In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('milk.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
0	6.6	35	1	0	1	0	254	high
1	6.6	36	0	1	0	1	253	high
2	8.5	70	1	1	1	1	246	low
3	9.5	34	1	1	0	1	255	low
4	6.6	37	0	0	0	0	255	medium

In [4]:

```
df.tail()
```

Out[4]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
1054	6.7	45	1	1	0	0	247	medium
1055	6.7	38	1	0	1	0	255	high
1056	3.0	40	1	1	1	1	255	low
1057	6.8	43	1	0	1	0	250	high
1058	8.6	55	0	1	1	1	255	low

In [5]:

```
df.shape
```

Out[5]:

```
(1059, 8)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['pH', 'Temprature', 'Taste', 'Odor', 'Fat ', 'Turbidity', 'Colour',
       'Grade'],
      dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
976
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
pH            0
Temprature    0
Taste         0
Odor          0
Fat           0
Turbidity     0
Colour        0
Grade         0
dtype: int64
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1059 entries, 0 to 1058
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   pH          1059 non-null   float64
 1   Temprature  1059 non-null   int64  
 2   Taste       1059 non-null   int64  
 3   Odor        1059 non-null   int64  
 4   Fat         1059 non-null   int64  
 5   Turbidity   1059 non-null   int64  
 6   Colour      1059 non-null   int64  
 7   Grade       1059 non-null   object 
dtypes: float64(1), int64(6), object(1)
memory usage: 66.3+ KB
```

In [10]:

```
df.describe()
```

Out[10]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	
count	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	1059.0
mean	6.630123	44.226629	0.546742	0.432483	0.671388	0.491029	251.8
std	1.399679	10.098364	0.498046	0.495655	0.469930	0.500156	4.3
min	3.000000	34.000000	0.000000	0.000000	0.000000	0.000000	240.0
25%	6.500000	38.000000	0.000000	0.000000	0.000000	0.000000	250.0
50%	6.700000	41.000000	1.000000	0.000000	1.000000	0.000000	255.0
75%	6.800000	45.000000	1.000000	1.000000	1.000000	1.000000	255.0
max	9.500000	90.000000	1.000000	1.000000	1.000000	1.000000	255.0

In [11]:

```
df.nunique()
```

Out[11]:

```
pH            16
Temprature    17
Taste          2
Odor           2
Fat            2
Turbidity      2
Colour         9
Grade          3
dtype: int64
```

In [12]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [13]:

```
for i in df.columns:
    print(i)
    print(df[i].unique())
    print('\n')
```

```
pH
[6.6 8.5 9.5 5.5 4.5 8.1 6.7 5.6 8.6 7.4 6.8 6.5 4.7 3.  9.  6.4]


Temprature
[35 36 70 34 37 45 60 66 50 55 90 38 40 43 42 41 65]


Taste
[1 0]


Odor
[0 1]


Fat
[1 0]


Turbidity
[0 1]


Colour
[254 253 246 255 250 247 245 240 248]


Grade
['high' 'low' 'medium']
```

In [14]:

```
for i in df.columns:
    print(i)
    print(df[i].value_counts())
    print('\n')
```

```
pH
6.8    249
6.5    189
6.6    159
6.7     82
3.0     70
9.0     61
8.6     40
7.4     39
4.5     37
9.5     24
8.1     24
5.5     23
8.5     22
4.7     20
5.6     19
6.4      1
Name: pH, dtype: int64


Temprature
45    219
38    179
40    132
37     83
43     77
36     66
50     58
55     48
34     40
41     30
66     24
35     23
70     22
65     22
60     18
90     17
42      1
Name: Temprature, dtype: int64


Taste
1    579
0    480
Name: Taste, dtype: int64


Odor
0    601
1    458
Name: Odor, dtype: int64


Fat
1    711
0    348
Name: Fat , dtype: int64


Turbidity
0    539
```

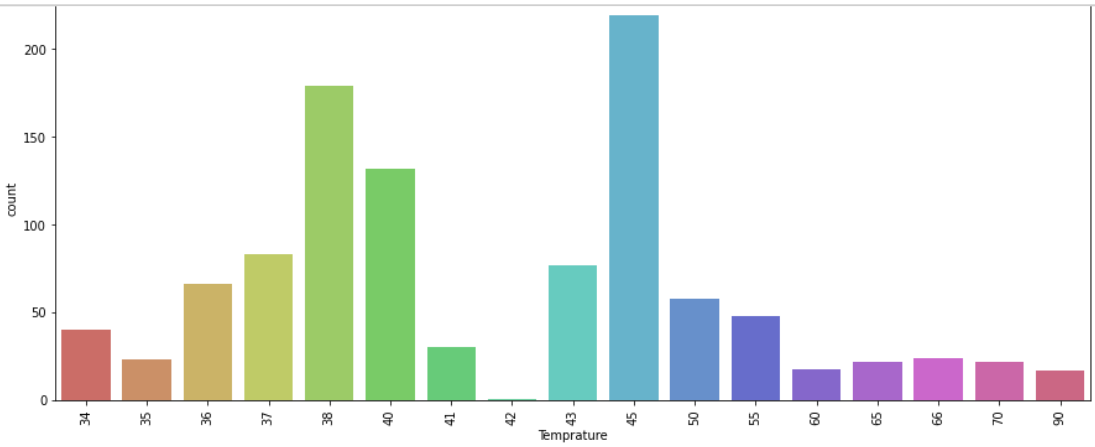
```
1    520
Name: Turbidity, dtype: int64


Colour
255    628
250    146
245    115
247     48
246     44
240     32
248     23
253     22
254      1
Name: Colour, dtype: int64


Grade
low       429
medium    374
high      256
Name: Grade, dtype: int64
```

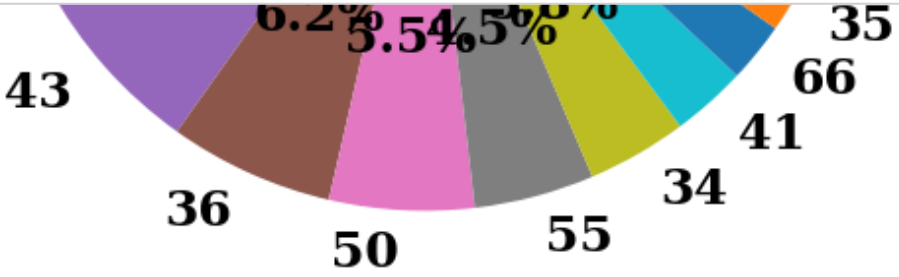
In [15]:

```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.countplot(df[i], data = df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



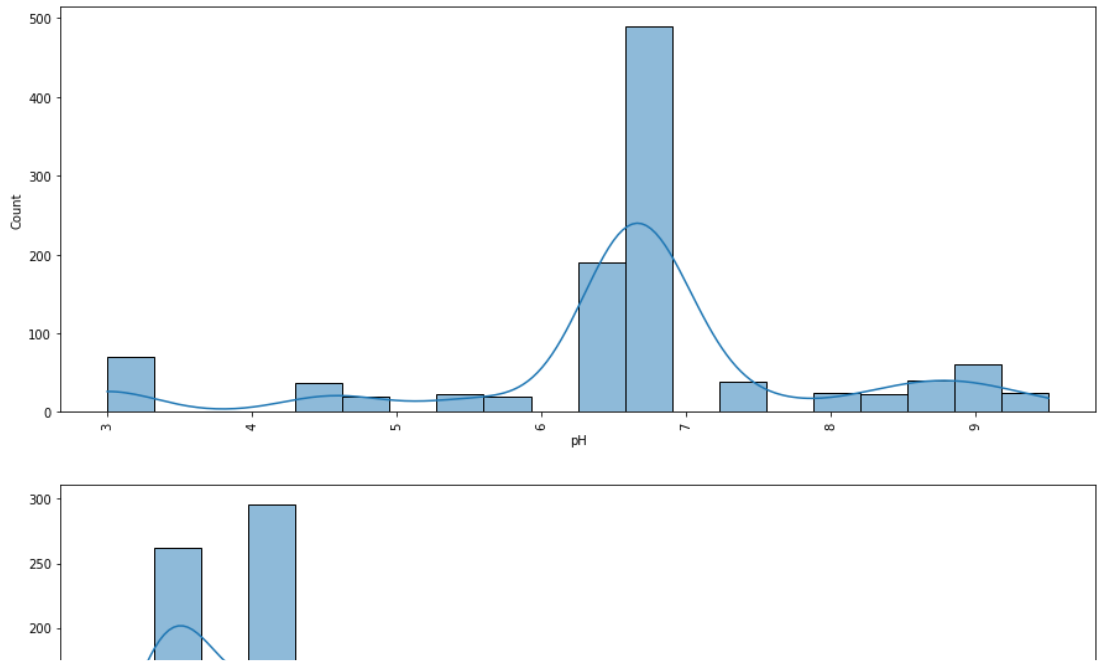
In [16]:

```
for i in df.columns:
    plt.figure(figsize=(30,10))
    plt.pie(df[i].value_counts(), labels=df[i].value_counts().index,
            autopct='%1.1f%%', textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
    hfont = {'fontname':'serif', 'weight': 'bold'}
    plt.title(i, size=20, **hfont)
    plt.show()
```



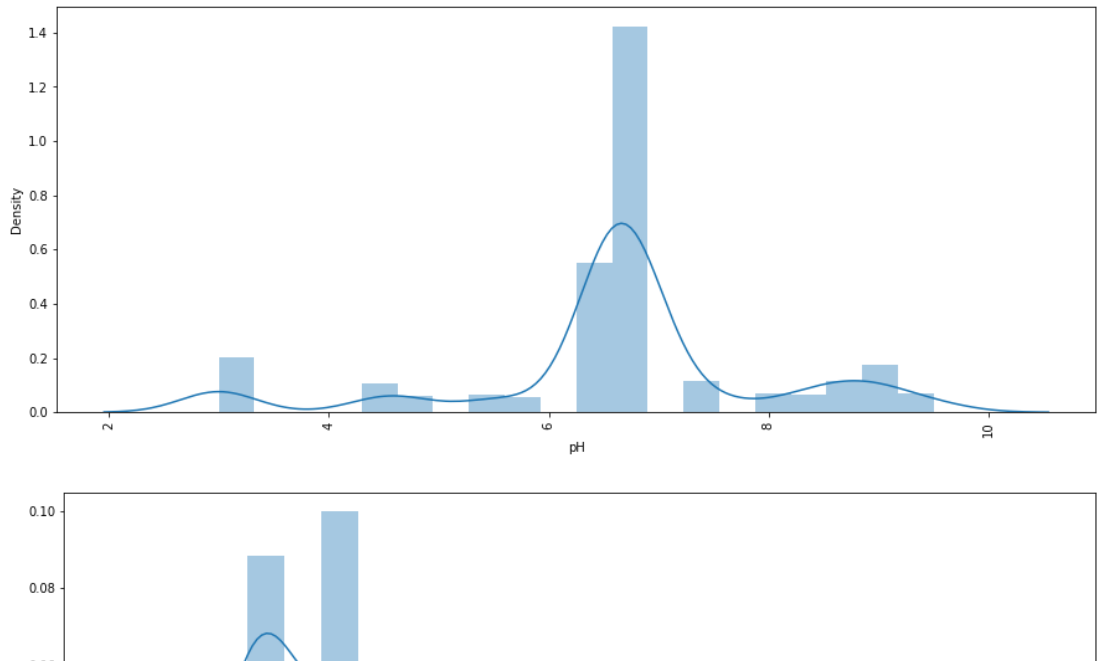
In [17]:

```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.histplot(df[i], kde = True, bins = 20, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



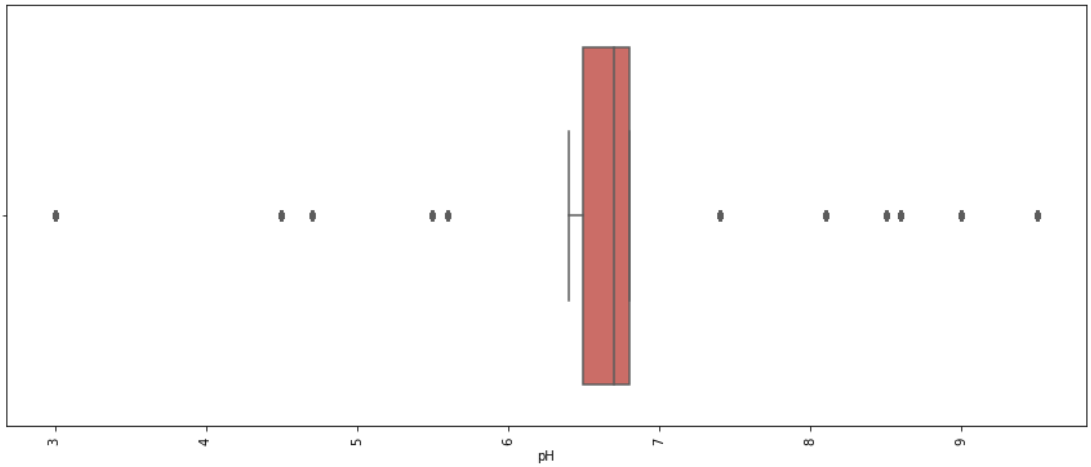
In [18]:

```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.distplot(df[i], kde = True, bins = 20)
    plt.xticks(rotation = 90)
    plt.show()
```



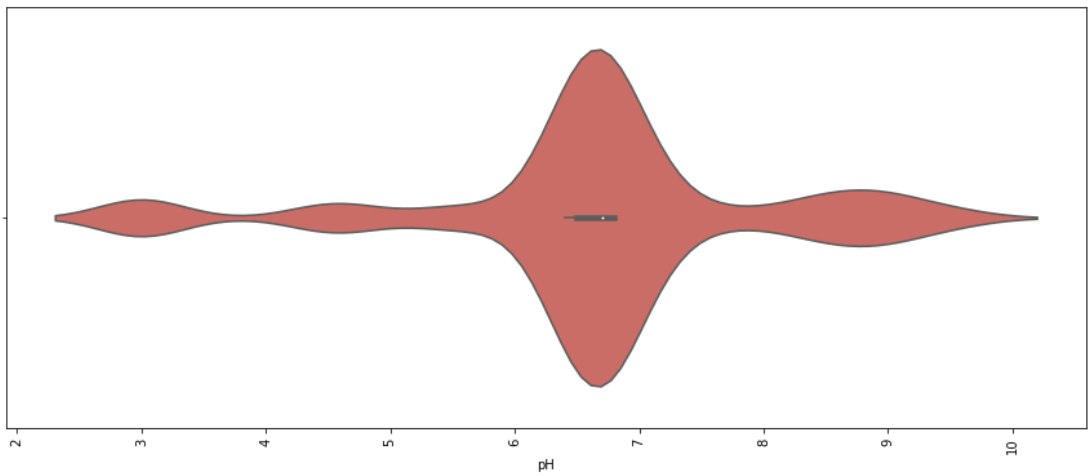
In [19]:

```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.boxplot(df[i], data = df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



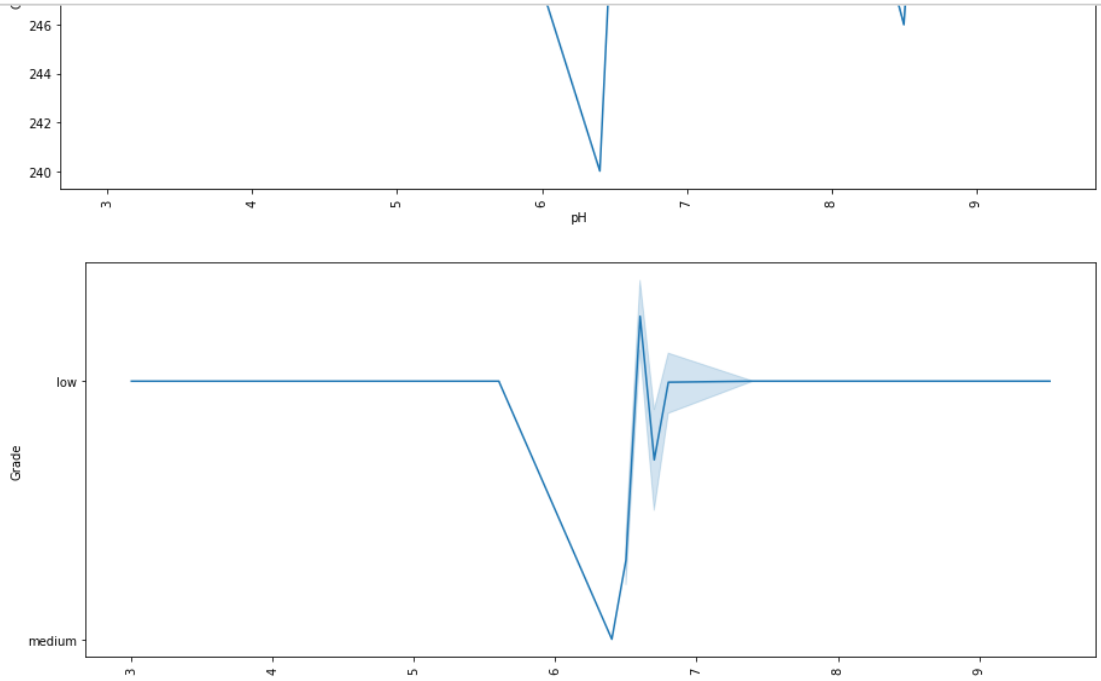
In [20]:

```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.violinplot(df[i], data = df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



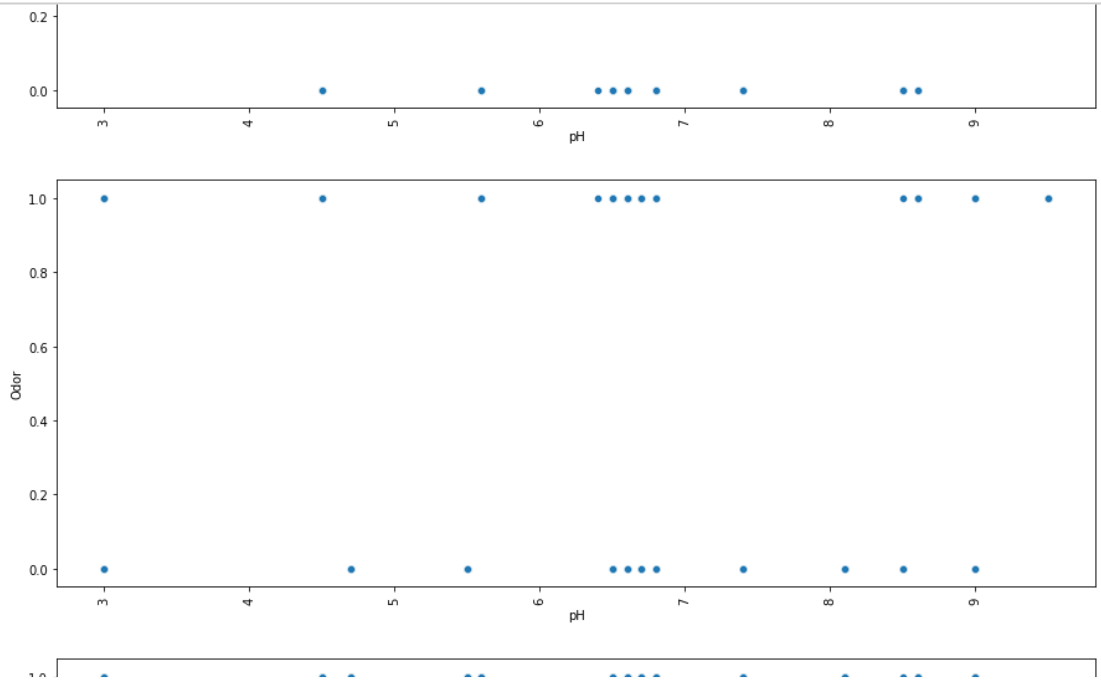
In [21]:

```
for i in df.columns:
    for j in df.columns:
        plt.figure(figsize=(15,6))
        sns.lineplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



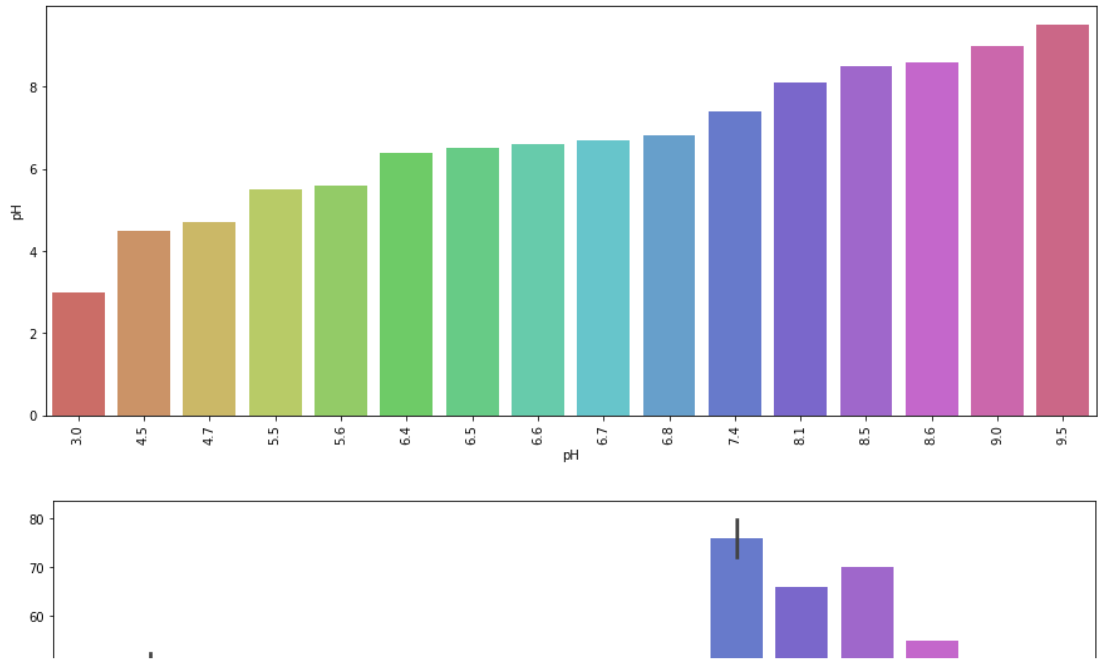
In [22]:

```
for i in df.columns:
    for j in df.columns:
        plt.figure(figsize=(15,6))
        sns.scatterplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



In [23]:

```
for i in df.columns:
    for j in df.columns:
        plt.figure(figsize=(15,6))
        sns.barplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



In [24]:

```
df_corr = df.corr()
```

In [25]:

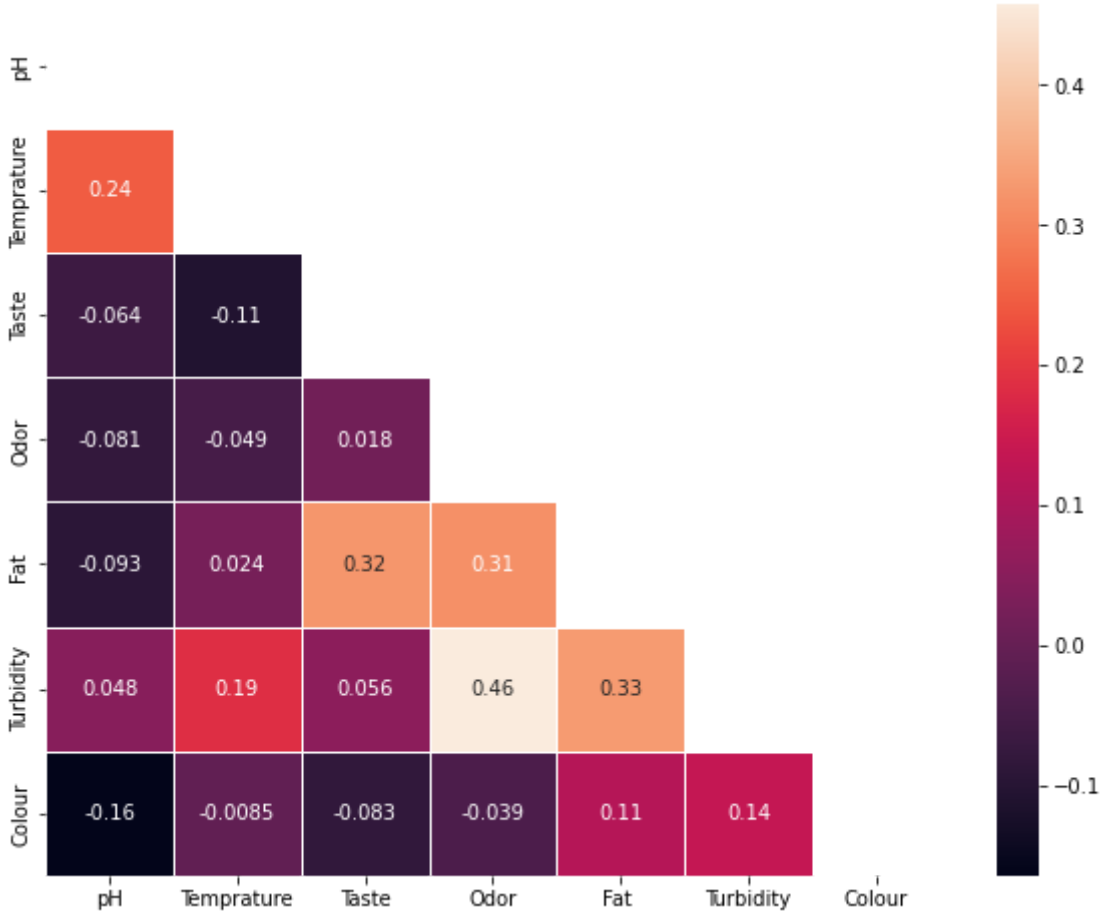
```
df_corr
```

Out[25]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour
pH	1.000000	0.244684	-0.064053	-0.081331	-0.093429	0.048384	-0.164565
Temprature	0.244684	1.000000	-0.109792	-0.048870	0.024073	0.185106	-0.008511
Taste	-0.064053	-0.109792	1.000000	0.017582	0.324149	0.055755	-0.082654
Odor	-0.081331	-0.048870	0.017582	1.000000	0.314505	0.457935	-0.039361
Fat	-0.093429	0.024073	0.324149	0.314505	1.000000	0.329264	0.114151
Turbidity	0.048384	0.185106	0.055755	0.457935	0.329264	1.000000	0.136436
Colour	-0.164565	-0.008511	-0.082654	-0.039361	0.114151	0.136436	1.000000

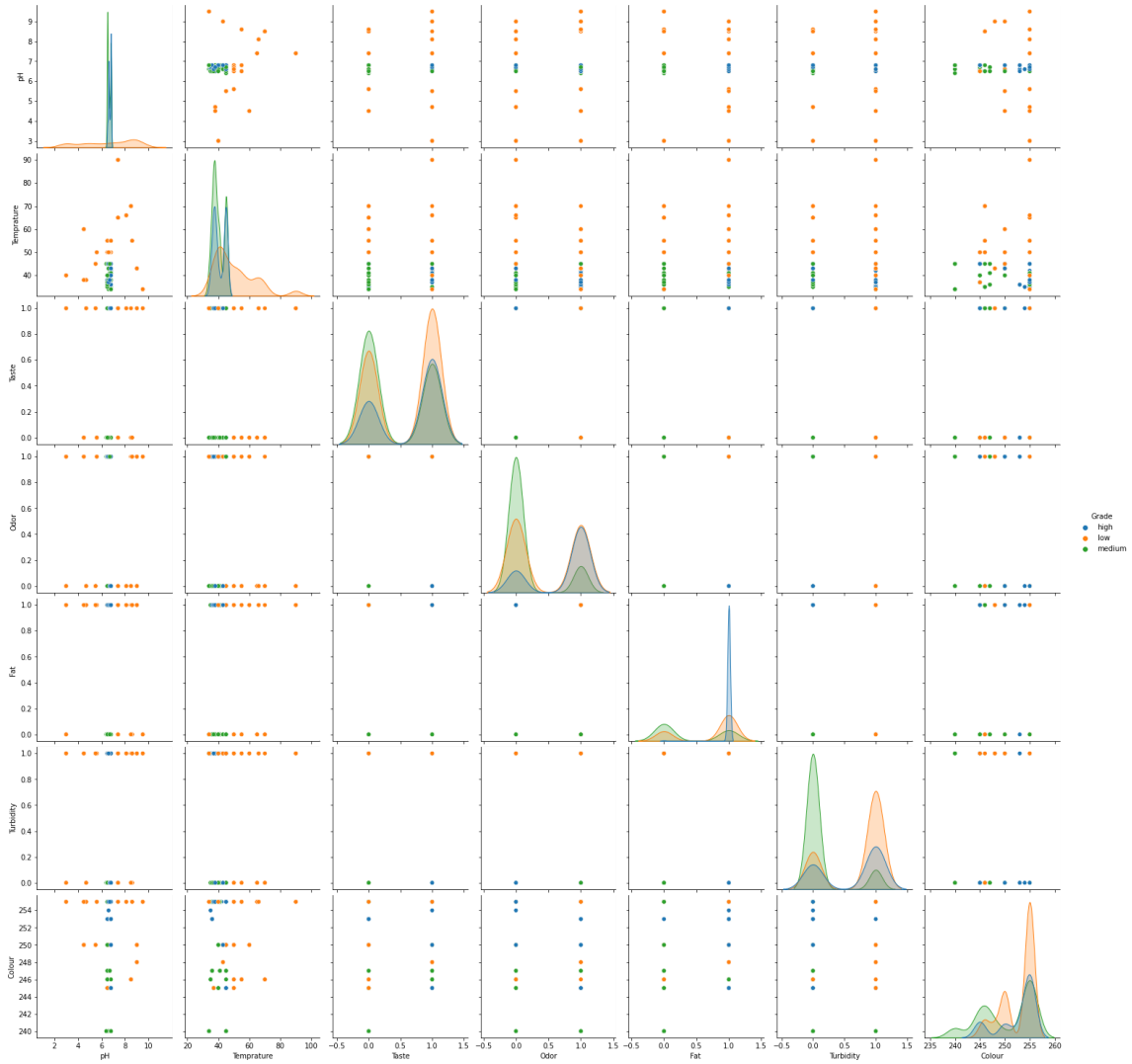
In [26]:

```
plt.figure(figsize=(10, 8))
matrix = np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap="rocket");
plt.show()
```



In [27]:

```
sns.pairplot(df,hue="Grade",height=3)
plt.show()
```



In [30]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Grade'] = le.fit_transform(df['Grade'])
```

In [31]:

```
X= df.drop("Temprature", axis = 1)
y= df["Temprature"]
```

In [32]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [33]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size = 0.80, random_state = 41)
```

In [34]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
```

Out[34]:

```
▾ LinearRegression

LinearRegression()
```

In [37]:

```
y_pred = lr.predict(X_test)
```

In [38]:

```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [39]:

```
print(mse)
print(r2)
```

```
95.6316242946285
0.14763496777448915
```

In [41]:

```
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor(max_depth=3, random_state=42)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
Mean Squared Error: 16.55803211456336
0.8524181965848553
```

In [42]:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, max_depth=3, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
Mean Squared Error: 13.642695673957737
0.8784026014035925
```