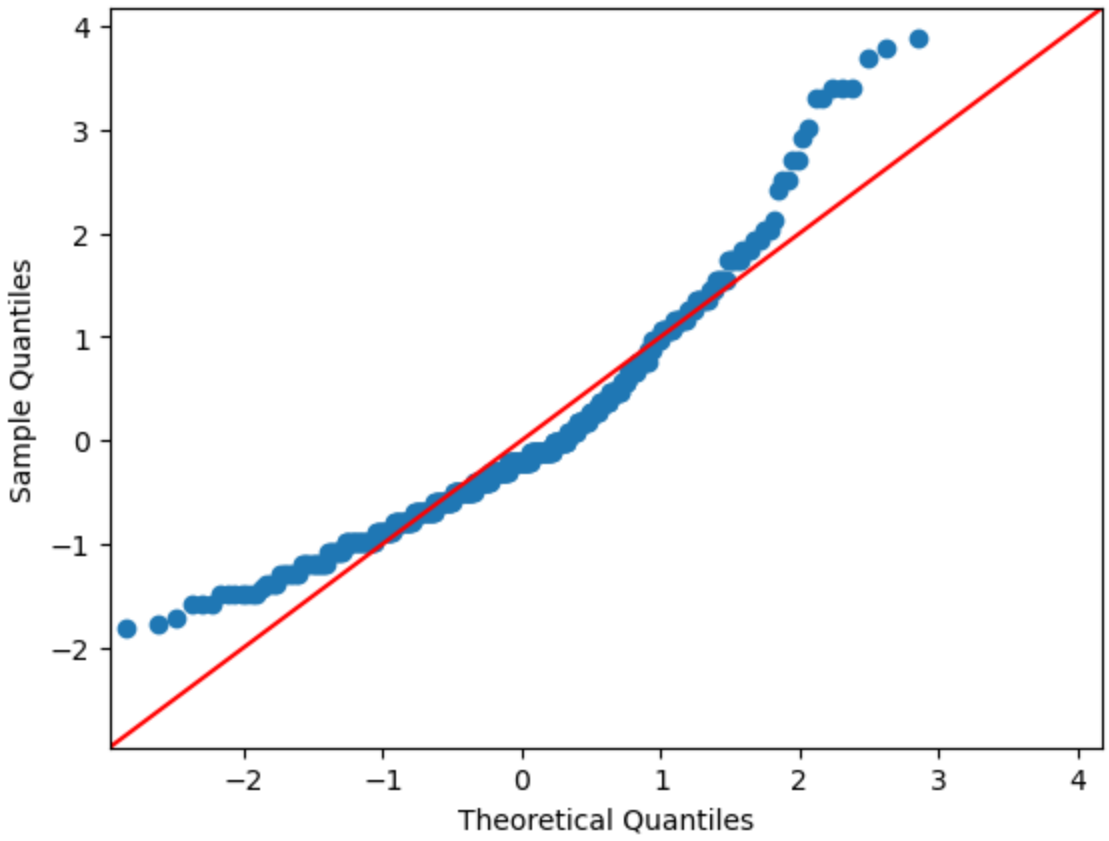
```
data = (x['age']-x['age'].min())/(x['age'].max()-x['age'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "age Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



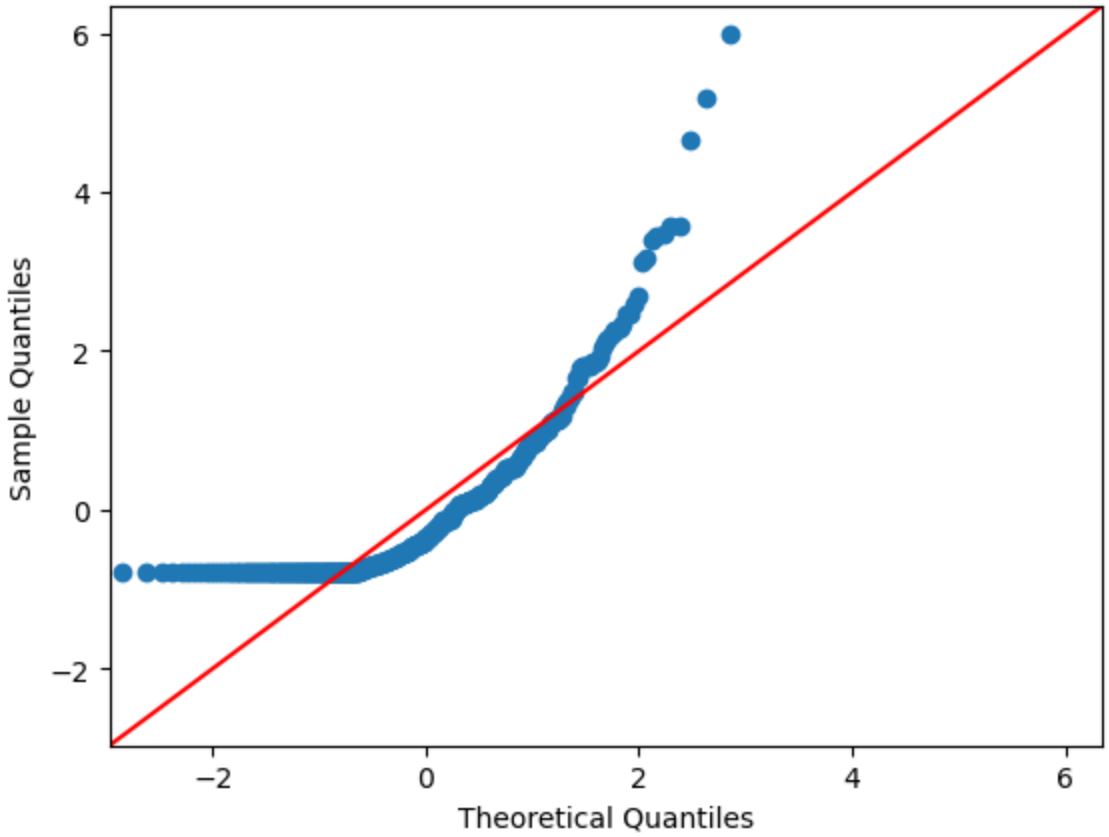
4) Produce quantile plot of each of the following variables: sbp, tobacco, ldl, adiposity, typea, obesity, alcohol, and age.

In [18]:
```
import statsmodels.api as sm
import pylab as py
```

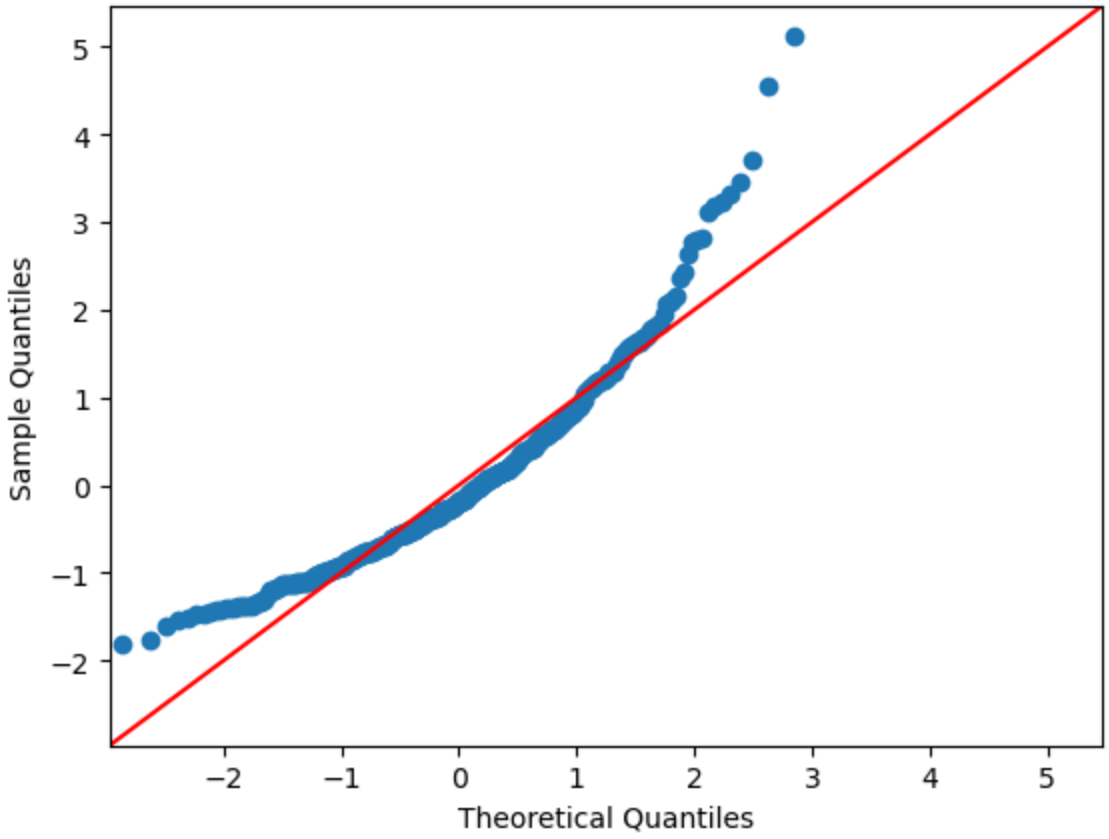
In [19]:
```
#normalize data
data = (x['sbp']-x['sbp'].mean())/x['sbp'].std()
sm.qqplot(data, line ='45')
py.show()
```



