In [86]:

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
```

In [87]:

```
data = pd.read_csv(r'C:\Users\91970\Desktop\DataScience\milknew.csv')
```

In [88]:

```
data.shape
```

Out[88]:

```
(1059, 8)
```

This dataset contains 1059 rows and 8 columns

In [89]:

```
data.size
```

Out[89]:

```
8472
```

In [90]:

```
data.head(10)
```

Out[90]:

	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
5	6.6	37	1	1	1	1	255	high
6	5.5	45	1	0	1	1	250	low
7	4.5	60	0	1	1	1	250	low
8	8.1	66	1	0	1	1	255	low
9	6.7	45	1	1	0	0	247	medium

In [91]:

```
data.tail(10)
```

Out[91]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
1049	6.5	37	0	0	0	0	255	medium
1050	6.6	37	1	1	1	1	255	high
1051	5.5	45	1	0	1	1	250	low
1052	6.5	40	1	0	0	0	250	medium
1053	8.1	66	1	0	1	1	255	low
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 [92]:

```
data.info()
```

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

In [93]:

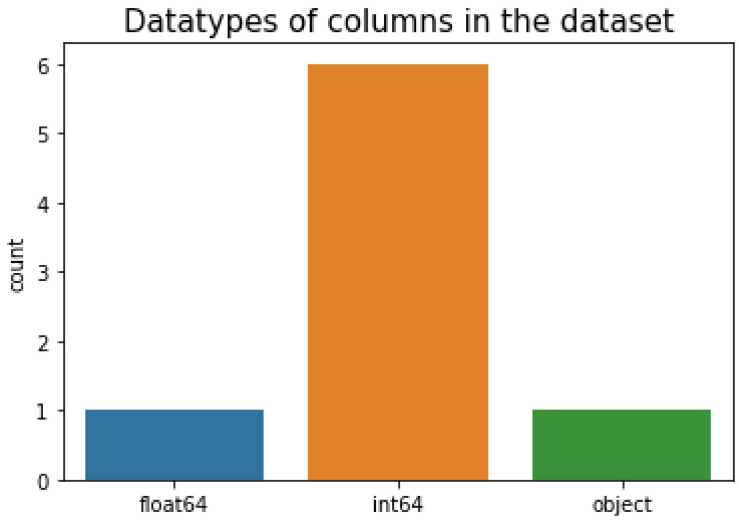
```
data.describe()
```

Out[93]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Co
count	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	1059.00(
mean	6.630123	44.226629	0.546742	0.432483	0.671388	0.491029	251.84(
std	1.399679	10.098364	0.498046	0.495655	0.469930	0.500156	4.307
min	3.000000	34.000000	0.000000	0.000000	0.000000	0.000000	240.00(
25%	6.500000	38.000000	0.000000	0.000000	0.000000	0.000000	250.00(
50%	6.700000	41.000000	1.000000	0.000000	1.000000	0.000000	255.00(
75%	6.800000	45.000000	1.000000	1.000000	1.000000	1.000000	255.00(
max	9.500000	90.000000	1.000000	1.000000	1.000000	1.000000	255.00(

In [94]:

```
sb.countplot(data.dtypes)
plt.title("Datatypes of columns in the dataset",fontsize = 15)
plt.show()
print("Count of the datatypes of columns")
print(data.dtypes.value_counts())
```




```
Count of the datatypes of columns
int64      6
object     1
float64    1
dtype: int64
```

In [95]:

```
data.isnull().sum()
```

Out[95]:

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

No null values present in the dataset

In [96]:

```
data.duplicated().sum()
```

Out[96]:

```
976
```

In [97]:

```
data.loc[data.duplicated(),:]
```

Out[97]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
35	6.8	45	0	1	1	1	255	high
48	9.5	34	1	1	0	1	255	low
50	6.6	37	1	1	1	1	255	high
51	5.5	45	1	0	1	1	250	low
52	4.5	60	0	1	1	1	250	low
53	8.1	66	1	0	1	1	255	low
54	6.7	45	1	1	0	0	247	medium
55	6.7	45	1	1	1	0	245	medium
56	5.6	50	0	1	1	1	255	low
57	8.6	55	0	1	1	1	255	low
58	7.4	90	1	0	1	1	255	low
62	3.0	40	1	1	1	1	255	low
63	9.0	43	1	0	1	1	250	low
64	8.6	55	0	1	1	1	255	low
66	6.8	45	0	1	1	1	255	high
67	6.5	38	1	0	0	0	255	medium
68	4.7	38	1	0	1	0	255	low
69	3.0	40	1	1	1	1	255	low
70	9.0	43	1	0	1	1	250	low
71	6.8	40	1	0	1	0	245	medium
72	6.6	45	0	1	1	1	250	high
73	6.5	36	0	0	1	0	255	medium
74	4.5	38	0	1	1	1	255	low
78	8.5	70	0	0	0	0	246	low
80	6.5	37	0	0	0	0	245	medium
81	6.6	37	1	0	1	0	255	high
83	7.4	65	0	0	0	0	255	low
85	6.5	38	1	1	1	1	255	high
86	6.6	38	0	0	0	0	255	medium
87	3.0	40	1	0	0	0	255	low
...	...	...	...	...	...	...	...	...
1029	6.8	45	0	1	0	0	240	medium
1030	7.4	65	0	0	0	0	255	low
1031	6.8	41	0	0	1	0	255	medium

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
1032	6.5	38	1	1	1	1	255	high
1033	6.6	38	0	0	0	0	255	medium
1034	3.0	40	1	0	0	0	255	low
1035	9.0	43	1	1	1	1	248	low
1036	6.8	40	1	1	1	1	255	high
1037	6.6	50	0	0	0	1	250	low
1038	6.5	36	0	0	0	0	247	medium
1039	6.6	50	0	0	0	0	255	low
1040	6.8	45	1	1	1	0	245	high
1041	6.6	40	1	0	0	0	255	medium
1042	9.0	43	1	1	1	1	248	low
1043	6.8	40	1	1	1	1	255	high
1044	6.6	50	0	0	0	1	250	low
1045	6.5	36	0	0	0	0	247	medium
1046	6.6	38	0	0	0	0	255	medium
1047	6.8	45	1	1	1	0	245	high
1048	9.5	34	1	1	0	1	255	low
1049	6.5	37	0	0	0	0	255	medium
1050	6.6	37	1	1	1	1	255	high
1051	5.5	45	1	0	1	1	250	low
1052	6.5	40	1	0	0	0	250	medium
1053	8.1	66	1	0	1	1	255	low
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

976 rows × 8 columns

Numerical Columns in the dataset

In [98]:

