Question no 7

In [89]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
```

In [11]:
```
df = pd.read_csv("Q7.csv")
```

In [12]:
```
df.head()
```

Out[12]:

	Unnamed: 0	Points	Score	Weigh
0	Mazda RX4	3.90	2.620	16.46
1	Mazda RX4 Wag	3.90	2.875	17.02
2	Datsun 710	3.85	2.320	18.61
3	Hornet 4 Drive	3.08	3.215	19.44
4	Hornet Sportabout	3.15	3.440	17.02

In [13]:
```
df.dtypes
```

Out[13]:
```
Unnamed: 0     object
Points        float64
Score         float64
Weigh         float64
dtype: object
```

In [14]:
```
df.mean()
```

```
/var/folders/9_/ckpgdd3s4qzg3w1zytsfvsmh0000gn/T/ipykernel_24091/3
698961737.py:1: FutureWarning: Dropping of nuisance columns in Dat
aFrame reductions (with 'numeric_only=None') is deprecated; in a f
uture version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.mean()
```

Out[14]:
```
Points     3.596563
Score      3.217250
Weigh     17.848750
dtype: float64
```

In [15]:
```
df.mode()
```

Out[15]:

	Unnamed: 0	Points	Score	Weigh
0	AMC Javelin	3.07	3.44	17.02

1	Cadillac Fleetwood	3.92	NaN	18.90
2	Camaro Z28	NaN	NaN	NaN
3	Chrysler Imperial	NaN	NaN	NaN
4	Datsun 710	NaN	NaN	NaN
5	Dodge Challenger	NaN	NaN	NaN
6	Duster 360	NaN	NaN	NaN
7	Ferrari Dino	NaN	NaN	NaN
8	Fiat 128	NaN	NaN	NaN
9	Fiat X1-9	NaN	NaN	NaN
10	Ford Pantera L	NaN	NaN	NaN
11	Honda Civic	NaN	NaN	NaN
12	Hornet 4 Drive	NaN	NaN	NaN
13	Hornet Sportabout	NaN	NaN	NaN
14	Lincoln Continental	NaN	NaN	NaN
15	Lotus Europa	NaN	NaN	NaN
16	Maserati Bora	NaN	NaN	NaN
17	Mazda RX4	NaN	NaN	NaN
18	Mazda RX4 Wag	NaN	NaN	NaN
19	Merc 230	NaN	NaN	NaN
20	Merc 240D	NaN	NaN	NaN
21	Merc 280	NaN	NaN	NaN
22	Merc 280C	NaN	NaN	NaN
23	Merc 450SE	NaN	NaN	NaN
24	Merc 450SL	NaN	NaN	NaN
25	Merc 450SLC	NaN	NaN	NaN
26	Pontiac Firebird	NaN	NaN	NaN
27	Porsche 914-2	NaN	NaN	NaN
28	Toyota Corolla	NaN	NaN	NaN
29	Toyota Corona	NaN	NaN	NaN
30	Valiant	NaN	NaN	NaN
31	Volvo 142E	NaN	NaN	NaN

In [16]:
```
df.median()
```

```
/var/folders/9_/ckpgdd3s4qzg3w1zytsfvsmh0000gn/T/ipykernel_24091/5
30051474.py:1: FutureWarning: Dropping of nuisance columns in Data
Frame reductions (with 'numeric_only=None') is deprecated; in a fu
ture version this will raise TypeError.  Select only valid columns
before calling the reduction.
  df.median()
```

Out[16]:
```
Points     3.695
Score      3.325
Weigh     17.710
dtype: float64
```

In [17]:
```
df.var()
```

```
/var/folders/9_/ckpgdd3s4qzg3w1zytsfvsmh0000gn/T/ipykernel_24091/1
568254755.py:1: FutureWarning: Dropping of nuisance columns in Dat
aFrame reductions (with 'numeric_only=None') is deprecated; in a f
uture version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.var()
```

Out[17]:
```
Points    0.285881
Score     0.957379
Weigh     3.193166
dtype: float64
```

In [18]:
```
df.std()
```

```
/var/folders/9_/ckpgdd3s4qzg3w1zytsfvsmh0000gn/T/ipykernel_24091/3
390915376.py:1: FutureWarning: Dropping of nuisance columns in Dat
aFrame reductions (with 'numeric_only=None') is deprecated; in a f
uture version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.std()
```

Out[18]:
```
Points    0.534679
Score     0.978457
Weigh     1.786943
dtype: float64
```

Question no 9

In [36]:
```
Q9a = pd.read_csv('Q9_a.csv')
Q9a
```

Out[36]:

	Index	speed	dist
0	1	4	2
1	2	4	10

