asktalos-intership-part-2-eta

September 4, 2023

[1]:
```
#importing the file using pandas
```

[2]:
```
import pandas as pd
```

[3]:
```
dataset = pd.read_csv('Material Compressive Strength Experimental Data.csv')
```

[4]:
```
dataset
```

[4]:

	Material Quantity (gm)	Additive Catalyst (gm)	Ash Component (gm)	\
0	486.42	180.60	21.26	
1	133.32	260.14	185.60	
2	559.97	2.84	111.76	
3	391.43	351.05	76.39	
4	394.78	352.61	194.35	
…	…	…	…	
6134	188.78	162.30	142.65	
6135	349.87	291.45	77.82	
6136	358.29	22.70	17.99	
6137	445.25	275.59	178.86	
6138	560.23	266.56	167.14	

	Water Mix (ml)	Plasticizer (gm)	Moderate Aggregator	\
0	201.66	16.11	1151.17	
1	175.99	6.27	1090.57	
2	295.23	11.95	1024.93	
3	299.14	19.00	1134.88	
4	235.54	17.02	1098.24	
…	…	…	…	
6134	163.66	15.98	1003.82	
6135	188.26	25.82	925.10	
6136	208.58	34.91	1081.07	
6137	191.77	18.07	865.15	
6138	175.49	10.63	1165.87	

	Refined Aggregator	Formulation Duration (hrs)	Compression Strength MPa
0	708.50	344.43	79.89
1	1010.25	28.86	59.80

```
2                 810.69                    237.68                    77.86
3                 881.34                    208.81                    71.74
4                 781.01                    266.84                    76.07
…                    …                         …                        …
6134             1002.47                    357.91                    50.61
6135             1005.31                    104.20                    54.24
6136              792.44                    302.76                    56.57
6137              833.10                    374.63                    58.21
6138              894.53                    360.96                    58.96

[6139 rows x 9 columns]
```

[5]:
```
#dimesdimesnsion of the data
```

[6]:
```
dataset.shape
```

[6]:
```
(6139, 9)
```

[7]:
```
#data description
```

[8]:
```
dataset.describe()
```

[8]:
```
       Material Quantity (gm)  Additive Catalyst (gm)  Ash Component (gm)  \
count             6030.000000             6030.000000         6030.000000
mean               383.642297              196.699846          111.856252
std                149.994316              133.329220           74.241117
min                124.440000                0.000000            0.000000
25%                256.030000               78.210000           44.582500
50%                377.405000              192.320000          115.250000
75%                511.522500              307.650000          174.257500
max                658.800000              438.470000          244.120000

       Water Mix (ml)  Plasticizer (gm)  Moderate Aggregator  \
count     6030.000000       6030.000000          6030.000000
mean       224.296955         17.651085           998.669332
std         41.545751         11.687965            97.732677
min        148.600000          0.000000           821.540000
25%        190.387500          7.922500           918.437500
50%        225.700000         16.345000           997.985000
75%        257.447500         27.667500          1079.827500
max        301.340000         39.280000          1174.360000

       Refined Aggregator  Formulation Duration (hrs)  \
count         6030.000000                 6030.000000
mean           811.832398                  174.408504
std            112.813539                  112.415173
min            609.230000                   16.250000
```

```
25%            717.447500                    70.300000
50%            810.260000                   163.105000
75%            905.857500                   272.602500
max           1018.050000                   380.250000

       Compression Strength MPa
count                6139.000000
mean                   56.851430
std                    16.124932
min                     2.610000
25%                    47.085000
50%                    59.790000
75%                    69.845000
max                    92.510000
```

[9]:
```
#dataset info
```

[10]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6139 entries, 0 to 6138
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Material Quantity (gm)       6030 non-null   float64
 1   Additive Catalyst (gm)       6030 non-null   float64
 2   Ash Component (gm)           6030 non-null   float64
 3   Water Mix (ml)               6030 non-null   float64
 4   Plasticizer (gm)             6030 non-null   float64
 5   Moderate Aggregator          6030 non-null   float64
 6   Refined Aggregator           6030 non-null   float64
 7   Formulation Duration (hrs)   6030 non-null   float64
 8   Compression Strength MPa     6139 non-null   float64
dtypes: float64(9)
memory usage: 431.8 KB
```

[11]:
```
#finding variant columns
```

[12]:
```
dataset.var()
```

[12]:
```
Material Quantity (gm)        22498.294732
Additive Catalyst (gm)        17776.680972
Ash Component (gm)             5511.743471
Water Mix (ml)                 1726.049445
Plasticizer (gm)                136.608535
Moderate Aggregator            9551.676176
Refined Aggregator            12726.894650
```

```
Formulation Duration (hrs)    12637.171095
Compression Strength MPa        260.013420
dtype: float64
```

[13]:
```
dataset.var().sort_values(ascending = False)
```

[13]:
```
Material Quantity (gm)        22498.294732
Additive Catalyst (gm)        17776.680972
Refined Aggregator            12726.894650
Formulation Duration (hrs)    12637.171095
Moderate Aggregator            9551.676176
Ash Component (gm)             5511.743471
Water Mix (ml)                 1726.049445
Compression Strength MPa        260.013420
Plasticizer (gm)                136.608535
dtype: float64
```

[14]:
```
#finding correlated columns
```

[15]:
```
corr_matrix = dataset.corr()
corr_matrix
```

[15]:
```
                            Material Quantity (gm)  Additive Catalyst (gm)  \
Material Quantity (gm)                    1.000000                0.009507
Additive Catalyst (gm)                    0.009507                1.000000
Ash Component (gm)                       -0.024180                0.053598
Water Mix (ml)                            0.004640                0.029818
Plasticizer (gm)                          0.048551                0.140246
Moderate Aggregator                      -0.009366               -0.022772
Refined Aggregator                       -0.016475                0.009807
Formulation Duration (hrs)                0.066251                0.162214
Compression Strength MPa                  0.130875                0.180811

                            Ash Component (gm)  Water Mix (ml)  \
Material Quantity (gm)               -0.024180        0.004640
Additive Catalyst (gm)                0.053598        0.029818
Ash Component (gm)                    1.000000       -0.006846
Water Mix (ml)                       -0.006846        1.000000
Plasticizer (gm)                      0.161667       -0.024760
Moderate Aggregator                  -0.003301       -0.029820
Refined Aggregator                    0.040000       -0.054666
Formulation Duration (hrs)            0.109820        0.031210
Compression Strength MPa              0.090961       -0.027051

                            Plasticizer (gm)  Moderate Aggregator  \
Material Quantity (gm)              0.048551            -0.009366
Additive Catalyst (gm)              0.140246            -0.022772
```

```
Ash Component (gm)                 0.161667           -0.003301
Water Mix (ml)                    -0.024760           -0.029820
Plasticizer (gm)                   1.000000           -0.020225
Moderate Aggregator               -0.020225            1.000000
Refined Aggregator                 0.056807           -0.006605
Formulation Duration (hrs)         0.156834            0.008240
Compression Strength MPa           0.207256           -0.032151

                           Refined Aggregator  Formulation Duration (hrs)  \
Material Quantity (gm)              -0.016475                    0.066251
Additive Catalyst (gm)               0.009807                    0.162214
Ash Component (gm)                   0.040000                    0.109820
Water Mix (ml)                      -0.054666                    0.031210
Plasticizer (gm)                     0.056807                    0.156834
Moderate Aggregator                 -0.006605                    0.008240
Refined Aggregator                   1.000000                    0.006408
Formulation Duration (hrs)           0.006408                    1.000000
Compression Strength MPa            -0.010762                    0.268032

                           Compression Strength MPa
Material Quantity (gm)                     0.130875
Additive Catalyst (gm)                     0.180811
Ash Component (gm)                         0.090961
Water Mix (ml)                            -0.027051
Plasticizer (gm)                           0.207256
Moderate Aggregator                       -0.032151
Refined Aggregator                        -0.010762
Formulation Duration (hrs)                 0.268032
Compression Strength MPa                   1.000000
```

[16]:
```
# finding columns that carry more than 50% of the same information by setting␣
↪threshold
```

[17]:
```
threshold = 0.5
correlated_columns = set()
```

[18]:
```
for row in range(len(corr_matrix)):
    for col in range(row):
        if abs (corr_matrix.iloc[row][col]) > threshold:

            corr = correlated_columns.add(corr_matrix.columns[row])
            print(f'correlated column',corr)
        else:
            print('There are no correlated columns')
```

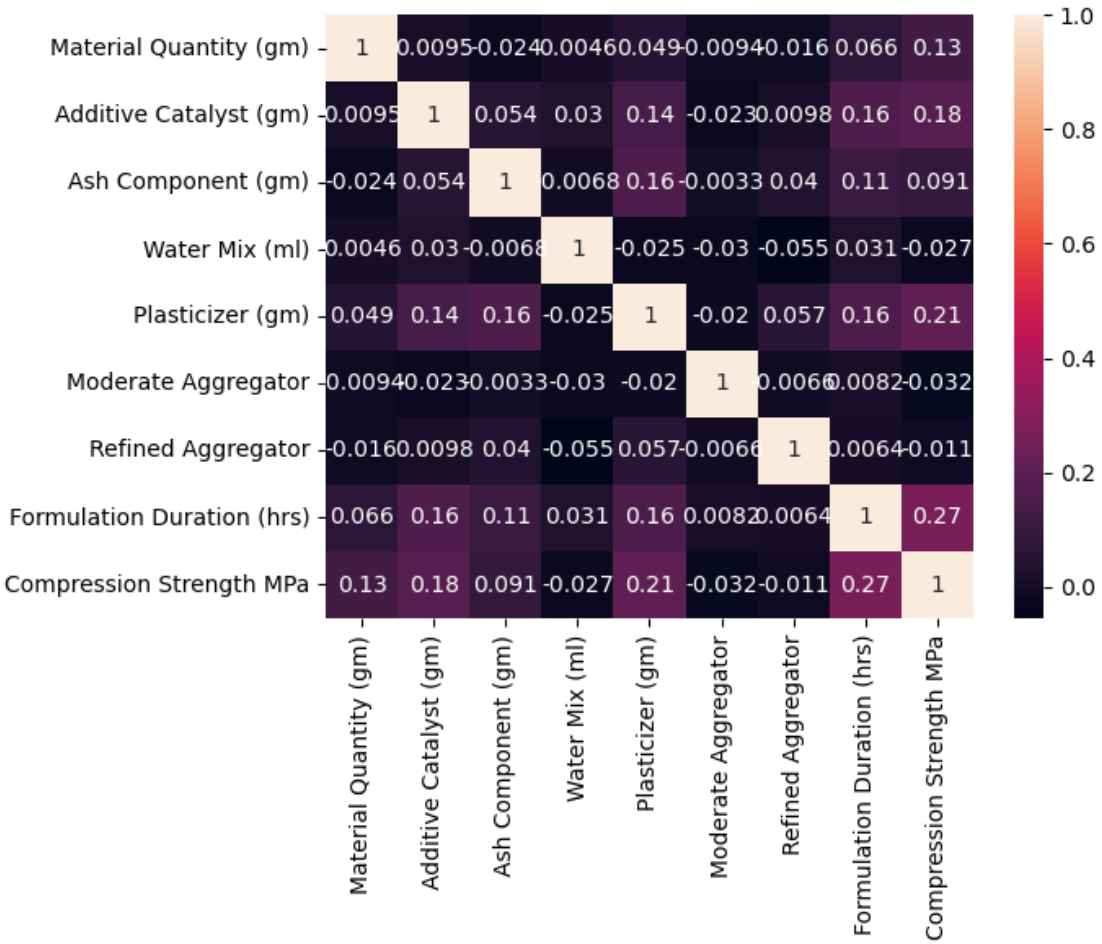
```
There are no correlated columns
There are no correlated columns
```

```
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
There are no correlated columns
```

OBSERVATION :

Thus, there are no columns that are related columns. All columns are mutually exclusive of each other.

[19]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(dataset.corr(),annot = True)
plt.show()
```



OBSERVATION:

The above heatmap clearly tells that most of the columns are very less likely to be related to each other by 30%

[20]:
```
#null values count
```

[21]:
```
dataset.isnull().sum()
```

[21]:
```
Material Quantity (gm)         109
Additive Catalyst (gm)         109
Ash Component (gm)             109
Water Mix (ml)                 109
Plasticizer (gm)               109
Moderate Aggregator            109
Refined Aggregator             109
Formulation Duration (hrs)     109
Compression Strength MPa         0
```

```
dtype: int64
```

[22]:
```
#109 null values ate present in 8 columns which is significant amount of data␣
↪to be dropped. So,filling with mean would be best here.
```

[23]:
```
#filling Material Quantity (gm)
```

[24]:
```
dataset['Material Quantity (gm)'].mean()
```

[24]:
```
383.6422968490888
```

[25]:
```
dataset['Material Quantity (gm)'] = dataset['Material Quantity (gm)'].
↪fillna(dataset['Material Quantity (gm)'].mean())
```

[26]:
```
dataset['Material Quantity (gm)'].isnull().sum()
```

[26]:
```
0
```

OBSERVATION :

The null values in 'Material Quantity (gm)' is filled with its mean value.

[27]:
```
#filling Additive Catalyst (gm)
```

[28]:
```
add_mean = dataset['Additive Catalyst (gm)'].mean()
```

[29]:
```
dataset['Additive Catalyst (gm)'] = dataset['Additive Catalyst (gm)'].
↪fillna(add_mean)
```

[30]:
```
dataset['Additive Catalyst (gm)'].isnull().sum()
```

[30]:
```
0
```

OBSERVATION :

The null values in 'Additive Catalyst (gm)' is filled with its mean value.

[31]:
```
#filling Ash Component (gm)
```

[32]:
```
ash_mean = dataset['Ash Component (gm)'].mean()
dataset['Ash Component (gm)'] = dataset['Ash Component (gm)'].fillna(ash_mean)
dataset['Ash Component (gm)'].isnull().sum()
```

[32]:
```
0
```

OBSERVATION : The null values in 'Ash Component (gm)' is filled with its mean value.

[33]:
```
#filling Water Mix (ml)
```

[34]:
```
wat_mean = dataset['Water Mix (ml)'].mean()
dataset['Water Mix (ml)'] = dataset['Water Mix (ml)'].fillna(wat_mean)
dataset['Water Mix (ml)'].isnull().sum()
```

[34]: 0

OBSERVATION :

The null values in 'Water Mix (ml)' is filled with its mean value.

[35]:
```
#filling Plasticizer (gm)
```

[36]:
```
pla_mean = dataset['Plasticizer (gm)'].mean()
dataset['Plasticizer (gm)'] = dataset['Plasticizer (gm)'].fillna(pla_mean)
dataset['Plasticizer (gm)'].isnull().sum()
```

[36]: 0

OBSERVATION : The null values in 'Plasticizer (gm)' is filled with its mean value.

[37]:
```
#filling Moderate Aggregator
```

[38]:
```
mod_mean = dataset['Moderate Aggregator'].mean()
dataset['Moderate Aggregator'] = dataset['Moderate Aggregator'].fillna(mod_mean)
dataset['Moderate Aggregator'].isnull().sum()
```

[38]: 0

OBSERVATION:

The null values in 'Moderate Aggregator' is filled with its mean value.

[39]:
```
#filling Refined Aggregator
```

[40]:
```
ref_mean = dataset['Refined Aggregator'].mean()
dataset['Refined Aggregator'] = dataset['Refined Aggregator'].fillna(ref_mean)
dataset['Refined Aggregator'].isnull().sum()
```

[40]: 0

OBSERVATION : The null values in 'Refined Aggregator' is filled with its mean value.

[41]:
```
#filling Formulation Duration (hrs)
```

[42]:
```
for_mean = dataset['Formulation Duration (hrs)'].mean()
dataset['Formulation Duration (hrs)'] = dataset['Formulation Duration (hrs)'].
  ↪fillna(for_mean)
dataset['Formulation Duration (hrs)'].isnull().sum()
```

[42]:
```
0
```

OBSERVATION:

The null values in 'Refined Aggregator' is filled with its mean value.

[43]:
```
dataset.isnull().sum()
```

[43]:
```
Material Quantity (gm)        0
Additive Catalyst (gm)        0
Ash Component (gm)            0
Water Mix (ml)                0
Plasticizer (gm)              0
Moderate Aggregator           0
Refined Aggregator            0
Formulation Duration (hrs)    0
Compression Strength MPa      0
dtype: int64
```

OBSERVATION:

All the null values are filled.

[44]:
```
#checking skeweness of data
```

[45]:
```
dataset.skew()
```

[45]:
```
Material Quantity (gm)        0.096605
Additive Catalyst (gm)        0.107584
Ash Component (gm)           -0.001224
Water Mix (ml)                0.024953
Plasticizer (gm)              0.182842
Moderate Aggregator          -0.020582
Refined Aggregator           -0.006749
Formulation Duration (hrs)    0.233290
Compression Strength MPa     -0.766954
dtype: float64
```

NOTES -0.5 and 0.5, the distribution of the value is almost symmetrical. -1 and -0.5, the data is negatively skewed. 0.5 to 1, the data is positively skewed.