```
num = []
for i in data.columns:
    if(data[i].dtypes != 'object'):
        num.append(i)
print(num)
```

```
['pH', 'Temprature', 'Taste', 'Odor', 'Fat ', 'Turbidity', 'Colour']
```

pH

In [99]:

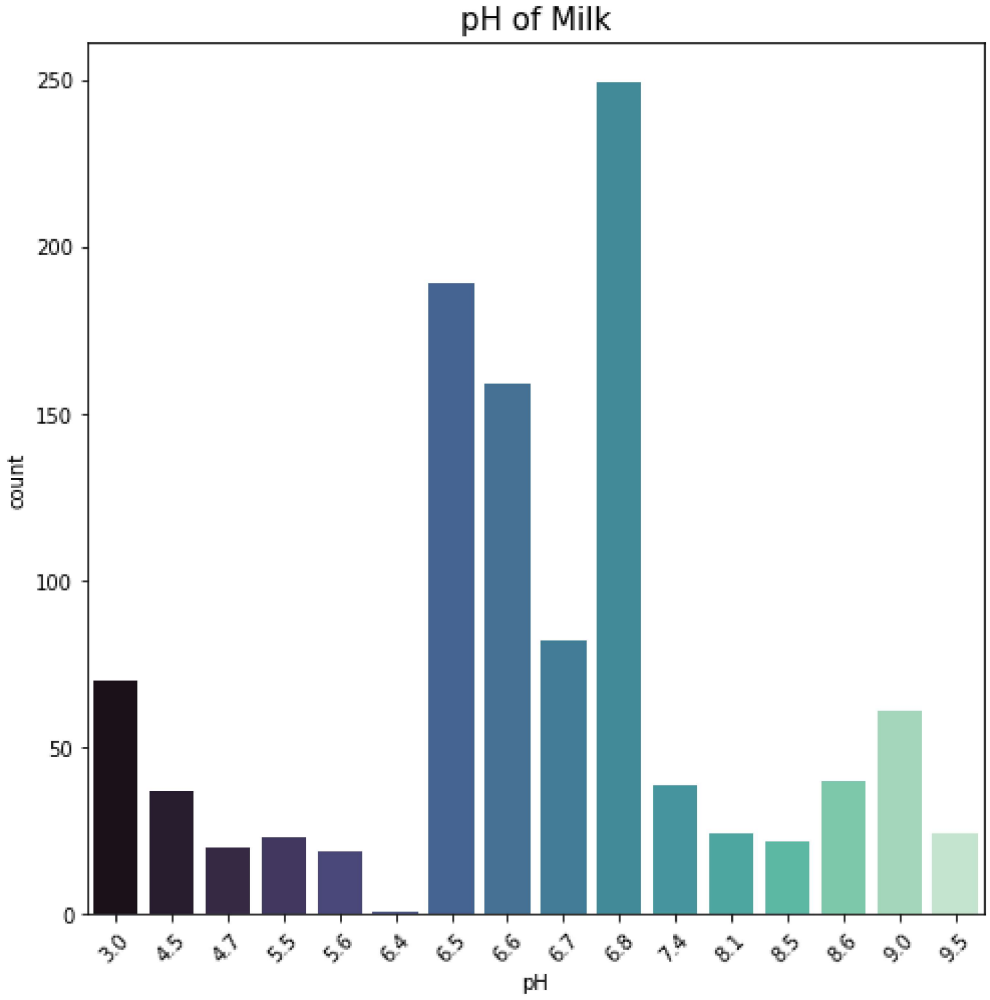
```
data['pH'].value_counts()
```

Out[99]:

```
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
8.1     24
9.5     24
5.5     23
8.5     22
4.7     20
5.6     19
6.4      1
Name: pH, dtype: int64
```

In [100]:

```
plt.figure(figsize = (8,8))
plt.title("pH of Milk", fontsize=15)
c1 = sb.countplot(x='pH', data = data, palette="mako")
plt.xticks(rotation=45)
plt.show()
```



Majority of the milk have pH less than 7 indicating that milk is slightly acidic in nature

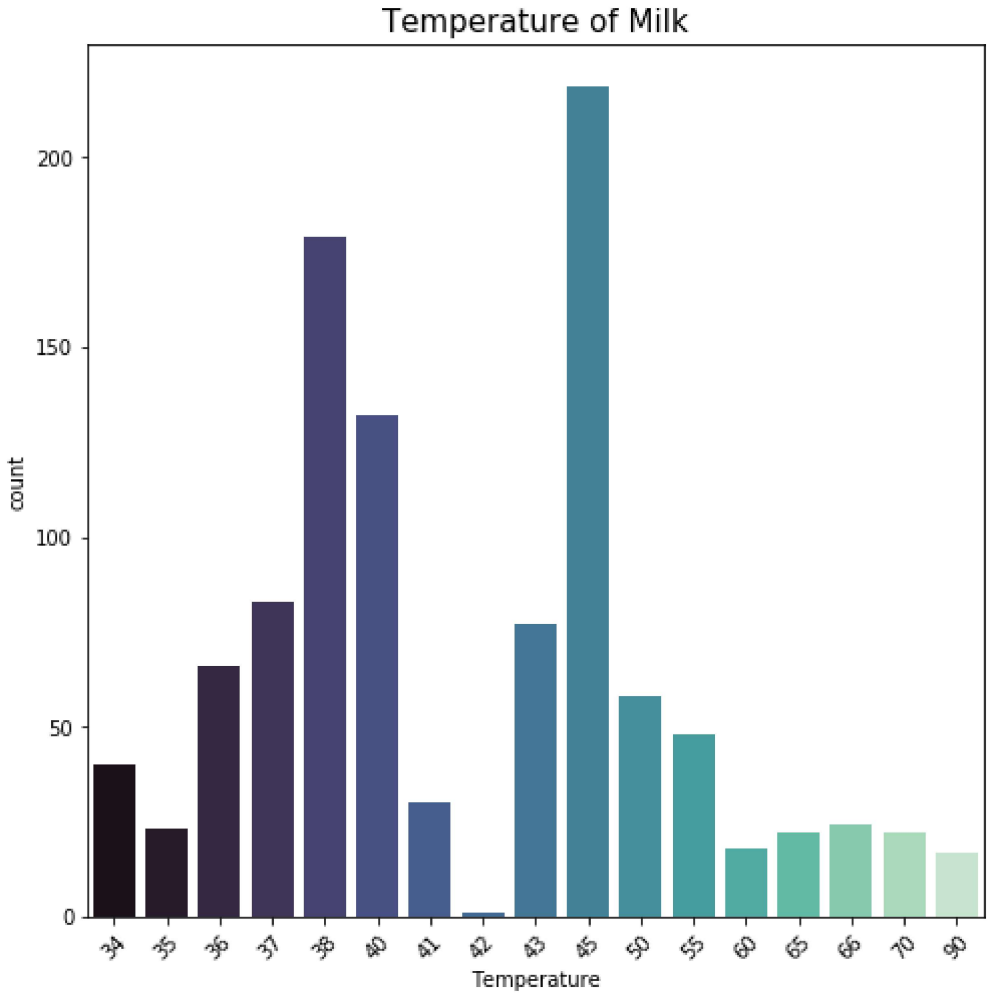
Temperature

In [101]:

```
data = data.rename(columns={'Temprature' : 'Temperature'})
```

In [102]:

```
plt.figure(figsize=(8,8))
plt.title("Temperature of Milk",fontsize = 15)
c1 = sb.countplot(x='Temperature', data=data, palette="mako")
plt.xticks(rotation=45)
plt.show()
```

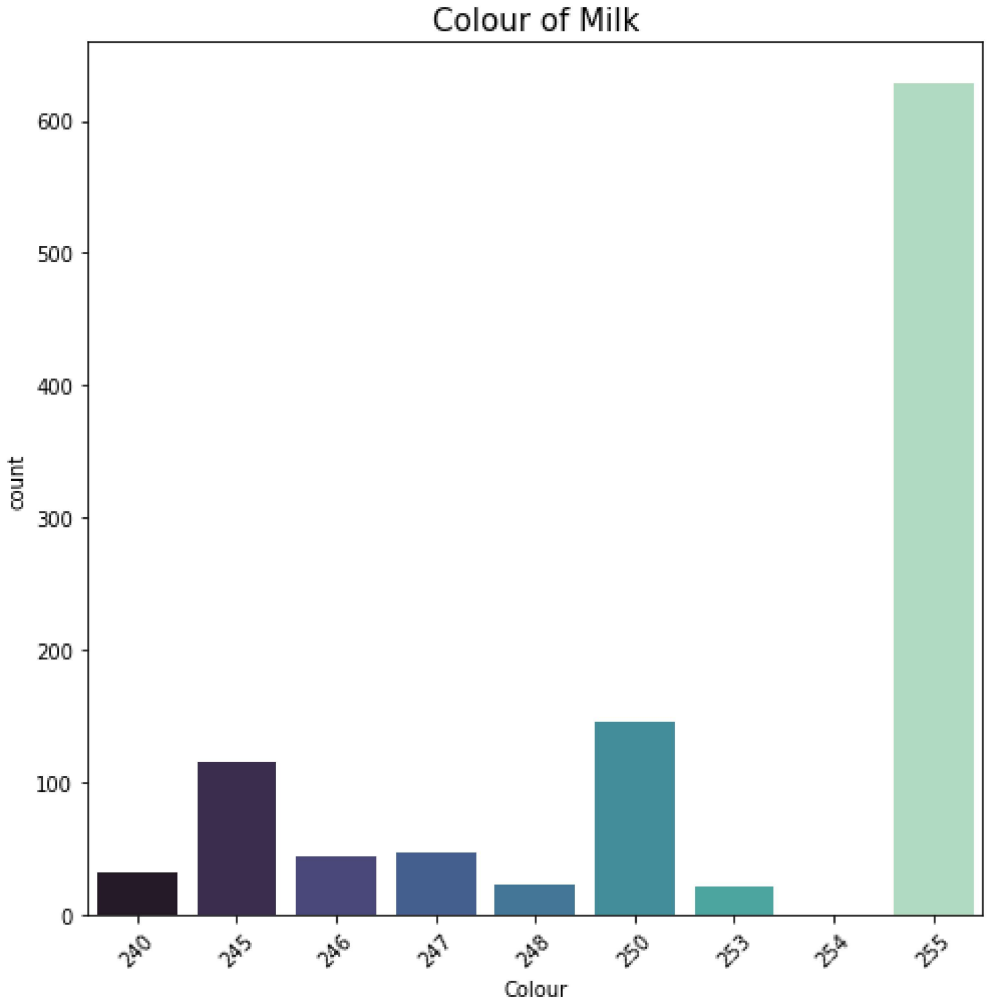


The milk is mostly produced under 50 degrees

Colour

In [103]:

```
plt.figure(figsize=(8,8))
plt.title("Colour of Milk",fontsize = 15)
c1 = sb.countplot(x='Colour', data=data, palette="mako")
plt.xticks(rotation=45)
plt.show()
```



Colour of the milk ranges from 240 to 255 and the value of colour moslty is 255

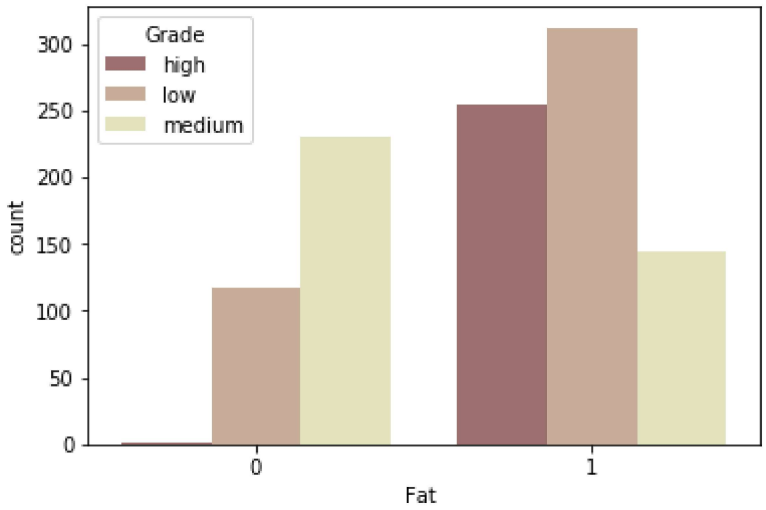
Fat

In [155]:

```
sb.countplot(x='Fat ', hue='Grade', data=data, palette="pink")
```

Out[155]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f4e4df128>
```



Fat is mandatory for high quality milk

In [105]:

```
data.iloc[:,:-1]
```

Out[105]:

	pH	Temperature	Taste	Odor	Fat	Turbidity	Colour
0	6.6	35	1	0	1	0	254
1	6.6	36	0	1	0	1	253
2	8.5	70	1	1	1	1	246
3	9.5	34	1	1	0	1	255
4	6.6	37	0	0	0	0	255
5	6.6	37	1	1	1	1	255
6	5.5	45	1	0	1	1	250
7	4.5	60	0	1	1	1	250
8	8.1	66	1	0	1	1	255
9	6.7	45	1	1	0	0	247
10	6.7	45	1	1	1	0	245
11	5.6	50	0	1	1	1	255
12	8.6	55	0	1	1	1	255
13	7.4	90	1	0	1	1	255
14	6.8	45	0	1	1	1	255
15	6.5	38	1	0	0	0	255
16	4.7	38	1	0	1	0	255
17	3.0	40	1	1	1	1	255
18	9.0	43	1	0	1	1	250
19	6.8	40	1	0	1	0	245
20	6.6	45	0	1	1	1	250
21	6.5	36	0	0	1	0	255
22	4.5	38	0	1	1	1	255
23	6.6	45	1	1	1	1	245
24	6.8	35	1	0	1	0	246
25	6.5	36	0	1	1	0	253
26	8.5	70	0	0	0	0	246
27	6.6	34	0	0	0	1	240
28	6.5	37	0	0	0	0	245
29	6.6	37	1	0	1	0	255
...	...	...	...	...	...	...	...
1029	6.8	45	0	1	0	0	240
1030	7.4	65	0	0	0	0	255
1031	6.8	41	0	0	1	0	255

	pH	Temperature	Taste	Odor	Fat	Turbidity	Colour
1032	6.5	38	1	1	1	1	255
1033	6.6	38	0	0	0	0	255
1034	3.0	40	1	0	0	0	255
1035	9.0	43	1	1	1	1	248
1036	6.8	40	1	1	1	1	255
1037	6.6	50	0	0	0	1	250
1038	6.5	36	0	0	0	0	247
1039	6.6	50	0	0	0	0	255
1040	6.8	45	1	1	1	0	245
1041	6.6	40	1	0	0	0	255
1042	9.0	43	1	1	1	1	248
1043	6.8	40	1	1	1	1	255
1044	6.6	50	0	0	0	1	250
1045	6.5	36	0	0	0	0	247
1046	6.6	38	0	0	0	0	255
1047	6.8	45	1	1	1	0	245
1048	9.5	34	1	1	0	1	255
1049	6.5	37	0	0	0	0	255
1050	6.6	37	1	1	1	1	255
1051	5.5	45	1	0	1	1	250
1052	6.5	40	1	0	0	0	250
1053	8.1	66	1	0	1	1	255
1054	6.7	45	1	1	0	0	247
1055	6.7	38	1	0	1	0	255
1056	3.0	40	1	1	1	1	255
1057	6.8	43	1	0	1	0	250
1058	8.6	55	0	1	1	1	255

1059 rows × 7 columns

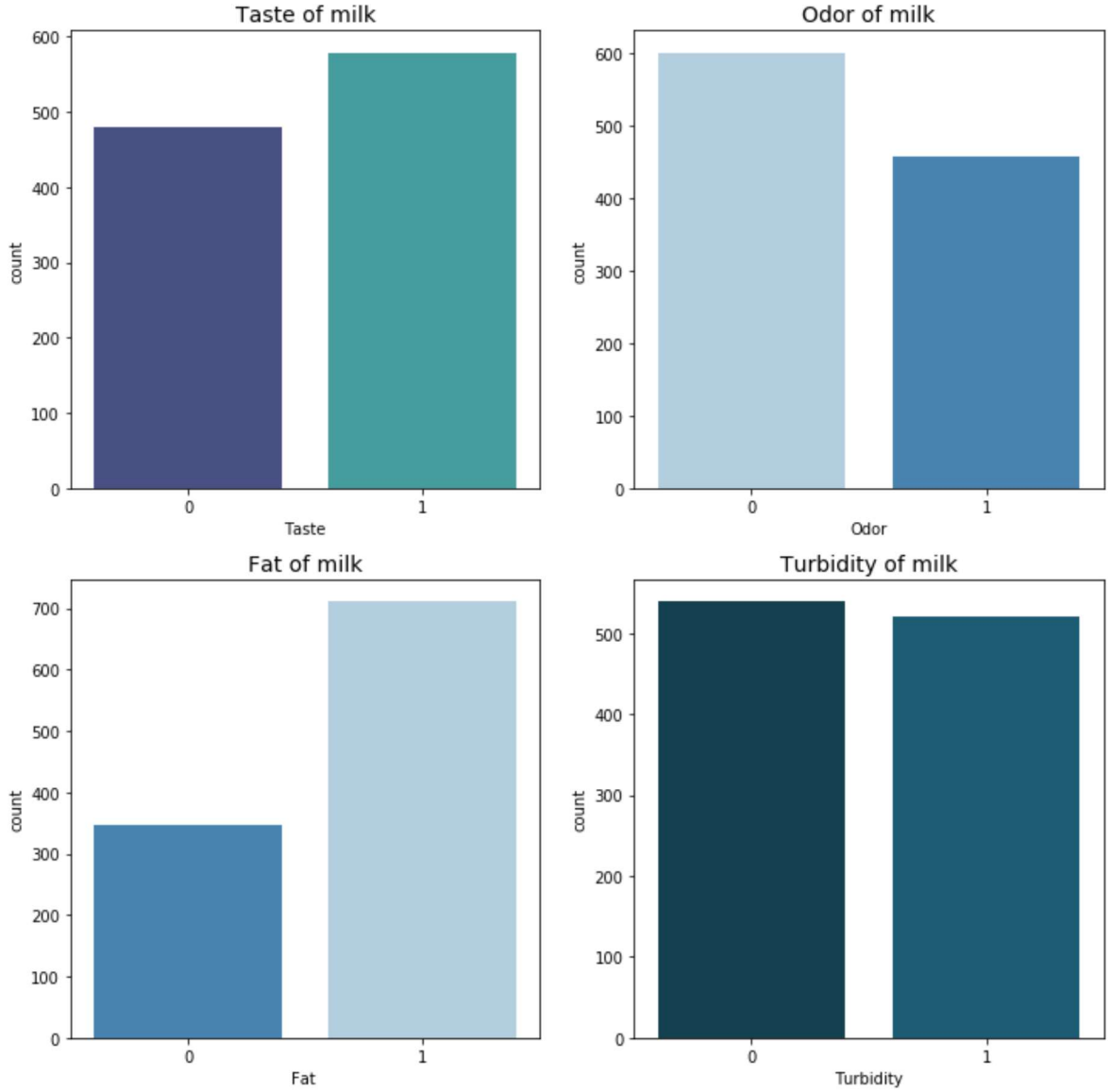
In [106]:

```
fig,ax=plt.subplots(2,2,figsize=(12,12))
ax[0,0].set_title("Taste of milk",fontsize=14)
c1=sb.countplot(data['Taste'],palette="mako",ax=ax[0][0])

ax[0,1].set_title("Odor of milk",fontsize=14)
c2=sb.countplot(data['Odor'],palette="Blues",ax=ax[0][1])

ax[1,0].set_title("Fat of milk",fontsize=14)
c3=sb.countplot(data['Fat '],palette="Blues_r",ax=ax[1][0])

ax[1,1].set_title("Turbidity of milk",fontsize=14)
c4=sb.countplot(data['Turbidity'],palette=["#0b445a","#0f6281"],ax=ax[1][1])
plt.show()
```



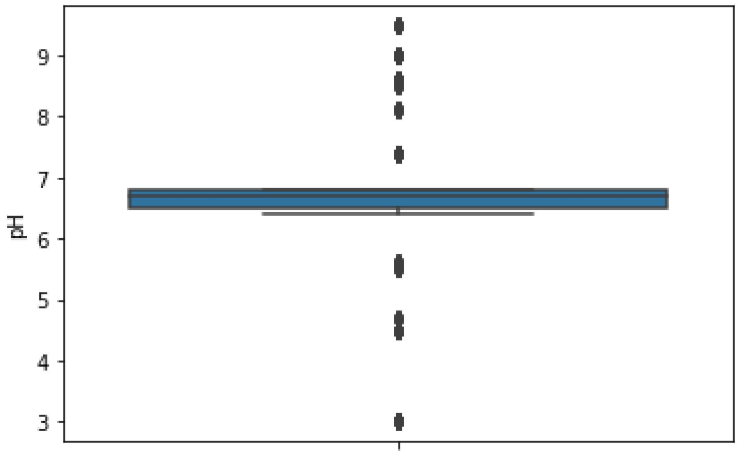
Box plot representation

In [107]:

```
sb.boxplot(y="pH",data=data)
```

Out[107]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f4b8d5a90>
```

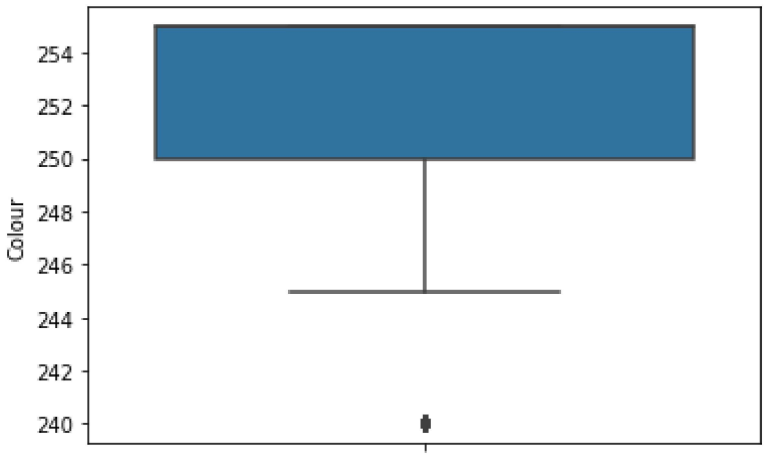


In [156]:

```
sb.boxplot(y="Colour",data=data)
```

Out[156]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f4e55c208>
```

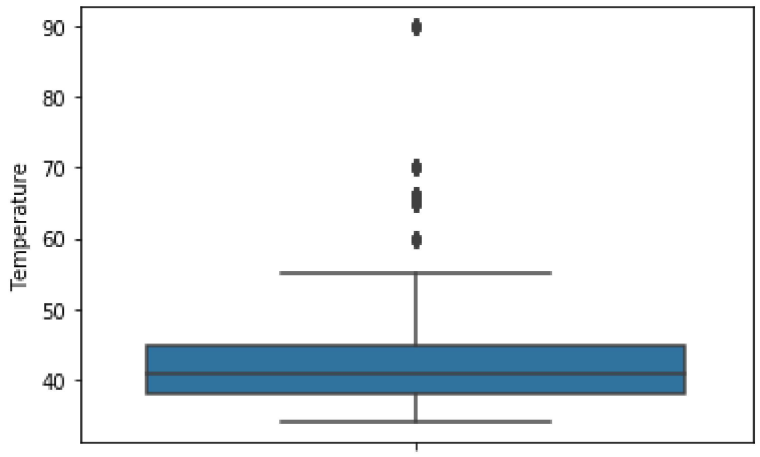


In [108]:

```
sb.boxplot(y="Temperature",data=data)
```

Out[108]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f4ba41860>
```

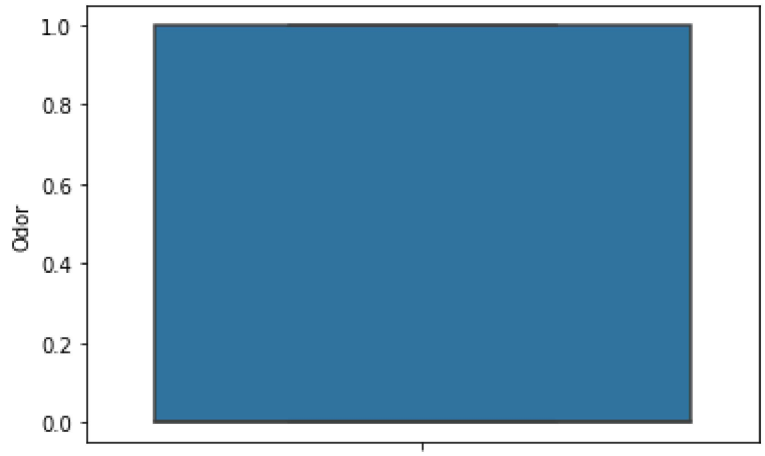


In [109]:

```
sb.boxplot(y="Odor",data=data)
```

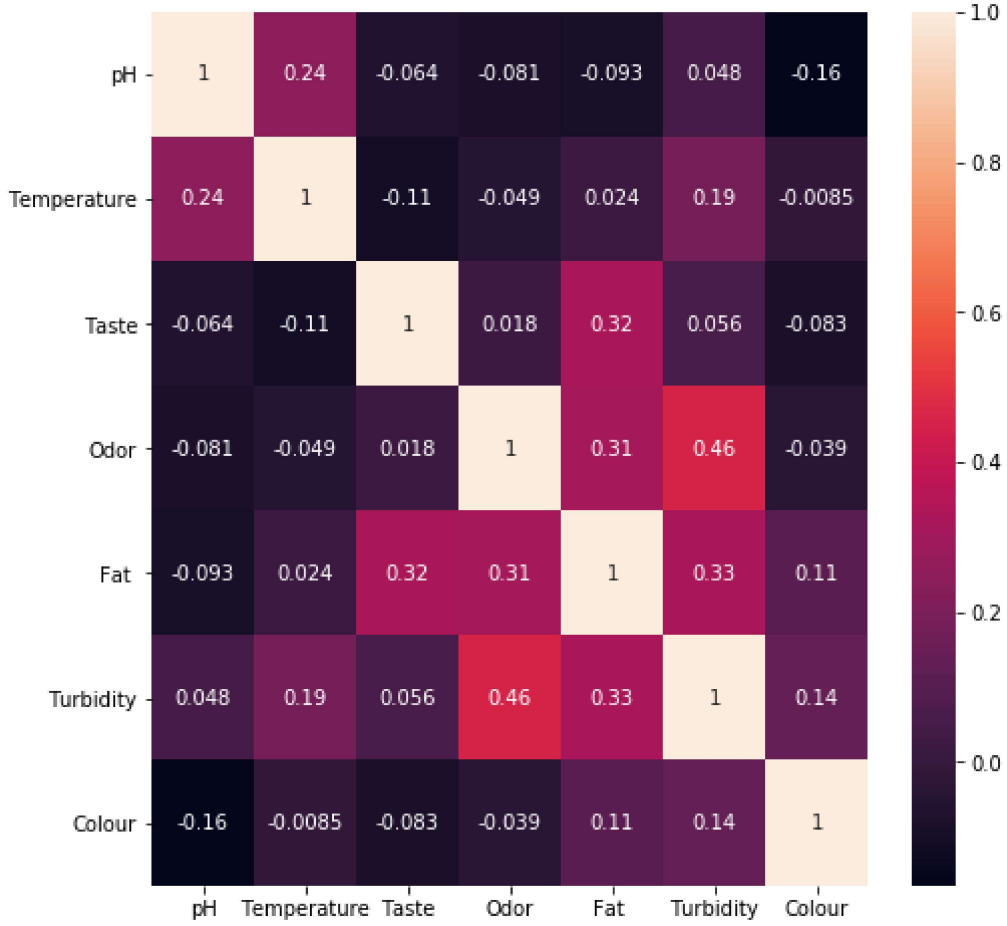
Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f4ba98198>
```



In [110]:

```
plt.figure(figsize=(8,8))
sb.heatmap(data.corr(),annot=True)
plt.show()
```



How odor of milk and turbidity of it are connected?

In [111]:

```
pd.DataFrame(data.groupby("Odor")["Turbidity"].mean())
```

Out[111]:

Odor	Turbidity
0	0.291181
1	0.753275

From this we can infer that bad odor milk has quite low turbidity and the opposite for good odor milk!

Finding the optimal temperature at which major features of milk are good?

In [112]:

```
pd.DataFrame(data.groupby("Temperature")["Taste","Odor","Turbidity"].mean())
```

Out[112]:

Temperature	Taste	Odor	Turbidity
34	0.600000	0.600000	1.000000
35	0.652174	0.347826	0.347826
36	0.000000	0.333333	0.015152
37	0.457831	0.289157	0.289157
38	0.692737	0.240223	0.240223
40	0.977273	0.530303	0.537879
41	0.266667	0.033333	0.000000
42	1.000000	1.000000	1.000000
43	0.974026	0.298701	0.922078
45	0.497717	0.716895	0.598174
50	0.172414	0.448276	0.534483
55	0.083333	0.833333	0.812500
60	0.000000	1.000000	1.000000
65	0.000000	0.000000	0.000000
66	1.000000	0.000000	1.000000
70	0.045455	0.045455	0.045455
90	1.000000	0.000000	1.000000

From the given Dataframe we can infer that 42°C as the optimal temperature at which the odor of the milk is good as well as the taste!

Finding the optimal colour value at which major features of milk are good?

In [113]:

```
pd.DataFrame(data.groupby("Colour")["Taste","Odor","Turbidity"].mean())
```

Out[113]:

Colour	Taste	Odor	Turbidity
240	0.000000	0.500000	0.500000
245	0.869565	0.600000	0.217391
246	0.522727	0.022727	0.022727
247	0.645833	0.500000	0.000000
248	1.000000	1.000000	1.000000
250	0.547945	0.273973	0.917808
253	0.000000	1.000000	0.045455
254	1.000000	0.000000	0.000000
255	0.511146	0.418790	0.509554

From the given Dataframe we can infer that 248 as the optimal colour range at which the odor of the milk is good as well as the taste!

Does Higher pH ensure we have better tasting milk?

In [114]:

```
pd.DataFrame(data.groupby("pH")["Taste","Odor"].mean())
```

Out[114]:

pH	Taste	Odor
3.0	1.000000	0.728571
4.5	0.000000	1.000000
4.7	1.000000	0.000000
5.5	1.000000	0.000000
5.6	0.000000	1.000000
6.4	0.000000	1.000000
6.5	0.518519	0.142857
6.6	0.383648	0.358491
6.7	1.000000	0.609756
6.8	0.393574	0.514056
7.4	0.435897	0.000000
8.1	1.000000	0.000000
8.5	0.045455	0.045455
8.6	0.000000	1.000000
9.0	1.000000	0.377049
9.5	1.000000	1.000000

We can infer from the given dataframe that it isn't necessary but at the same time milk should not have low pH values

Having higher fat content makes milk taste better?

In [115]:

```
pd.DataFrame(data.groupby("Fat ")["Taste"].mean())
```

Out[115]:

Fat	Taste
0	0.316092
1	0.659634

Well it does seem that higher fat content in milk does make the milk taste quite better.

Having higher turbidity content makes milk taste better?

In [153]:

```
pd.DataFrame(data.groupby("Turbidity")["Taste"].mean())
```

Out[153]:

Turbidity	Taste
0	0.519481
1	0.575000

We infer from the given dataframe that it isn't necessary to have higher turbidity for better milk taste.

Having higher odor content makes milk taste better?

In [154]:

```
pd.DataFrame(data.groupby("Odor")["Taste"].mean())
```

Out[154]:

Odor	Taste
0	0.539101
1	0.556769

We infer from the given dataframe that it isn't necessary to have good odor for better milk taste.

Performing groupby operation

In [116]:

```
B=data.groupby("Grade")
B
```

Out[116]:

```
<pandas.core.groupby.groupby.DataFrameGroupBy object at 0x0000022F4BD1C7B8>
```

In [158]:

```
B1=data.groupby("Grade")[["pH"]].mean().reset_index()
B1
```

Out[158]:

	Grade	pH
0	high	6.692578
1	low	6.588578
2	medium	6.635027

So the milk having the grade high has high pH and color value so we select this grade to analyze and create dataframe

In [118]:

```
X=B.get_group("high")
X
```

Out[118]:

	pH	Temperature	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
5	6.6	37	1	1	1	1	255	high
14	6.8	45	0	1	1	1	255	high
20	6.6	45	0	1	1	1	250	high
23	6.6	45	1	1	1	1	245	high
25	6.5	36	0	1	1	0	253	high
29	6.6	37	1	0	1	0	255	high
32	6.8	42	1	1	1	1	255	high
35	6.8	45	0	1	1	1	255	high
39	6.5	38	1	1	1	1	255	high
43	6.8	40	1	1	1	1	255	high
47	6.8	45	1	1	1	0	245	high
50	6.6	37	1	1	1	1	255	high
61	6.7	38	1	0	1	0	255	high
66	6.8	45	0	1	1	1	255	high
72	6.6	45	0	1	1	1	250	high
75	6.8	45	1	1	1	1	245	high
77	6.8	36	0	1	1	0	253	high
81	6.6	37	1	0	1	0	255	high
85	6.5	38	1	1	1	1	255	high
89	6.8	40	1	1	1	1	255	high
93	6.8	45	1	1	1	0	245	high
98	6.6	37	1	1	1	1	255	high
105	6.6	35	0	1	1	1	255	high
109	6.7	38	1	0	1	0	255	high
111	6.6	43	1	0	1	1	250	high
114	6.8	45	0	1	1	1	255	high
116	6.6	38	1	0	1	0	255	high
123	6.8	45	1	1	1	1	245	high
...	...	...	...	...	...	...	...	...
950	6.8	36	0	1	1	0	253	high
954	6.6	37	1	0	1	0	255	high
957	6.8	41	1	1	1	0	255	high

	pH	Temperature	Taste	Odor	Fat	Turbidity	Colour	Grade
958	6.5	38	1	1	1	1	255	high
963	6.7	38	1	0	1	0	255	high
965	6.6	43	1	0	1	1	250	high
970	6.7	38	1	0	1	0	255	high
972	6.8	43	1	0	1	0	250	high
975	6.8	45	0	1	1	1	255	high
981	6.6	45	0	1	1	1	250	high
984	6.8	45	1	1	1	1	245	high
986	6.8	36	0	1	1	0	253	high
990	6.6	37	1	0	1	0	255	high
994	6.5	38	1	1	1	1	255	high
996	6.7	38	1	0	1	0	255	high
1001	6.8	45	0	1	1	1	255	high
1011	6.6	45	0	1	1	1	250	high
1013	6.8	45	0	1	1	1	255	high
1019	6.6	45	0	1	1	1	250	high
1022	6.8	45	1	1	1	1	245	high
1024	6.8	36	0	1	1	0	253	high
1028	6.6	37	1	0	1	0	255	high
1032	6.5	38	1	1	1	1	255	high
1036	6.8	40	1	1	1	1	255	high
1040	6.8	45	1	1	1	0	245	high
1043	6.8	40	1	1	1	1	255	high
1047	6.8	45	1	1	1	0	245	high
1050	6.6	37	1	1	1	1	255	high
1055	6.7	38	1	0	1	0	255	high
1057	6.8	43	1	0	1	0	250	high

256 rows × 8 columns

In [119]:

```
grade_high=pd.DataFrame(X)
grade_high.describe()
```

Out[119]:

	pH	Temperature	Taste	Odor	Fat	Turbidity	Colour
count	256.000000	256.000000	256.000000	256.000000	256.000000	256.000000	256.000000
mean	6.692578	40.648438	0.664062	0.750000	0.996094	0.632812	252.539062
std	0.108752	3.739749	0.473242	0.433861	0.062500	0.482982	3.711989
min	6.500000	35.000000	0.000000	0.000000	0.000000	0.000000	245.000000
25%	6.600000	37.000000	0.000000	0.750000	1.000000	0.000000	250.000000
50%	6.700000	40.000000	1.000000	1.000000	1.000000	1.000000	255.000000
75%	6.800000	45.000000	1.000000	1.000000	1.000000	1.000000	255.000000
max	6.800000	45.000000	1.000000	1.000000	1.000000	1.000000	255.000000

In [120]:

```
fat_value=grade_high.groupby("Fat ")[["Colour"]].mean().reset_index()
fat_value
```

Out[120]:

	Fat	Colour
0	0	253.000000
1	1	252.537255

There is no relation between fat content and colour of milk.

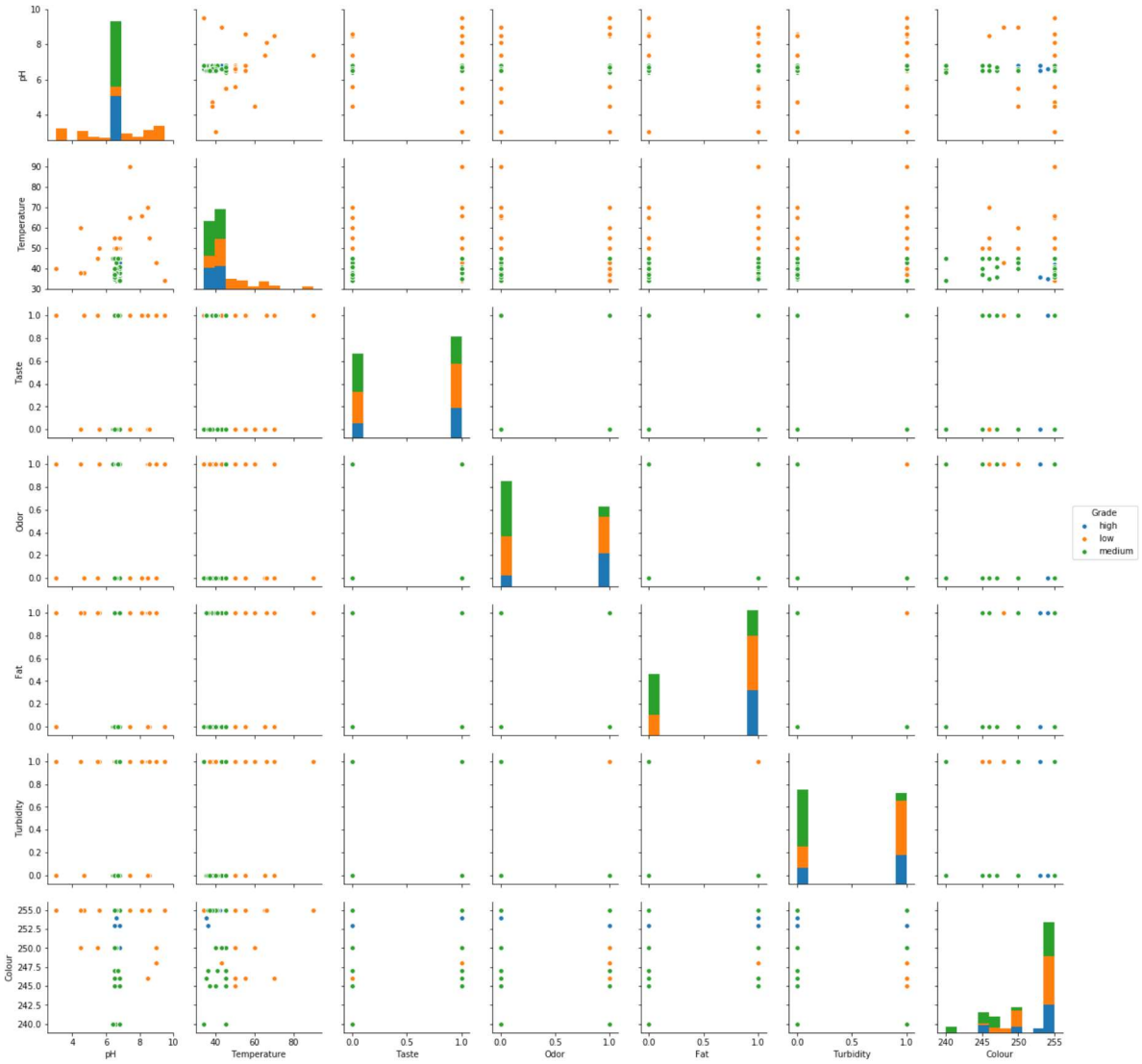
Splitting the dataset into Training and Test:

In [121]:

```
X=data.drop("Grade",axis=1).values
y=data[["Grade"]]
```

In [122]:

```
p=sb.pairplot(data,hue="Grade")
plt.show()
```



Scaling the data

In [129]:

```
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

In [130]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=43)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(847, 7)
(847,)
(212, 7)
(212,)
```

```
Splitting training data and testing data
```

Building a Machine Learning Model

Logistic Regression

In [131]:

```
from sklearn.linear_model import LogisticRegression
```

In [132]:

```
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

In [133]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=5)
```

In [134]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(xtrain,ytrain)
```

Out[134]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [135]:

```
ypred=model.predict(xtest)
```

In [136]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest, ypred)*100)
```

```
78.77358490566037
```

In [137]:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(ytest, ypred))
```

```
[[28 20  6]
 [ 5 71  7]
 [ 3  4 68]]
```

In [138]:

```
print(model.predict([[6.5,34,1,0,1,220,1]]))
```

```
['low']
```

Accuracy Score of Logistic Regression is 78.77%

K-Nearest Neighbours Classifier

In [139]:

```
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

In [140]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=43)
```

In [141]:

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=1)
model.fit(xtrain,ytrain)
```

Out[141]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

In [142]:

```
ypred=model.predict(xtest)
```

In [143]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest, ypred)*100)
```

```
99.05660377358491
```

In [144]:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(ytest, ypred))
```

```
[[50  0  1]
 [ 0 82  0]
 [ 1  0 78]]
```

In [145]:

```
print(model.predict([[6.5,34,1,0,1,220,1]]))
```

```
['medium']
```

Accuracy Score of K-Nearest Neighbours Classifier is 99.05%

Decision Tree Classifier

In [146]:

```
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

In [147]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=47)
```

In [148]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
model.fit(xtrain,ytrain)
```

Out[148]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [149]:

```
ypred=model.predict(xtest)
```

In [150]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest, ypred)*100)
```

```
99.05660377358491
```

In [151]:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(ytest, ypred))
```

```
[[46  0  2]
 [ 0 94  0]
 [ 0  0 70]]
```

In [152]:

```
print(model.predict([[5.5,45,1,0,1,1,250]]))
```

```
['low']
```

Accuracy Score of Decision Tree Classifier is 99.05%