In [1]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\11_winequality-red - 11_winequality-red.csv",low_memor
df
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alco
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1495	7.0	0.430	0.02	1.9	0.080	15.0	28.0	0.99492	3.35	0.81	1
1496	7.7	0.540	0.26	1.9	0.089	23.0	147.0	0.99636	3.26	0.59	
1497	6.9	0.740	0.03	2.3	0.054	7.0	16.0	0.99508	3.45	0.63	1
1498	6.6	0.895	0.04	2.3	0.068	7.0	13.0	0.99582	3.53	0.58	1
1499	6.9	0.740	0.03	2.3	0.054	7.0	16.0	0.99508	3.45	0.63	1

1500 rows × 12 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1500 non-null   float64
 1   volatile acidity      1500 non-null   float64
 2   citric acid           1500 non-null   float64
 3   residual sugar        1500 non-null   float64
 4   chlorides             1500 non-null   float64
 5   free sulfur dioxide   1500 non-null   float64
 6   total sulfur dioxide  1500 non-null   float64
 7   density               1500 non-null   float64
 8   pH                    1500 non-null   float64
 9   sulphates             1500 non-null   float64
 10  alcohol               1500 non-null   float64
 11  quality               1500 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 140.8 KB
```

In [4]:
```
#to display top 5 rows
df.head()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

Data cleaning and Pre-Processing

In [5]:
```
#To find null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1500 non-null   float64
 1   volatile acidity      1500 non-null   float64
 2   citric acid           1500 non-null   float64
 3   residual sugar        1500 non-null   float64
 4   chlorides             1500 non-null   float64
 5   free sulfur dioxide   1500 non-null   float64
 6   total sulfur dioxide  1500 non-null   float64
 7   density               1500 non-null   float64
 8   pH                    1500 non-null   float64
 9   sulphates             1500 non-null   float64
 10  alcohol               1500 non-null   float64
 11  quality               1500 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 140.8 KB
```

In [6]:
```
# To display summary of statistics
df.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulf dioxi
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.0000
mean	8.415267	0.526203	0.274980	2.543900	0.088241	15.608667	46.7853
std	1.741944	0.179831	0.195533	1.404742	0.048021	10.463230	33.2505
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0000
25%	7.200000	0.390000	0.100000	1.900000	0.071000	7.000000	22.0000
50%	8.000000	0.520000	0.260000	2.200000	0.080000	13.000000	38.0000
75%	9.300000	0.635000	0.430000	2.600000	0.091000	21.000000	63.0000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.0000

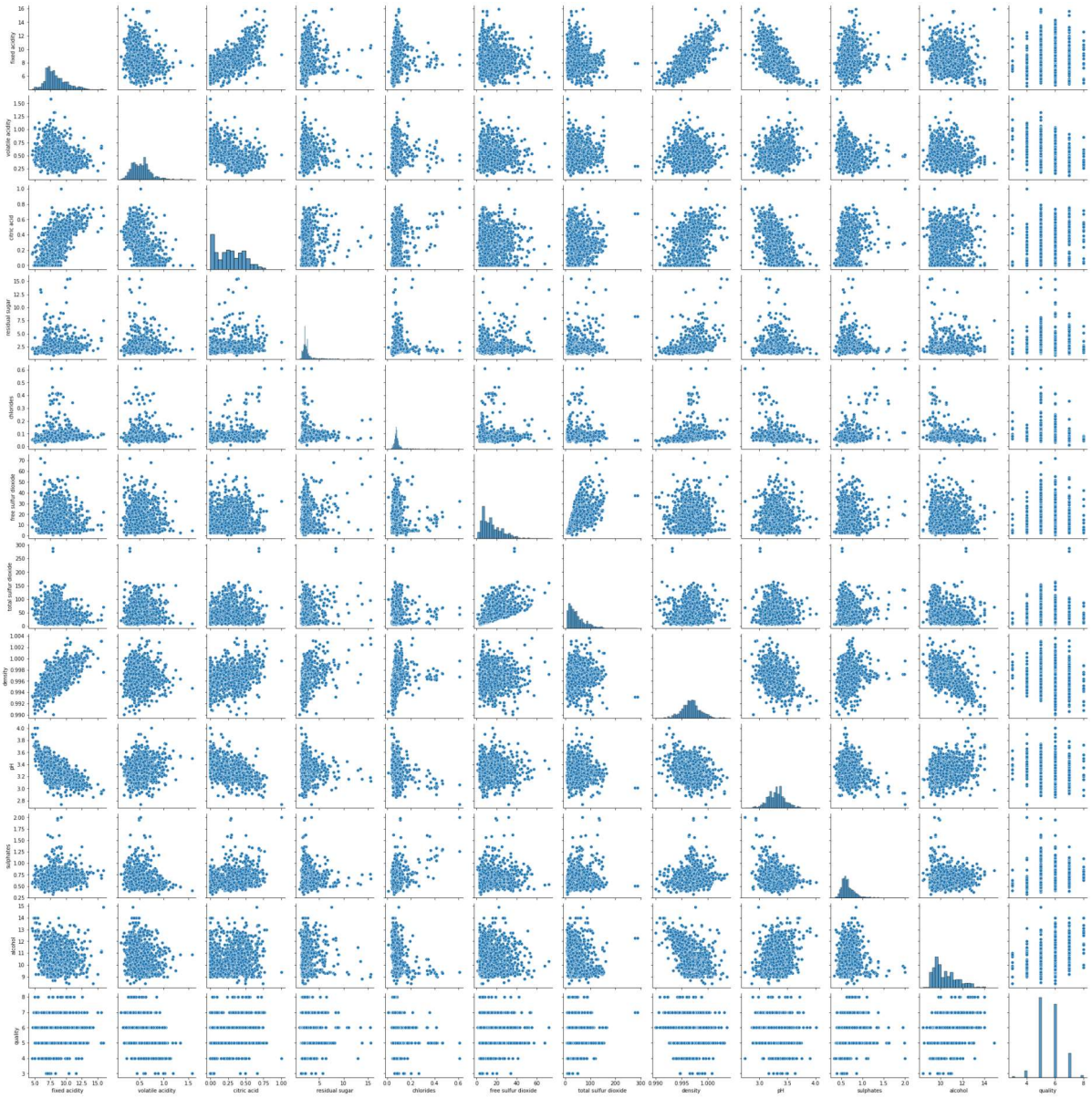
In [7]:
```
#To Display column heading
df.columns
```

Out[7]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

EDA and VISUALIZATION

In [8]:
```
sns.pairplot(df)
```

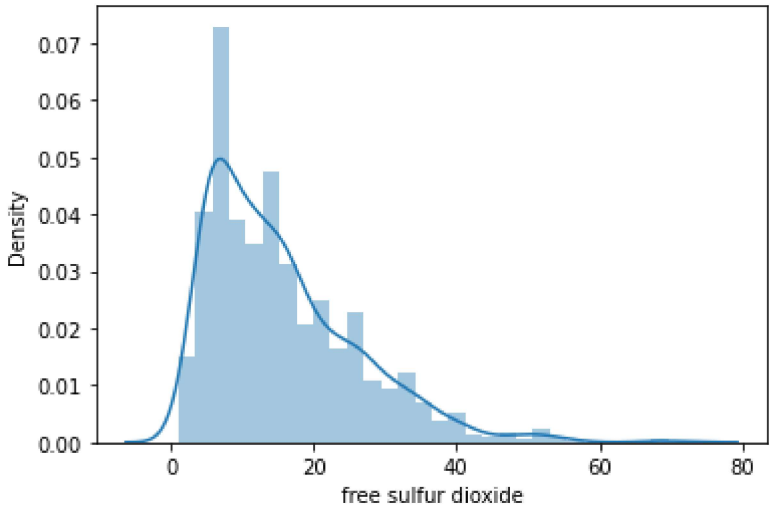
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1e5a5d13b20>
```