OBSERVATION

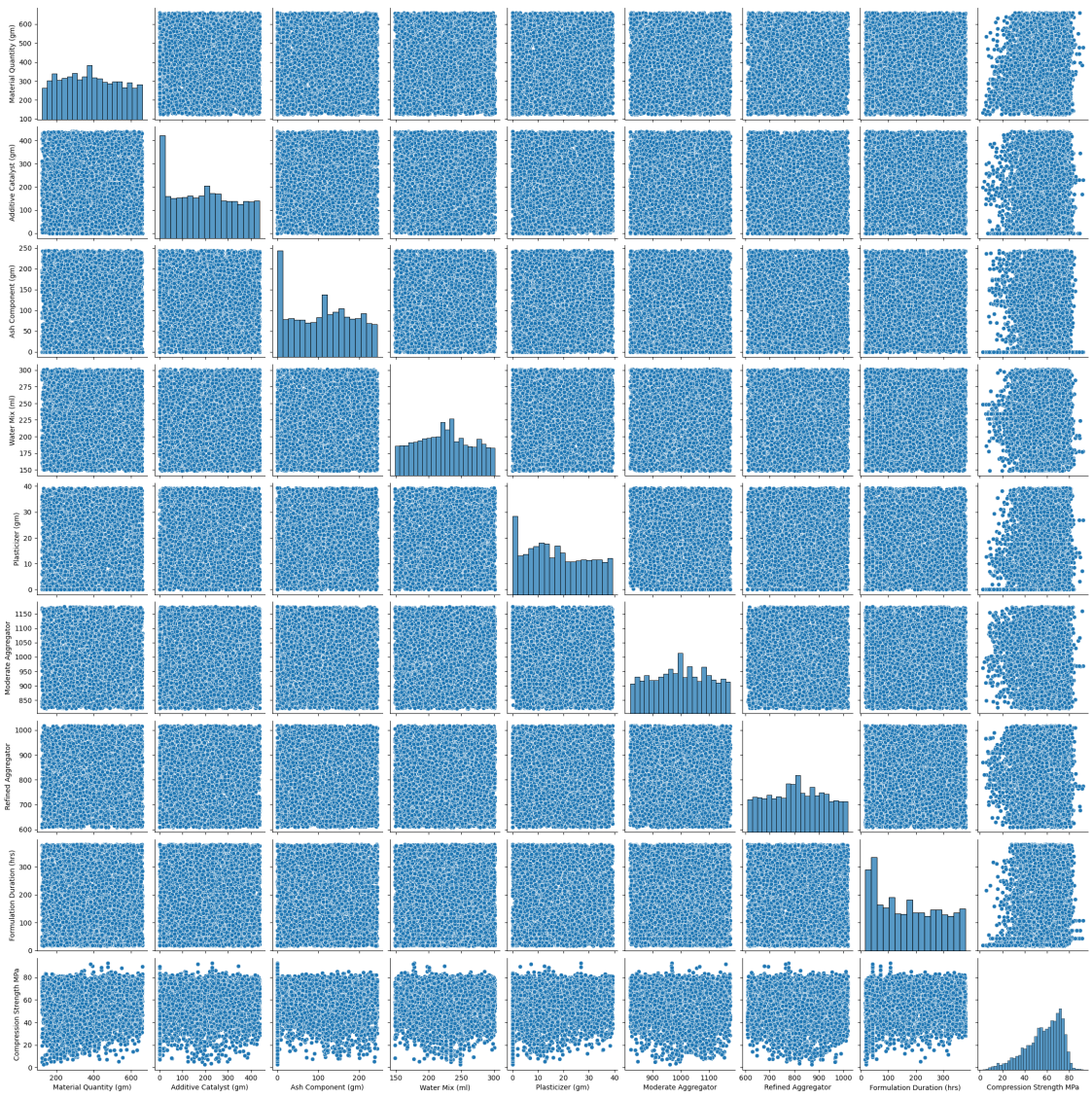
Almost symmetrical - Compression Strength MPa Negatively skewed - Compression Strength MPa

we can say Compression Strength MPa is almost symmetrical but negatively screwed

[46]:
```
import seaborn as sns
```

[47]:
```
sns.pairplot(dataset)
```

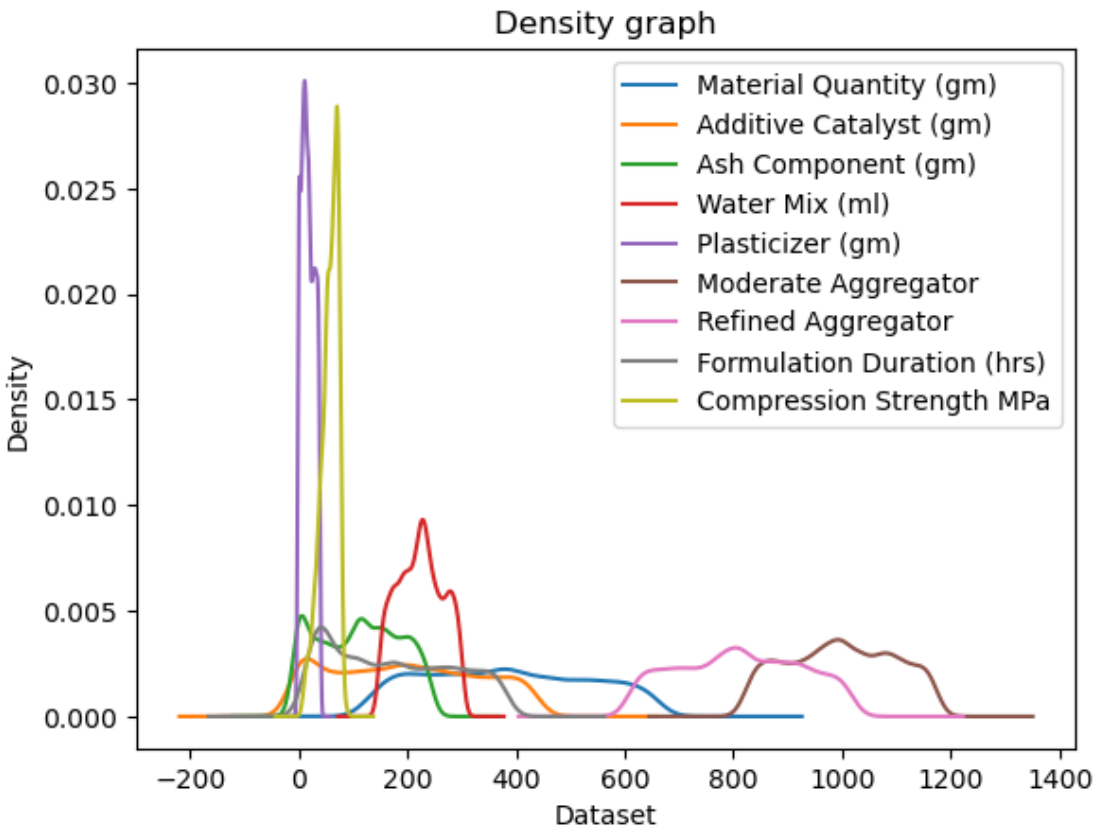
[47]: <seaborn.axisgrid.PairGrid at 0x1c7e99802b0>



OBSERVATION:

The above graph explains relationship between two variables

[48]:
```
import matplotlib.pyplot as plt
dataset.plot(kind = 'density')
plt.title('Density graph')
plt.xlabel('Dataset')
plt.show()
```

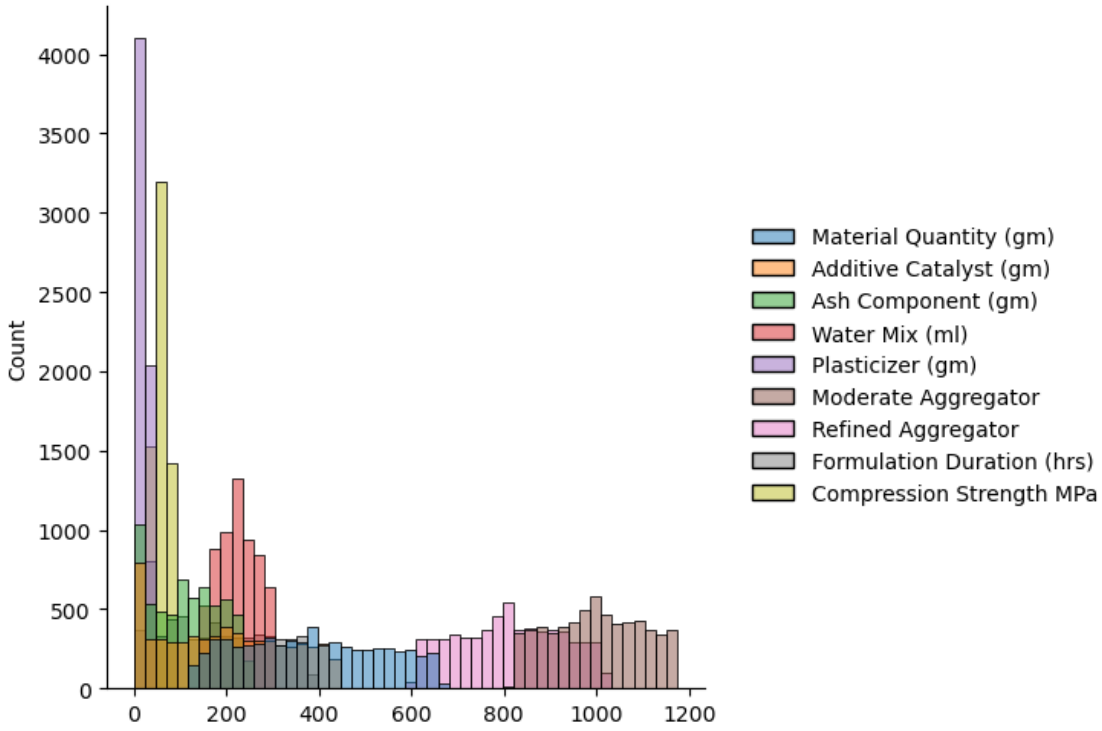


OBSERVATION:

Data is distributed highly in Plasticizer(gm) and Compression Strength Mpa. Data is distributed less in Addictive Catalyst (gm) and Material Qunatity (gm)

[ ]:
```
#distance plot
```

[112]:
```
sns.displot(dataset, legend=True)
plt.show()
```

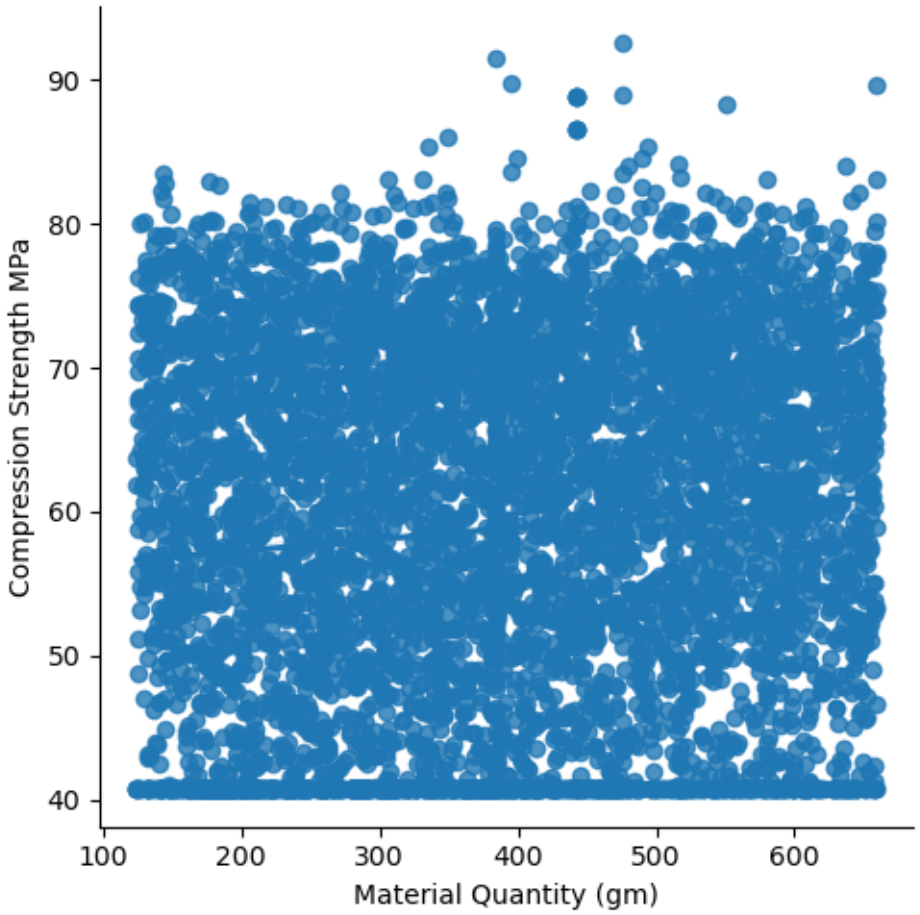


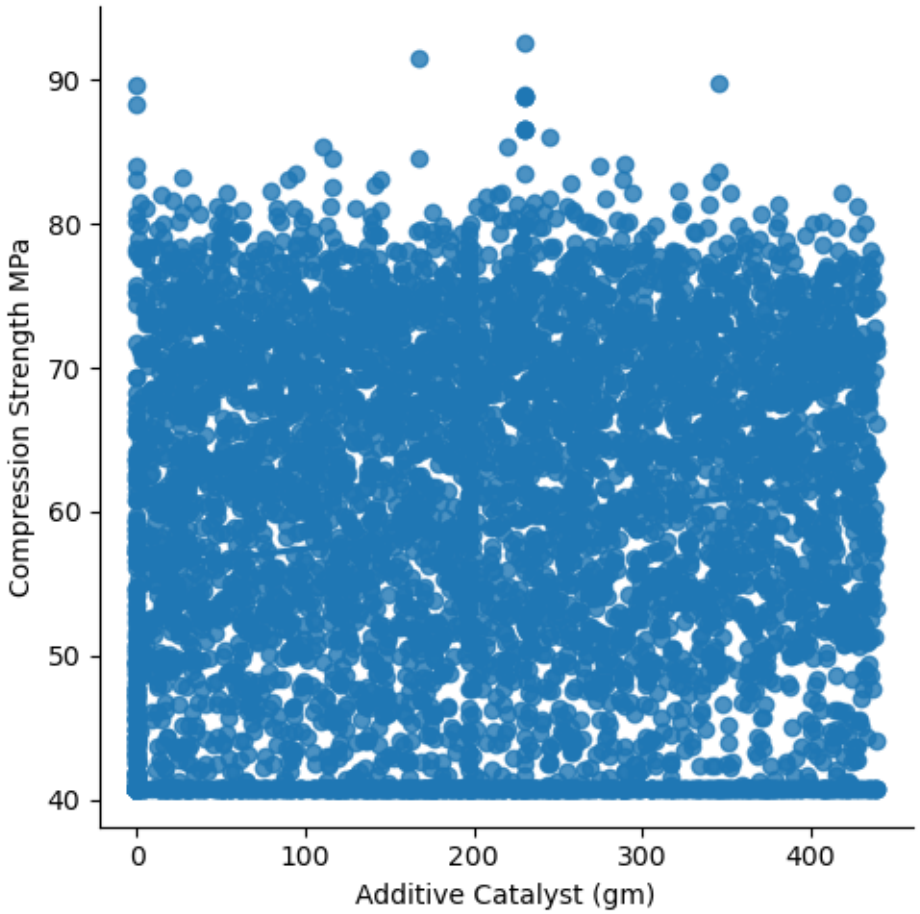
OBSERVATION:

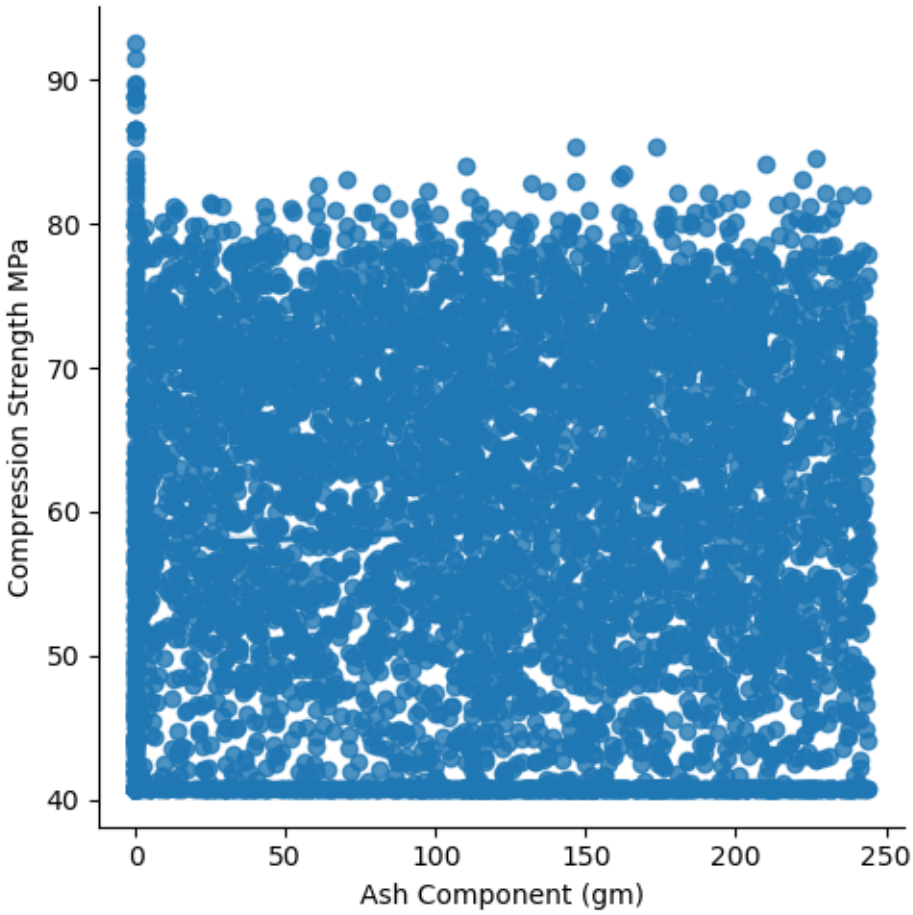
The data distribution of all the variables against the density distribution is observed.

