DATA VISUALIZATION with *Seaborn* & *Matplotlib*:

Objective: To showcase how data can be visualized in various ways, using *Python*.

In [4]:
```
# Matplotlib: Matplotlib is a comprehensive library for creating static, animate
```

Importing Libraries:

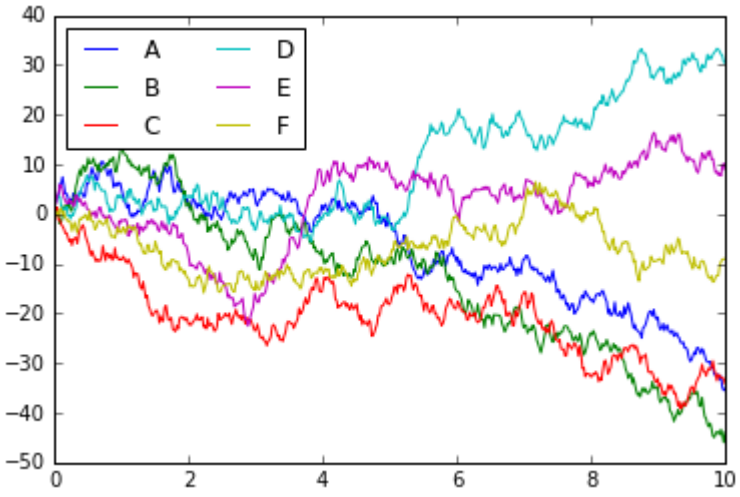
In [5]:
```
import matplotlib.pyplot as plt
plt.style.use('classic')
%matplotlib inline
import numpy as np
import pandas as pd
```

In [6]:
```
#Now we create some random walk data:
```

In [7]:
```
# Create some data
rng = np.random.RandomState(0)
x = np.linspace(0, 10, 500)
y = np.cumsum(rng.randn(500, 6), 0)
```

In [8]:
```
#And do a simple plot:
```

In [9]:
```
# Plot the data with Matplotlib defaults
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
```



In [10]:
```
# Seaborn:  -  Seaborn is an open-source Python library built on top of matplotli
#           -  Seaborn works easily with dataframes and the Pandas library. The gr
```

Importing Seaborn:

In [11]:
```
import seaborn as sns
sns.set()
```

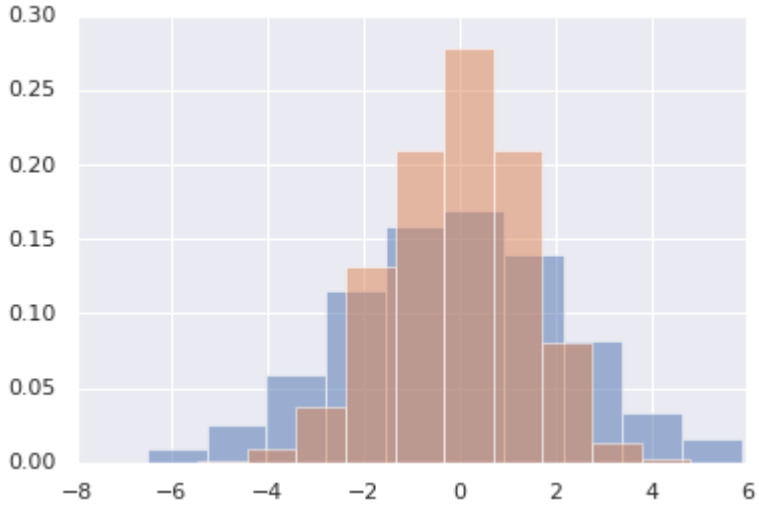
In [12]:
```
# same plotting code as above!
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
```

In [13]:
```
#Exploring Seaborn Plots:
```

In [14]:
```
#Histograms, KDE, and densities
```

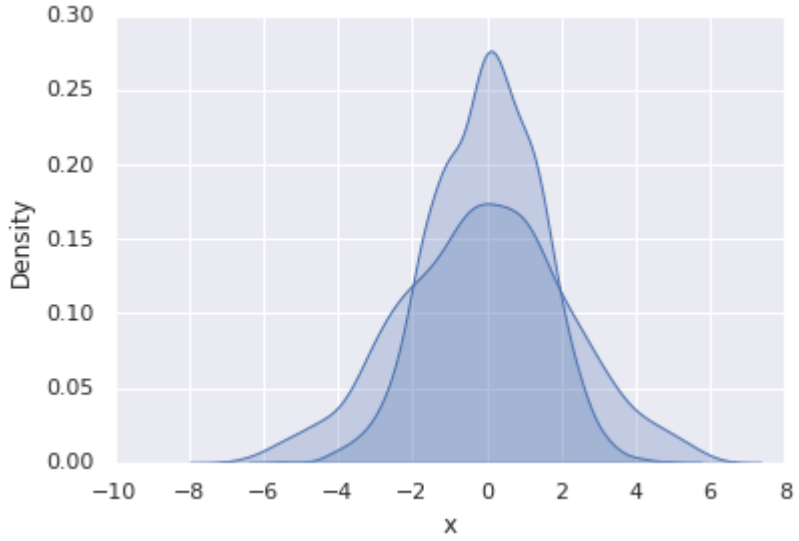
Plotting univariate or bivariate histogram to show distributions of datasets:

In [15]:
```
data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])

for col in 'xy':
    plt.hist(data[col], density=True, alpha=0.5)
```



Rather than a histogram, we can get a smooth estimate of the distribution using a kernel density estimation

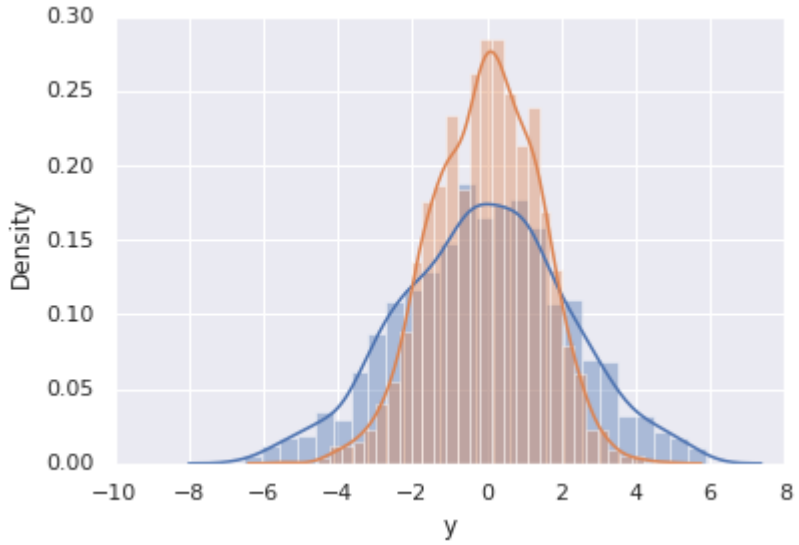
In [16]:
```
for col in 'xy':
    sns.kdeplot(data[col], shade=True)
```



Histograms and KDE can be combined using distplot:

In [17]:
```
sns.distplot(data['x'])
sns.distplot(data['y']);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

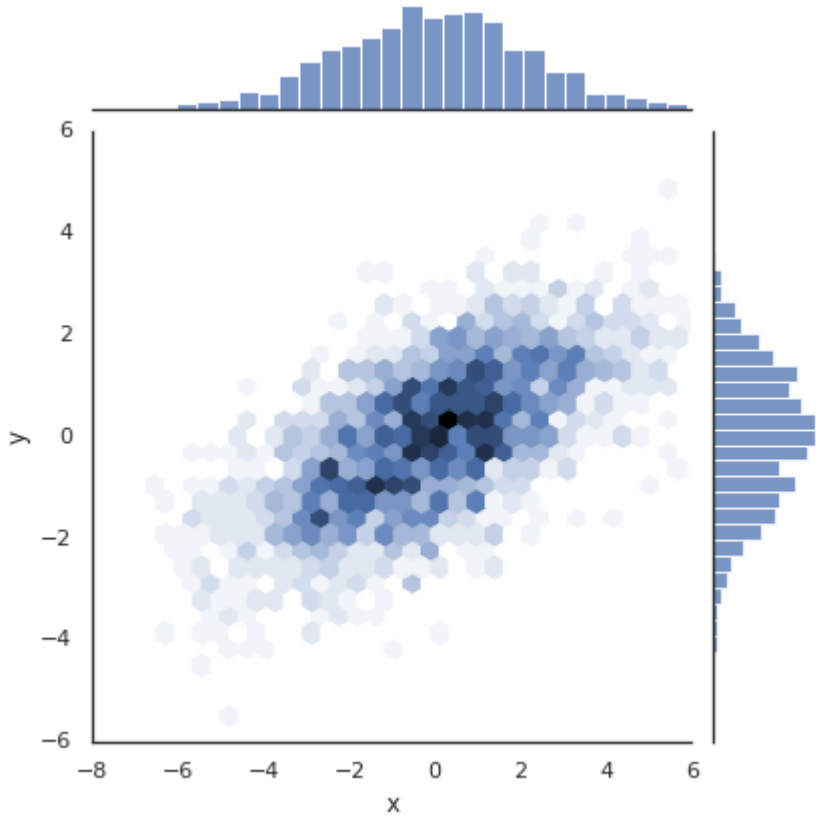


Jointplot() in Seaborn library creates a scatter plot with two histograms at the top and right margins

of the graph by default.

In [18]:
```
with sns.axes_style('white'):
    sns.jointplot("x", "y", data, kind='hex')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y, data. From version 0.12,
the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



Working with Wine quality Dataset:

In [19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
%matplotlib inline
```

In [20]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving winequalityN.csv to winequalityN.csv
```

In [21]:
```
import io
df = pd.read_csv(io.BytesIO(uploaded['winequalityN.csv']))


# Dataset is now stored in a Pandas Dataframe
```

Viewing the Data:

In [22]:
```
df.head(20)
```

Out[22]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	white	7.0	0.27	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	
1	white	6.3	0.30	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	
2	white	8.1	0.28	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	
3	white	7.2	0.23	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	
4	white	7.2	0.23	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40	
5	white	8.1	0.28	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44	
6	white	6.2	0.32	0.16	7.00	0.045	30.0	136.0	0.9949	3.18	0.47	
7	white	7.0	0.27	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45	
8	white	6.3	0.30	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49	
9	white	8.1	0.22	0.43	1.50	0.044	28.0	129.0	0.9938	3.22	0.45	
10	white	8.1	0.27	0.41	1.45	0.033	11.0	63.0	0.9908	2.99	0.56	
11	white	8.6	0.23	0.40	4.20	0.035	17.0	109.0	0.9947	3.14	0.53	
12	white	7.9	0.18	0.37	1.20	0.040	16.0	75.0	0.9920	3.18	0.63	
13	white	6.6	0.16	0.40	1.50	0.044	48.0	143.0	0.9912	3.54	0.52	
14	white	8.3	0.42	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67	
15	white	6.6	0.17	0.38	1.50	0.032	28.0	112.0	0.9914	3.25	0.55	
16	white	6.3	0.48	0.04	1.10	0.046	30.0	99.0	0.9928	3.24	0.36	
17	white	NaN	0.66	0.48	1.20	0.029	29.0	75.0	0.9892	3.33	0.39	
18	white	7.4	0.34	0.42	1.10	0.033	17.0	171.0	0.9917	3.12	0.53	
19	white	6.5	0.31	0.14	7.50	0.044	34.0	133.0	0.9955	3.22	0.50	

Cleaning Null values:

In [23]:
```
df=df.fillna(df.mean())
```

Statistical Description of the dataset:

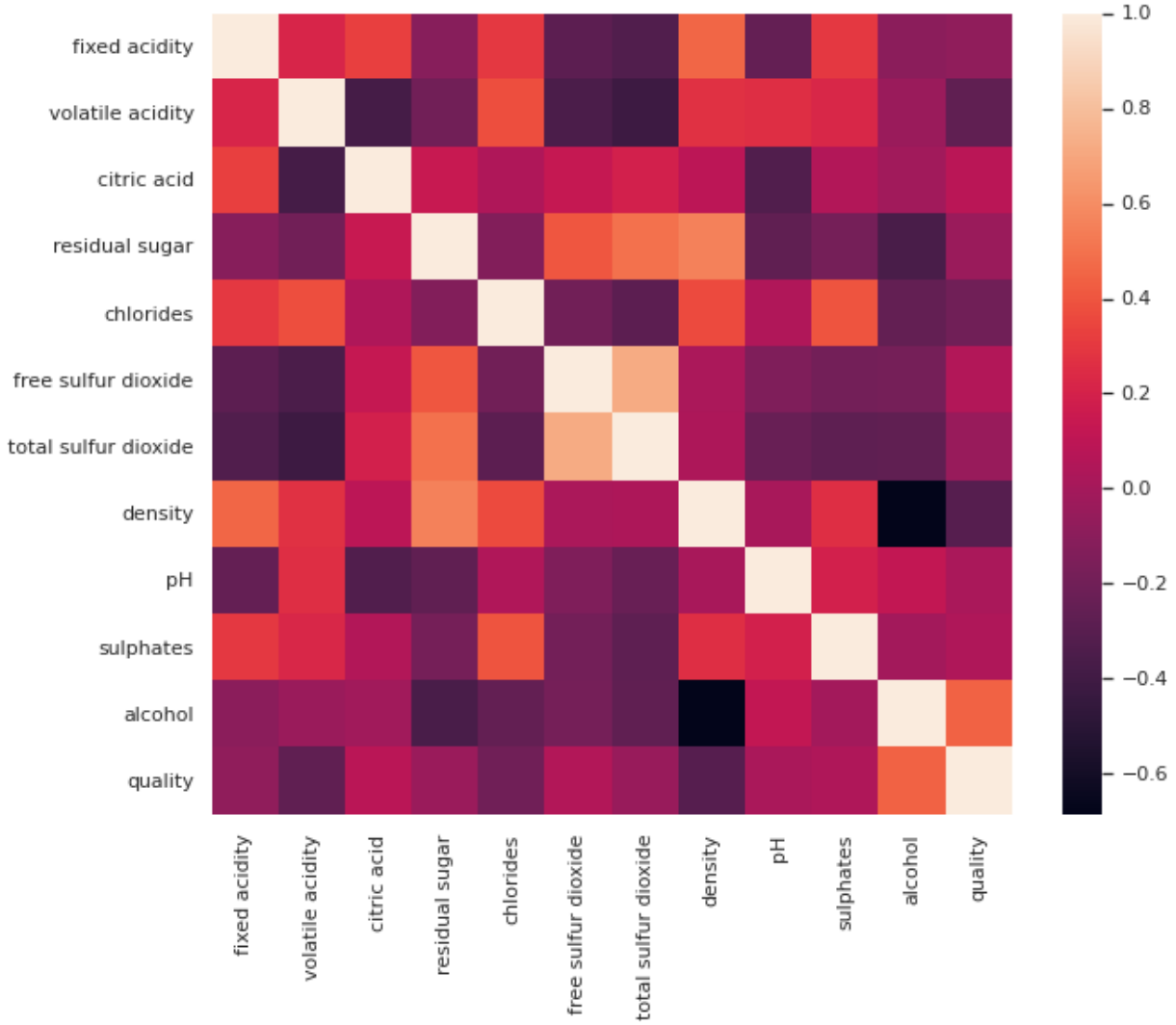
In [24]:
```
df.describe()
```

Out[24]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574
std	1.295751	0.164548	0.145231	4.757392	0.035031	17.749400	56.521855
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000

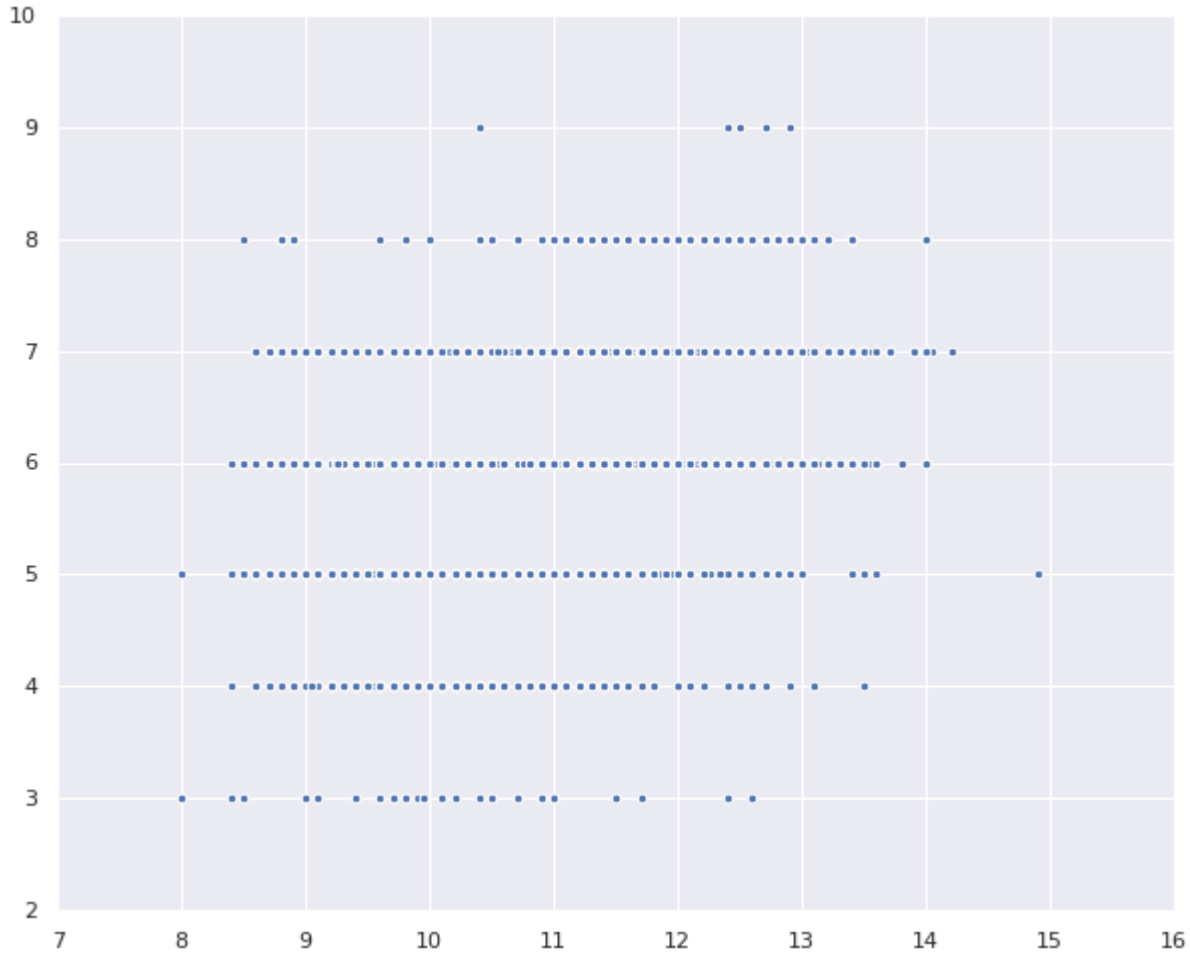
Heatmap is defined as a graphical representation of data using colors to visualize the values.

In [25]:
```
import seaborn as sns
sns.set(rc={'figure.figsize':(10,8)})
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
plt.show()
```



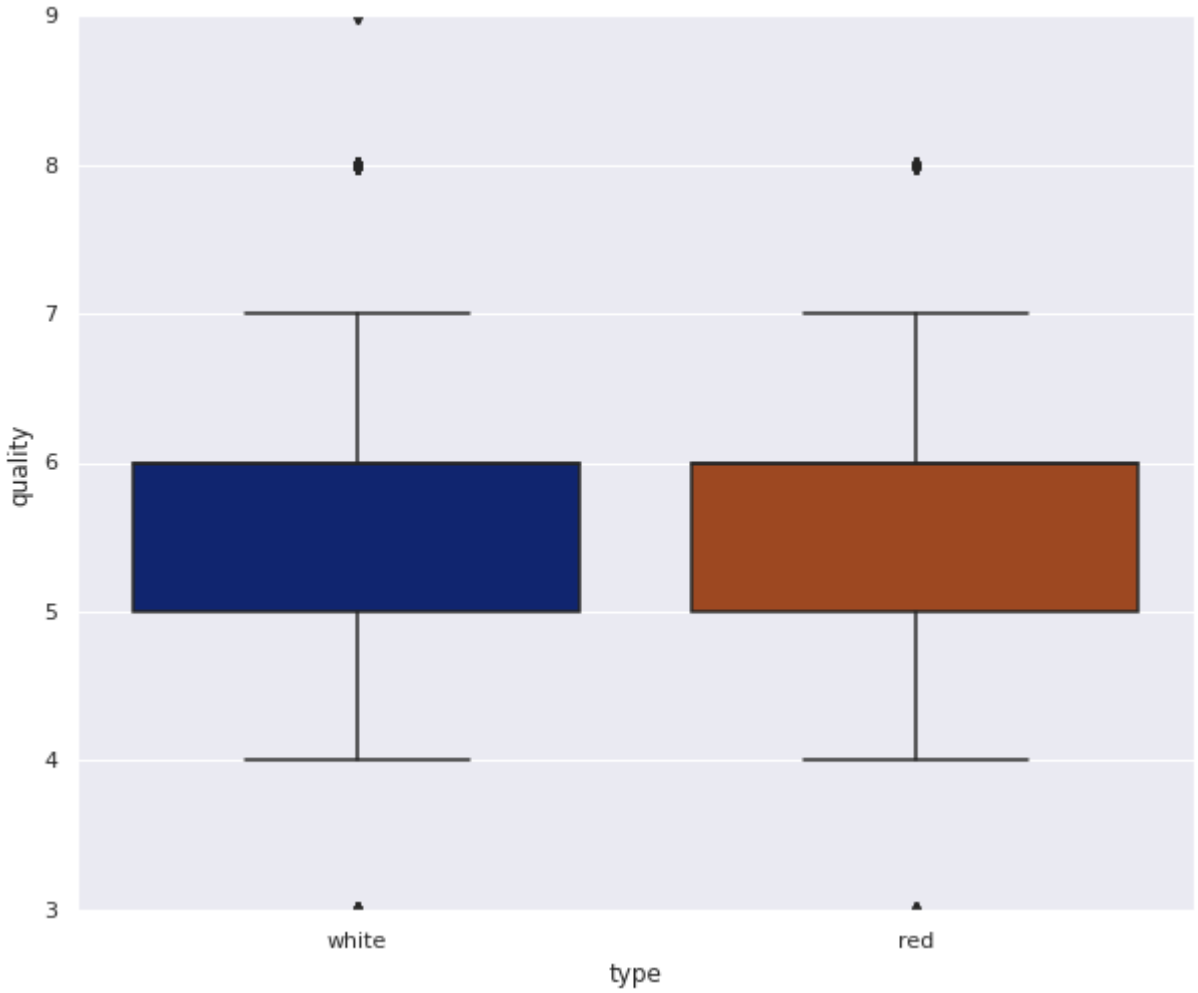
The **scatter plot** is a mainstay of statistical visualization.

In [26]:
```
plt.scatter("alcohol","quality",data=df)
plt.show()
```



Box Plot is the visual representation of the depicting groups of numerical data through their quartiles.

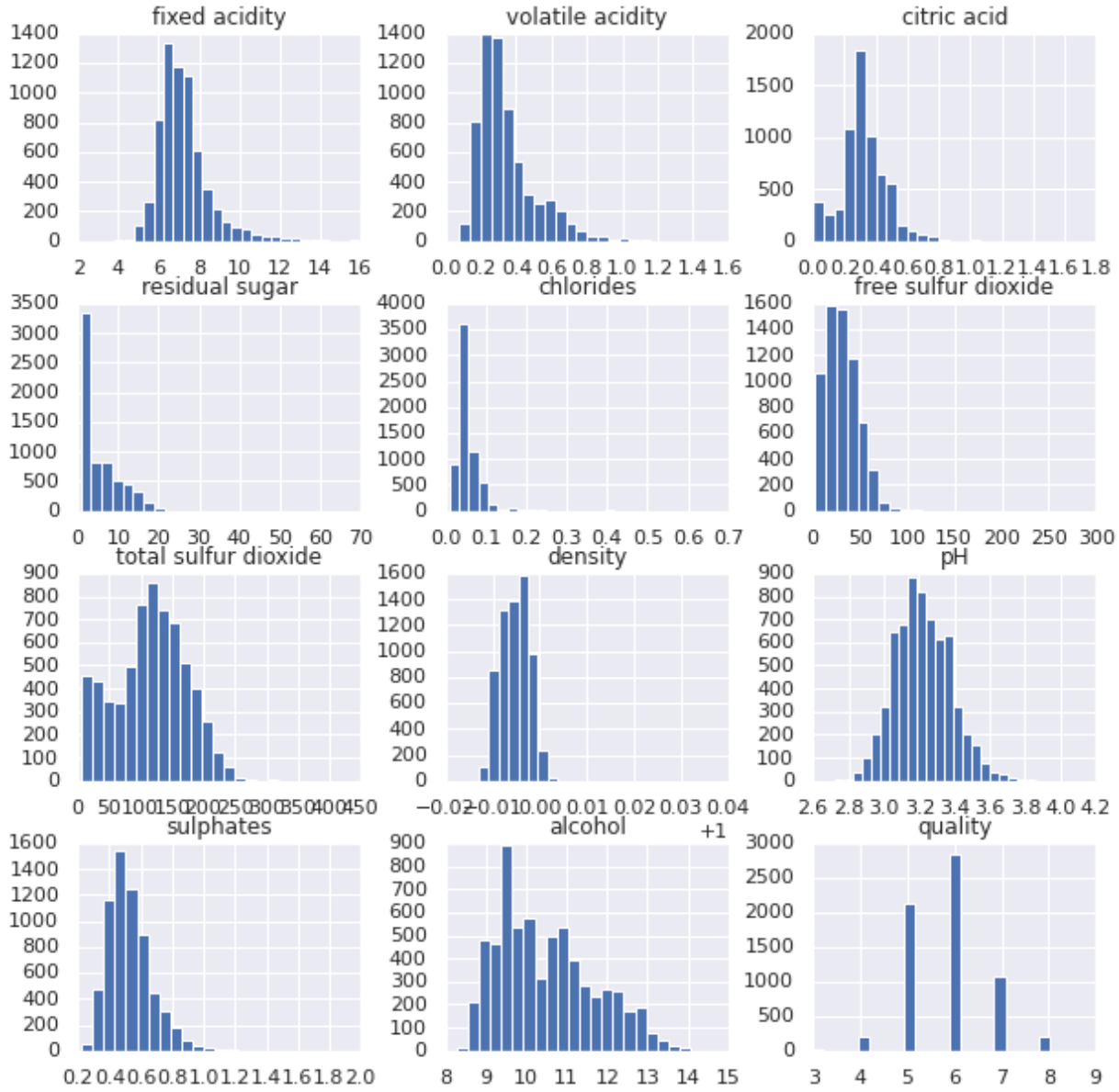
In [27]:
```
sns.boxplot(x="type",y="quality",data=df, palette="dark")
plt.show()
```



In [28]:
```
df=df[df.columns.drop('type')]
```

Histograms are used to display the distribution of one or several numerical variables.

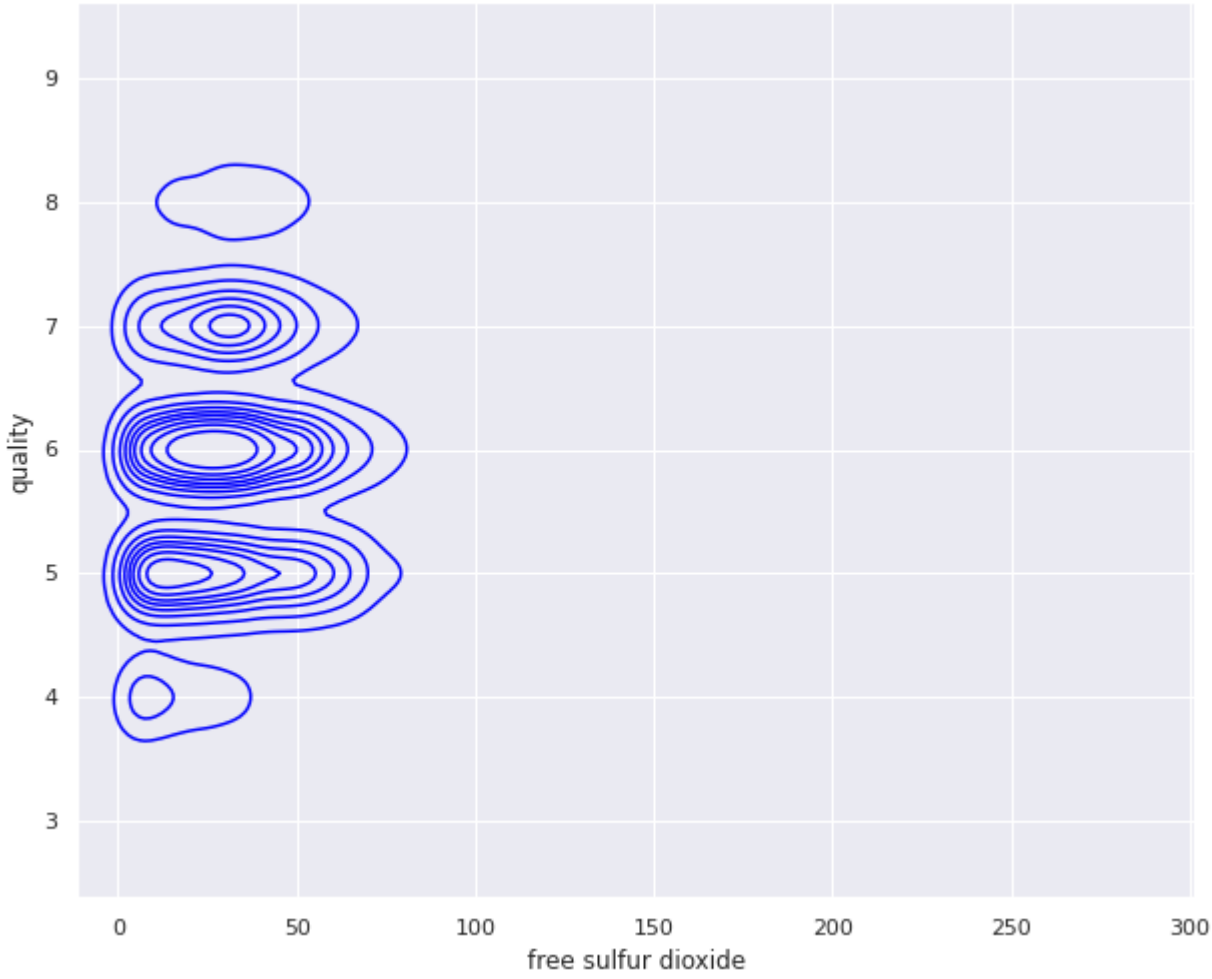
In [29]:
```
df.hist(bins=25,figsize=(10,10))
# display histogram
plt.show()
```



Kdeplot is a *Kernel Distribution Estimation Plot* which depicts the probability density function of the data variables

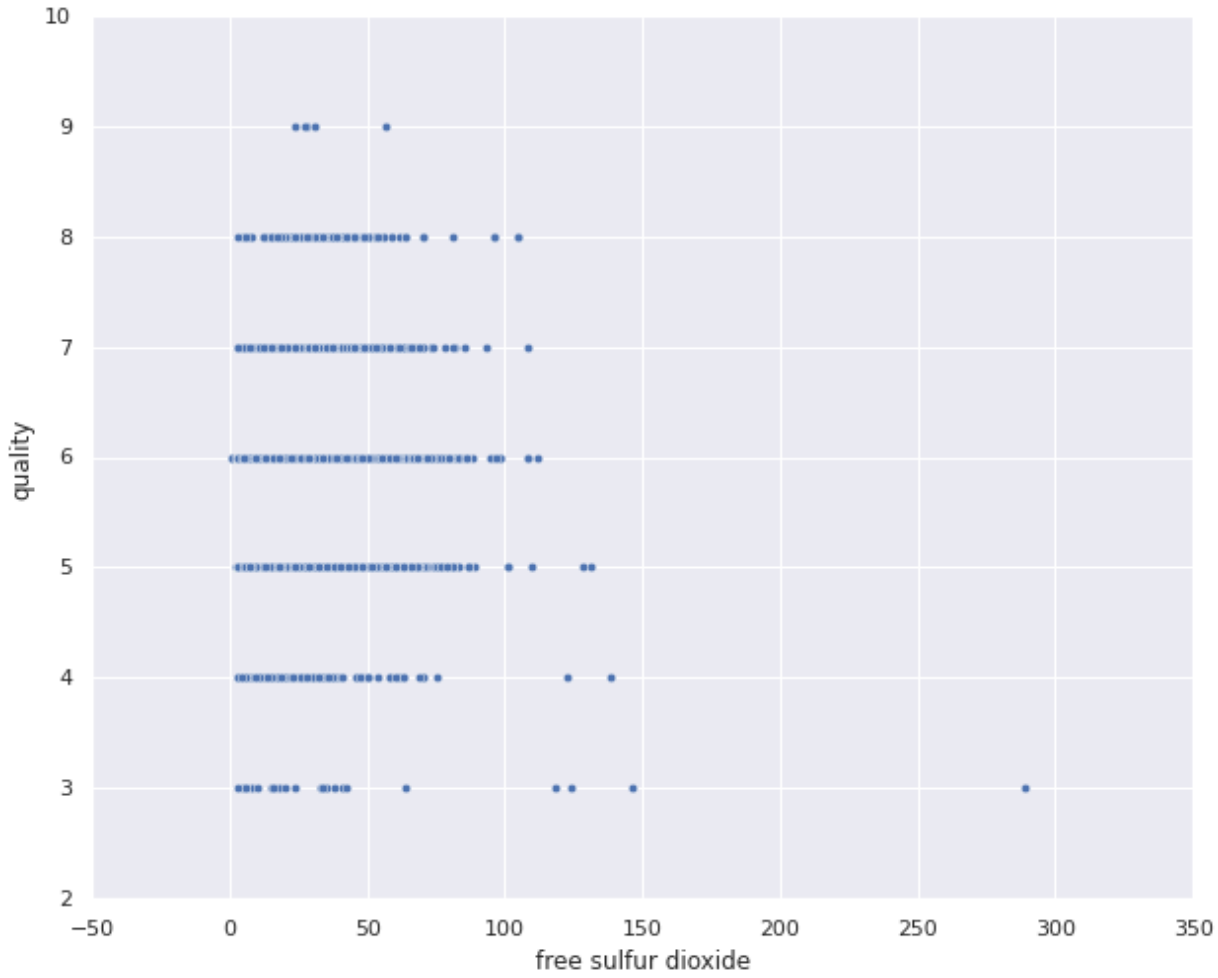
In [30]:
```
sns.kdeplot(x = 'free sulfur dioxide', y='quality' , data = df , color = 'blue')
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7f8ec5fb90>
```



In [31]:
```
sns.scatterplot(x='free sulfur dioxide', y ='quality' ,
data = df )
```

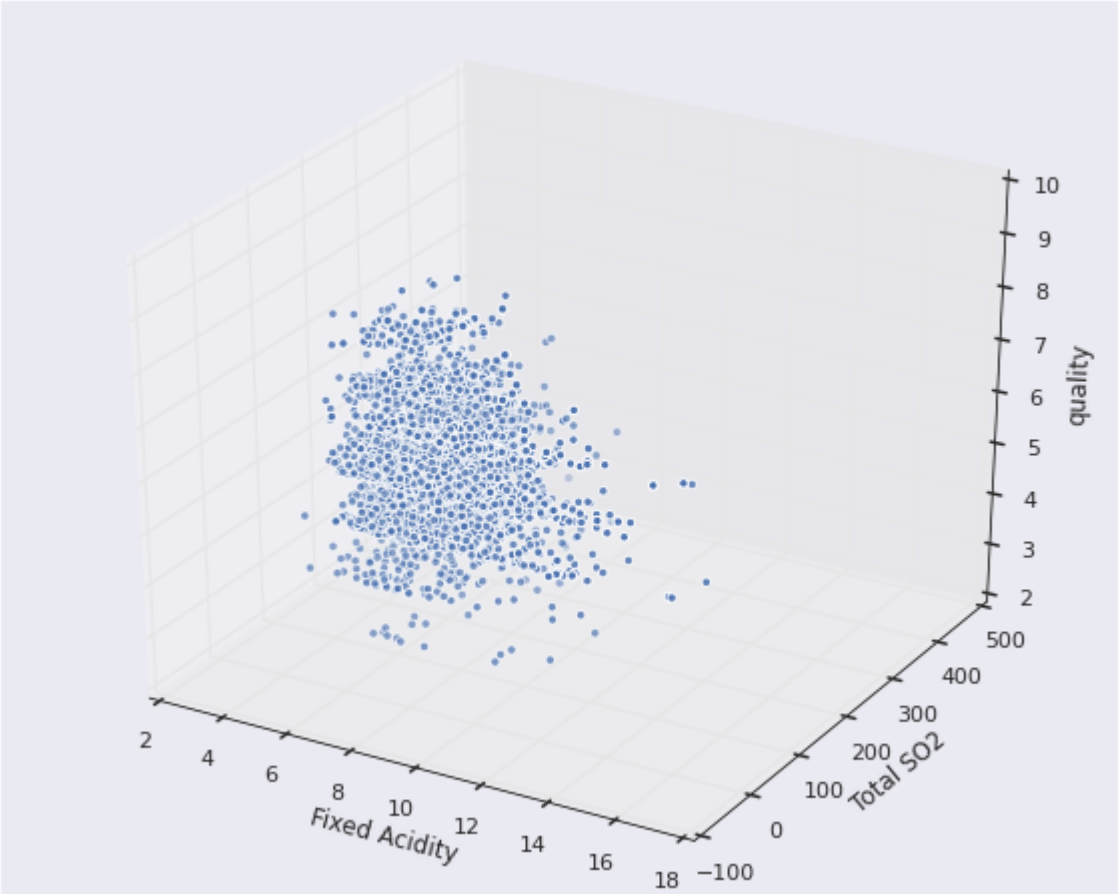
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7f8e9c75d0>



Interactive 3D plot to visualize 3 data dimensions :

In [32]:
```
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
```

In [33]:
```
sns.set(style = "darkgrid")

fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')

x = df['fixed acidity']
y = df['total sulfur dioxide']
z = df['quality']

ax.set_xlabel("Fixed Acidity")
ax.set_ylabel("Total SO2")
ax.set_zlabel("quality")

ax.scatter(x, y, z)

plt.show()
```



In [33]:
```

```