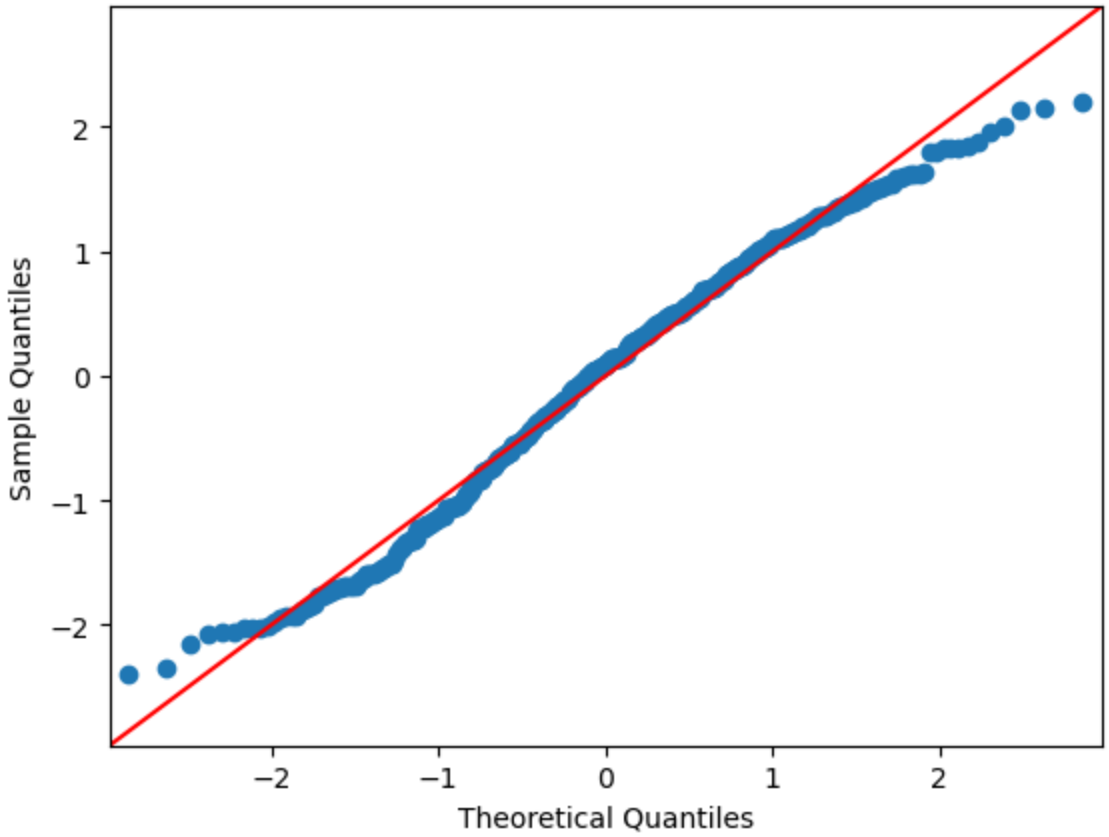
In [20]:
```
#normalize data
data = (x['tobacco']-x['tobacco'].mean())/x['tobacco'].std()
sm.qqplot(data, line ='45')
py.show()
```



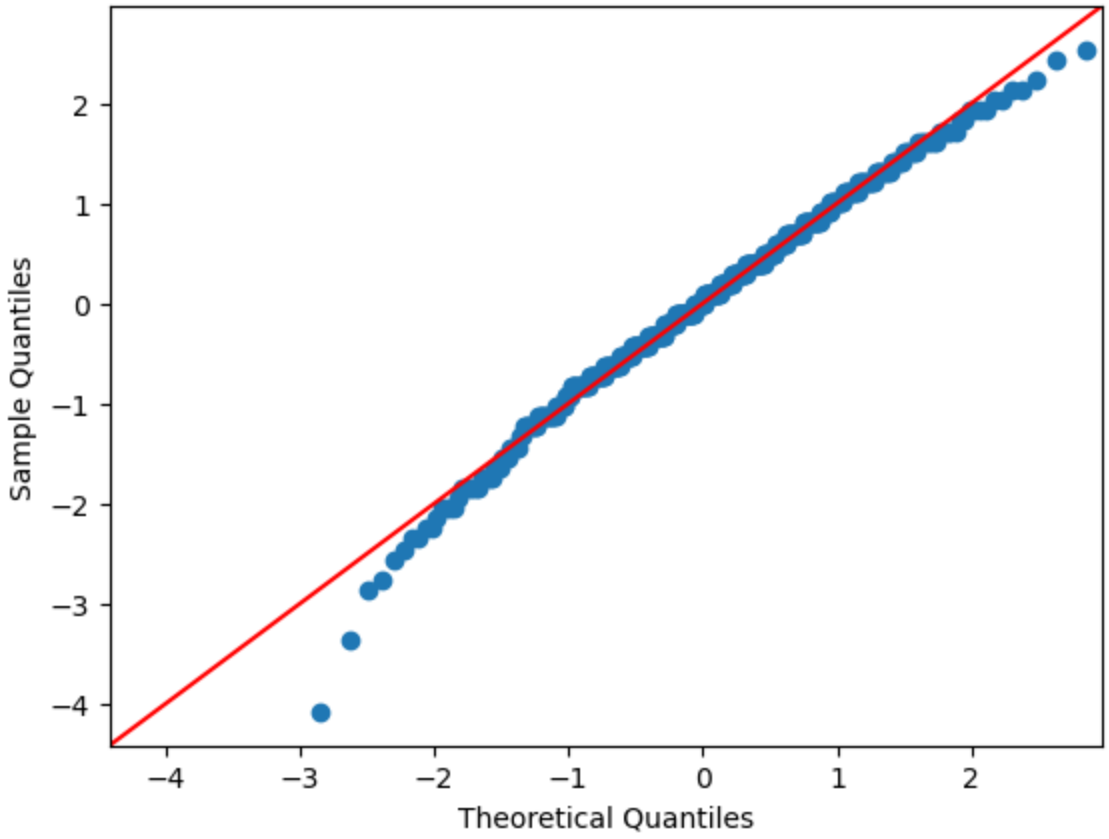
In [21]:
```
#normalize data
data = (x['ldl']-x['ldl'].mean())/x['ldl'].std()
sm.qqplot(data, line ='45')
py.show()
```



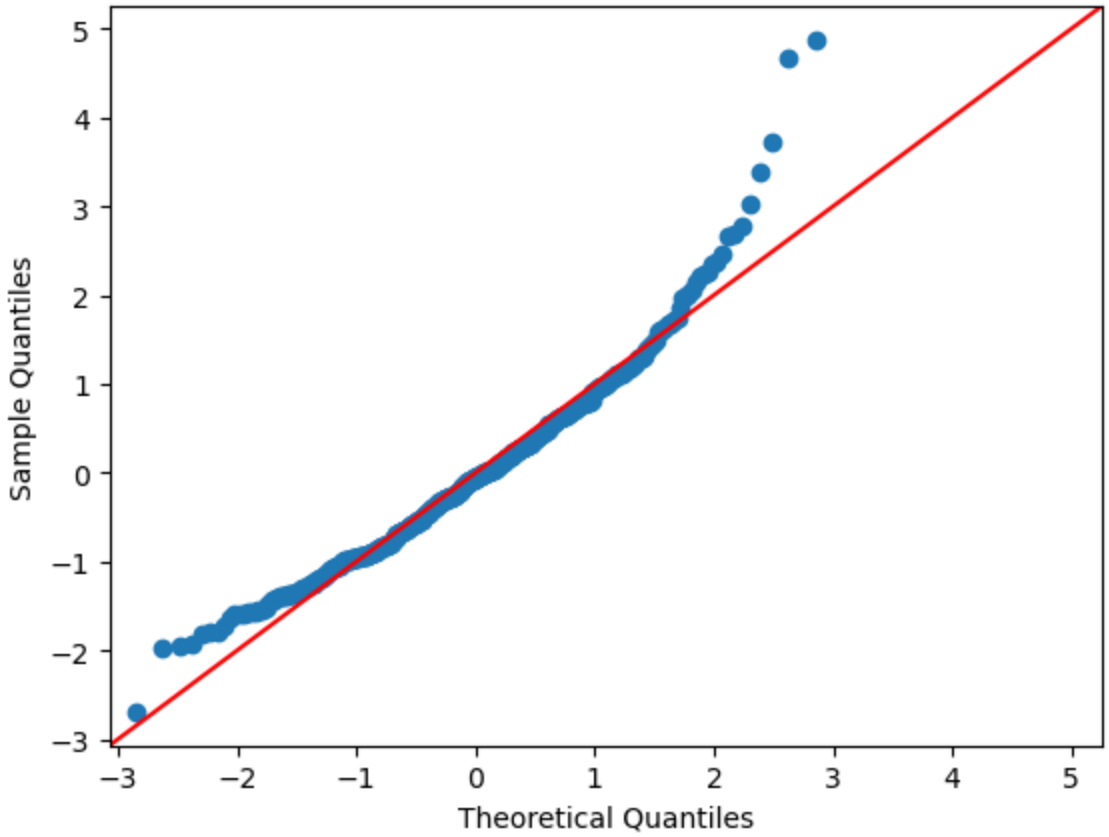
In [22]:
```
#normalize data
data = (x['adiposity']-x['adiposity'].mean())/x['adiposity'].std()
sm.qqplot(data, line ='45')
py.show()
```



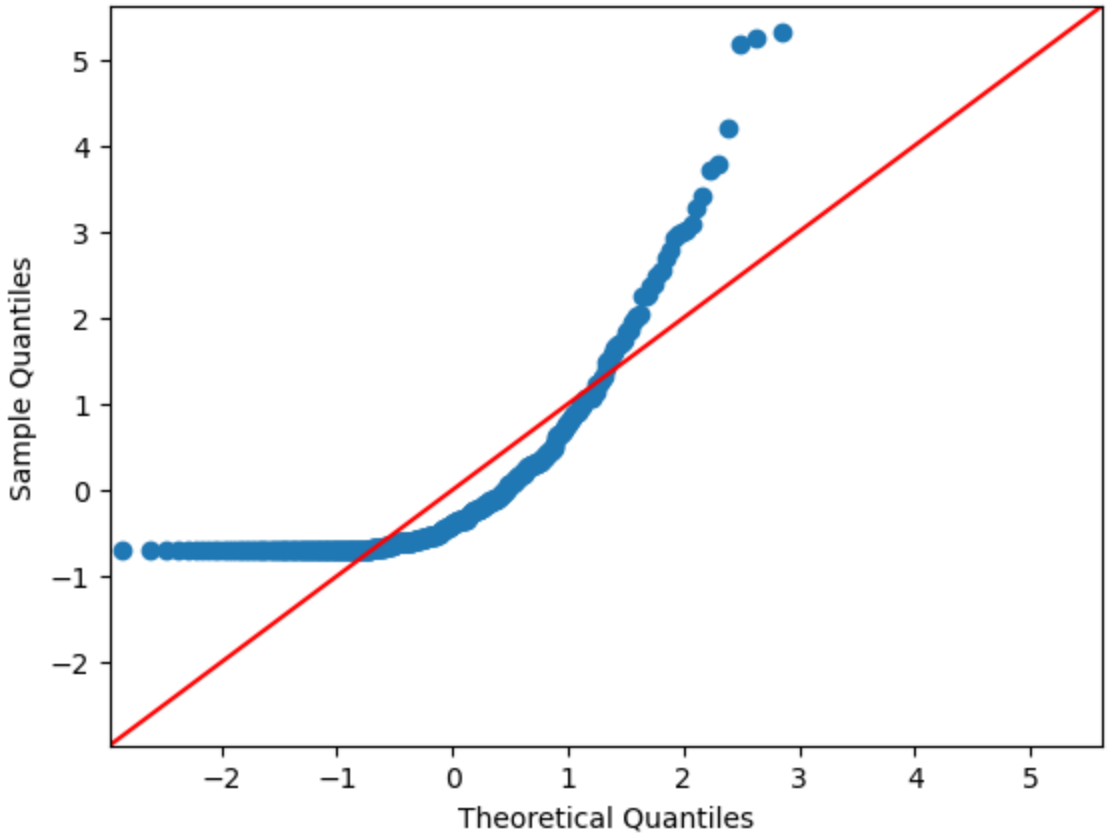
In [23]:
```
#normalize data
data = (x['typea']-x['typea'].mean())/x['typea'].std()