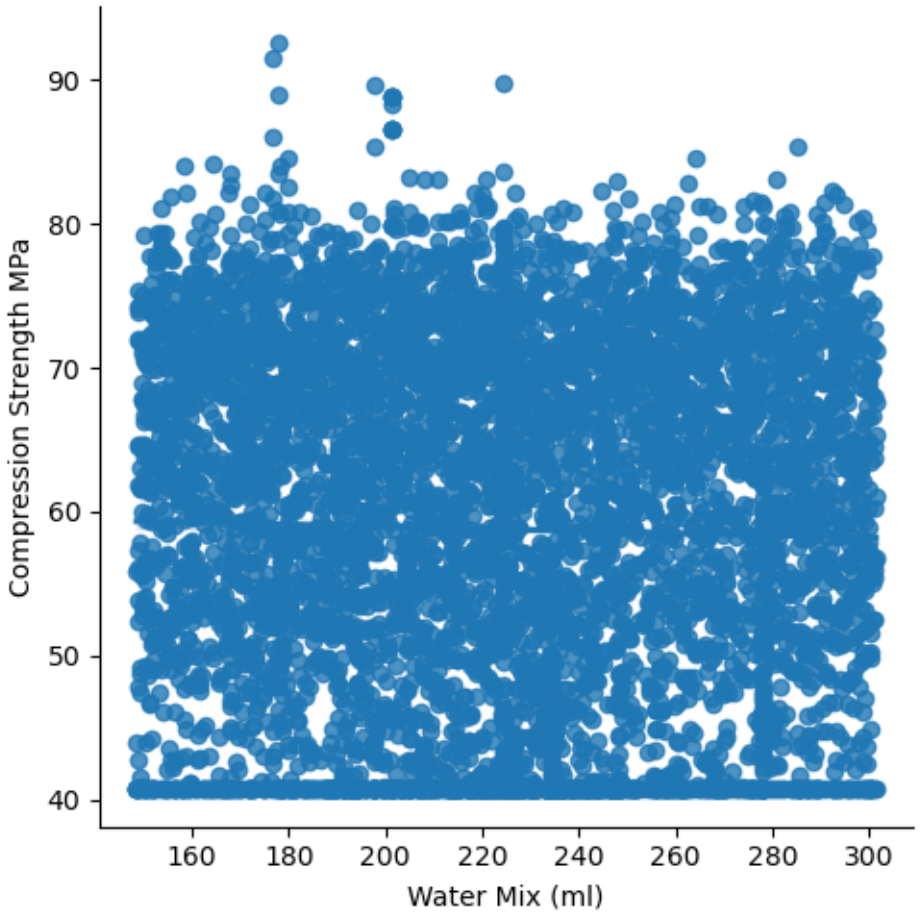
[ ]:
```
#relationship between all the Columns(IV) Vs Compression Strength MPa (DV)
```

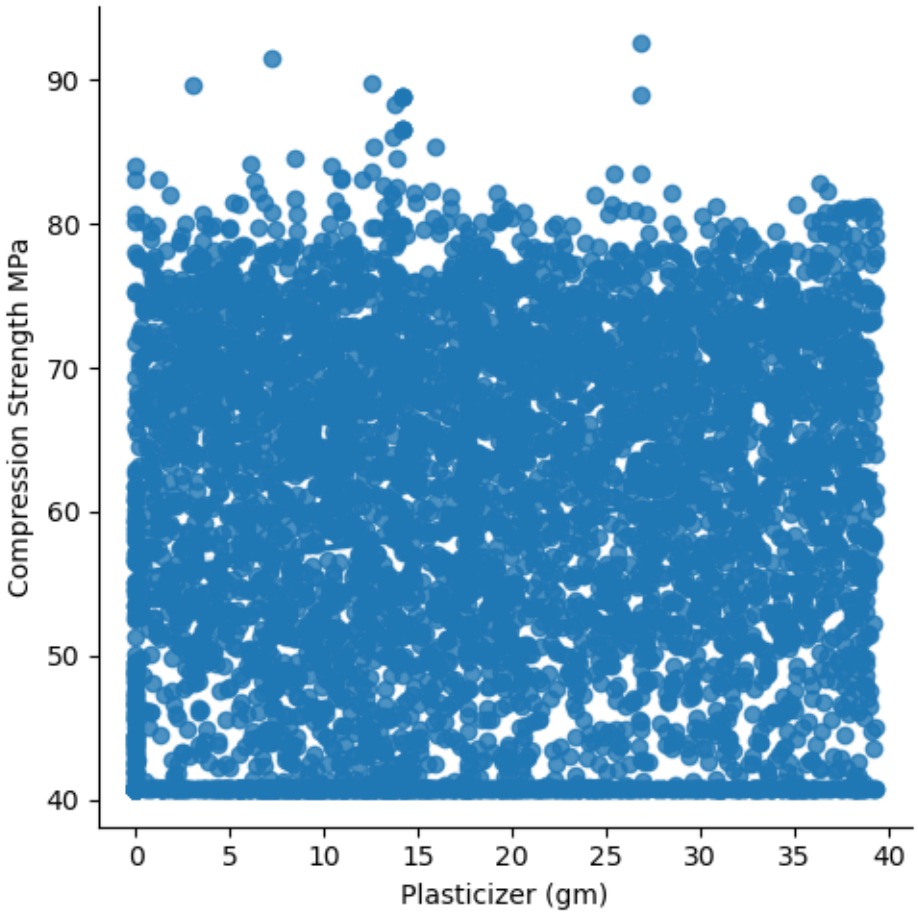
[120]:
```
for col in dataset:
    sns.lmplot(x=col, y= 'Compression Strength MPa', data=dataset, order=1)
    plt.ylabel("Compression Strength MPa")
    plt.xlabel(col)
```

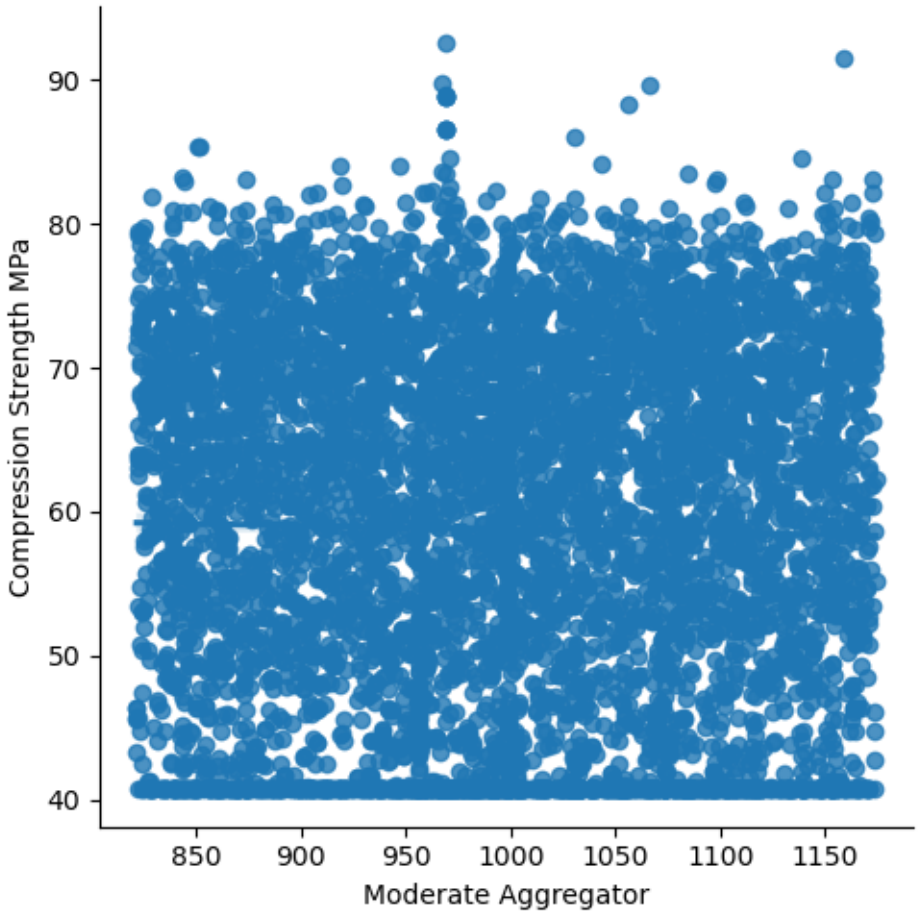

Compression Strength MPa
90
80
70
60
50
40
100
200
300
400
500
600
Material Quantity (gm)

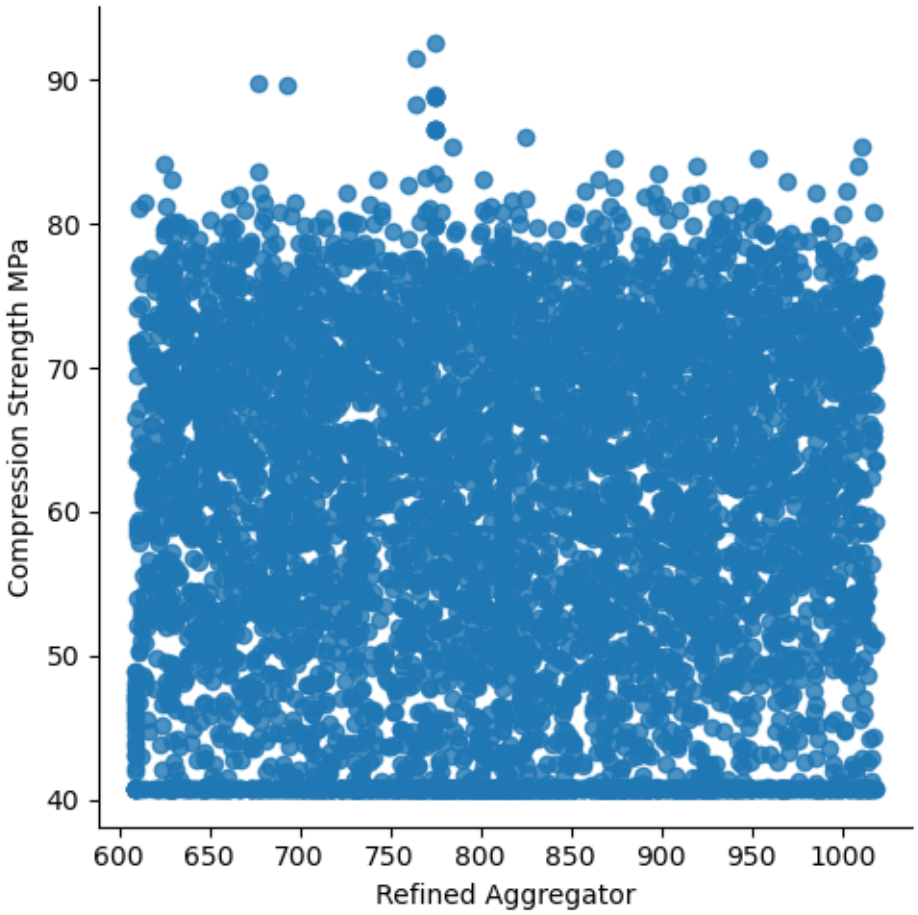

Compression Strength MPa
90
80
70
60
50
40
0
100
200
300
400
Additive Catalyst (gm)

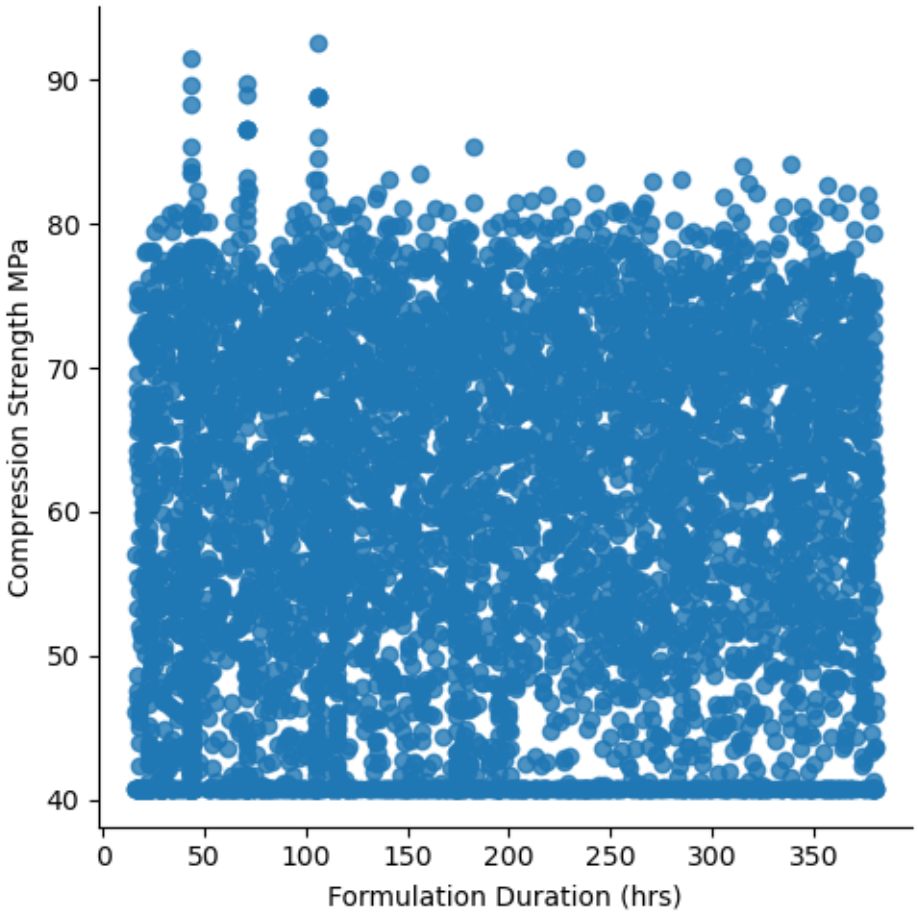

Compression Strength MPa
90
80
70
60
50
40
0
50
100
150
200
250
Ash Component (gm)

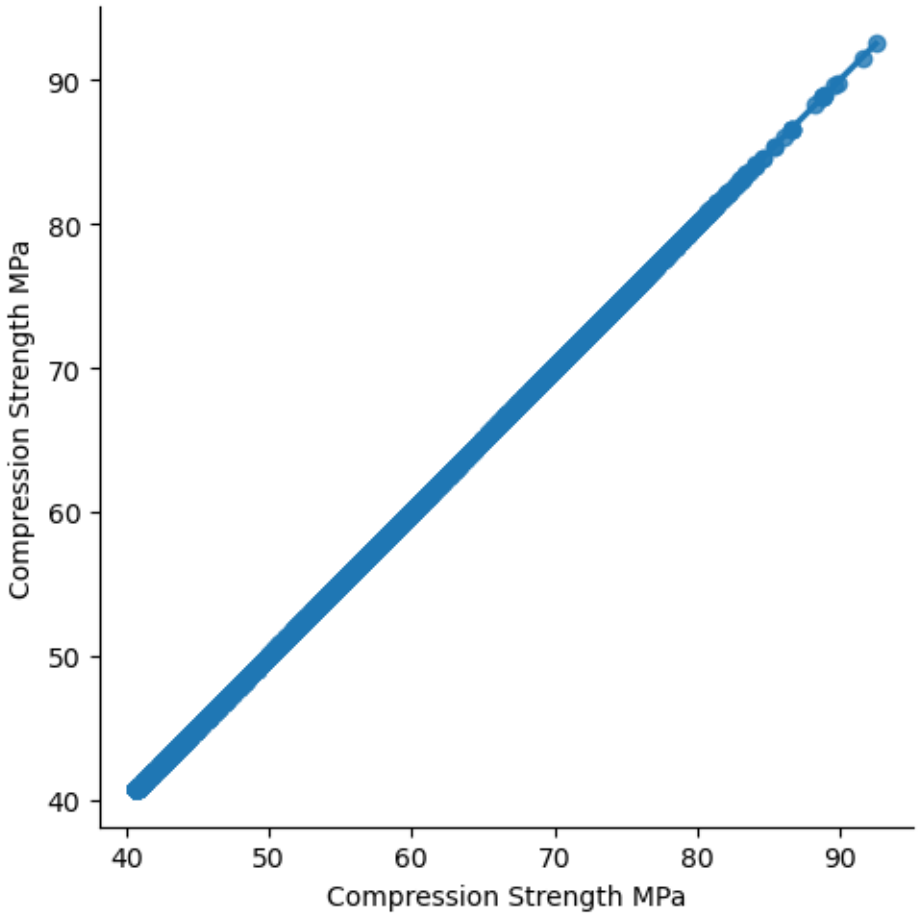

Compression Strength MPa
90
80
70
60
50
40
160
180
200
220
240
260
280
300
Water Mix (ml)


Compression Strength MPa
90
80
70
60
50
40
0
5
10
15
20
25
30
35
40
Plasticizer (gm)


Compression Strength MPa
90
80
70
60
50
40
850
900
950
1000
1050
1100
1150
Moderate Aggregator


Compression Strength MPa
90
80
70
60
50
40
600
650
700
750
800
850
900
950
1000
Refined Aggregator


Compression Strength MPa
90
80
70
60
50
40
0
50
100
150
200
250
300
350
Formulation Duration (hrs)



OBSERVATION:

The IV(s) and DV shows non linear relationship with each other except itself.

[50]:
```
#Detecting outliers
```

[51]:
```
def find_outliers_IQR(dataset):

   q1= dataset.quantile(0.25)

   q3= dataset.quantile(0.75)

   IQR=q3-q1

   outliers = dataset[((dataset<(q1-1.5*IQR)) | (dataset>(q3+1.5*IQR)))]

   return outliers
```

```
outliers = find_outliers_IQR(dataset)

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min outlier value: "+ str(outliers.min()))

outliers


outliers = find_outliers_IQR(dataset)

outliers
```

```
number of outliers: 6139
max outlier value: Material Quantity (gm)          NaN
Additive Catalyst (gm)           NaN
Ash Component (gm)               NaN
Water Mix (ml)                   NaN
Plasticizer (gm)                 NaN
Moderate Aggregator              NaN
Refined Aggregator               NaN
Formulation Duration (hrs)       NaN
Compression Strength MPa       12.86
dtype: float64
min outlier value: Material Quantity (gm)          NaN
Additive Catalyst (gm)           NaN
Ash Component (gm)               NaN
Water Mix (ml)                   NaN
Plasticizer (gm)                 NaN
Moderate Aggregator              NaN
Refined Aggregator               NaN
Formulation Duration (hrs)       NaN
Compression Strength MPa       2.61
dtype: float64
```

[51]:

	Material Quantity (gm)	Additive Catalyst (gm)	Ash Component (gm)	\
0	NaN	NaN	NaN	
1	NaN	NaN	NaN	
2	NaN	NaN	NaN	
3	NaN	NaN	NaN	
4	NaN	NaN	NaN	
…	…	…	…	
6134	NaN	NaN	NaN	
6135	NaN	NaN	NaN	
6136	NaN	NaN	NaN	

```
6137                    NaN                    NaN                  NaN
6138                    NaN                    NaN                  NaN

      Water Mix (ml)  Plasticizer (gm)  Moderate Aggregator  \
0                NaN               NaN                  NaN
1                NaN               NaN                  NaN
2                NaN               NaN                  NaN
3                NaN               NaN                  NaN
4                NaN               NaN                  NaN
…                …                 …                    …
6134             NaN               NaN                  NaN
6135             NaN               NaN                  NaN
6136             NaN               NaN                  NaN
6137             NaN               NaN                  NaN
6138             NaN               NaN                  NaN

      Refined Aggregator  Formulation Duration (hrs)  Compression Strength MPa
0                    NaN                         NaN                       NaN
1                    NaN                         NaN                       NaN
2                    NaN                         NaN                       NaN
3                    NaN                         NaN                       NaN
4                    NaN                         NaN                       NaN
…                    …                           …                         …
6134                 NaN                         NaN                       NaN
6135                 NaN                         NaN                       NaN
6136                 NaN                         NaN                       NaN
6137                 NaN                         NaN                       NaN
6138                 NaN                         NaN                       NaN

[6139 rows x 9 columns]
```

OBSERVATION:

number of outliers: 6139

Dropping the outliers will completely diminish the data.