In [9]:
```
sns.distplot(df['free sulfur dioxide'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:
```
<AxesSubplot:xlabel='free sulfur dioxide', ylabel='Density'>
```

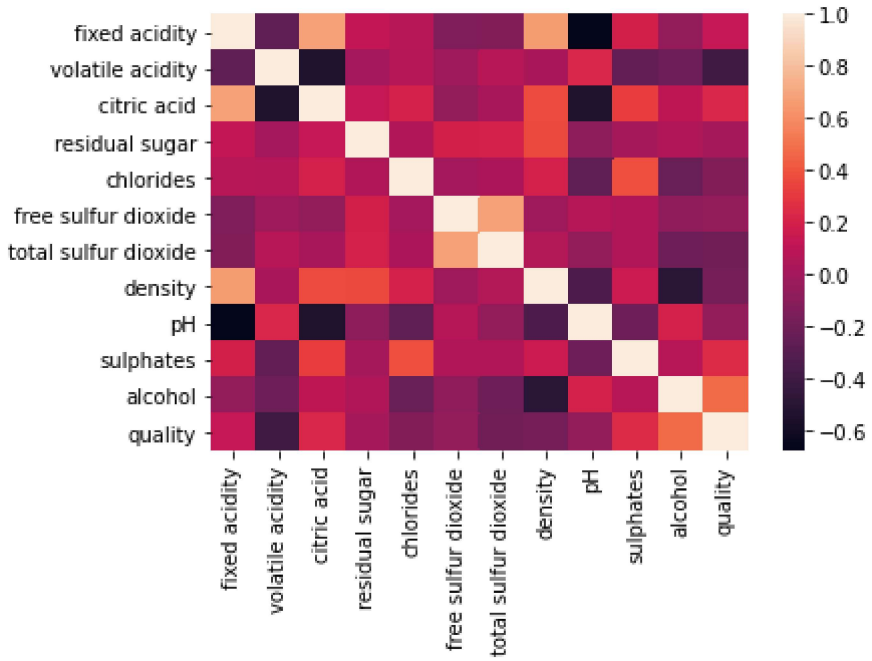


In [10]:
```
df1=df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

Out[11]: <AxesSubplot:>



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [12]:
```
x=df1[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide','density',
       'pH', 'sulphates', 'alcohol', 'quality']]
y=df1[ 'total sulfur dioxide']
```

To Split my dataset into training and test data

In [13]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]: LinearRegression()

In [15]:
```
lr.intercept_
```

Out[15]:
```
-3429.7606159594256
```

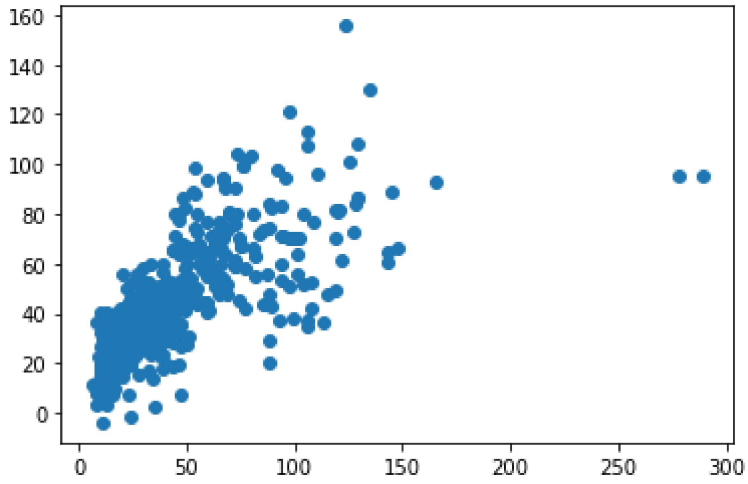
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
fixed acidity	-9.009331
volatile acidity	32.956666
citric acid	48.345679
residual sugar	-0.992319
chlorides	-97.049679
free sulfur dioxide	1.992317
density	3713.697683
pH	-51.404871
sulphates	14.095426
alcohol	-1.777702
quality	-3.811668

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1e5ae573430>
```



In [18]:
```
lr.score(x_test,y_test)
```

Out[18]:
```
0.4984946684392494
```

In [ ]:
```

```

In [ ]: