Diamonds

July 11, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style('darkgrid')
```

Here, the "diamonds.csv" file has been downloaded.

[2]:
```
dia = pd.read_csv('diamonds.csv')
dia.head()
```

[2]:
```
   Unnamed: 0  carat      cut color clarity  depth  table  price     x     y  \
0           1   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98
1           2   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84
2           3   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07
3           4   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23
4           5   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35

      z
0  2.43
1  2.31
2  2.31
3  2.63
4  2.75
```

A correlation matrix and heatmap for the same have been created.

[3]:
```
dia.corr()
```

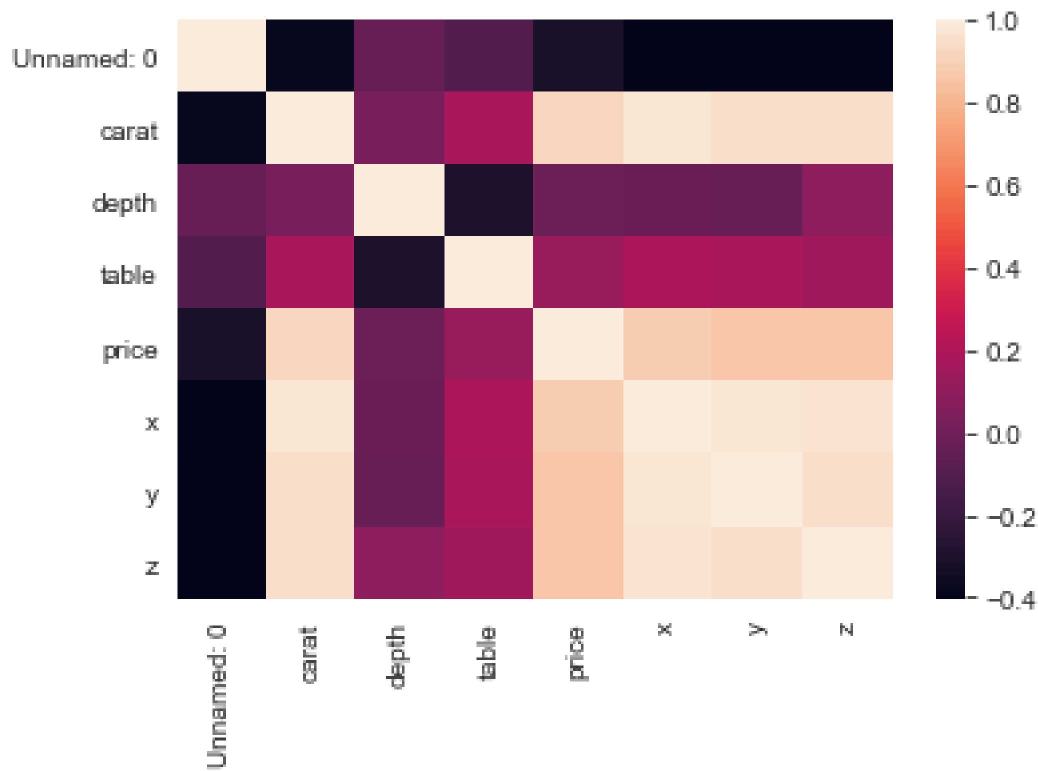
[3]:
```
            Unnamed: 0     carat     depth     table     price         x  \
Unnamed: 0    1.000000 -0.377983 -0.034800 -0.100830 -0.306873 -0.405440
carat        -0.377983  1.000000  0.028224  0.181618  0.921591  0.975094
depth        -0.034800  0.028224  1.000000 -0.295779 -0.010647 -0.025289
table        -0.100830  0.181618 -0.295779  1.000000  0.127134  0.195344
price        -0.306873  0.921591 -0.010647  0.127134  1.000000  0.884435
x            -0.405440  0.975094 -0.025289  0.195344  0.884435  1.000000
y            -0.395843  0.951722 -0.029341  0.183760  0.865421  0.974701
```

```
z            -0.399208  0.953387  0.094924  0.150929  0.861249  0.970772

                   y         z
Unnamed: 0 -0.395843 -0.399208
carat       0.951722  0.953387
depth      -0.029341  0.094924
table       0.183760  0.150929
price       0.865421  0.861249
x           0.974701  0.970772
y           1.000000  0.952006
z           0.952006  1.000000
```

[4]:
```
sns.heatmap(dia.corr())
```

[4]: <matplotlib.axes._subplots.AxesSubplot at 0x168f7edd308>



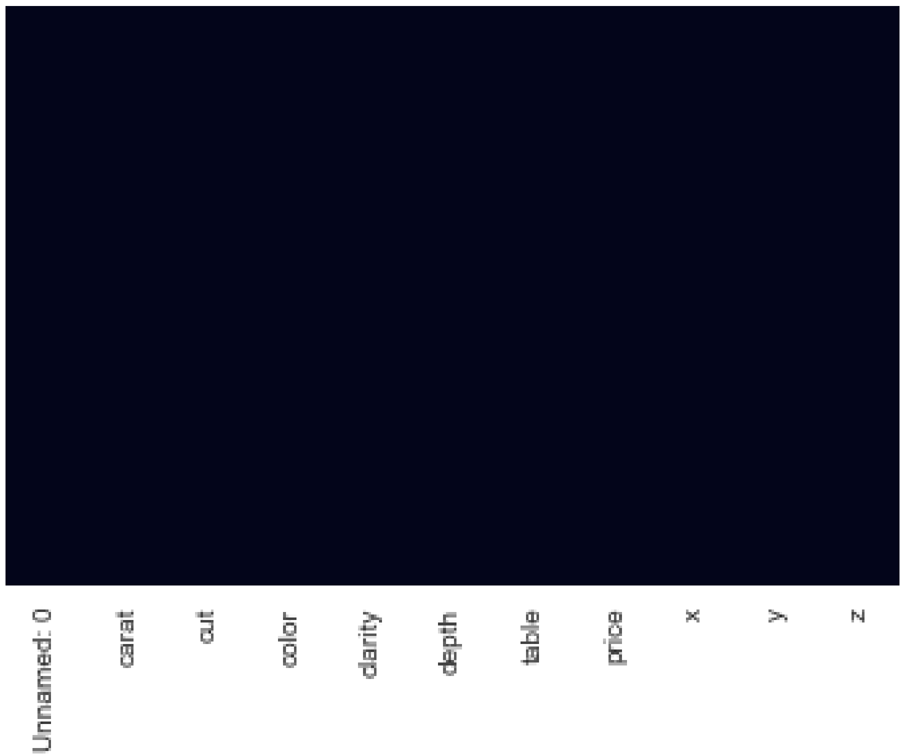
These matrices are an intersection between the various variables with each other (including themselves). Depending on the magnitude of the intersection, one can easily conclude whether a correlation exists between the two variables or not.

For instance, variable pairs such as - x,y,z and carat; x and y; y and z; z and x; price and carat etc. share a strong positive correlation.

[5]:

```python
sns.heatmap(dia.isnull(), yticklabels = False, cbar = False)
```

[5]: <matplotlib.axes._subplots.AxesSubplot at 0x168fb64e3c8>



The plot above indicates that the data set has no missing values.

[6]:

```python
dia.describe()
```

[6]:

```
         Unnamed: 0         carat         depth         table         price  \
count  53940.000000  53940.000000  53940.000000  53940.000000  53940.000000
mean   26970.500000      0.797940     61.749405     57.457184   3932.799722
std    15571.281097      0.474011      1.432621      2.234491   3989.439738
min        1.000000      0.200000     43.000000     43.000000    326.000000
25%    13485.750000      0.400000     61.000000     56.000000    950.000000
50%    26970.500000      0.700000     61.800000     57.000000   2401.000000
75%    40455.250000      1.040000     62.500000     59.000000   5324.250000
max    53940.000000      5.010000     79.000000     95.000000  18823.000000

                  x             y             z
count  53940.000000  53940.000000  53940.000000
mean       5.731157      5.734526      3.538734
std        1.121761      1.142135      0.705699
min        0.000000      0.000000      0.000000
```

```
25%        4.710000     4.720000     2.910000
50%        5.700000     5.710000     3.530000
75%        6.540000     6.540000     4.040000
max       10.740000    58.900000    31.800000
```

[7]:

```
dia.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  53940 non-null  int64
 1   carat       53940 non-null  float64
 2   cut         53940 non-null  object
 3   color       53940 non-null  object
 4   clarity     53940 non-null  object
 5   depth       53940 non-null  float64
 6   table       53940 non-null  float64
 7   price       53940 non-null  int64
 8   x           53940 non-null  float64
 9   y           53940 non-null  float64
 10  z           53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

Most of the variables are numeric in nature.

[40]:

```
dia.columns
```

[40]:

```
Index(['Unnamed: 0', 'carat', 'cut', 'color', 'clarity', 'depth', 'table',
       'price', 'x', 'y', 'z'],
      dtype='object')
```

Fig-1: Scatter plot for carat and price.

[21]:

```
plt.figure(figsize = (10,10))
sns.lmplot(data = dia, x = 'carat', y = 'price')
```

[21]:

```
<seaborn.axisgrid.FacetGrid at 0x168fe63a6c8>
```

```
<Figure size 720x720 with 0 Axes>
```

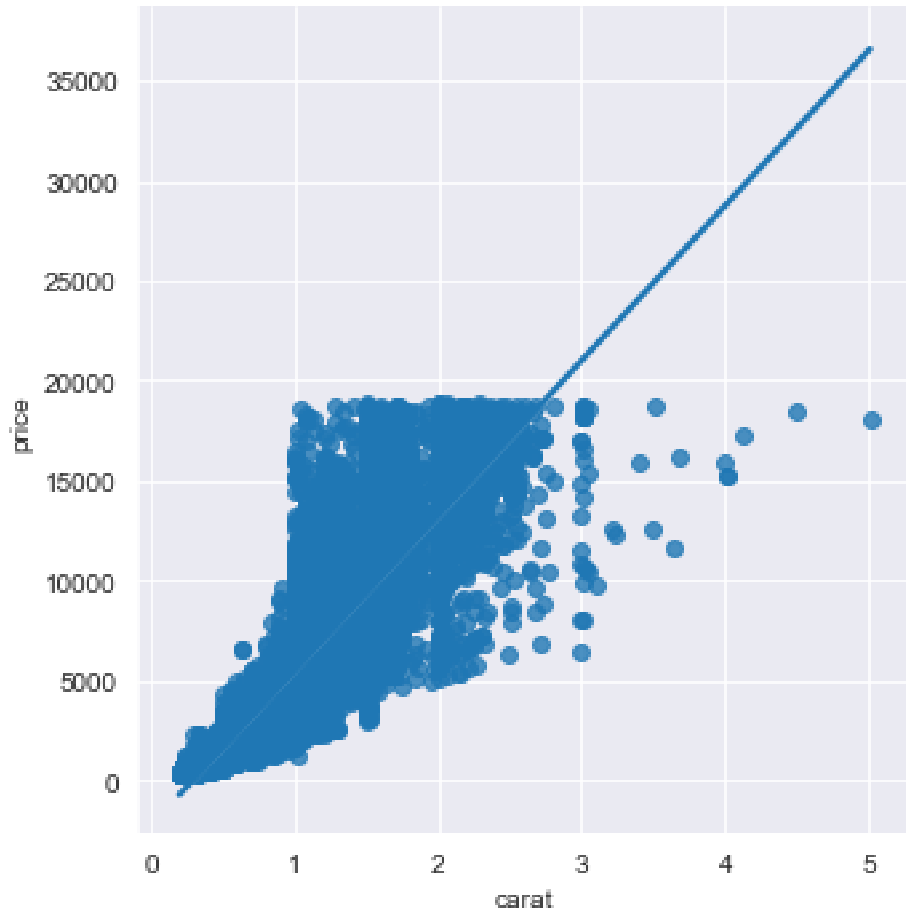
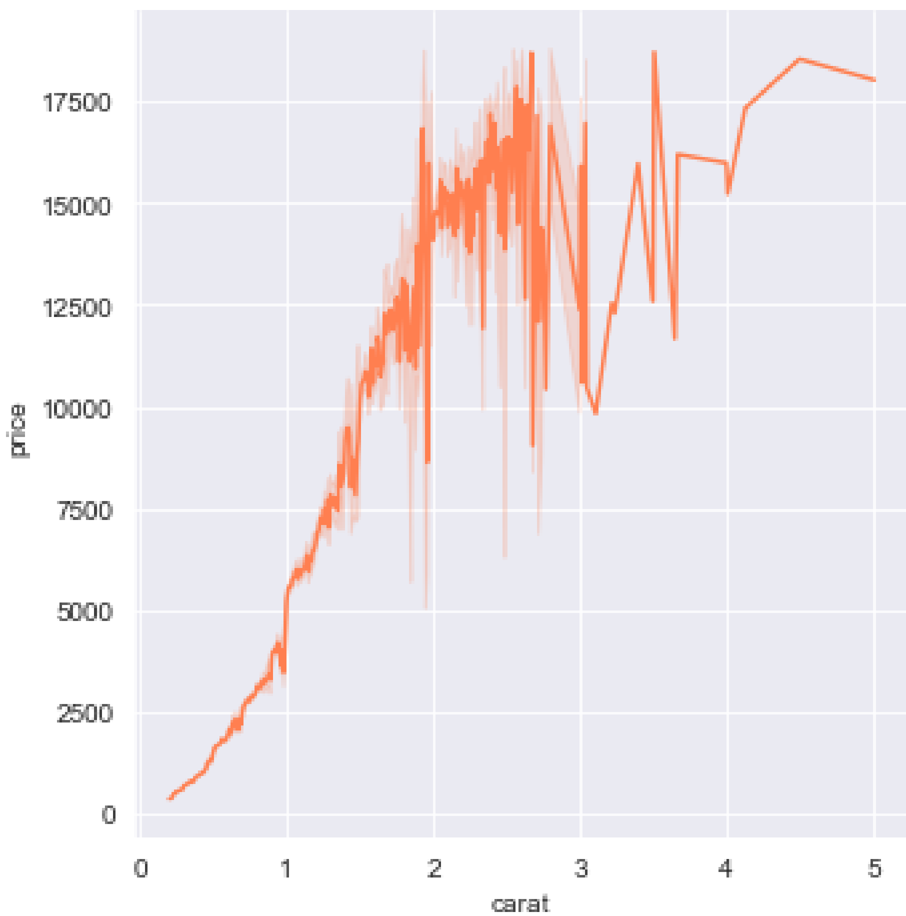


Fig-2: Relational plot for carat and price.

[22]:
```
plt.figure(figsize = (10,10))
sns.relplot(data = dia, x = 'carat', y = 'price', kind = 'line', color =␣
 ↪'coral')
```

[22]:
```
<seaborn.axisgrid.FacetGrid at 0x168fe69dfc8>
```

```
<Figure size 720x720 with 0 Axes>
```



COMMENT : We see that the trend is generally positive. As the number of carats increases, the price subsequently increases too. Even in real life, it is an established fact that the diamonds of a higher carat value will be more expensive.

[53]:
```
sns.distplot(dia['price'], color = 'darkgreen')
```

```
C:\Users\Yutika\miniconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will
be interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

[53]: <matplotlib.axes._subplots.AxesSubplot at 0x168837c5a48>

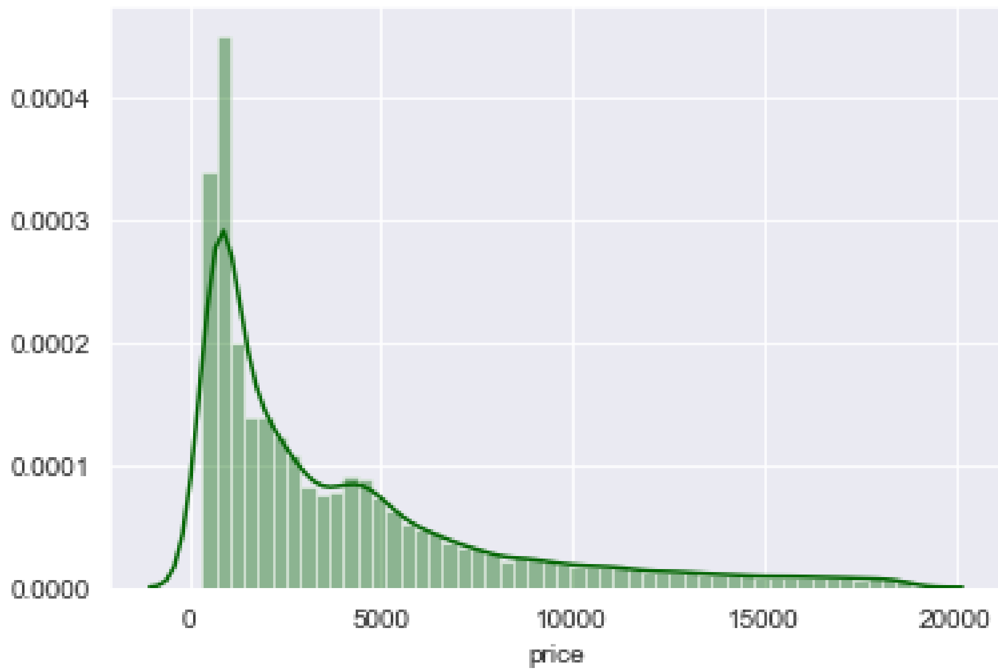
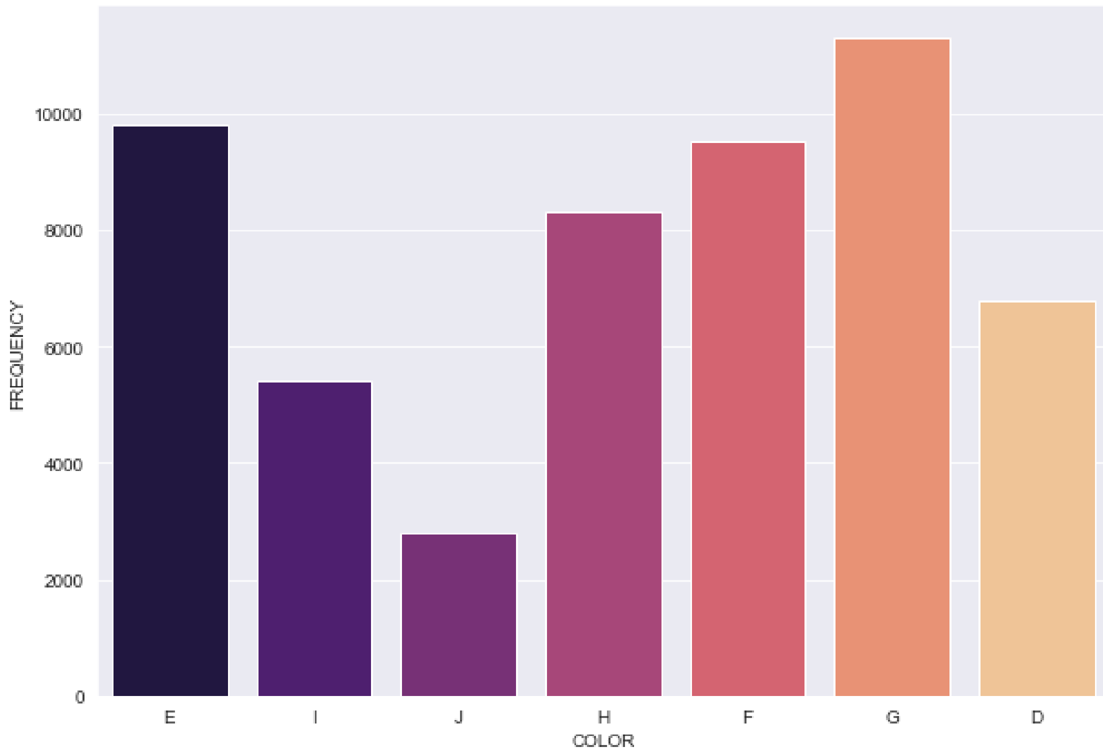


Fig-3: Countplots for color, cut and clarity.

i) Color:

[10]:
```
plt.figure(figsize = (10,7))
g1 = sns.countplot(dia['color'], palette = 'magma')
g1.set_xlabel('COLOR')
g1.set_ylabel('FREQUENCY')
```

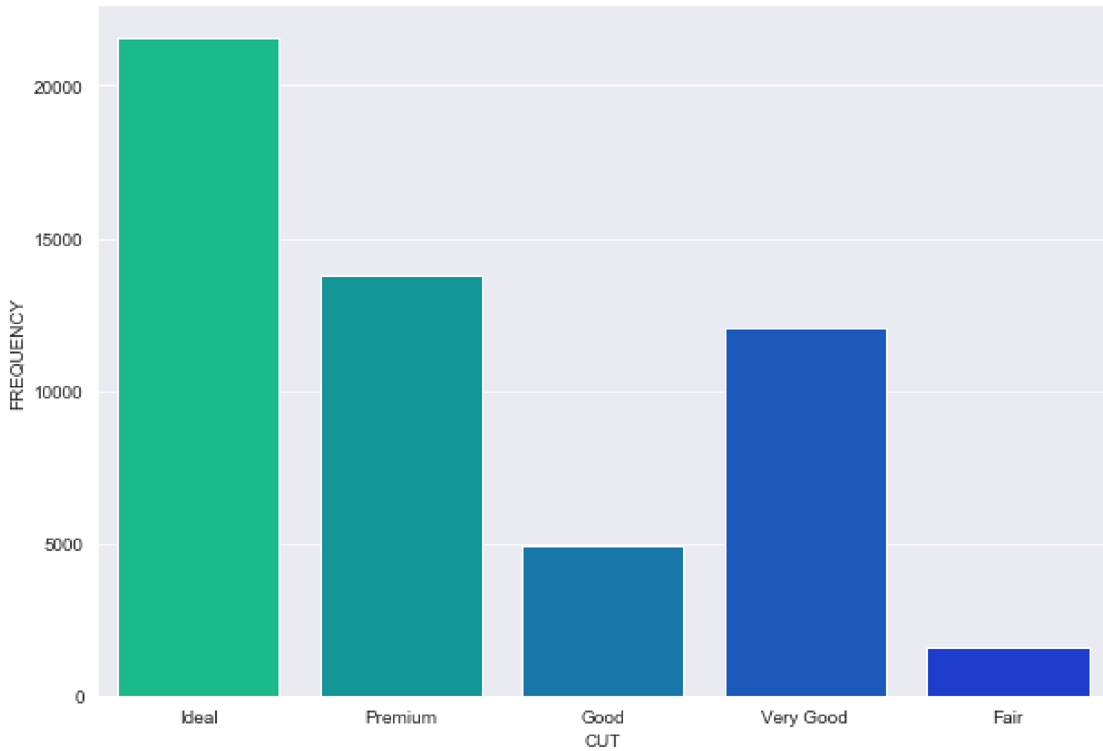
[10]: Text(0, 0.5, 'FREQUENCY')



ii) Cut:

[11]:
```
plt.figure(figsize=(10,7))
g2 = sns.countplot(dia['cut'], palette = 'winter_r')
g2.set_xlabel('CUT')
g2.set_ylabel('FREQUENCY')
```

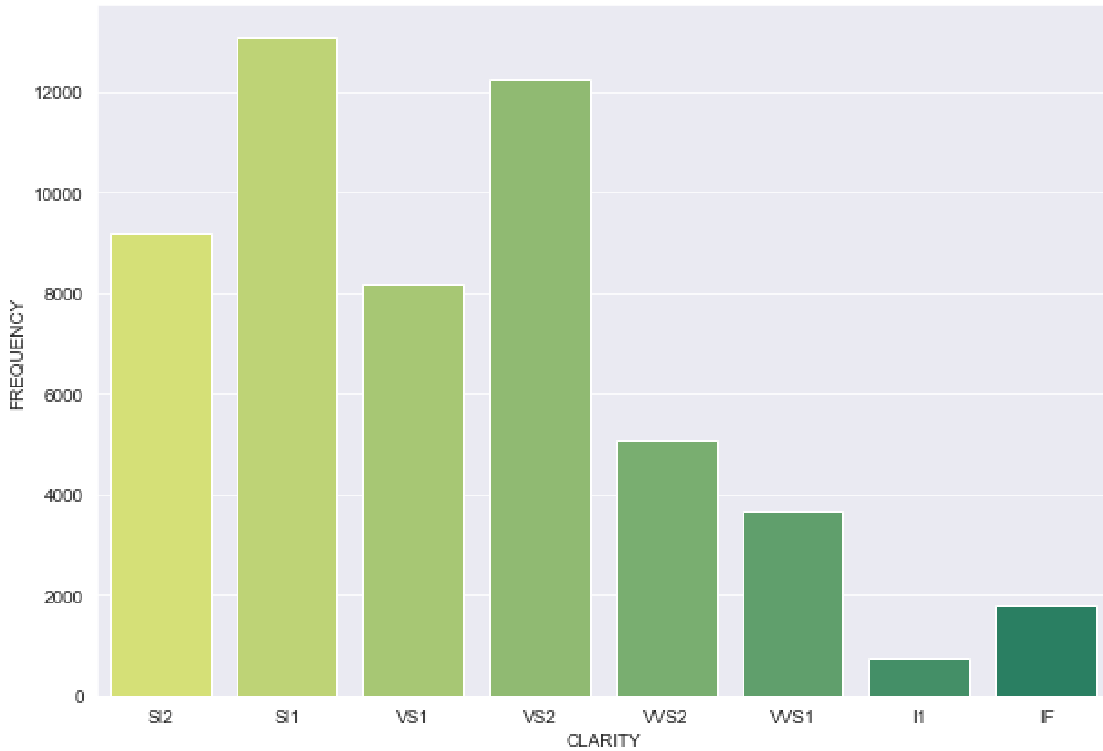
[11]: Text(0, 0.5, 'FREQUENCY')



iii) Clarity:

[12]:
```
plt.figure(figsize = (10,7))
g3 = sns.countplot(dia['clarity'], palette = 'summer_r')
g3.set_xlabel('CLARITY')
g3.set_ylabel('FREQUENCY')
```

[12]: `Text(0, 0.5, 'FREQUENCY')`



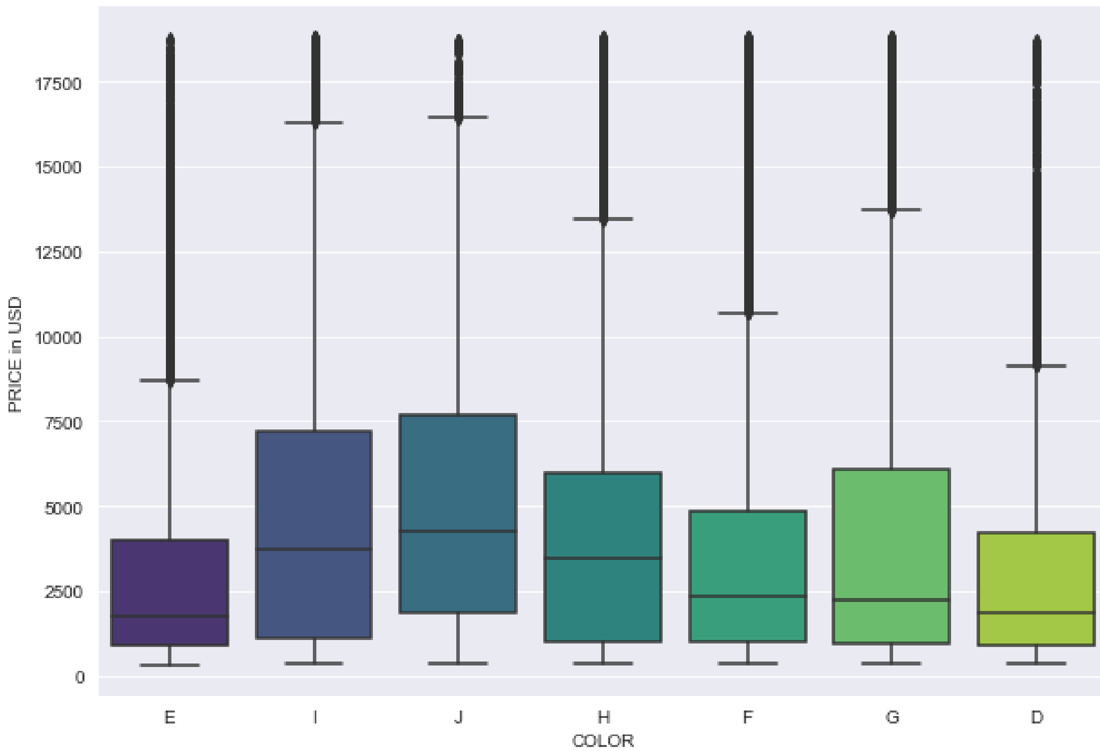
COMMENT : According to these charts, the sale of diamonds of color = 'G', cut = 'ideal' and clarity = 'SI1' is the most as opposed to the sale of diamonds of color = 'J', cut = 'Fair' and clarity = 'I1'; which is the least.

Fig-4: Boxplots for price as per color, cut and clarity.

i) Color v/s price :

[13]:
```
plt.figure(figsize = (10,7))
g4 = sns.boxplot(data = dia, x = 'color', y = 'price', palette = 'viridis')
g4.set_xlabel('COLOR')
g4.set_ylabel('PRICE in USD')
```

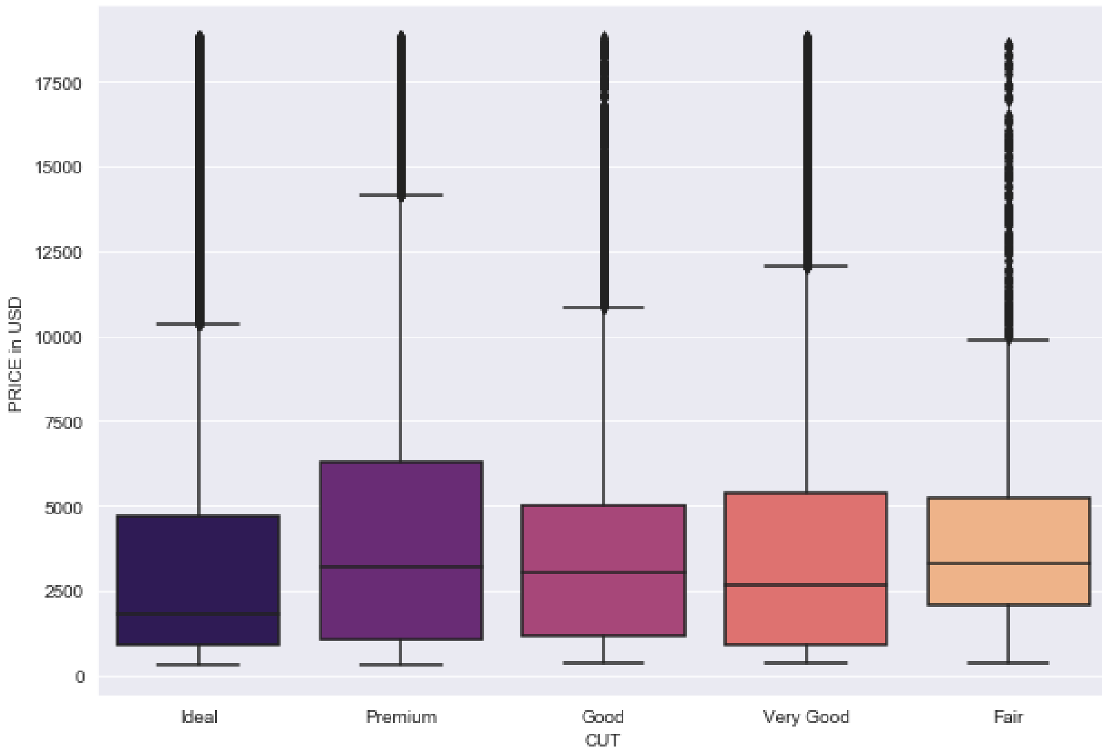
[13]:
```
Text(0, 0.5, 'PRICE in USD')
```



i) Cut v/s price :

[45]:
```
plt.figure(figsize = (10,7))
g5 = sns.boxplot(data = dia, x = 'cut', y = 'price', palette = 'magma')
g5.set_xlabel('CUT')
g5.set_ylabel('PRICE in USD')
```

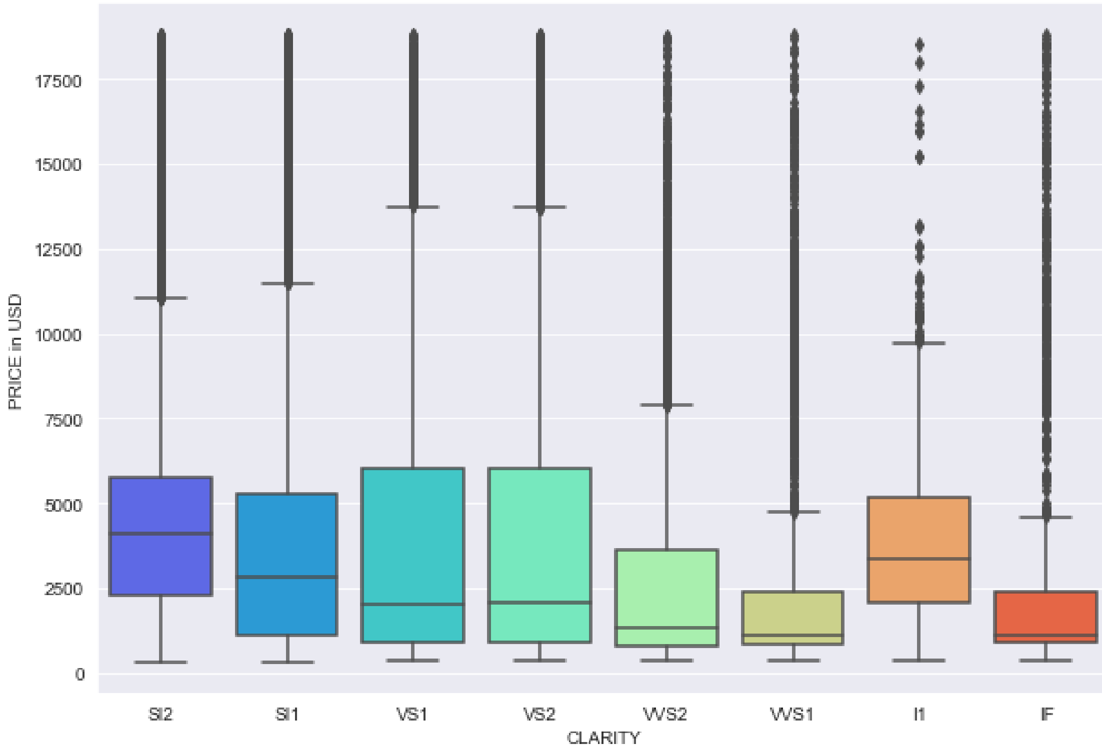
[45]: Text(0, 0.5, 'PRICE in USD')



iii) Clarity v/s price:

[15]:
```
plt.figure(figsize = (10,7))
g6 = sns.boxplot(data = dia, x = 'clarity', y = 'price', palette = 'rainbow')
g6.set_xlabel('CLARITY')
g6.set_ylabel('PRICE in USD')
```

[15]: Text(0, 0.5, 'PRICE in USD')



Statistical analysis :-

[46]:
```
import scipy.stats as stats
from scipy.stats import kurtosis
from scipy.stats import skew
```

Determining the type of distribution for variables- x,y and z.

i) Distribution for x :

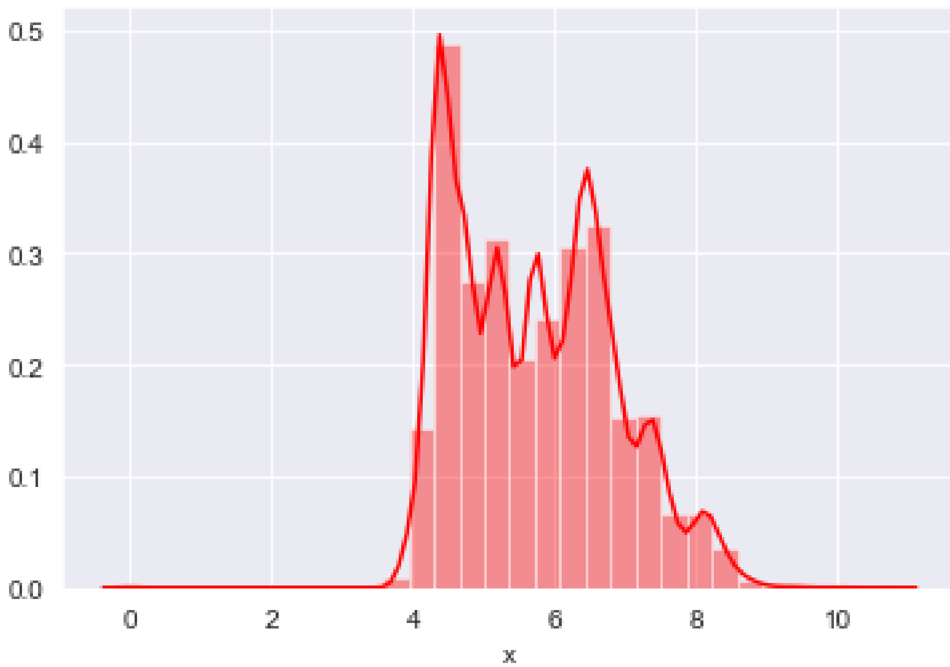
[48]:
```
s1 = skew(dia['x'])
k1 = kurtosis(dia['x'])
print("Skewness for x = ", s1)
print("Kurtosis for x = ", k1)
```

```
Skewness for x =  0.3786658120772097
Kurtosis for x =  -0.6182146042773282
```

The distribution for x is **positively skewed** and **highly platykurtic.**

[16]:
```
sns.distplot(dia['x'], color = 'red', bins = 30)
```

[16]: <matplotlib.axes._subplots.AxesSubplot at 0x168fdecf3c8>



ii) **Distribution for y :**

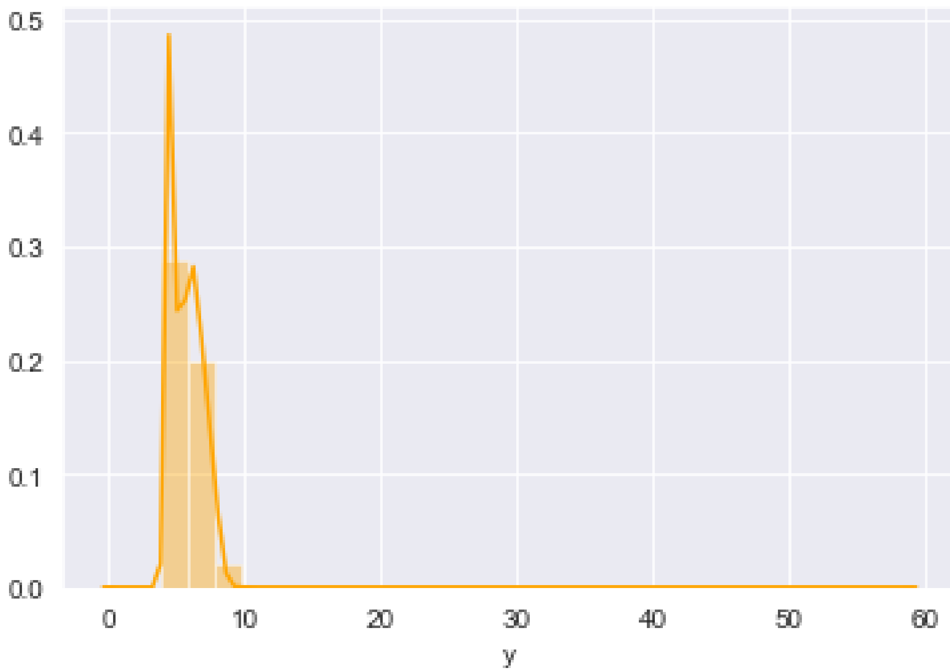
[49]:
```
s2 = skew(dia['y'])
k2 = kurtosis(dia['y'])
print("Skewness for y = ", s2)
print("Kurtosis for y = ", k2)
```

```
Skewness for y =  2.4340990250113648
Kurtosis for y =  91.20599095863467
```

The distribution for y is **positively skewed** and **highly leptokurtic.**

[17]:
```
sns.distplot(dia['y'], color = 'orange', bins = 30)
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x168fdf81148>



iii) Distribution for z :

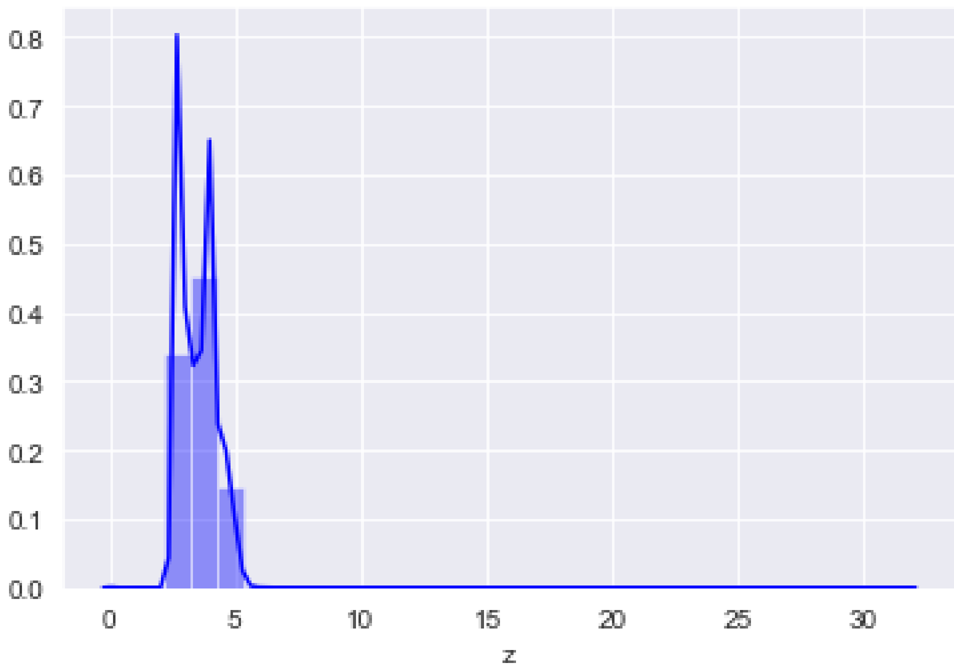
[50]:
```
s3 = skew(dia['z'])
k3 = kurtosis(dia['z'])
print("Skewness for z = ", s3)
print("Kurtosis for z = ", k3)
```

```
Skewness for z =  1.5223802221853722
Kurtosis for z =  47.08214348390816
```

The distribution for z is **positively skewed** and **highly leptokurtic.**

[18]:
```
sns.distplot(dia['z'], color = 'blue', bins = 30)
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x168fe291e88>



Hence, it is fair to conclude that neither x nor y nor z are normally distributed .

Cross tabulation for the various categorical variables:

i) Color and Clarity :

[37]:
```
pd.crosstab(dia['color'],dia['clarity'])
```

[37]:

clarity color	I1	IF	SI1	SI2	VS1	VS2	VVS1	VVS2
D	42	73	2083	1370	705	1697	252	553
E	102	158	2426	1713	1281	2470	656	991
F	143	385	2131	1609	1364	2201	734	975
G	150	681	1976	1548	2148	2347	999	1443
H	162	299	2275	1563	1169	1643	585	608
I	92	143	1424	912	962	1169	355	365
J	50	51	750	479	542	731	74	131

For this contingency table, it is observed that the sale of diamonds with a combination of **color** = **'E'** and **clarity** = **'VS2'** is maximum.

ii) Clarity and Cut :

[38]:
```
pd.crosstab(dia['clarity'],dia['cut'])
```

[38]:

cut clarity	Fair	Good	Ideal	Premium	Very Good

I1	210	96	146	205	84
IF	9	71	1212	230	268
SI1	408	1560	4282	3575	3240
SI2	466	1081	2598	2949	2100
VS1	170	648	3589	1989	1775
VS2	261	978	5071	3357	2591
VVS1	17	186	2047	616	789
VVS2	69	286	2606	870	1235

For this cross tabulation, the sale of diamonds of a combination of **clarity** = **'VS2'** and **cut** = **'Ideal'** is the most.

iii) Color and Cut :

[39]:
```
pd.crosstab(dia['color'],dia['cut'])
```

[39]:

cut	Fair	Good	Ideal	Premium	Very Good
color					
D	163	662	2834	1603	1513
E	224	933	3903	2337	2400
F	312	909	3826	2331	2164
G	314	871	4884	2924	2299
H	303	702	3115	2360	1824
I	175	522	2093	1428	1204
J	119	307	896	808	678

Diamonds of **color** = **'E'** and **cut** = **'ideal'** have the maximum sale.

FINAL CONCLUSION: The sale of diamonds which are a combination of **color** = **'E'**, **clarity** = **'VS2'** and **cut** = **'ideal'** have the maximum sale. This differs from the conclusion that was derived from the countplots which stated *"the sale of diamonds of color = 'G', cut = 'ideal' and clarity = 'SI1' is the most as opposed to the sale of diamonds of color = 'J', cut = 'Fair' and clarity = 'I1'; which is the least."*