[52]:
```
#checking number of outliers in each column
```

[53]:
```
for col in dataset:
    outliers = find_outliers_IQR(dataset[col])
    print(outliers.dtype)
    print(f"the total number of outliers present in {col} is", len(outliers))
```

```
float64
the total number of outliers present in Material Quantity (gm) is 0
float64
the total number of outliers present in Additive Catalyst (gm) is 0
float64
```

```
the total number of outliers present in Ash Component (gm) is 0
float64
the total number of outliers present in Water Mix (ml) is 0
float64
the total number of outliers present in Plasticizer (gm) is 0
float64
the total number of outliers present in Moderate Aggregator is 0
float64
the total number of outliers present in Refined Aggregator is 0
float64
the total number of outliers present in Formulation Duration (hrs) is 0
float64
the total number of outliers present in Compression Strength MPa is 64
```

OBSERVATION:

Compression Strength MPa is the only column with 64 outliers

[54]:
```
#specifically observing the outliers in the Compression Strength MPa columnn
```

[55]:
```
outliers = find_outliers_IQR(dataset['Compression Strength MPa'])
outliers
```

[55]:
```
34      11.64
81      10.77
106      8.29
210      7.77
214      9.15
        …
5750     9.37
5869    10.23
5886     8.41
6101    11.58
6118    12.08
Name: Compression Strength MPa, Length: 64, dtype: float64
```

[56]:
```
#detecting outliers using scaling
```

[57]:
```
#Trial 1: Using robust scaler which is more effective in case of outliers
```

[58]:
```
from sklearn.preprocessing import RobustScaler
```

[59]:
```
transformer = RobustScaler().fit(dataset)
transformer
```

[59]:
```
RobustScaler()
```

[60]:
```
RobustScaler()
ro_scaler = transformer.transform(dataset)
ro_scaler
```

[60]:
```
array([[ 4.16084681e-01, -7.04739843e-02, -7.25732077e-01, …,
        -5.62812625e-01,  8.88348175e-01,  8.83128295e-01],
       [-9.89036789e-01,  2.83856023e-01,  5.69660663e-01, …,
         1.08070589e+00, -6.94845102e-01,  4.39367311e-04],
       [ 7.08768579e-01, -8.62348539e-01, -1.23753596e-02, …,
        -6.22221138e-03,  3.52790669e-01,  7.93936731e-01],
       …,
       [-9.37941463e-02, -7.73877406e-01, -7.51507508e-01, …,
        -1.05623083e-01,  6.79292613e-01, -1.41476274e-01],
       [ 2.52253328e-01,  3.52681753e-01,  5.16533323e-01, …,
         1.15836612e-01,  1.03985953e+00, -6.94200351e-02],
       [ 7.09803219e-01,  3.12455453e-01,  4.24151657e-01, …,
         4.50422669e-01,  9.71278063e-01, -3.64674868e-02]])
```

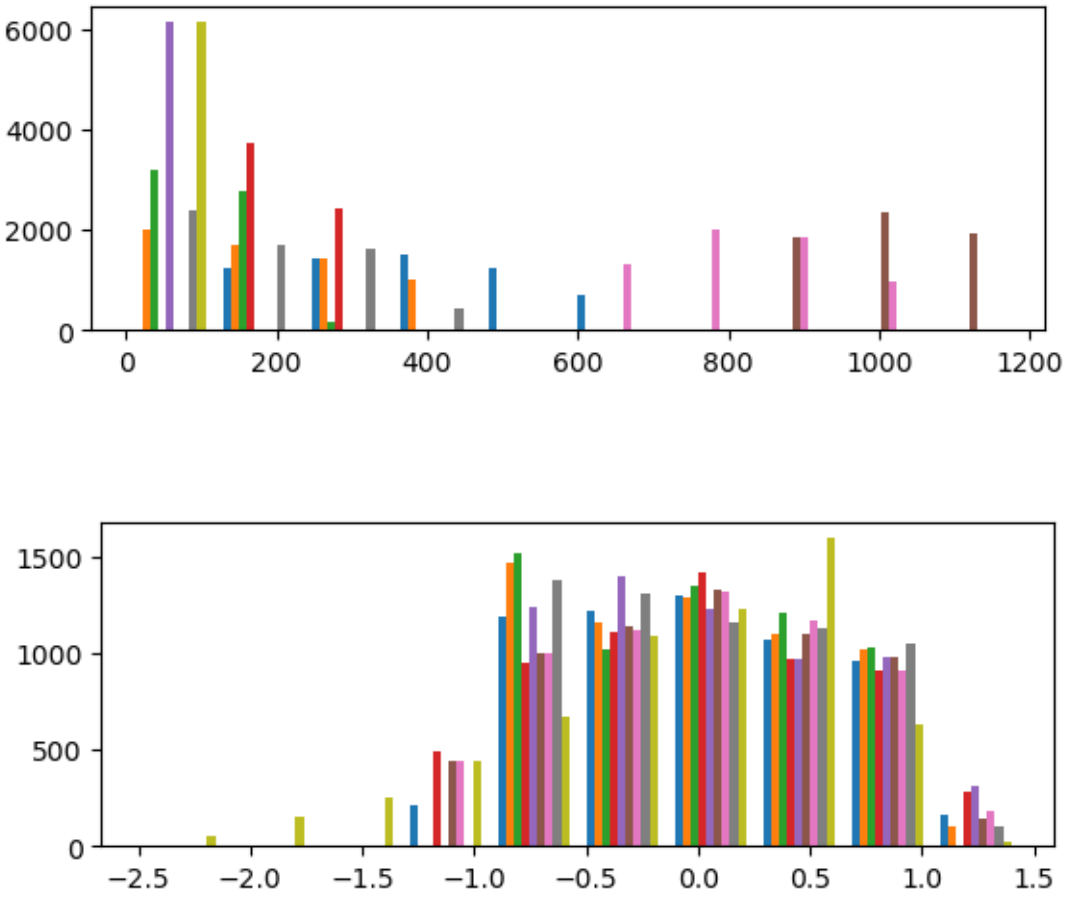
[61]:
```
for col in dataset:
    outliers = find_outliers_IQR(dataset[col])
    print(outliers.dtype)
    print(f"the total number of outliers present in {col} is", len(outliers))
```

```
float64
the total number of outliers present in Material Quantity (gm) is 0
float64
the total number of outliers present in Additive Catalyst (gm) is 0
float64
the total number of outliers present in Ash Component (gm) is 0
float64
the total number of outliers present in Water Mix (ml) is 0
float64
the total number of outliers present in Plasticizer (gm) is 0
float64
the total number of outliers present in Moderate Aggregator is 0
float64
the total number of outliers present in Refined Aggregator is 0
float64
the total number of outliers present in Formulation Duration (hrs) is 0
float64
the total number of outliers present in Compression Strength MPa is 64
```

[62]:
```
#Method: Using histogram
```

[63]:
```
plt.subplot(2,1,1)
plt.hist(dataset)
plt.show()
```

```
plt.subplot(2,1,2)
plt.hist(ro_scaler)
plt.show()
```



OBSERVATION:

Although the robust scaler did not reduce the outlier, it has normalised the dataset (Gausssian distribution)

[64]:
```
#Method 2: Box plot
```

[65]:
```
import plotly.express as px
```

[66]:
```
px.box(ro_scaler)
```

OBSERVATION:

The ouliers are most significant in Compression Strength Mpa

[67]:
```
px.scatter(ro_scaler)
```

OBSERVATION:

Most of the outliers are scattered in the range -1 to -2.5 which is of Variable 8, that is Compression Strength Mpa

OBSERVATION:

The data is scaled between the range -1 to 1

[68]:
```
#resolving outliers
```

[69]:
```
#Capping
```

[70]:
```
upper_limit = dataset['Compression Strength MPa'].mean() +␣
 ↪3*dataset['Compression Strength MPa'].std()

print(upper_limit)

lower_limit = dataset['Compression Strength MPa'].mean() - dataset['Compression␣
 ↪Strength MPa'].std()

print(lower_limit)
```

```
105.22622504246476
40.72649858632283
```

[71]:
```
#before capping
dataset['Compression Strength MPa'].describe()
```

[71]:
```
count    6139.000000
mean       56.851430
std        16.124932
min         2.610000
25%        47.085000
50%        59.790000
75%        69.845000
max        92.510000
Name: Compression Strength MPa, dtype: float64
```

[72]:
```
#applying the limits
```

[73]:
```
import numpy as np
```

[74]:
```
dataset['Compression Strength MPa'] = np.where(dataset['Compression Strength␣
 ↪MPa'] > upper_limit,
                                                upper_limit,
```

```
      np.where(dataset['Compression Strength MPa'] < lower_limit,  lower_limit,

    dataset['Compression Strength MPa']))
```

[75]:
```
dataset['Compression Strength MPa'].describe()
```

[75]:
```
count    6139.000000
mean       58.777734
std        12.604705
min        40.726499
25%        47.085000
50%        59.790000
75%        69.845000
max        92.510000
Name: Compression Strength MPa, dtype: float64
```

OBSERVATION:

Limits are applied.

[76]:
```
#inspecting outliers

outliers = find_outliers_IQR(dataset['Compression Strength MPa'])
outliers
```

[76]:
```
Series([], Name: Compression Strength MPa, dtype: float64)
```

OBSERVATION:

The limits are applied on the outliers in Compression Strength MPa making it a part of the data and not outliers anymore.

[77]:
```
for col in dataset:
    outliers = find_outliers_IQR(dataset[col])
    print(outliers.dtype)
    print(f"the total number of outliers present in {col} is", len(outliers))
```

```
float64
the total number of outliers present in Material Quantity (gm) is 0
float64
the total number of outliers present in Additive Catalyst (gm) is 0
float64
the total number of outliers present in Ash Component (gm) is 0
float64
the total number of outliers present in Water Mix (ml) is 0
float64
the total number of outliers present in Plasticizer (gm) is 0
float64
the total number of outliers present in Moderate Aggregator is 0
```

```
float64
the total number of outliers present in Refined Aggregator is 0
float64
the total number of outliers present in Formulation Duration (hrs) is 0
float64
the total number of outliers present in Compression Strength MPa is 0
```

OBSERVATION:

Thus, the outliers has not affected other columns but Compression Strength MPa. There are no outliers in Compression Strength MPa

[78]:
```
#scaling the converted outlier
```

[79]:
```
transformer = RobustScaler().fit_transform(dataset)
transformer
```

[79]:
```
array([[ 4.16084681e-01, -7.04739843e-02, -7.25732077e-01, …,
        -5.62812625e-01,  8.88348175e-01,  8.83128295e-01],
       [-9.89036789e-01,  2.83856023e-01,  5.69660663e-01, …,
         1.08070589e+00, -6.94845102e-01,  4.39367311e-04],
       [ 7.08768579e-01, -8.62348539e-01, -1.23753596e-02, …,
        -6.22221138e-03,  3.52790669e-01,  7.93936731e-01],
       …,
       [-9.37941463e-02, -7.73877406e-01, -7.51507508e-01, …,
        -1.05623083e-01,  6.79292613e-01, -1.41476274e-01],
       [ 2.52253328e-01,  3.52681753e-01,  5.16533323e-01, …,
         1.15836612e-01,  1.03985953e+00, -6.94200351e-02],
       [ 7.09803219e-01,  3.12455453e-01,  4.24151657e-01, …,
         4.50422669e-01,  9.71278063e-01, -3.64674868e-02]])
```

[80]:
```
scaled_dataset = pd.DataFrame(transformer, columns = dataset.columns)
```

[81]:
```
scaled_dataset
```

[81]:

	Material Quantity (gm)	Additive Catalyst (gm)	Ash Component (gm)	\
0	0.416085	-0.070474	-0.725732	
1	-0.989037	0.283856	0.569661	
2	0.708769	-0.862349	-0.012375	
3	0.038083	0.688836	-0.291176	
4	0.051414	0.695786	0.638632	
…	…	…	…	
6134	-0.768340	-0.151996	0.231112	
6135	-0.127301	0.423334	-0.279904	
6136	-0.093794	-0.773877	-0.751508	
6137	0.252253	0.352682	0.516533	
6138	0.709803	0.312455	0.424152	

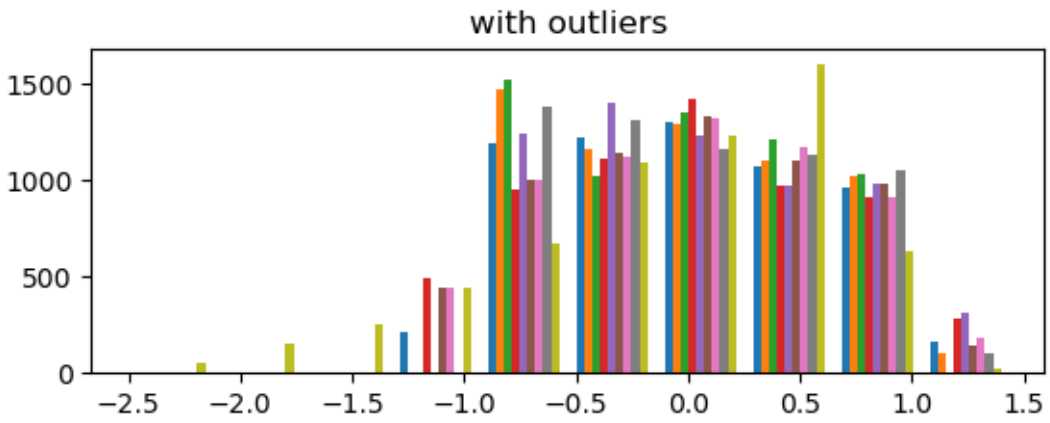
```
      Water Mix (ml)  Plasticizer (gm)  Moderate Aggregator  \
0          -0.346316         -0.034012             0.960604
1          -0.739034         -0.541098             0.578884
2           1.085184         -0.248390             0.165416
3           1.145002          0.114919             0.857993
4           0.172004          0.012883             0.627197
…                …                 …                    …
6134       -0.927667         -0.040711             0.032444
6135       -0.551319          0.466375            -0.463414
6136       -0.240449          0.934811             0.519043
6137       -0.497620          0.066993            -0.841040
6138       -0.746683         -0.316413             1.053199

      Refined Aggregator  Formulation Duration (hrs)  Compression Strength MPa
0              -0.562813                    0.888348                  0.883128
1               1.080706                   -0.694845                  0.000439
2              -0.006222                    0.352791                  0.793937
3               0.378582                    0.207952                  0.525044
4              -0.167878                    0.499084                  0.715290
…                    …                           …                         …
6134            1.038331                    0.955976                 -0.403339
6135            1.053800                   -0.316869                 -0.243849
6136           -0.105623                    0.679293                 -0.141476
6137            0.115837                    1.039860                 -0.069420
6138            0.450423                    0.971278                 -0.036467

[6139 rows x 9 columns]
```

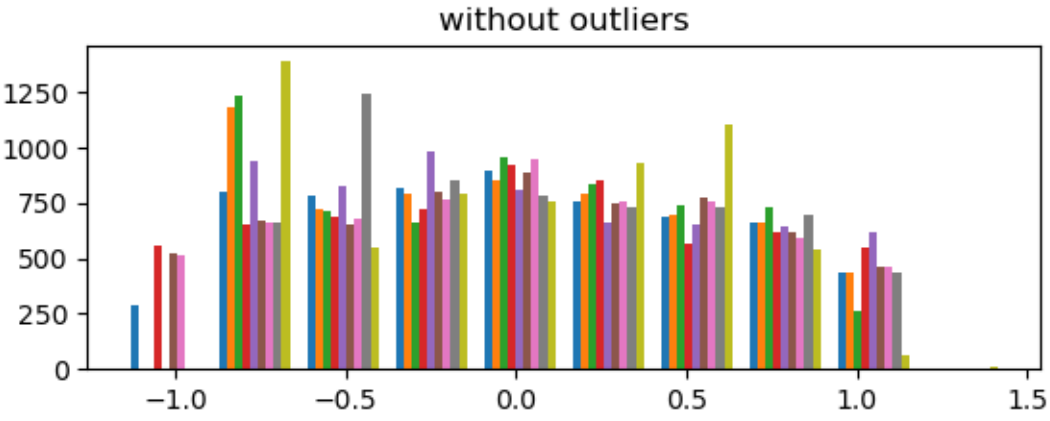
OBSERVATION

All the columns have been scaled accordingly between -1 to 1.

[82]:
```
#Plotting scaled data
```

[83]:
```
plt.subplot(2,1,1)
plt.hist(ro_scaler)
plt.title("with outliers")
plt.show()

plt.subplot(2,1,2)
plt.hist(scaled_dataset)
plt.title("without outliers")
plt.show()
```





OBSERVATION:

All the outliers present has been resolved.

[84]:
```
import plotly.express as px
df = px.data.tips()
fig1 = px.box(ro_scaler, title = "with outliers")
fig2 = px.box(scaled_dataset, title = "without outliers")
fig1.show()
fig2.show()
```

OBSERVATION:

All the outliers present has been resolved.

[85]:
```
import plotly.express as px
df = px.data.tips()
fig1 = px.scatter(ro_scaler, title = "with outliers")
```

```
fig2 = px.scatter(scaled_dataset, title = "without outliers")
fig1.show()
fig2.show()
```

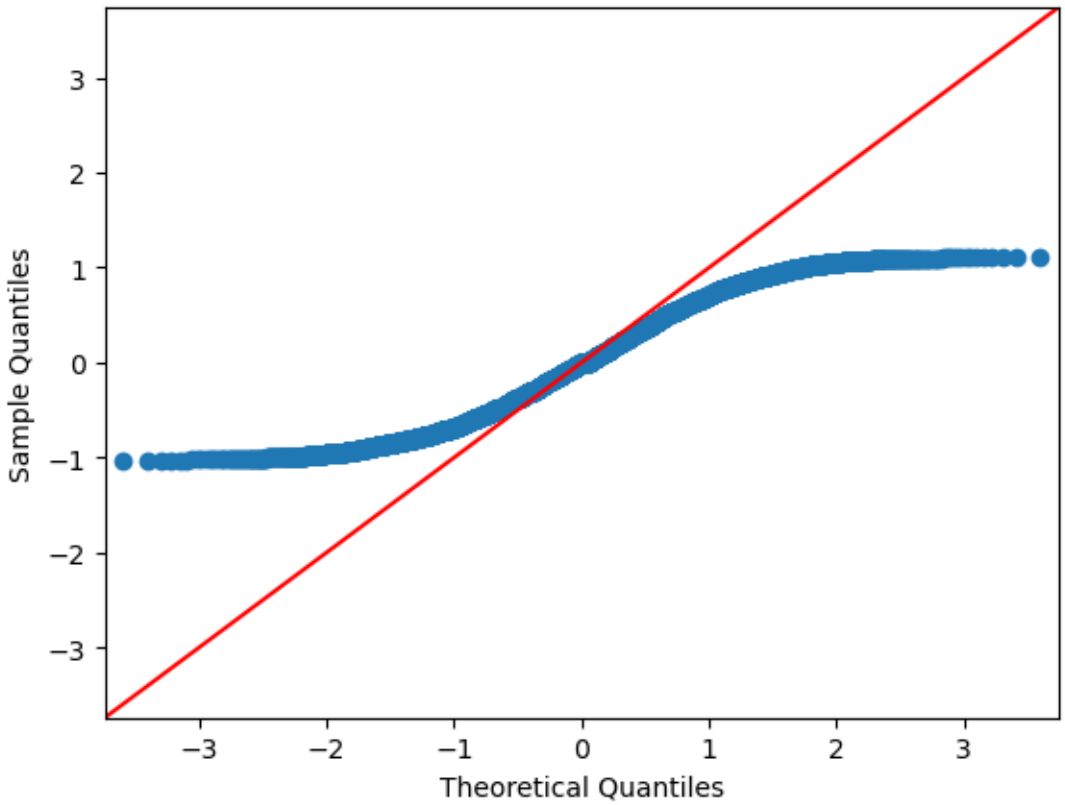
OBSERVATION:

All the outliers present has been resolved.