sm.qqplot(data, line ='45')
py.show()
```



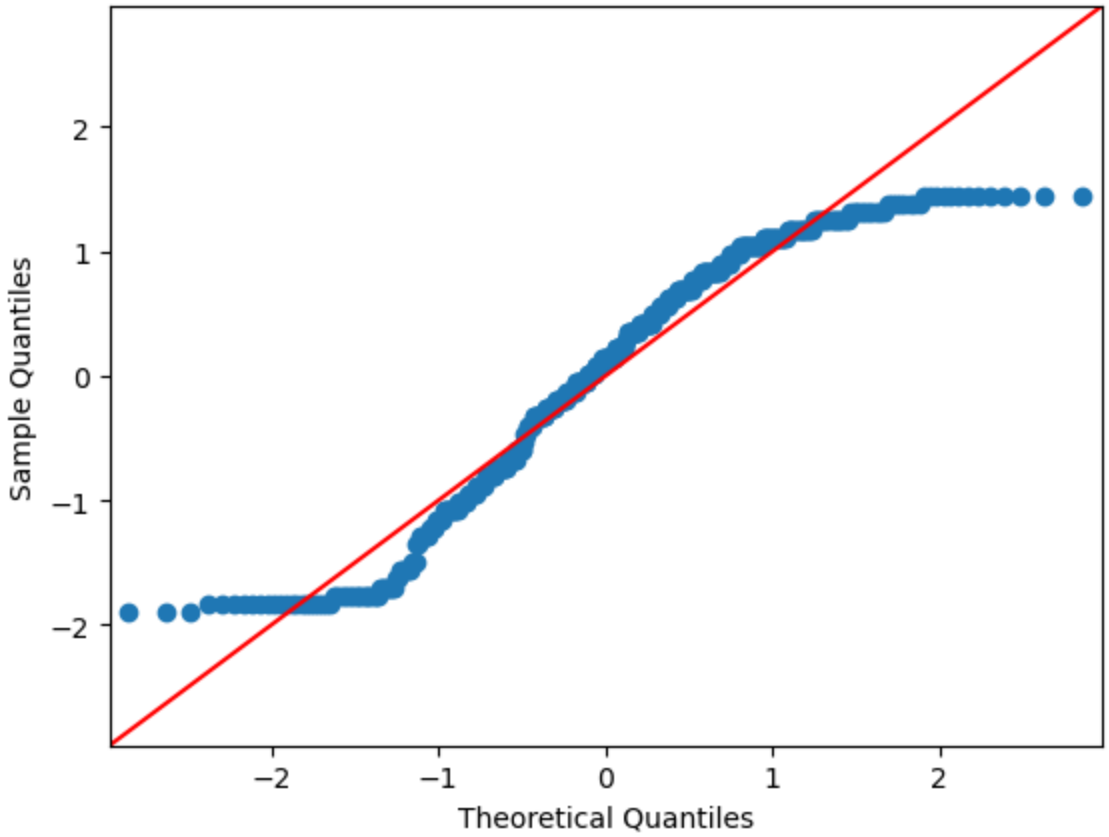
In [24]:
```
#normalize data
data = (x['obesity']-x['obesity'].mean())/x['obesity'].std()
sm.qqplot(data, line ='45')
py.show()
```



In [25]:
```
#normalize data
data = (x['alcohol']-x['alcohol'].mean())/x['alcohol'].std()
sm.qqplot(data, line ='45')
py.show()
```



In [26]:
```
#normalize data
data = (x['age']-x['age'].mean())/x['age'].std()
sm.qqplot(data, line ='45')
py.show()
```



5) Build a logistic regression model with all predictors.

In [27]:
```
from sklearn.linear_model import LogisticRegression
```

```
import statsmodels.api as sm
logit_model=sm.Logit(y,x)
result=logit_model.fit()
print(result.summary2())

params = result.params
conf = result.conf_int()
conf['Odds Ratio'] = params
conf.columns = ['5%', '95%', 'Odds Ratio']
print(np.exp(conf))

from sklearn.metrics import roc_auc_score
roc_auc_score(y, result.predict(x))
```

```
Optimization terminated successfully.
         Current function value: 0.533084
         Iterations 6
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.174
Dependent Variable: chd              AIC:              512.5694
Date:               2022-11-01 21:27 BIC:              553.9251
No. Observations:   462              Log-Likelihood:   -246.28
Df Model:           9                LL-Null:          -298.05
Df Residuals:       452              LLR p-value:      3.0212e-18
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
----------------------------------------------------------------
               Coef.   Std.Err.     z     P>|z|    [0.025   0.975]
----------------------------------------------------------------
names         -0.0016    0.0009  -1.8545  0.0637  -0.0033   0.0001
sbp           -0.0041    0.0052  -0.7963  0.4258  -0.0142   0.0060
tobacco        0.0834    0.0262   3.1843  0.0015   0.0321   0.1348
ldl            0.1566    0.0591   2.6511  0.0080   0.0408   0.2724
adiposity      0.0673    0.0277   2.4305  0.0151   0.0130   0.1216
typea          0.0087    0.0102   0.8500  0.3953  -0.0113   0.0287
obesity       -0.1765    0.0404  -4.3686  0.0000  -0.2557  -0.0973
alcohol        0.0004    0.0045   0.0796  0.9365  -0.0085   0.0092
age            0.0266    0.0109   2.4308  0.0151   0.0051   0.0480
famhist        0.9512    0.2252   4.2234  0.0000   0.5098   1.3926
================================================================

                 5%       95%  Odds Ratio
names      0.996749  1.000090    0.998418
sbp        0.985890  1.006017    0.995903
tobacco    1.032604  1.144314    1.087025
ldl        1.041665  1.313052    1.169513
adiposity  1.013117  1.129298    1.069631
typea      0.988743  1.029073    1.008706
obesity    0.774374  0.907265    0.838190
alcohol    0.991584  1.009210    1.000358
age        1.005157  1.049134    1.026910
famhist    1.664927  4.025447    2.588837
```

Out[27]:
```
0.7716887417218542
```

6) Perform power transformation on the following variables: sbp (power = -2), tobacco (power = 0.4), ldl (power = 0.1), obesity (power = -0.4), and alcohol (power = 0.4).

In [28]:
```
x['sbp'] = 1 / (x['sbp']**(2))
x['tobacco'] = (x['tobacco']**(0.4))
x['ldl'] = (x['ldl']**(0.1))
```

```
x['obesity'] = 1 / (x['obesity']**(0.4))
x['alcohol'] = (x['alcohol']**(0.4))

print(x[['sbp','tobacco','ldl','obesity','alcohol']])

print('\nsbp mean', x['sbp'].mean())
print('sbp median', x['sbp'].median())
print('sbp mode', x['sbp'].mode())
print('sbp skewness', x['sbp'].skew())

print('\ntobacco mean', x['tobacco'].mean())
print('tobacco median', x['tobacco'].median())
print('tobacco mode', x['tobacco'].mode())
print('tobacco skewness', x['tobacco'].skew())

print('\nldl mean', x['ldl'].mean())
print('ldl median', x['ldl'].median())
print('ldl mode', x['ldl'].mode())
print('ldl skewness', x['ldl'].skew())

print('\nobesity mean', x['obesity'].mean())
print('obesity median', x['obesity'].median())
print('obesity mode', x['obesity'].mode())
print('obesity skewness', x['obesity'].skew())

print('\nalcohol mean', x['alcohol'].mean())
print('alcohol median', x['alcohol'].median())
print('alcohol mode', x['alcohol'].mode())
print('alcohol skewness', x['alcohol'].skew())
```

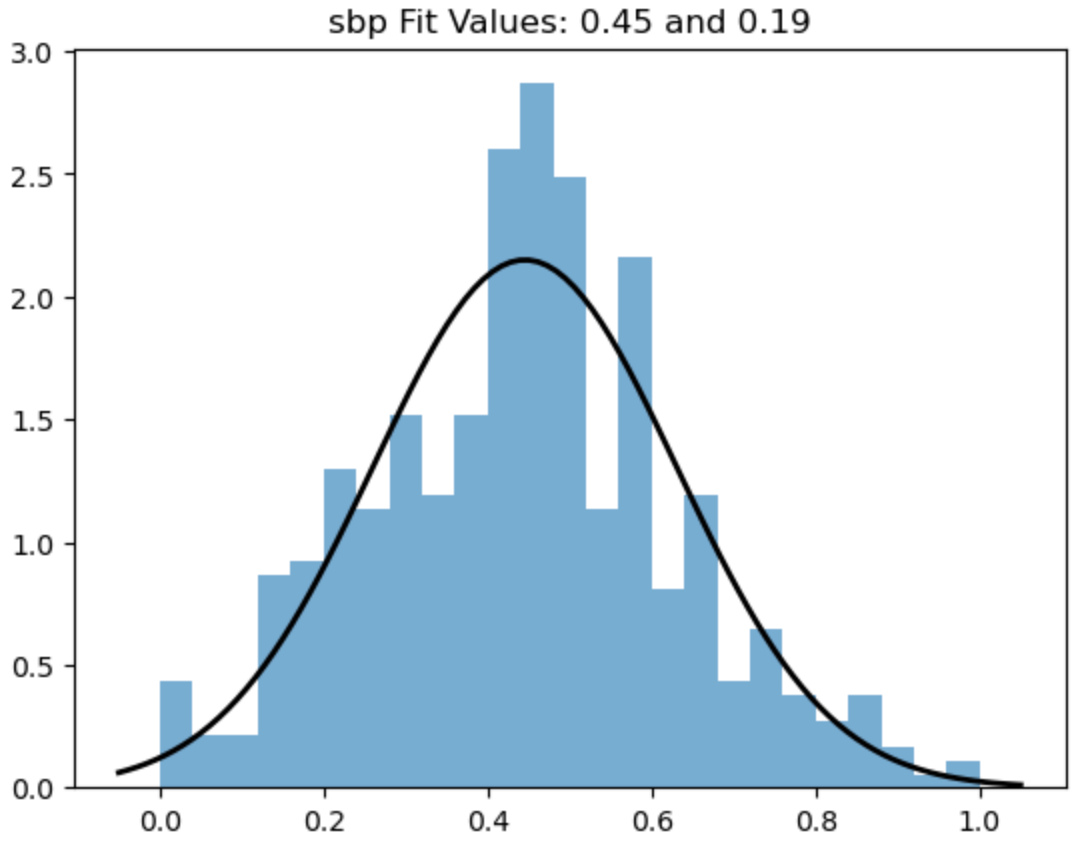
```
          sbp   tobacco       ldl   obesity   alcohol
0    0.000039  2.701920  1.190736  0.274632  6.238304
1    0.000048  0.158489  1.159962  0.260508  1.335202
2    0.000072  0.364113  1.132812  0.259540  1.707536
3    0.000035  2.238847  1.204164  0.250031  3.580604
4    0.000056  2.840636  1.133462  0.271692  5.051066
..        ...       ...       ...       ...       ...
457  0.000022  0.693145  1.195832  0.262040  0.000000
458  0.000030  1.775414  1.159962  0.261453  3.227917
459  0.000086  1.551846  1.047465  0.301167  3.717181
460  0.000072  1.963168  1.277860  0.266206  3.563422
461  0.000057  0.000000  1.170320  0.341250  0.000000

[462 rows x 5 columns]

sbp mean 5.532204943991282e-05
sbp median 5.569169079973268e-05
sbp mode 0    0.000054
1    0.000056
Name: sbp, dtype: float64
sbp skewness 0.06330375900307444

tobacco mean 1.2609696848977374
tobacco median 1.3195079107728942
tobacco mode 0    0.0
Name: tobacco, dtype: float64
tobacco skewness 0.13472830540878233

ldl mean 1.1590867107219038
ldl median 1.158107763468698
ldl mode 0    1.135708
1    1.147254
2    1.158906
Name: ldl, dtype: float64
ldl skewness 0.0287454615618759

obesity mean 0.2733598884063176
obesity median 0.27246985410716074
obesity mode 0    0.271275
1    0.276566
Name: obesity, dtype: float64
obesity skewness -0.00027314324477181024

alcohol mean 2.2635692905221894
alcohol median 2.239993374876412
alcohol mode 0    0.0
Name: alcohol, dtype: float64
alcohol skewness 0.40094492105179225
```

7) Produce histogram of each of the following transformed variables with imposing normal curve: sbp, tobacco, ldl, obesity, and alcohol.

In [29]:

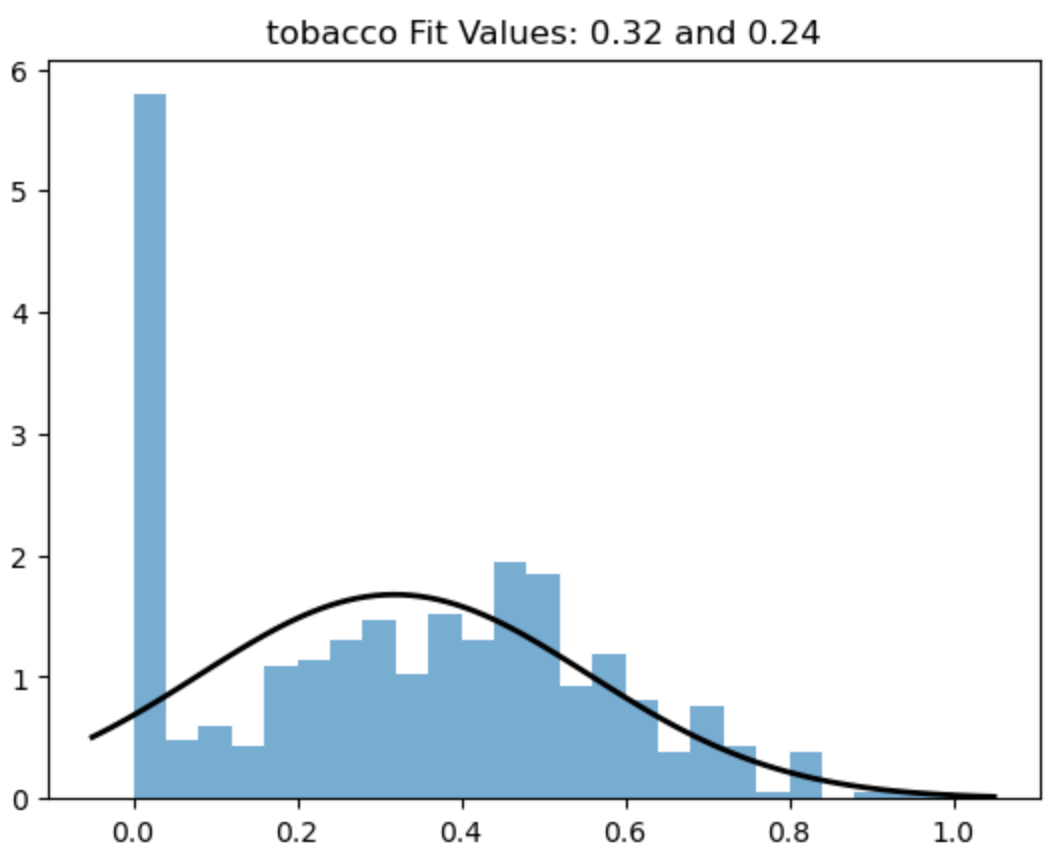
```
data = (x['sbp']-x['sbp'].min())/(x['sbp'].max()-x['sbp'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
```

```
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "sbp Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



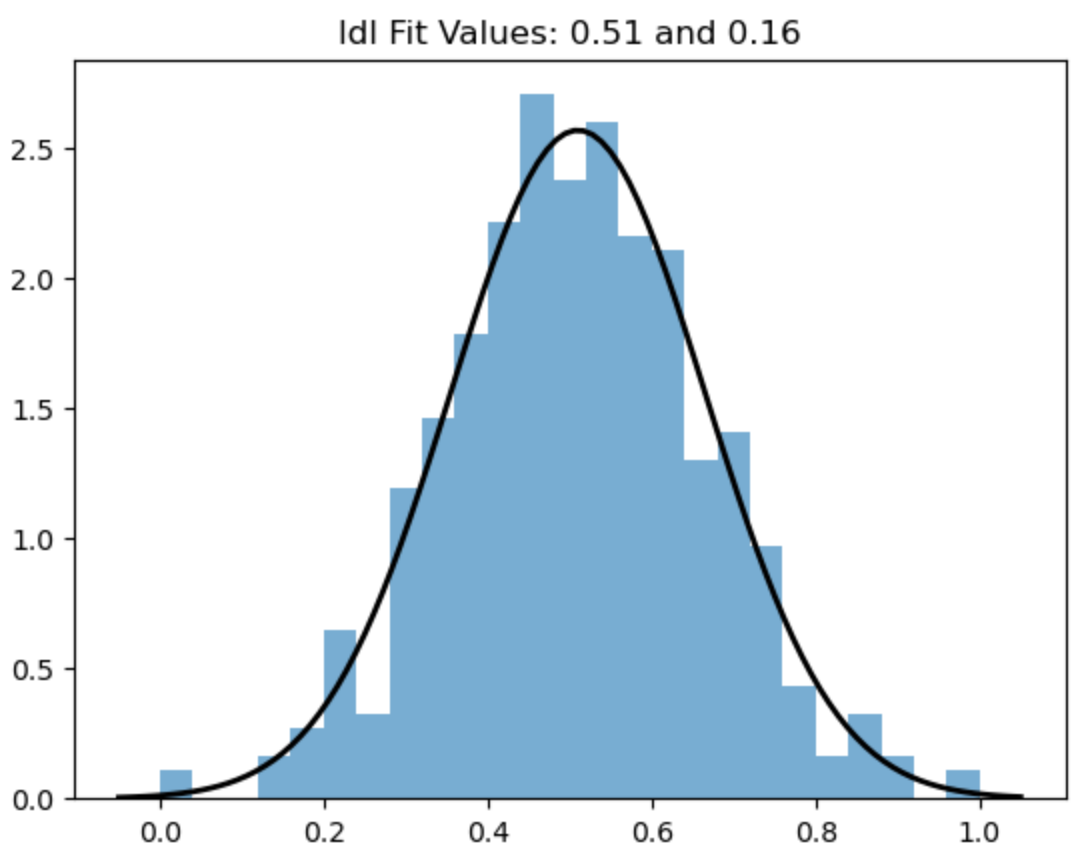
In [30]:

```
data = (x['tobacco']-x['tobacco'].min())/(x['tobacco'].max()-x['tobacco'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "tobacco Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



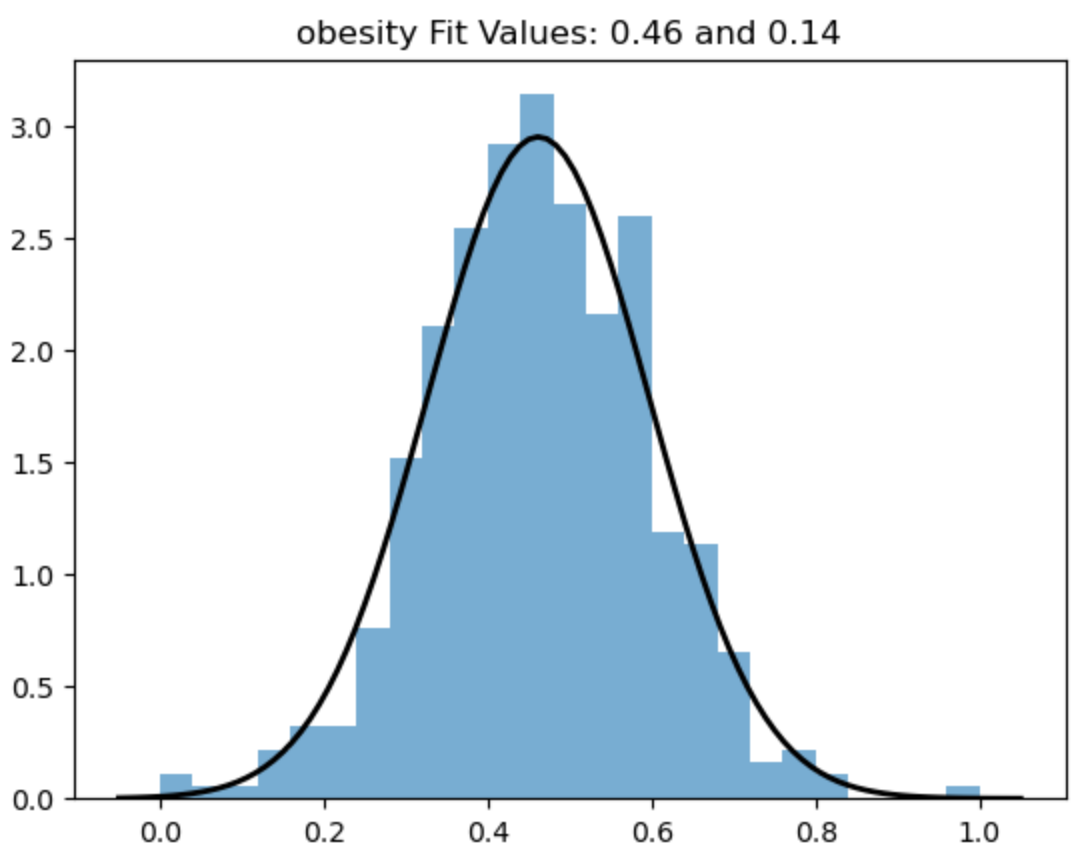
In [31]:

```
data = (x['ldl']-x['ldl'].min())/(x['ldl'].max()-x['ldl'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "ldl Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



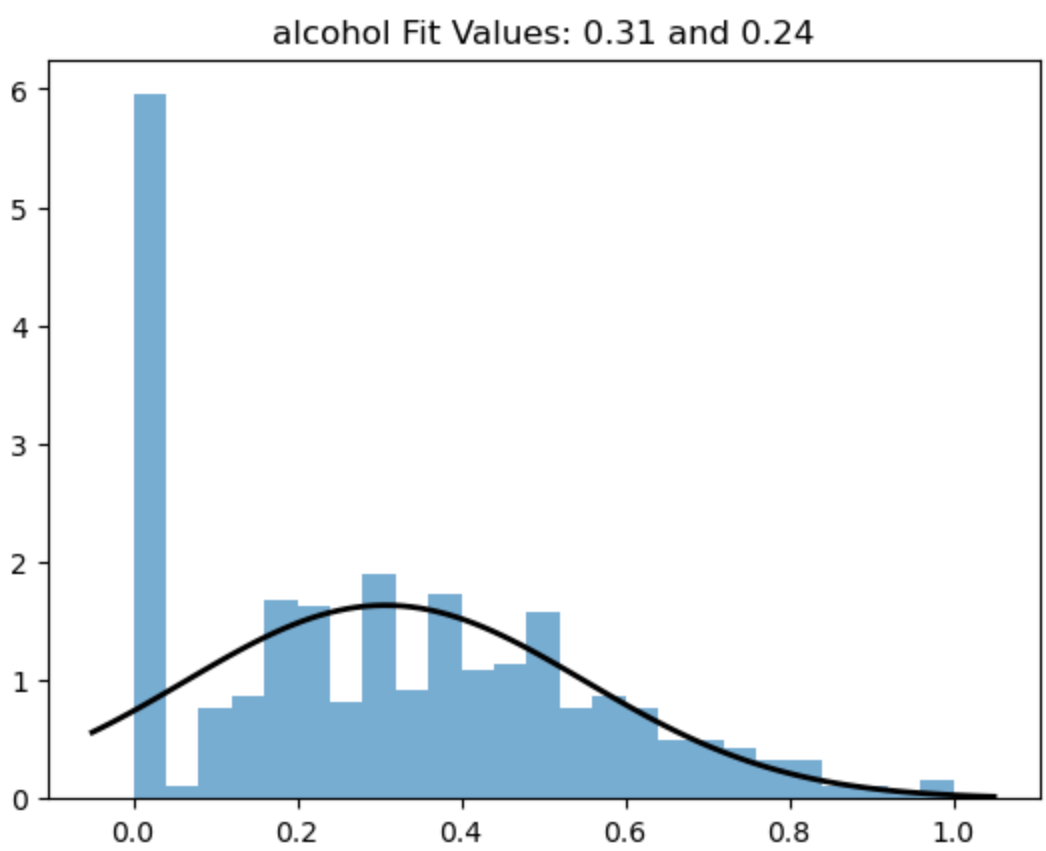
In [32]:

```
data = (x['obesity']-x['obesity'].min())/(x['obesity'].max()-x['obesity'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "obesity Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```



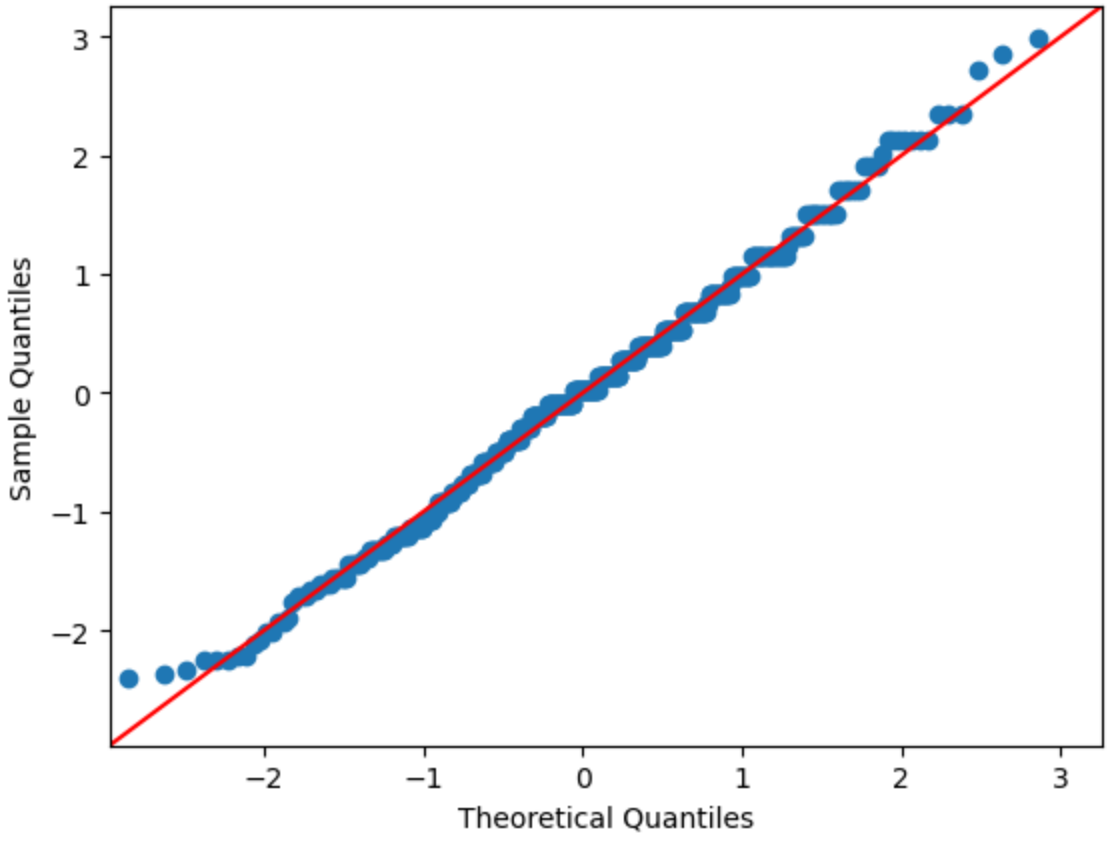
In [33]:

```
data = (x['alcohol']-x['alcohol'].min())/(x['alcohol'].max()-x['alcohol'].min())

# Fit a normal distribution to
# the data:
# mean and standard deviation
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=25, density=True, alpha=0.6)

# Plot the PDF.
xmin, xmax = plt.xlim()
x2 = np.linspace(xmin, xmax, 100)
p = norm.pdf(x2, mu, std)

plt.plot(x2, p, 'k', linewidth=2)
title = "alcohol Fit Values: {:.2f} and {:.2f}".format(mu, std)
plt.title(title)

plt.show()
```

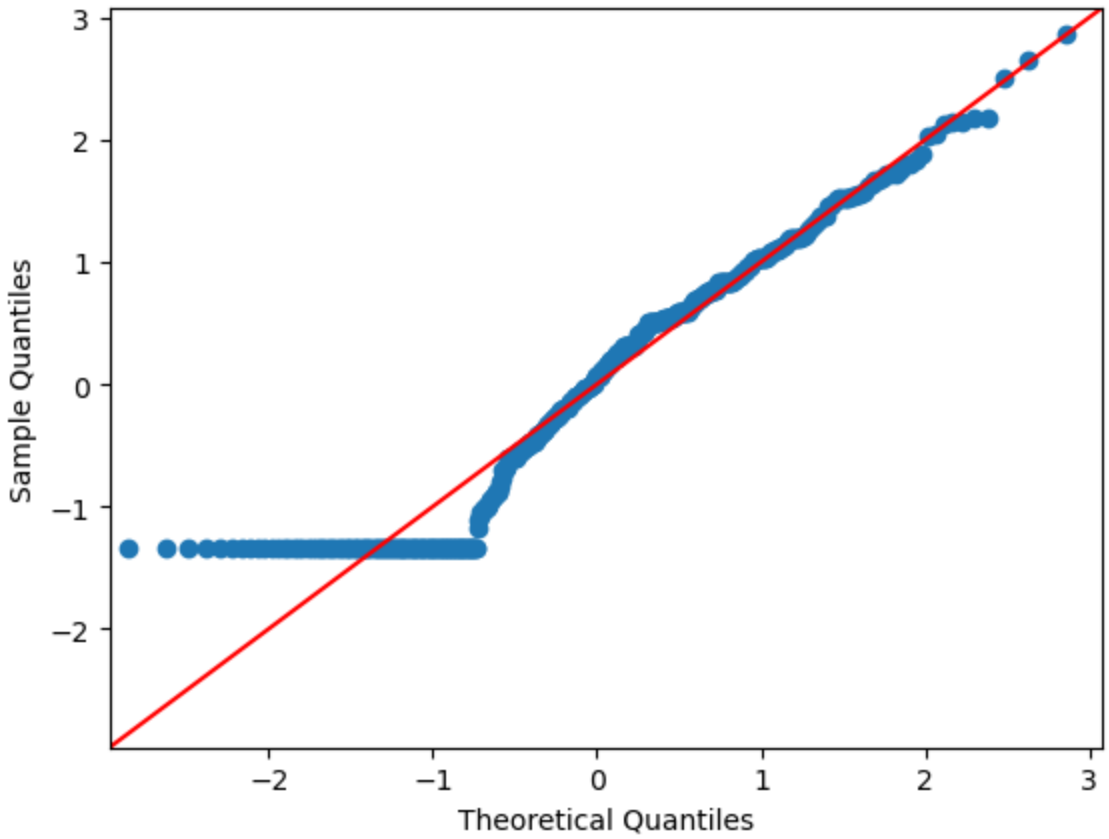


8) Produce quantile plot of each of the following transformed variables: sbp, tobacco, ldl, obesity, and alcohol.

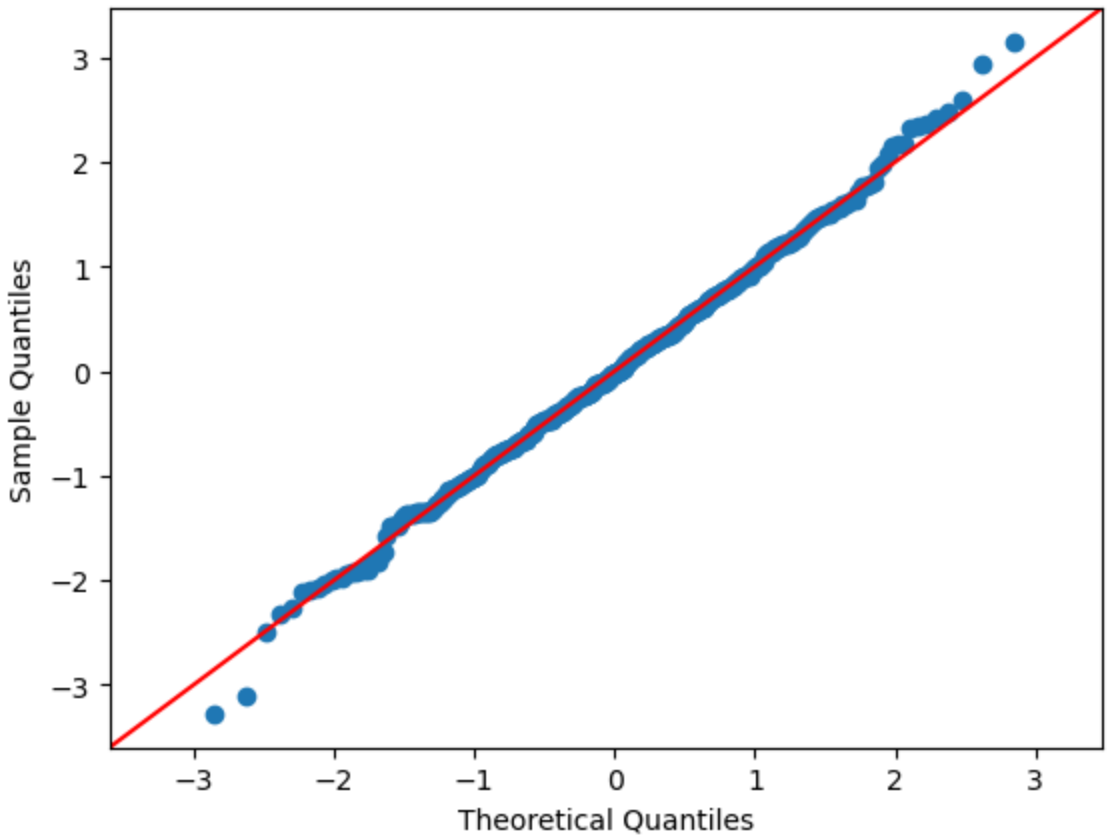
In [34]:
```
#normalize data
data = (x['sbp']-x['sbp'].mean())/x['sbp'].std()
sm.qqplot(data, line ='45')
py.show()
```



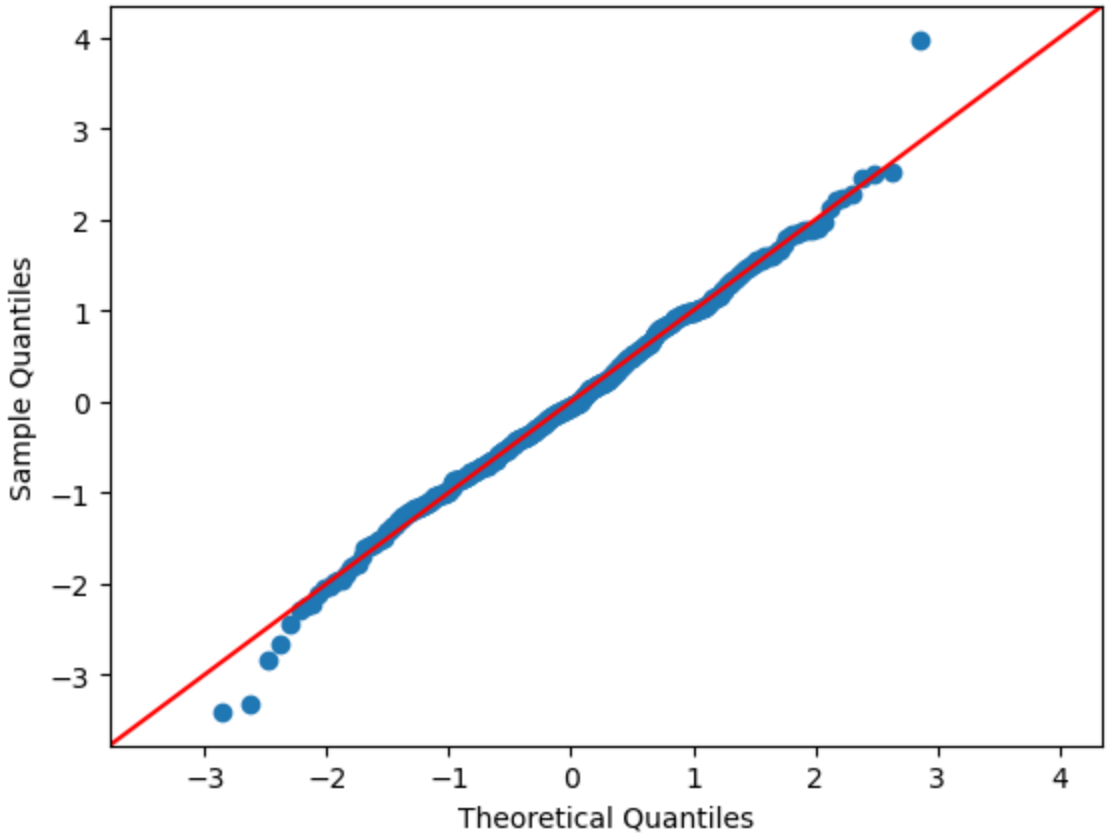
In [35]:
```
#normalize data
data = (x['tobacco']-x['tobacco'].mean())/x['tobacco'].std()
sm.qqplot(data, line ='45')
py.show()
```



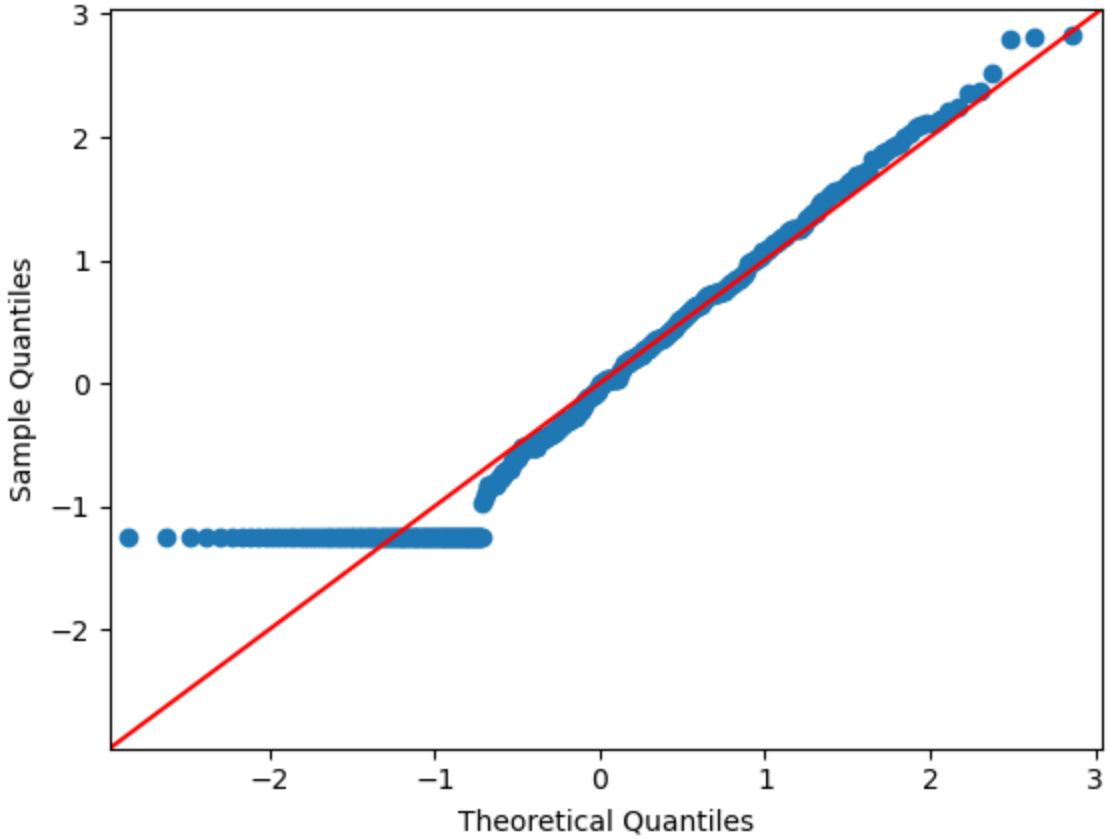
In [36]:
```
#normalize data
data = (x['ldl']-x['ldl'].mean())/x['ldl'].std()
sm.qqplot(data, line ='45')
py.show()
```



In [37]:
```
#normalize data
data = (x['obesity']-x['obesity'].mean())/x['obesity'].std()
sm.qqplot(data, line ='45')
py.show()
```



In [38]:
```
#normalize data
data = (x['alcohol']-x['alcohol'].mean())/x['alcohol'].std()
sm.qqplot(data, line ='45')
py.show()
```



9) Build a logistic regression model with all predictors (transformed and three remaining original variables do not perform any transformation).

In [39]:
```
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
logit_model=sm.Logit(y,x)
result=logit_model.fit()

print(result.summary2())
from sklearn.metrics import roc_auc_score
roc_auc_score(y, result.predict(x))
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 5.512707
         Iterations: 35
                            Results: Logit
==========================================================================
Model:                  Logit               Pseudo R-squared:    -7.545
Dependent Variable:     chd                 AIC:                 5113.7409
Date:                   2022-11-01 21:27    BIC:                 5155.0965
No. Observations:       462                 Log-Likelihood:      -2546.9
Df Model:               9                   LL-Null:             -298.05
Df Residuals:           452                 LLR p-value:         1.0000
Converged:              0.0000              Scale:               1.0000
No. Iterations:         35.0000
--------------------------------------------------------------------------
             Coef.       Std.Err.     z    P>|z|     [0.025      0.975]
--------------------------------------------------------------------------
names           0.0134      0.0036  3.7624 0.0002      0.0064       0.0204
sbp       1274525.2191 257129.5210  4.9567 0.0000 770560.6185 1778489.8196
tobacco         4.8739      1.0906  4.4688 0.0000      2.7363       7.0115
ldl           -29.7285      9.2654 -3.2086 0.0013    -47.8883     -11.5687
adiposity      -0.2110      0.1040 -2.0281 0.0426     -0.4149      -0.0071
typea           0.1969      0.0563  3.4950 0.0005      0.0865       0.3073
obesity      -383.7912     83.6670 -4.5871 0.0000   -547.7755    -219.8070
alcohol         0.3914      0.2316  1.6903 0.0910     -0.0624       0.8453
age             0.9850      0.2066  4.7680 0.0000      0.5801       1.3899
famhist         9.3157      1.9718  4.7245 0.0000      5.4510      13.1803
==========================================================================
```

```
C:\Users\danma\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Out[39]:
```
0.6427152317880794
```

10) Build another logistic regression model with all predictors as in Part 9 except using significant predictors only.

In [41]:
```
print("alcohol had a p-value higher than 0.05 therefore was removed from the model.")
x_new = x.loc[:, x.columns != 'alcohol']

from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
logit_model=sm.Logit(y,x_new)
result=logit_model.fit()

print(result.summary2())
from sklearn.metrics import roc_auc_score
roc_auc_score(y, result.predict(x_new))
```

```
alcohol had a p-value higher than 0.05 therefore was removed from the model.
Warning: Maximum number of iterations has been exceeded.
         Current function value: 5.507143
         Iterations: 35
                            Results: Logit
================================================================================
Model:                  Logit               Pseudo R-squared:    -7.536
Dependent Variable:     chd                 AIC:                 5106.6005
Date:                   2022-11-01 21:29    BIC:                 5143.8206
No. Observations:       462                 Log-Likelihood:      -2544.3
Df Model:               8                   LL-Null:             -298.05
Df Residuals:           453                 LLR p-value:         1.0000
Converged:              0.0000              Scale:               1.0000
No. Iterations:         35.0000
--------------------------------------------------------------------------------
              Coef.       Std.Err.     z    P>|z|     [0.025      0.975]
--------------------------------------------------------------------------------
names           0.0145      0.0037  3.9390 0.0001      0.0073       0.0217
sbp       1272406.2402 246202.7002  5.1681 0.0000 789857.8148 1754954.6656
tobacco         4.7808      1.0060  4.7525 0.0000      2.8091       6.7524
ldl           -32.3842      9.5933 -3.3757 0.0007    -51.1868     -13.5817
adiposity      -0.1749      0.0994 -1.7591 0.0786     -0.3698       0.0200
typea           0.1973      0.0550  3.5843 0.0003      0.0894       0.3052
obesity      -370.5931     76.9381 -4.8168 0.0000   -521.3891    -219.7971
age             0.9693      0.1939  4.9984 0.0000      0.5892       1.3494
famhist         9.4535      1.9232  4.9154 0.0000      5.6840      13.2230
================================================================================
```

```
C:\Users\danma\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Out[41]:

```
0.6401386589403973
```