```
 2   3   7   4
 3   4   7  22
 4   5   8  16
 5   6   9  10
 6   7  10  18
 7   8  10  26
 8   9  10  34
 9  10  11  17
10  11  11  28
11  12  12  14
12  13  12  20
13  14  12  24
14  15  12  28
15  16  13  26
16  17  13  34
17  18  13  34
18  19  13  46
19  20  14  26
20  21  14  36
21  22  14  60
22  23  14  80
23  24  15  20
24  25  15  26
25  26  15  54
26  27  16  32
27  28  16  40
28  29  17  32
29  30  17  40
30  31  17  50
31  32  18  42
32  33  18  56
33  34  18  76
34  35  18  84
```

```
35   36   19   36
36   37   19   46
37   38   19   68
38   39   20   32
39   40   20   48
40   41   20   52
41   42   20   56
42   43   20   64
43   44   22   66
44   45   23   54
45   46   24   70
46   47   24   92
47   48   24   93
48   49   24  120
49   50   25   85
```

In [37]:

```
q9aa= pd.read_csv('Q9_a.csv', usecols=[1,2])
q9aa
```

Out[37]:

```
    speed  dist
 0      4     2
 1      4    10
 2      7     4
 3      7    22
 4      8    16
 5      9    10
 6     10    18
 7     10    26
 8     10    34
 9     11    17
10     11    28
11     12    14
12     12    20
13     12    24
14     12    28
```

15	13	26
16	13	34
17	13	34
18	13	46
19	14	26
20	14	36
21	14	60
22	14	80
23	15	20
24	15	26
25	15	54
26	16	32
27	16	40
28	17	32
29	17	40
30	17	50
31	18	42
32	18	56
33	18	76
34	18	84
35	19	36
36	19	46
37	19	68
38	20	32
39	20	48
40	20	52
41	20	56
42	20	64
43	22	66
44	23	54
45	24	70
46	24	92
47	24	93
48	24	120

49	25	85

In [38]:
```
print(q9aa.skew())
```

```
speed   -0.117510
dist     0.806895
dtype: float64
```

In [39]:
```
print(q9aa.kurt())
```

```
speed   -0.508994
dist     0.405053
dtype: float64
```

In [40]:
```
Q9b=pd.read_csv('Q9_b.csv')
Q9b
```

Out[40]:

	Unnamed: 0	SP	WT
0	1	104.185353	28.762059
1	2	105.461264	30.466833
2	3	105.461264	30.193597
3	4	113.461264	30.632114
4	5	104.461264	29.889149
...	...	...	...
76	77	169.598513	16.132947
77	78	150.576579	37.923113
78	79	151.598513	15.769625
79	80	167.944460	39.423099
80	81	139.840817	34.948615

81 rows × 3 columns

In [41]:
```
Q9bb=pd.read_csv('Q9_b.csv', usecols=[1,2])
Q9bb
```

Out[41]:

	SP	WT
0	104.185353	28.762059
1	105.461264	30.466833
2	105.461264	30.193597
3	113.461264	30.632114
4	104.461264	29.889149
...	...	...
76	169.598513	16.132947
77	150.576579	37.923113
78	151.598513	15.769625
79	167.944460	39.423099
80	139.840817	34.948615

81 rows × 2 columns

In [43]:
```
np.round(Q9bb.SP.skew(), 2)
```

Out[43]:
```
1.61
```

In [44]:
```
np.round(Q9bb.SP.kurt(), 2)
```

Out[44]:
```
2.98
```

Question no.8

In [46]:
```
l=[108, 110, 123, 134, 135, 145, 167, 187, 199]
np.average(l)
```

Out[46]:
```
145.33333333333334
```

Question no. 11

In [47]:
```
from scipy import stats
```

```
In [48]: AVG_WGT1 = stats.norm.interval(0.97, loc = 200, scale = 30)
         print('Average weight of adult in Mexico at 94% confidence interval
```

```
Average weight of adult in Mexico at 94% confidence interval [134.
897 265.103]
```

```
In [49]: AVG_WGT2 = stats.norm.interval(0.99, loc = 200, scale = 30)
         print('Average weight of adult in Mexico at 98% confidence interval
```

```
Average weight of adult in Mexico at 98% confidence interval [122.
725 277.275]
```

```
In [50]: AVG_WGT3 = stats.norm.interval(0.98, loc = 200, scale = 30)
         print('Average weight of adult in Mexico at 96% confidence interval
```

```
Average weight of adult in Mexico at 96% confidence interval [130.
21 269.79]
```

Question no. 12

```
In [51]: df = [34,36,36,38,38,39,39,40,40,41,41,41,41,42,42,45,49,56]
         df = pd.DataFrame(df)
```

```
In [52]: df.mean()
```

```
Out[52]: 0    41.0
         dtype: float64
```

```
In [53]: df.median()
```

```
Out[53]: 0    40.5
         dtype: float64
```

```
In [54]: df.std()
```

```
Out[54]: 0    5.052664
         dtype: float64
```

```
In [55]: df.var()
```

```
Out[55]: 0    25.529412
         dtype: float64
```

Question no. 20

```
In [56]: import pandas as pd
```

In [58]:
```
q20=pd.read_csv("Cars.csv")
q20
```

Out[58]:

	HP	MPG	VOL	SP	WT
0	49	53.700681	89	104.185353	28.762059
1	55	50.013401	92	105.461264	30.466833
2	55	50.013401	92	105.461264	30.193597
3	70	45.696322	92	113.461264	30.632114
4	53	50.504232	92	104.461264	29.889149
...	...	...	...	...	...
76	322	36.900000	50	169.598513	16.132947
77	238	19.197888	115	150.576579	37.923113
78	263	34.000000	50	151.598513	15.769625
79	295	19.833733	119	167.944460	39.423099
80	236	12.101263	107	139.840817	34.948615

81 rows × 5 columns

In [60]:
```
q20.mean()
```

Out[60]:
```
HP       117.469136
MPG       34.422076
VOL       98.765432
SP       121.540272
WT        32.412577
dtype: float64
```

In [61]:
```
q20.std()
```

Out[61]:
```
HP       57.113502
MPG       9.131445
VOL      22.301497
SP       14.181432
WT        7.492813
dtype: float64
```

In [63]:
```
import scipy.stats as ss
```

In [64]:
```
round(1 - ss.norm.cdf(38, 34.422076,9.131445), 4)
```

Out[64]:
```
0.3476
```

In [65]:
```
round(ss.norm.cdf(40, 34.422076,9.131445), 4)
```

Out[65]:
```
0.7293
```

In [66]:
```
round(ss.norm.cdf(50, 34.422076,9.131445) - (1 - ss.norm.cdf(20, 34
```

Out[66]:
```
0.0131
```

Question no. 21

In [79]:
```
import matplotlib.pyplot as plt
import scipy.stats as ss
import pylab
import statsmodels.api as sm
from scipy.stats import norm
```

In [80]:
```
Qn21 = pd.read_csv('Cars.csv')
Qn21
```

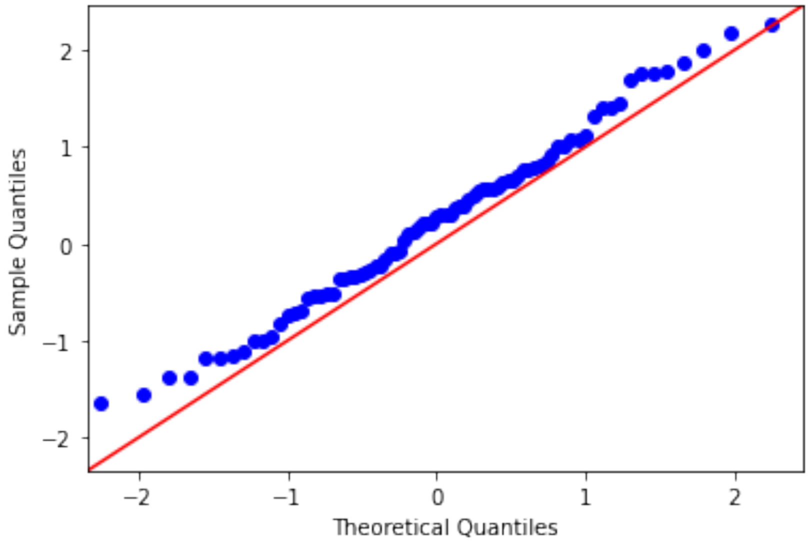
Out[80]:

	HP	MPG	VOL	SP	WT
0	49	53.700681	89	104.185353	28.762059
1	55	50.013401	92	105.461264	30.466833
2	55	50.013401	92	105.461264	30.193597
3	70	45.696322	92	113.461264	30.632114
4	53	50.504232	92	104.461264	29.889149
...	...	...	...	...	...
76	322	36.900000	50	169.598513	16.132947
77	238	19.197888	115	150.576579	37.923113
78	263	34.000000	50	151.598513	15.769625
79	295	19.833733	119	167.944460	39.423099
80	236	12.101263	107	139.840817	34.948615

81 rows × 5 columns

In [81]:
```
Qn21.MPG = norm.rvs(size=81)
sm.qqplot(Qn21.MPG, line='45')
pylab.show()
```

```
/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/go
fplots.py:993: UserWarning: marker is redundantly defined by the '
marker' keyword argument and the fmt string "bo" (-> marker='o').
The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



In [82]:
```
from scipy.stats import shapiro
```

In [83]:
```
Qn21.MPG = norm.rvs(size=81)
ss.shapiro(Qn21.MPG)
```

Out[83]:
```
ShapiroResult(statistic=0.9838477969169617, pvalue=0.4018413424491
8823)
```

In [86]:

```
Qn21a = pd.read_csv('wc-at.csv')
Qn21a
```

Out[86]:

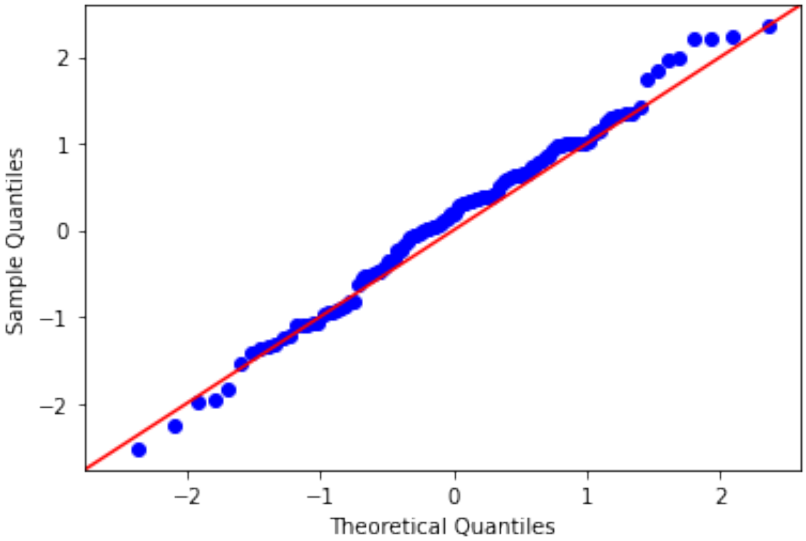
	Waist	AT
0	74.75	25.72
1	72.60	25.89
2	81.80	42.60
3	83.95	42.80
4	74.65	29.84
...	...	...
104	100.10	124.00
105	93.30	62.20
106	101.80	133.00
107	107.90	208.00
108	108.50	208.00

109 rows × 2 columns

In [87]:

```
Qn21a.AT = norm.rvs(size=109)
sm.qqplot(Qn21a.AT, line = '45')
pylab.show()
```

```
/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/go
fplots.py:993: UserWarning: marker is redundantly defined by the '
marker' keyword argument and the fmt string "bo" (-> marker='o').
The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



```
In [88]: Qn21a.AT = norm.rvs(size=109)
         ss.shapiro(Qn21a.AT)

Out[88]: ShapiroResult(statistic=0.9915087819099426, pvalue=0.7359000444412
         231)
```

Question no 22

```
In [69]: import scipy.stats as st
```

```
In [70]: def zscore(x):
             y=(1-x)/2
             s = st.norm.ppf(1-y)
             print(s)
```

```
In [71]: zscore(0.94)

         1.8807936081512509
```

```
In [72]: zscore(0.6)

         0.8416212335729143
```

```
In [73]: zscore(0.9)

         1.6448536269514722
```

Question no. 23

```
In [74]: import scipy.stats as st
```

```
In [75]: def tscore (x,s):
             y = (1-x)/2
             ts = st.t.ppf(1-y,s-1)
             print(ts)
```

```
In [76]: tscore(.95, 25)

         2.0638985616280205
```

```
In [77]: tscore(.96, 25)

         2.1715446760080677
```

```
In [78]: tscore(.99, 25)

         2.796939504772804

In [ ]:
```