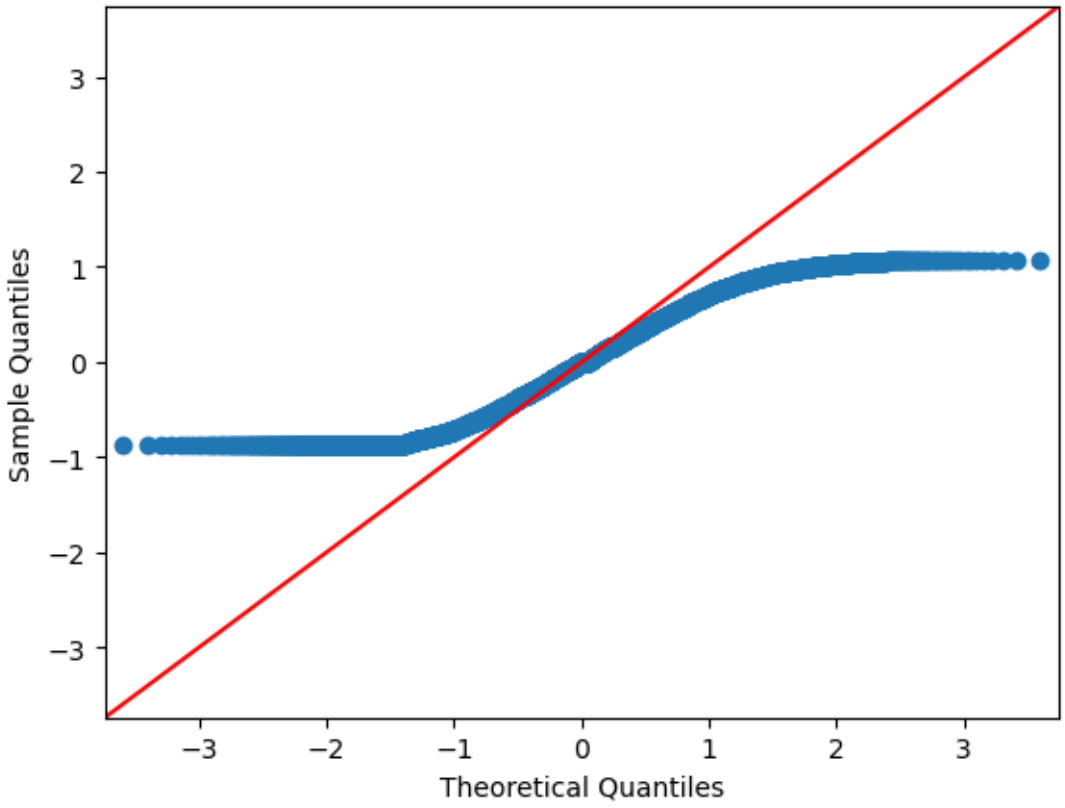
[86]:
```
#QQ Plot

#NEED: To check if all columns of the data is normally distributed.
```

[87]:
```
import statsmodels.api as sm
for col in scaled_dataset:
    print(col)
    sm.qqplot(scaled_dataset[col], line = '45')
    plt.show()
```

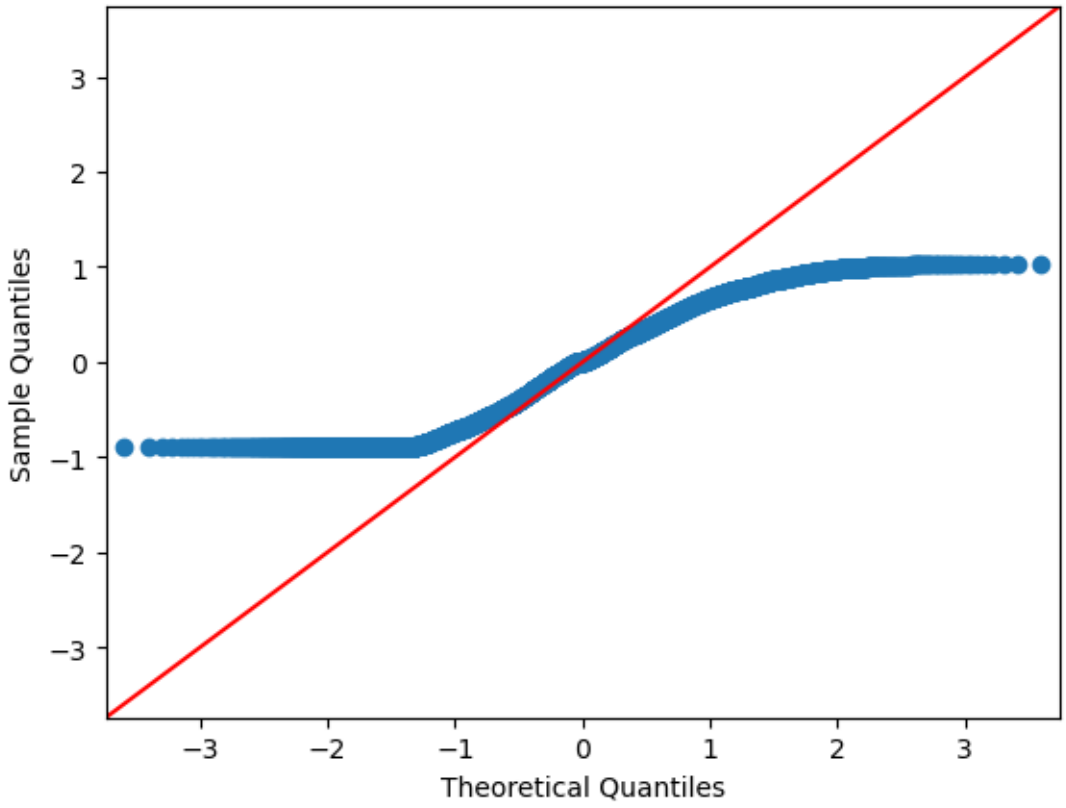
```
Material Quantity (gm)
```



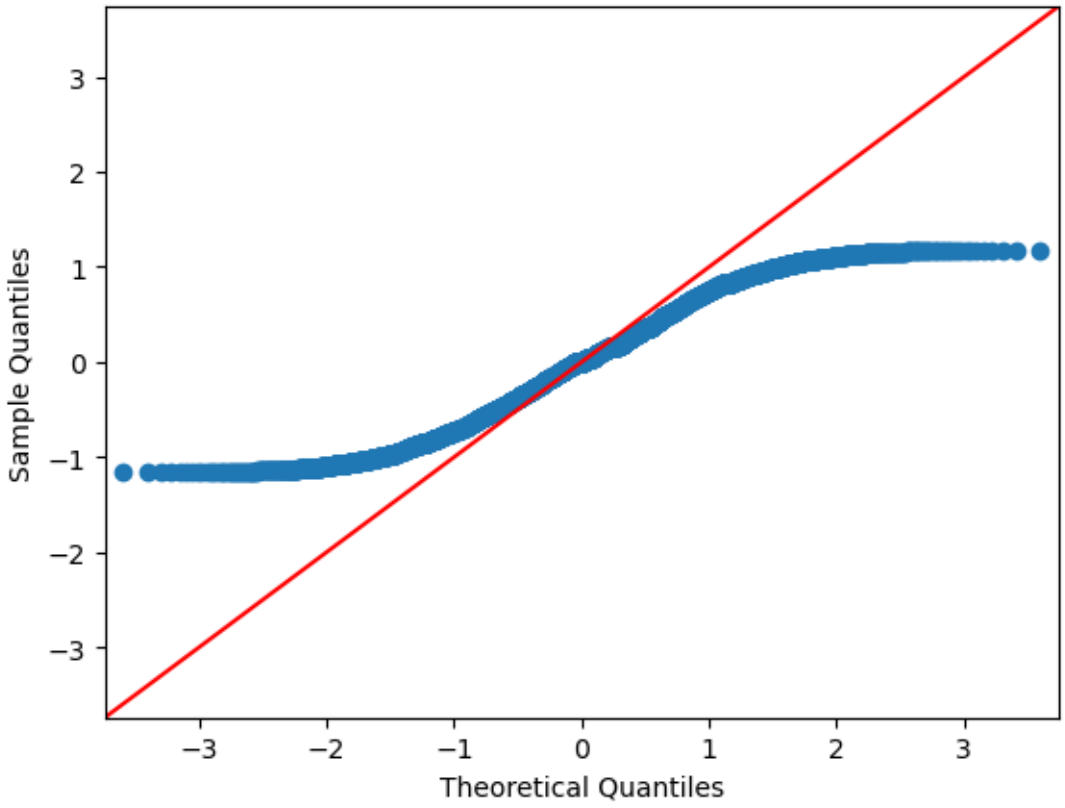

```
Additive Catalyst (gm)
```



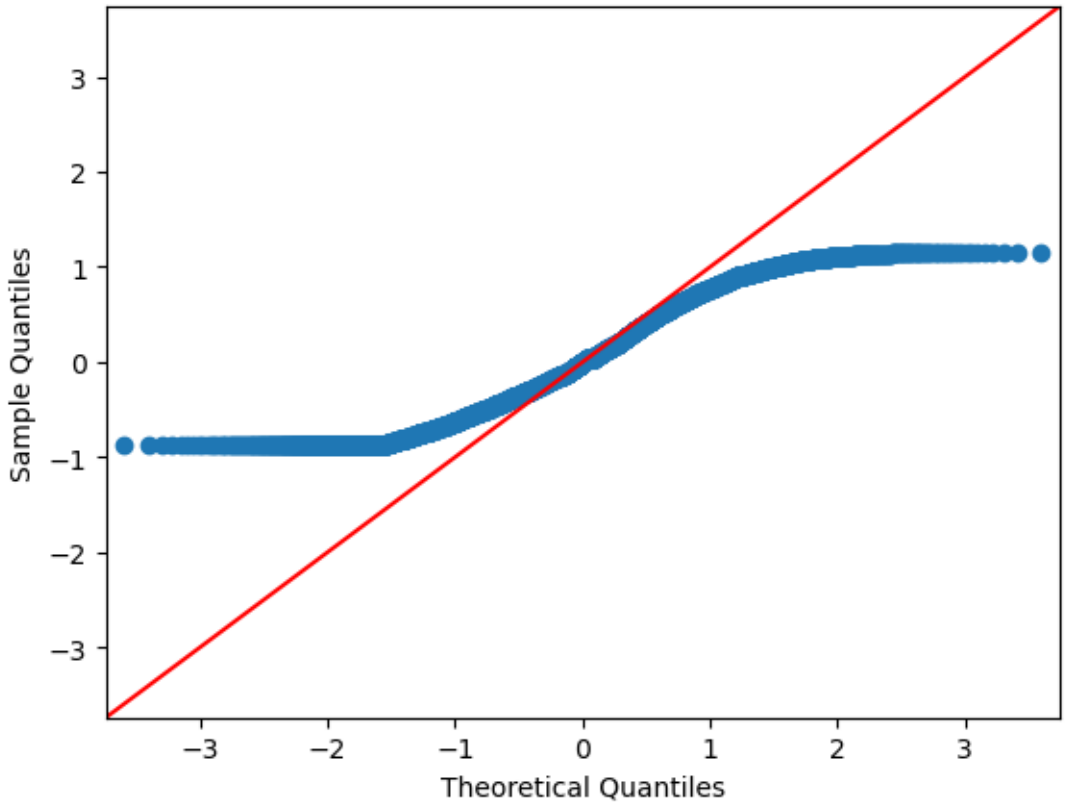
Ash Component (gm)



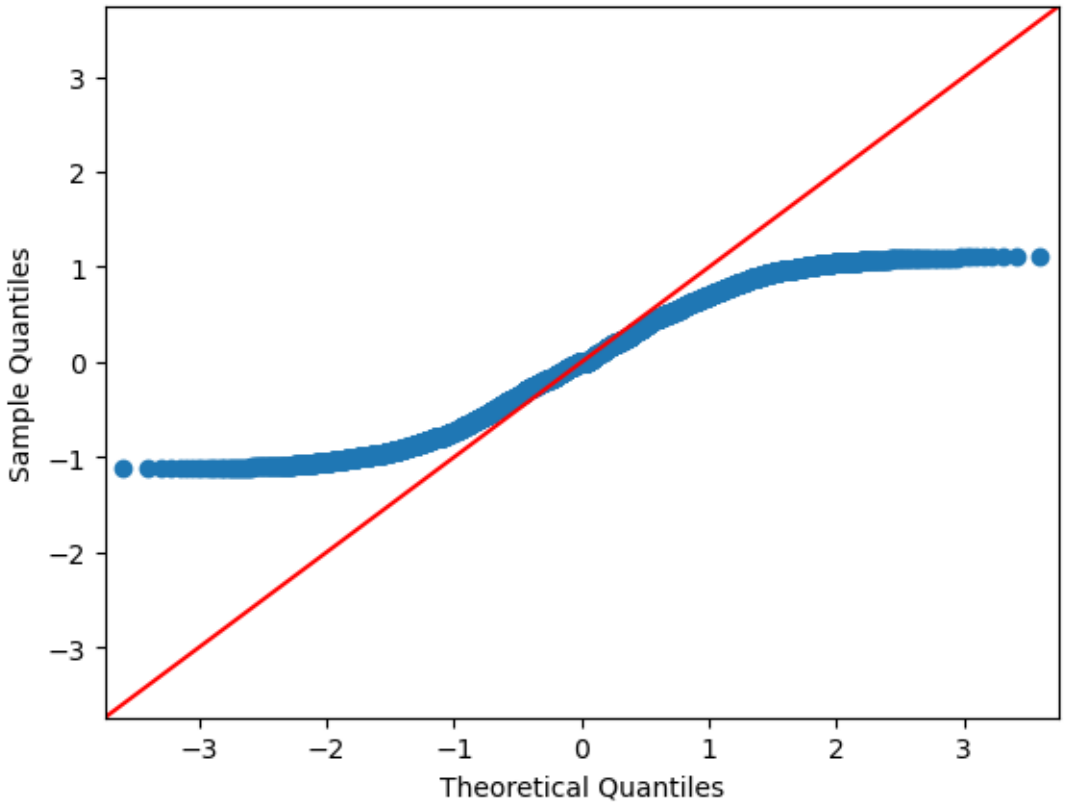
`Water Mix (ml)`



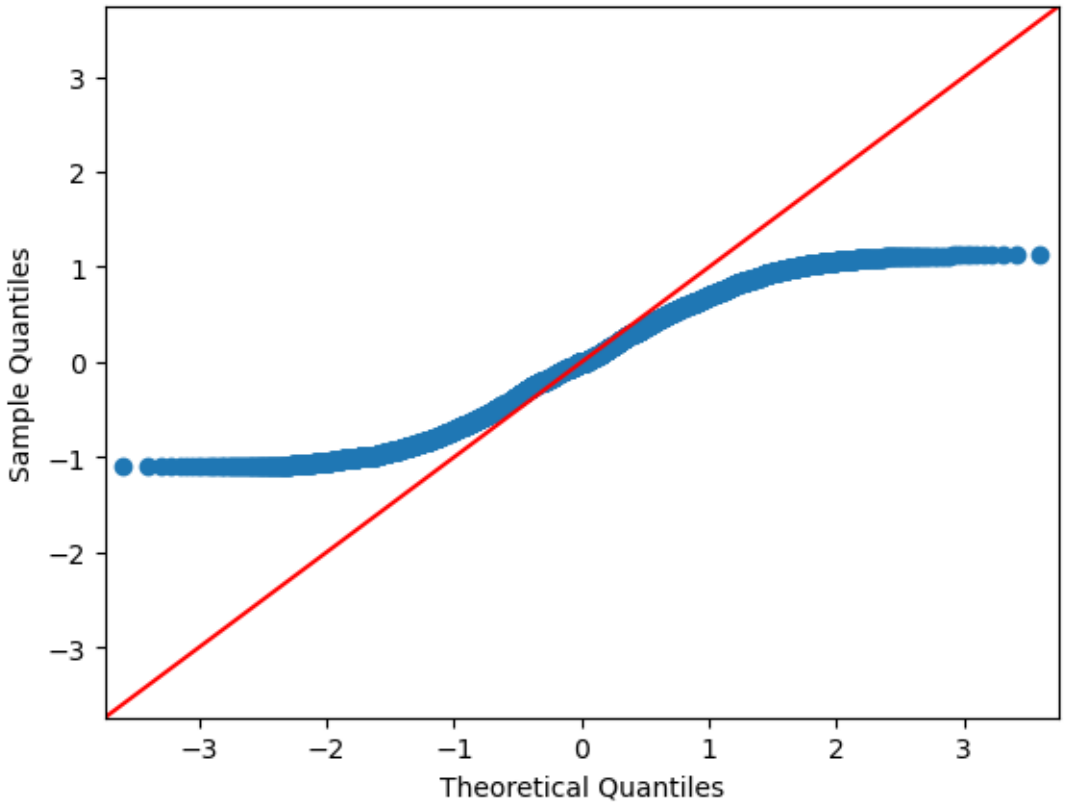
Plasticizer (gm)



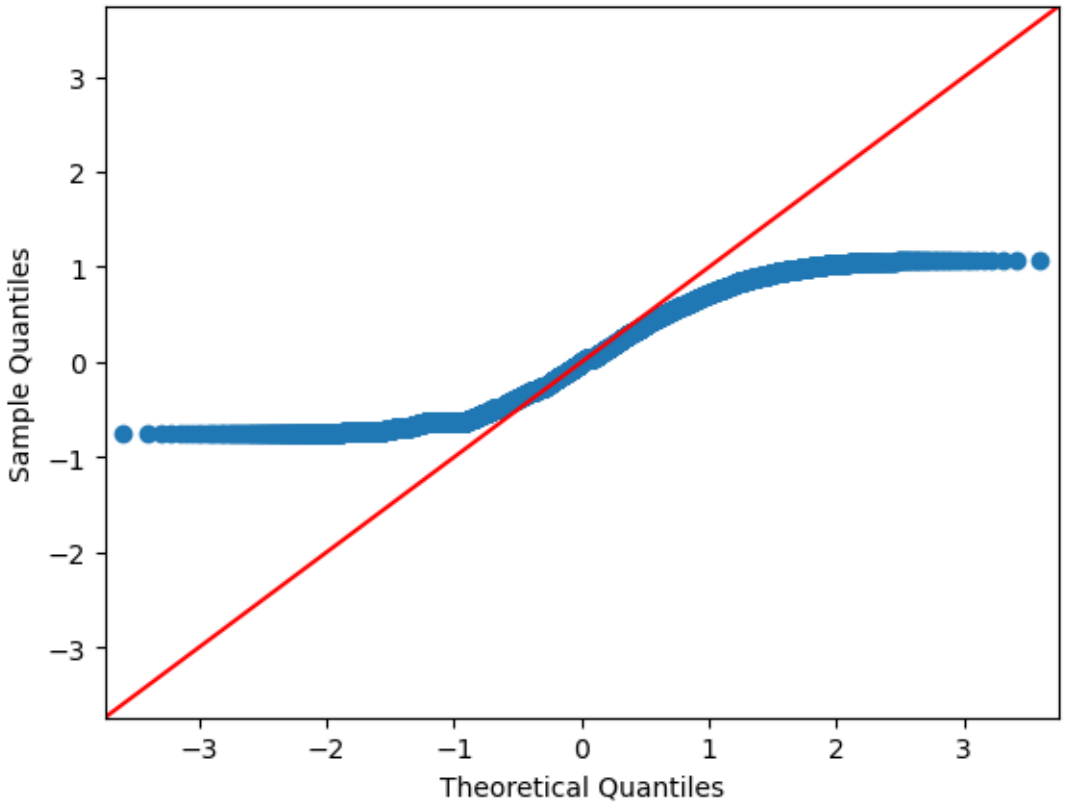
Moderate Aggregator



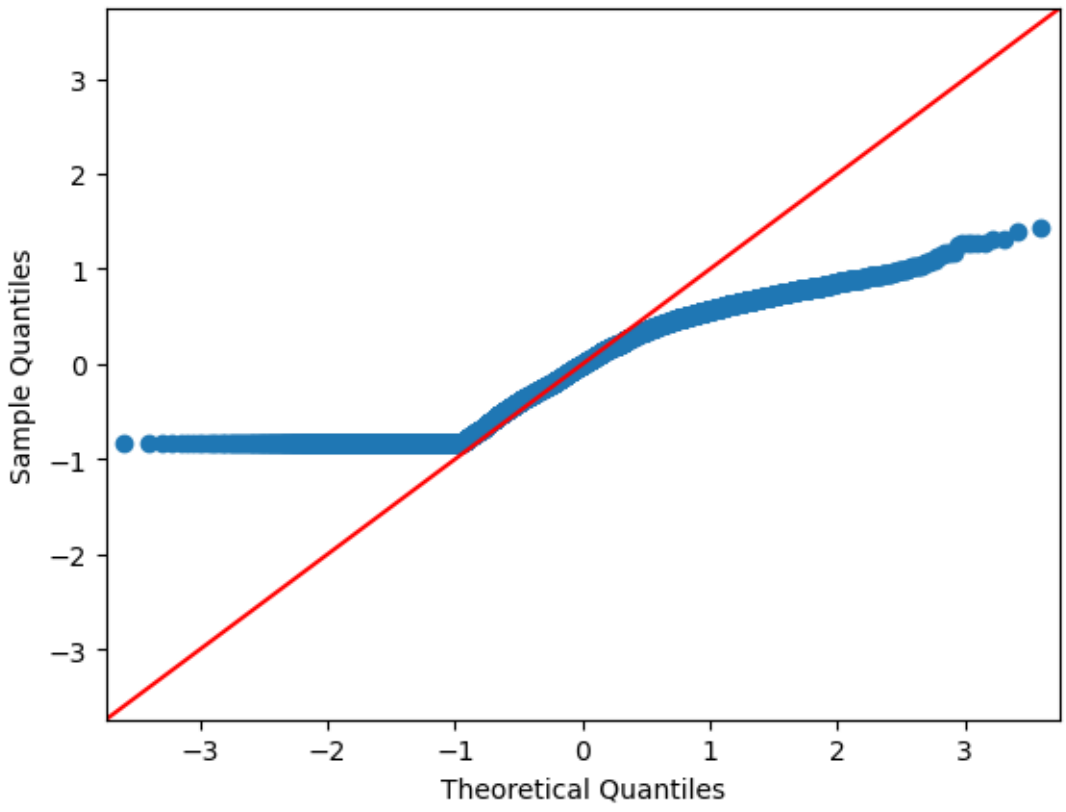
Refined Aggregator



Formulation Duration (hrs)



Compression Strength MPa



OBSERVATION:

All the columns are partially normally distributed.

[88]:
```
#converting the columns as normally distributed as possible
```

[89]:
```
from scipy.stats import norm
import statistics
```

[90]:
```
scaled_dataset.describe()
```

[90]:

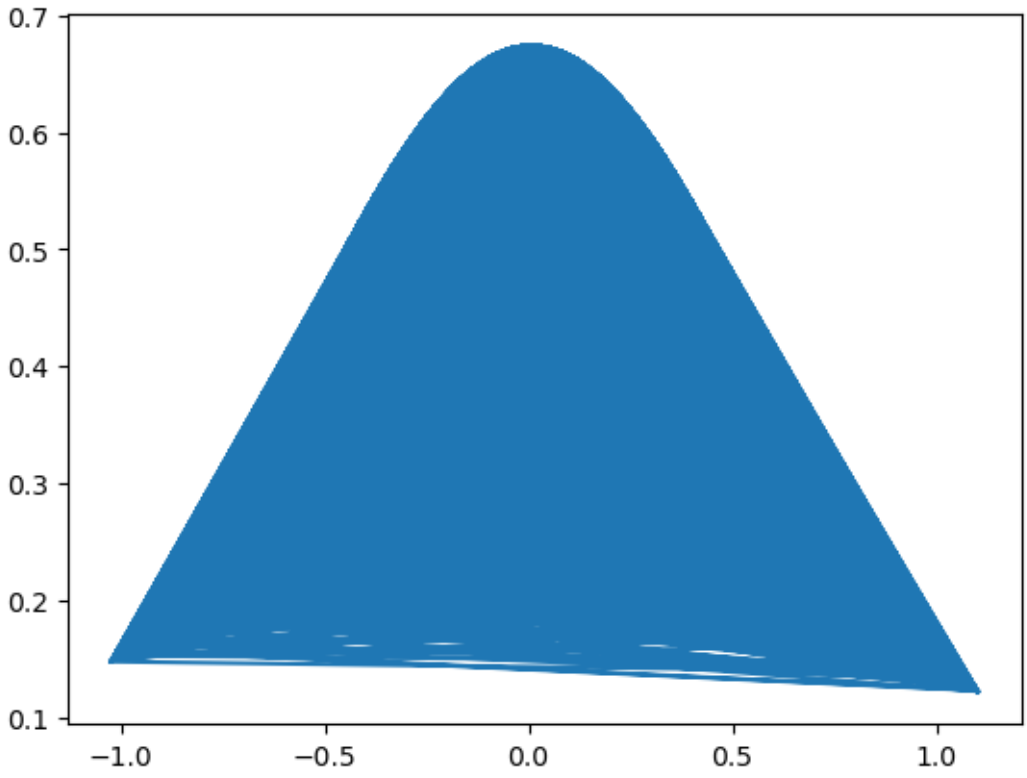
	Material Quantity (gm)	Additive Catalyst (gm)	Ash Component (gm)	\
count	6139.000000	6139.000000	6139.000000	
mean	0.007092	0.001247	-0.011617	
std	0.591562	0.588650	0.579978	
min	-1.024374	-0.875000	-0.893312	
25%	-0.490937	-0.514233	-0.531510	
50%	0.000000	0.000000	0.000000	
75%	0.509063	0.485767	0.468490	
max	1.102051	1.078270	1.030938	

	Water Mix (ml)	Plasticizer (gm)	Moderate Aggregator	\

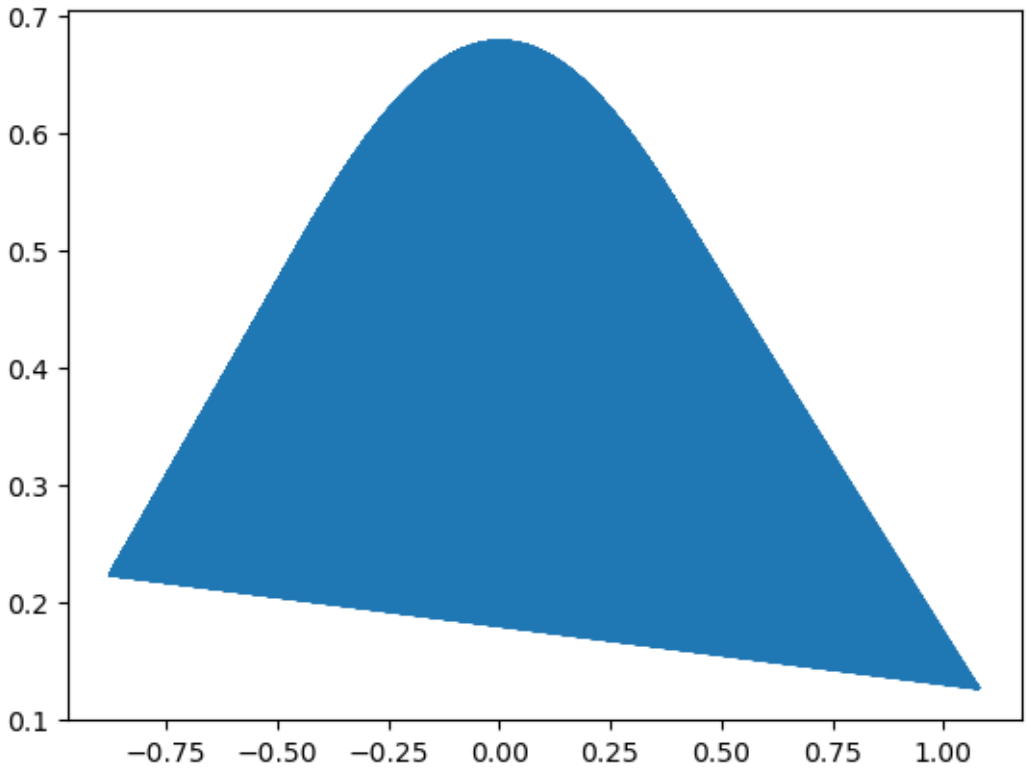
```
count    6.139000e+03       6139.000000         6.139000e+03
mean     6.121725e-15          0.045405        -7.326568e-15
std      6.299275e-01          0.596945         6.101289e-01
min     -1.158066e+00         -0.864210        -1.115740e+00
25%     -5.048873e-01         -0.450657        -4.960117e-01
50%      0.000000e+00          0.000000         0.000000e+00
75%      4.951127e-01          0.549343         5.039883e-01
max      1.178659e+00          1.160010         1.106678e+00

       Refined Aggregator  Formulation Duration (hrs)  \
count        6.139000e+03                 6139.000000
mean        -7.802504e-15                    0.035362
std          6.089726e-01                    0.558949
min         -1.103499e+00                   -0.758109
25%         -5.005033e-01                   -0.482177
50%          0.000000e+00                    0.000000
75%          4.994967e-01                    0.517823
max          1.123190e+00                    1.068055

       Compression Strength MPa
count                6139.000000
mean                   -0.044476
std                     0.553810
min                    -0.837588
25%                    -0.558216
50%                     0.000000
75%                     0.441784
max                     1.437610
```

[91]:
```
for col in dataset:
    print(col)
    mean = statistics.mean(scaled_dataset[col])
    sd = statistics.stdev(scaled_dataset[col])
    plt.plot(scaled_dataset[col], norm.pdf(scaled_dataset[col], mean, sd))
    plt.show()
```

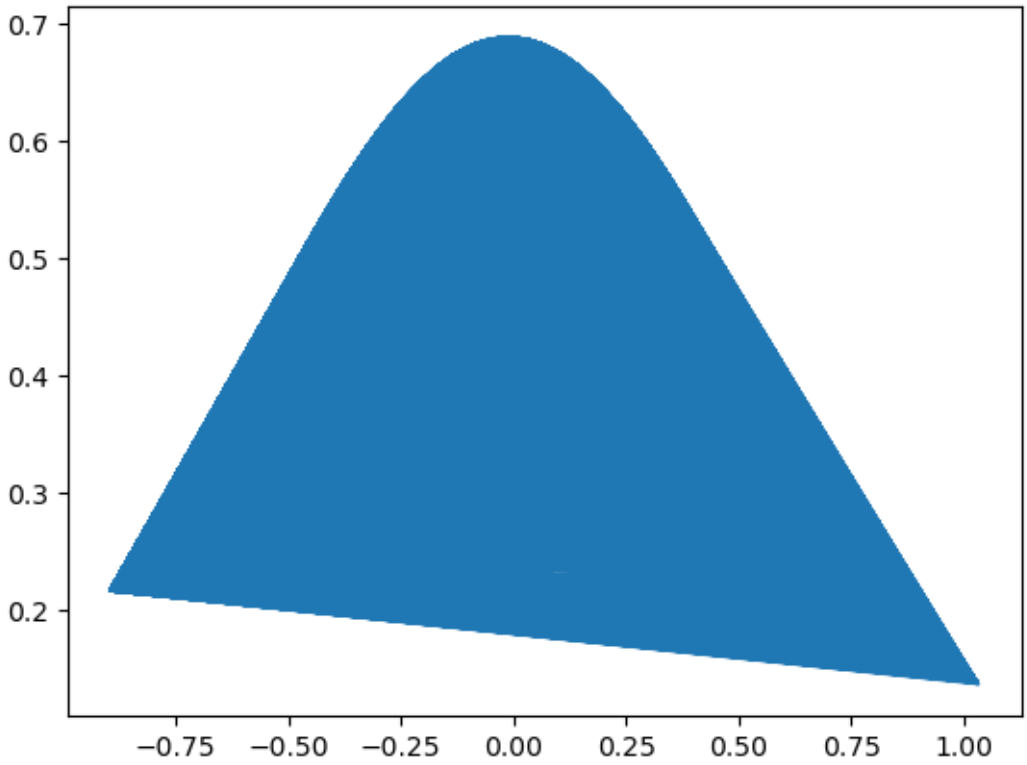
```
Material Quantity (gm)
```



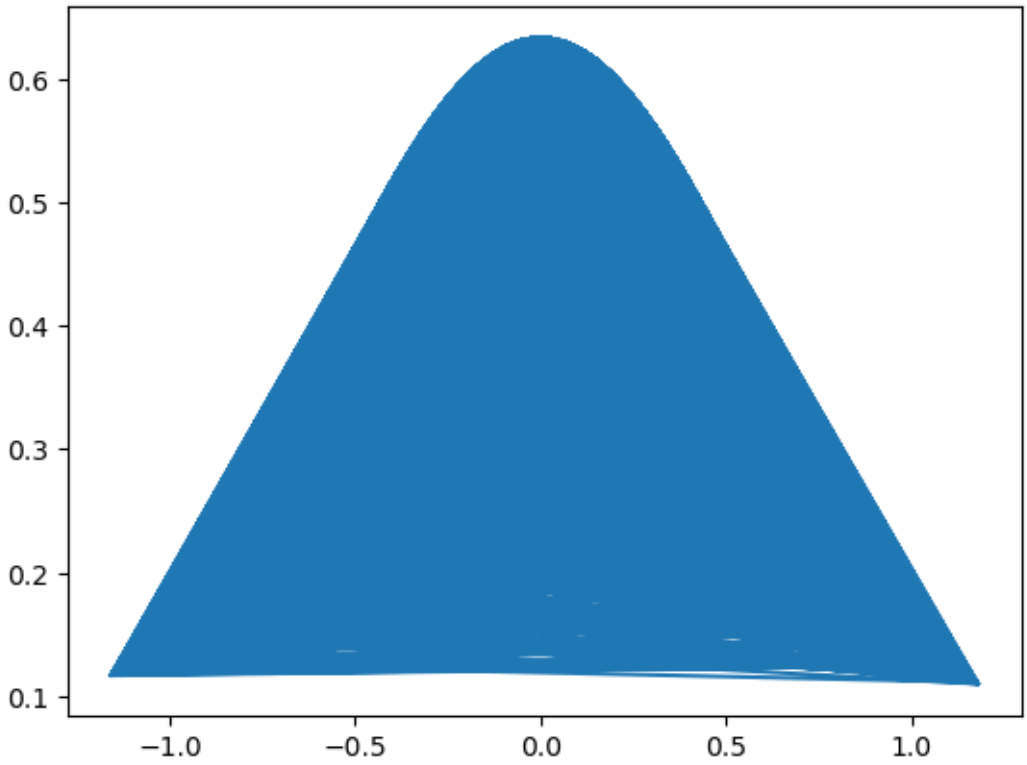
Additive Catalyst (gm)



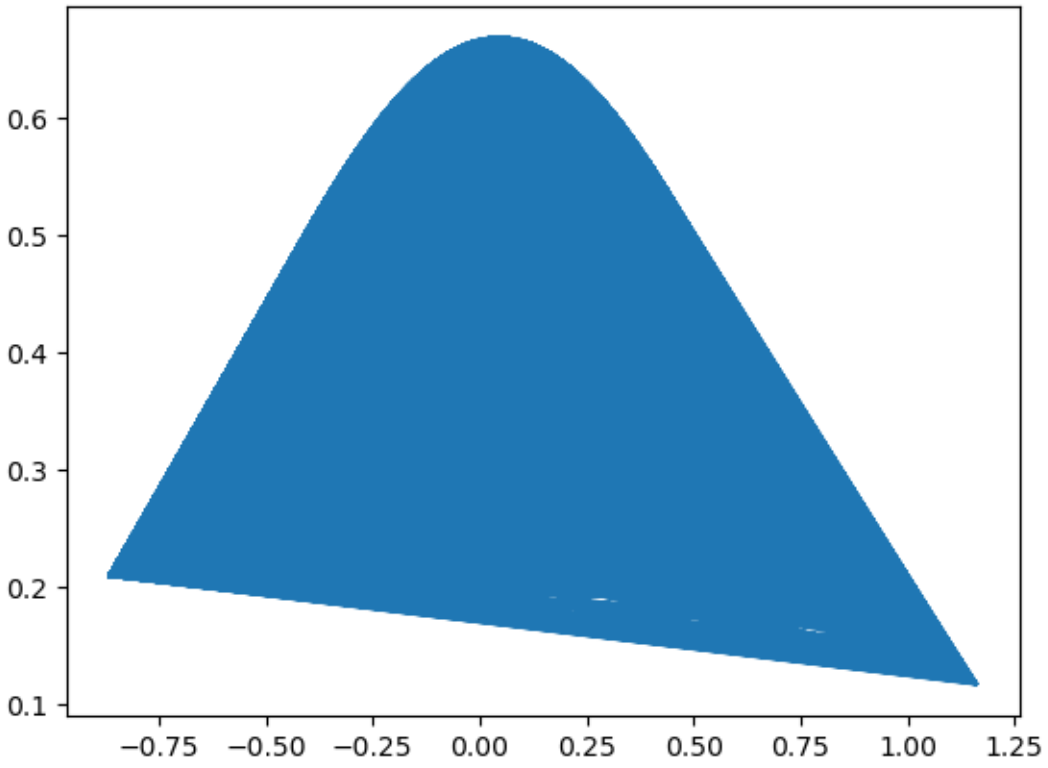
Ash Component (gm)



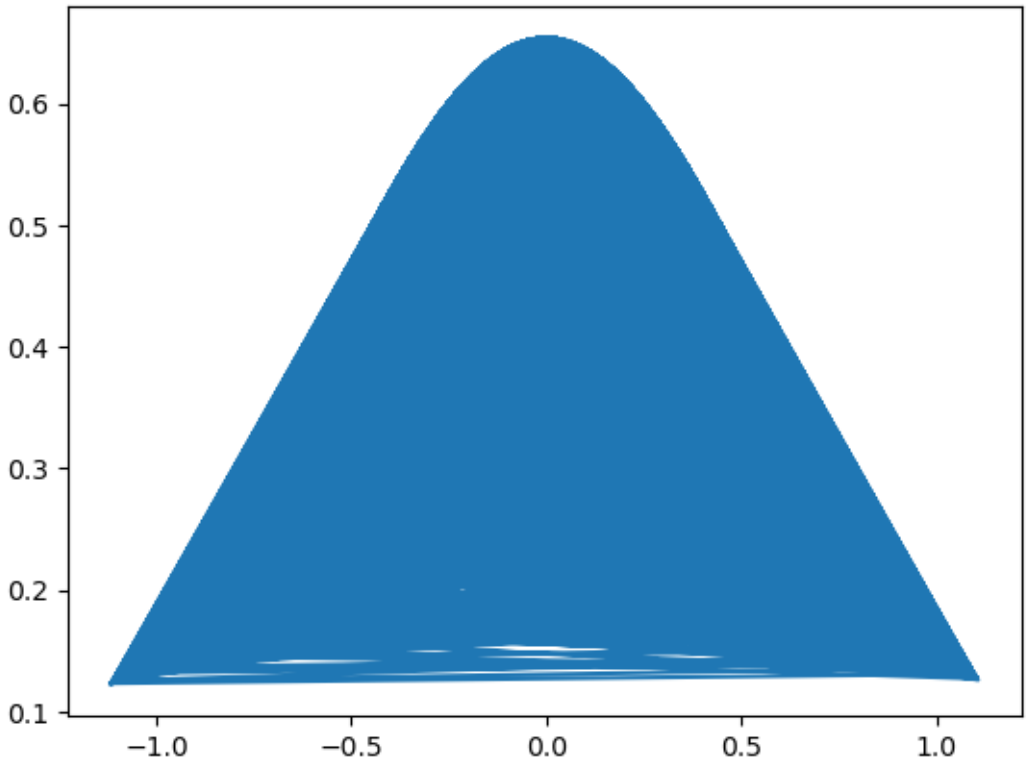
Water Mix (ml)



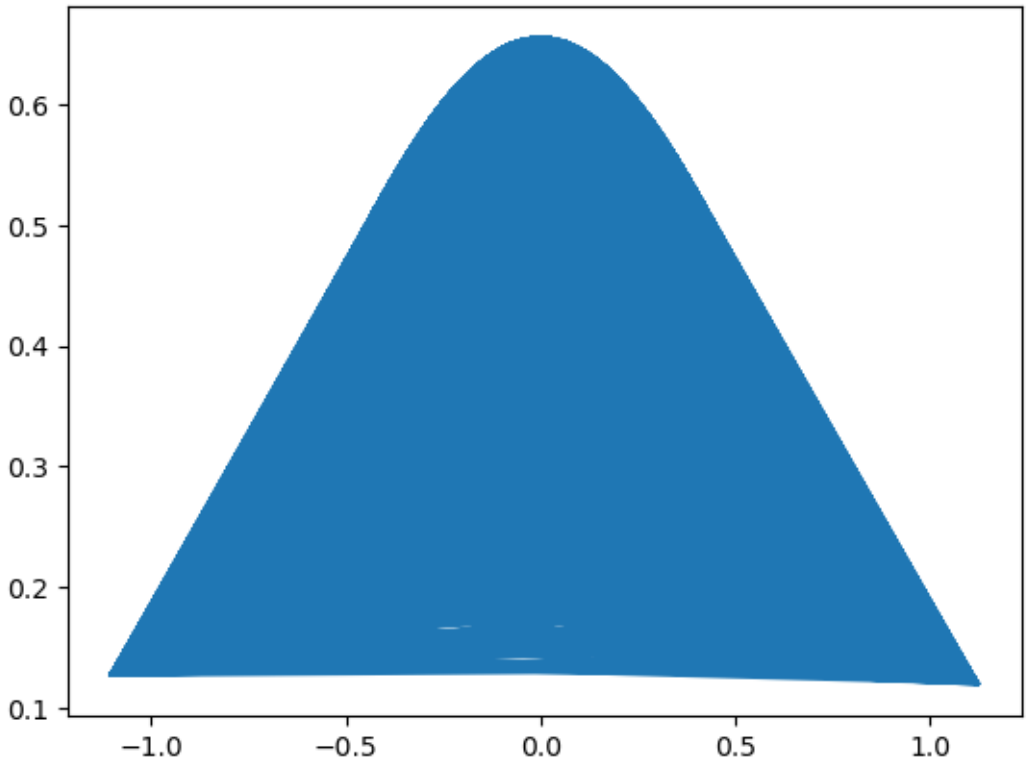
Plasticizer (gm)



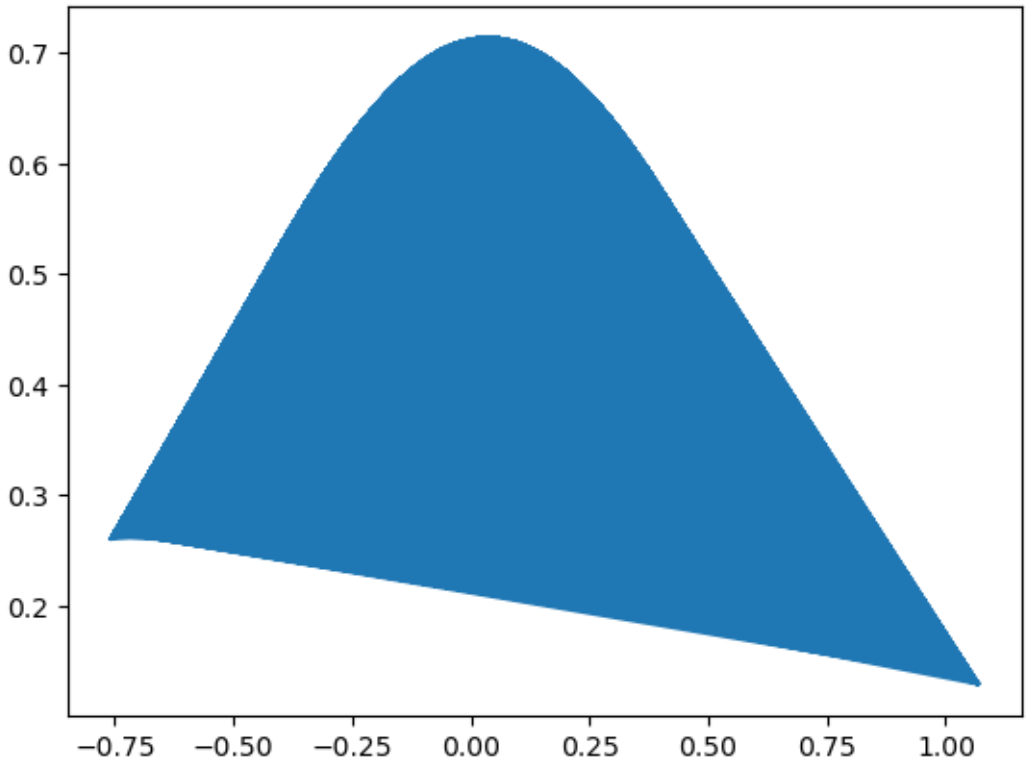
Moderate Aggregator



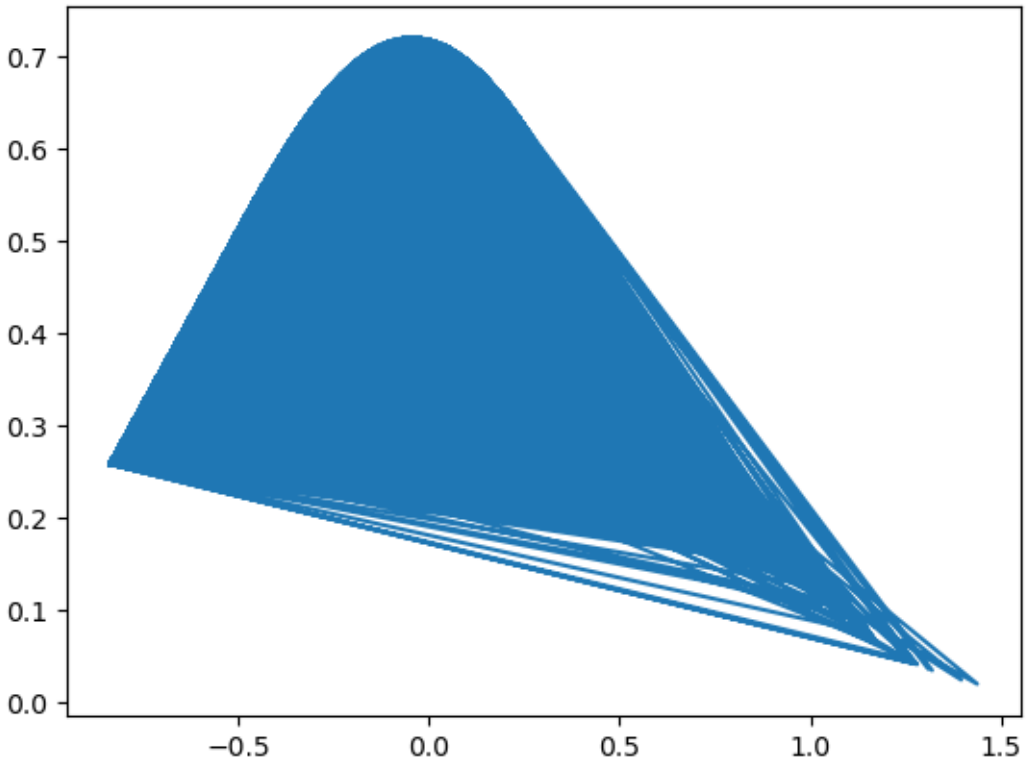
Refined Aggregator



Formulation Duration (hrs)



Compression Strength MPa



OBSERVATION:

All the columns are now normally distributed with some skewness.

[92]:
```
dataset.skew()
```

[92]:
```
Material Quantity (gm)         0.096605
Additive Catalyst (gm)         0.107584
Ash Component (gm)            -0.001224
Water Mix (ml)                 0.024953
Plasticizer (gm)               0.182842
Moderate Aggregator           -0.020582
Refined Aggregator            -0.006749
Formulation Duration (hrs)     0.233290
Compression Strength MPa      -0.066850
dtype: float64
```

OBSERVATION:

The skewness of the data has not increased and remains the same which is good.

[94]:
```
#power transformation
```

[95]:
```
from sklearn.preprocessing import PowerTransformer
```

[124]:
```
pt = PowerTransformer(method= 'yeo-johnson', standardize=True)

skl_yeojohnson = pt.fit(dataset)

skl_yeojohnson = pt.transform(dataset)

skl_yeojohnson
```

[124]:
```
array([[ 7.20371102e-01,  6.97379005e-02, -1.20102790e+00, …,
        -9.21750952e-01,  1.34372875e+00,  1.65877466e+00],
       [-1.86476786e+00,  5.67781576e-01,  9.47296921e-01, …,
         1.75068625e+00, -1.52836582e+00,  8.99988955e-02],
       [ 1.15464554e+00, -1.80899695e+00,  1.59480464e-01, …,
         1.43125659e-03,  6.61698860e-01,  1.50190895e+00],
       …,
       [-9.71197757e-02, -1.39226267e+00, -1.27445179e+00, …,
        -1.61933994e-01,  1.09236856e+00, -1.66020126e-01],
       [ 4.67111016e-01,  6.56777001e-01,  8.81033845e-01, …,
         2.01193701e-01,  1.51638200e+00, -3.58810726e-02],
       [ 1.15614339e+00,  6.05019545e-01,  7.63530128e-01, …,
         7.44246035e-01,  1.43913965e+00,  2.35315024e-02]])
```

[125]:
```
#transforming into dataframe

power_dataset = pd.DataFrame(data=skl_yeojohnson, columns=dataset.columns)

power_dataset
```

[125]:

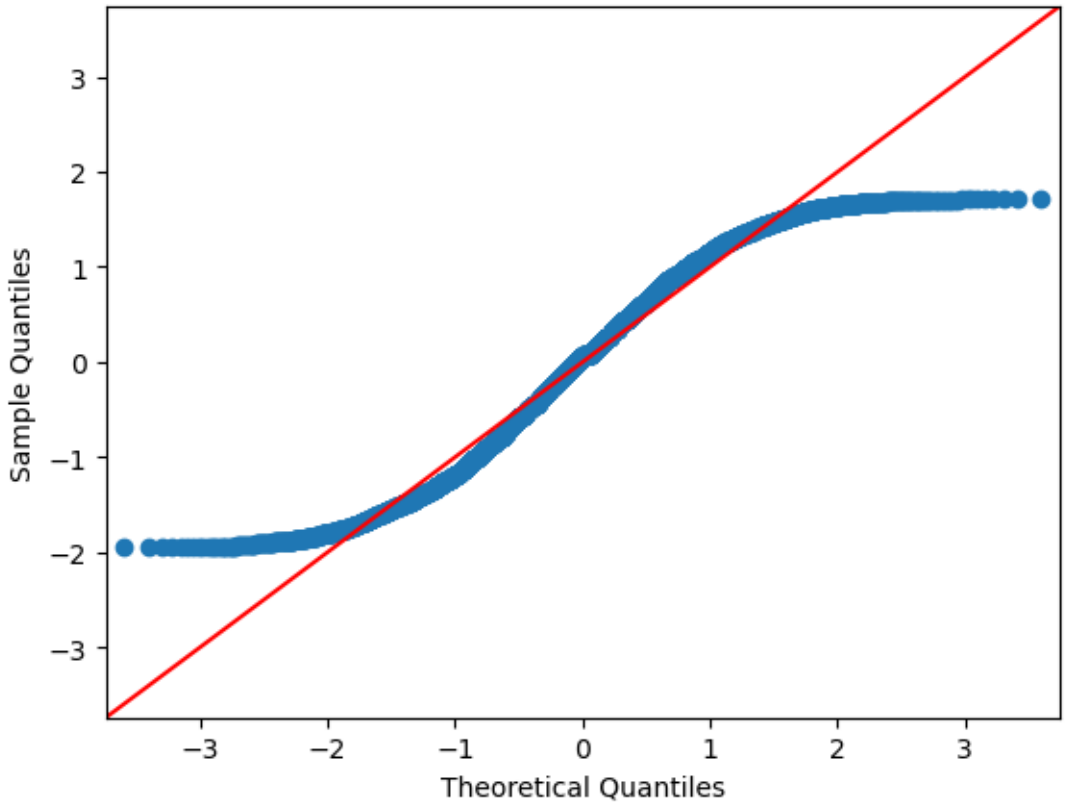
	Material Quantity (gm)	Additive Catalyst (gm)	Ash Component (gm)	\
0	0.720371	0.069738	-1.201028	
1	-1.864768	0.567782	0.947297	
2	1.154646	-1.808997	0.159480	
3	0.123089	1.064805	-0.287432	
4	0.144974	1.072836	1.031987	
…	…	…	…	
6134	-1.366528	-0.056941	0.507475	
6135	-0.154205	0.746021	-0.268029	
6136	-0.097120	-1.392263	-1.274452	
6137	0.467111	0.656777	0.881034	
6138	1.156143	0.605020	0.763530	

	Water Mix (ml)	Plasticizer (gm)	Moderate Aggregator	\
0	-0.531744	0.011636	1.568624	
1	-1.181059	-0.925826	0.949050	
2	1.679431	-0.349741	0.274745	
3	1.767549	0.243249	1.402344	
4	0.294030	0.086055	1.027621	

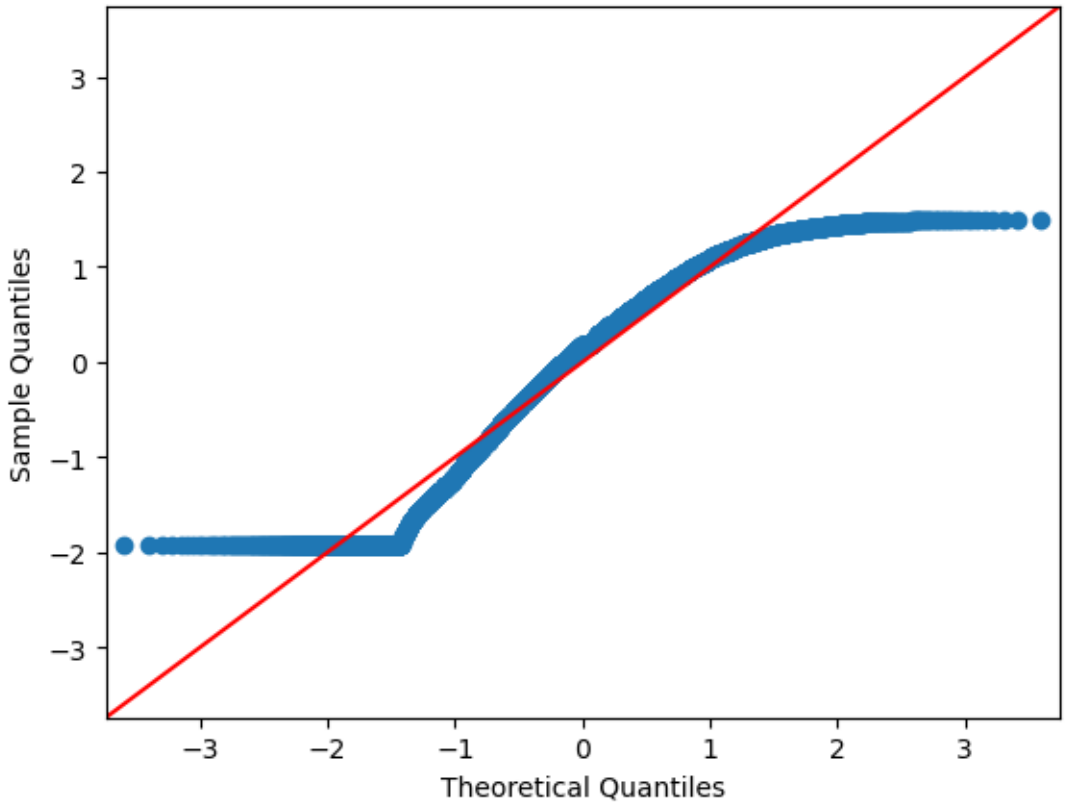
```
…              …               …                  …
6134       -1.501392         0.000884            0.057143
6135       -0.867893         0.744619           -0.757687
6136       -0.360384         1.343057            0.851669
6137       -0.779271         0.170183           -1.382034
6138       -1.193936        -0.473275            1.718509

      Refined Aggregator  Formulation Duration (hrs)  Compression Strength MPa
0              -0.921751                    1.343729                  1.658775
1               1.750686                   -1.528366                  0.089999
2               0.001431                    0.661699                  1.501909
3               0.628186                    0.451540                  1.026880
4              -0.264571                    0.861219                  1.363305
…                    …                           …                         …
6134            1.683653                    1.421704                 -0.641690
6135            1.708133                   -0.472742                 -0.351456
6136           -0.161934                    1.092369                 -0.166020
6137            0.201194                    1.516382                 -0.035881
6138            0.744246                    1.439140                  0.023532

[6139 rows x 9 columns]
```

[97]:
```
import statsmodels.api as sm
for col in power_dataset:
    print(col)
    sm.qqplot(power_dataset[col], line = '45')
    plt.show()
```

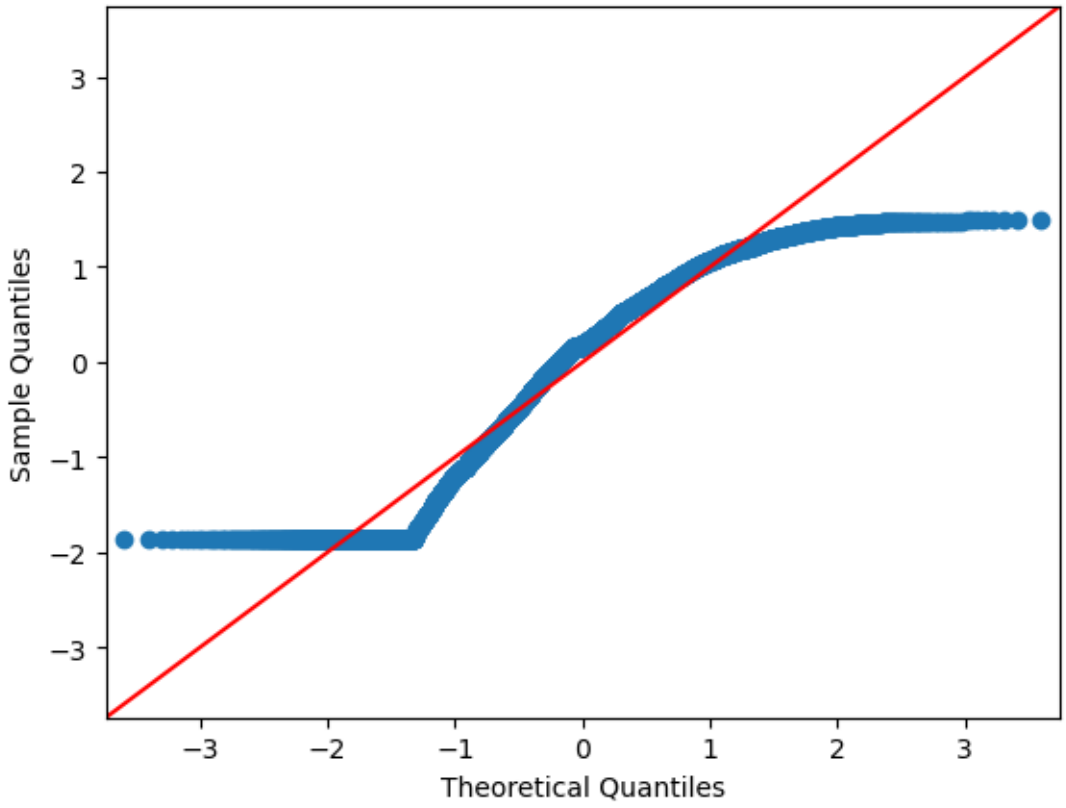
```
Material Quantity (gm)
```



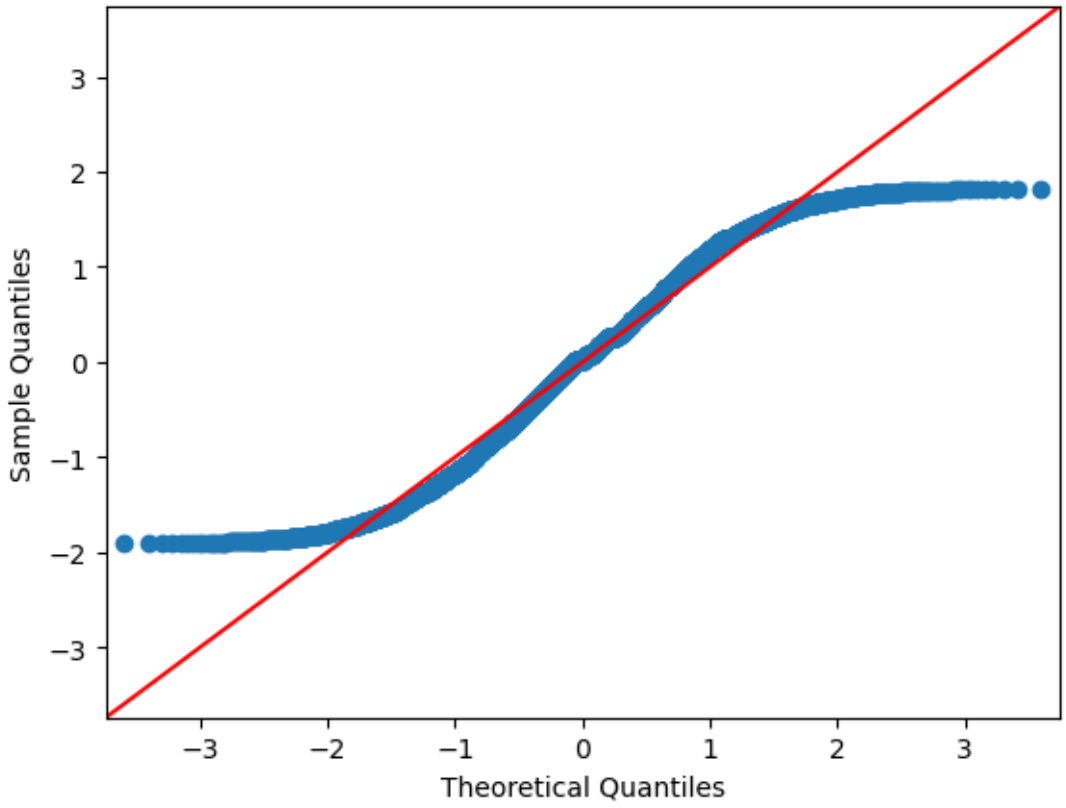
Additive Catalyst (gm)



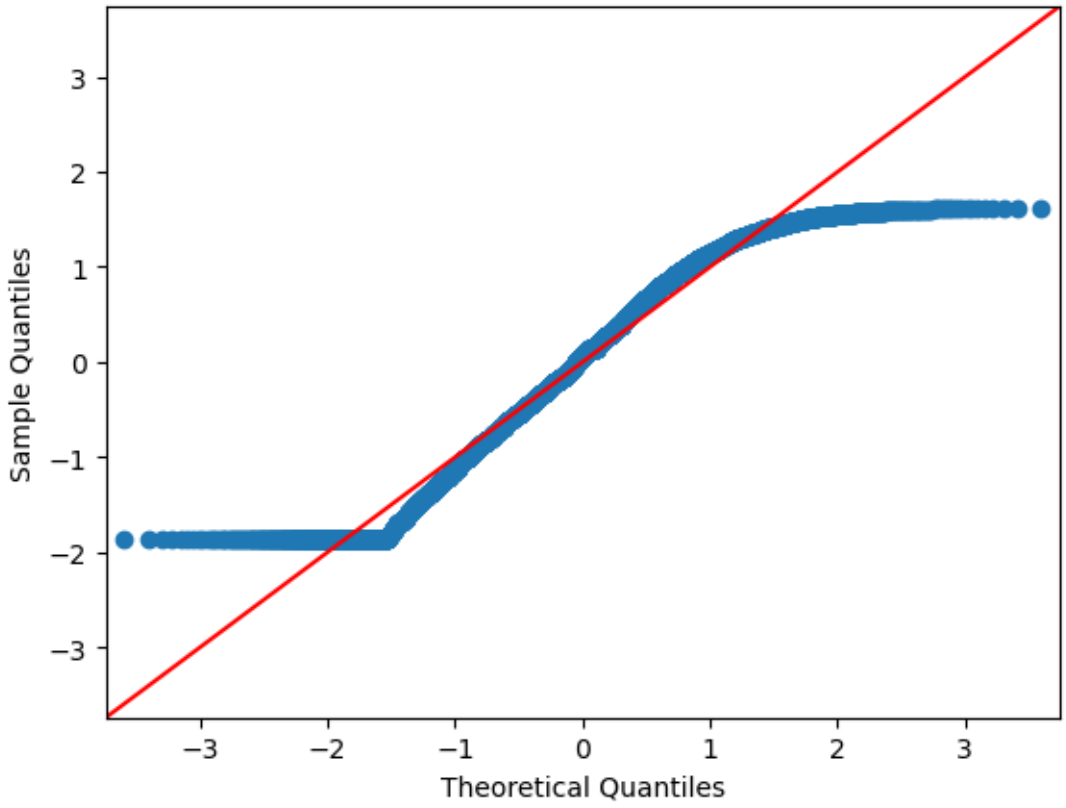
Ash Component (gm)



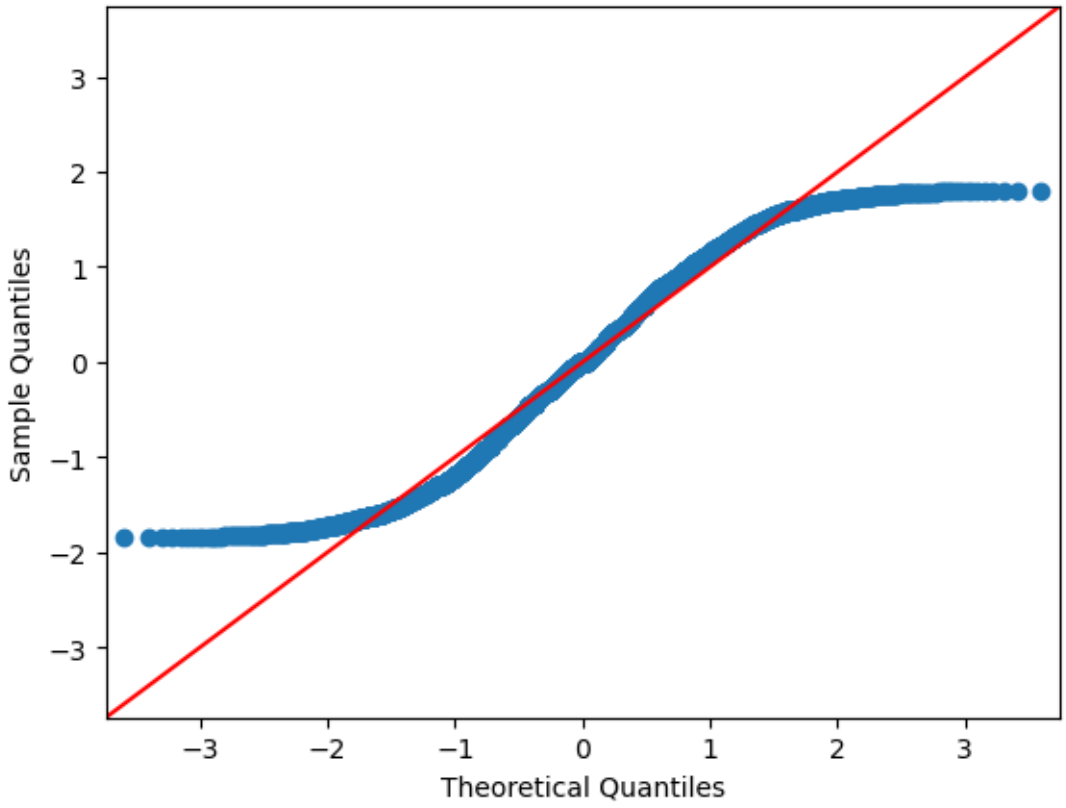
`Water Mix (ml)`



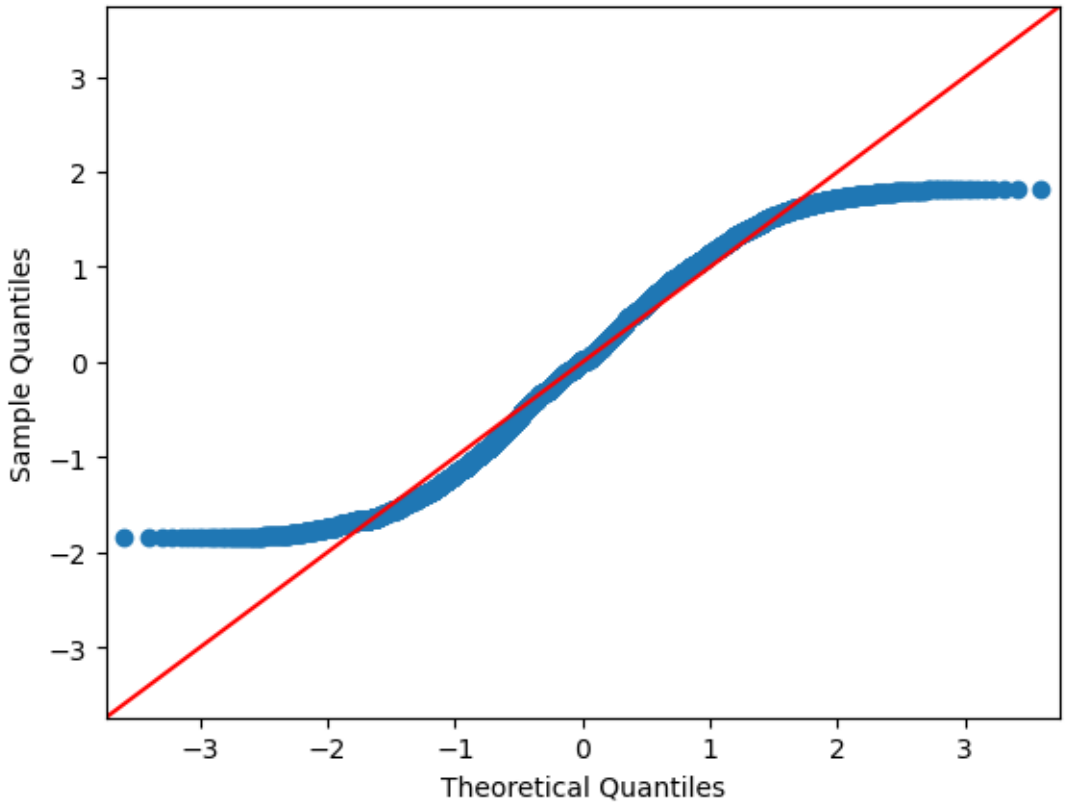
Plasticizer (gm)



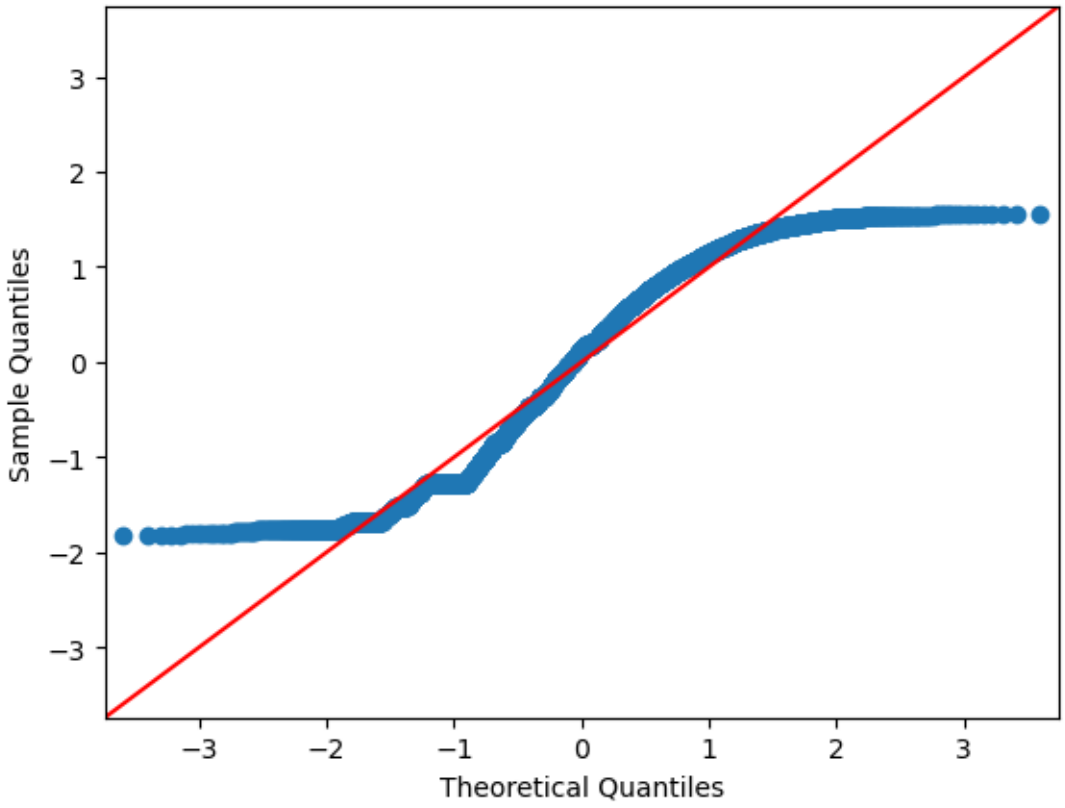
Moderate Aggregator



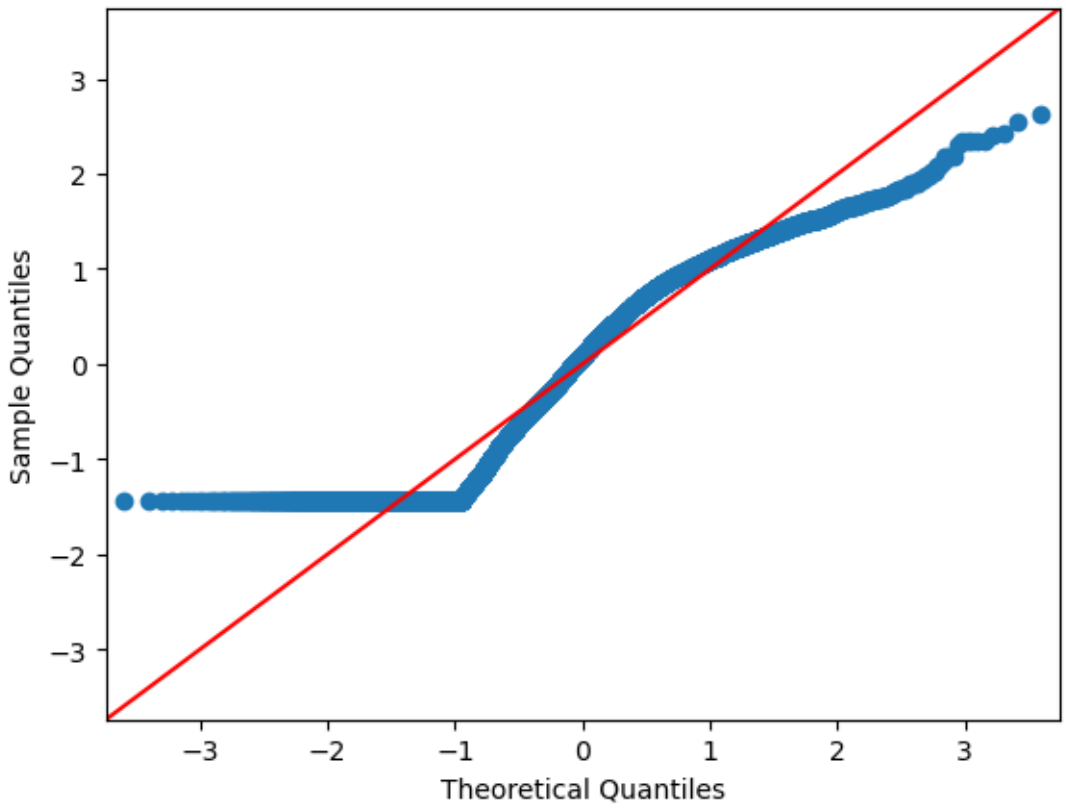
Refined Aggregator



`Formulation Duration (hrs)`



Compression Strength MPa



OBSERVATION:

The columns are more gaussian distributed than before

[ ]: