In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
diamond_data = pd.read_csv('diamonds.csv')
```

In [3]:

```
diamond_data.head()
```

Out[3]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [4]:

```
diamond_data.tail()
```

Out[4]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

In [8]:

```
diamond_data = diamond_data.drop('Unnamed: 0', axis = 1)
```

In [9]:

```
diamond_data.shape
```

Out[9]:

```
(53940, 10)
```

In [10]:

```
diamond_data.columns
```

Out[10]:

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

In [11]:

```
diamond_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   carat    53940 non-null  float64
 1   cut      53940 non-null  object 
 2   color    53940 non-null  object 
 3   clarity  53940 non-null  object 
 4   depth    53940 non-null  float64
 5   table    53940 non-null  float64
 6   price    53940 non-null  int64  
 7   x        53940 non-null  float64
 8   y        53940 non-null  float64
 9   z        53940 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.1+ MB
```

In [12]:

```
diamond_data.describe()
```

Out[12]:

	carat	depth	table	price	x	y
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000

In [13]:

```
diamond_data.duplicated().sum()
```

Out[13]:

146

In [16]:

```
diamond_data = diamond_data.drop_duplicates()
```

In [17]:

```
diamond_data.shape
```

Out[17]:

(53794, 10)

In [18]:

```
diamond_data.isnull().sum()
```

Out[18]:

```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

In [20]:

```
diamond_data.nunique()
```

Out[20]:

```
carat        273
cut            5
color          7
clarity        8
depth        184
table        127
price      11602
x            554
y            552
z            375
dtype: int64
```

In [21]:

```
diamond_data['cut'].unique()
```

Out[21]:

```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

In [22]:

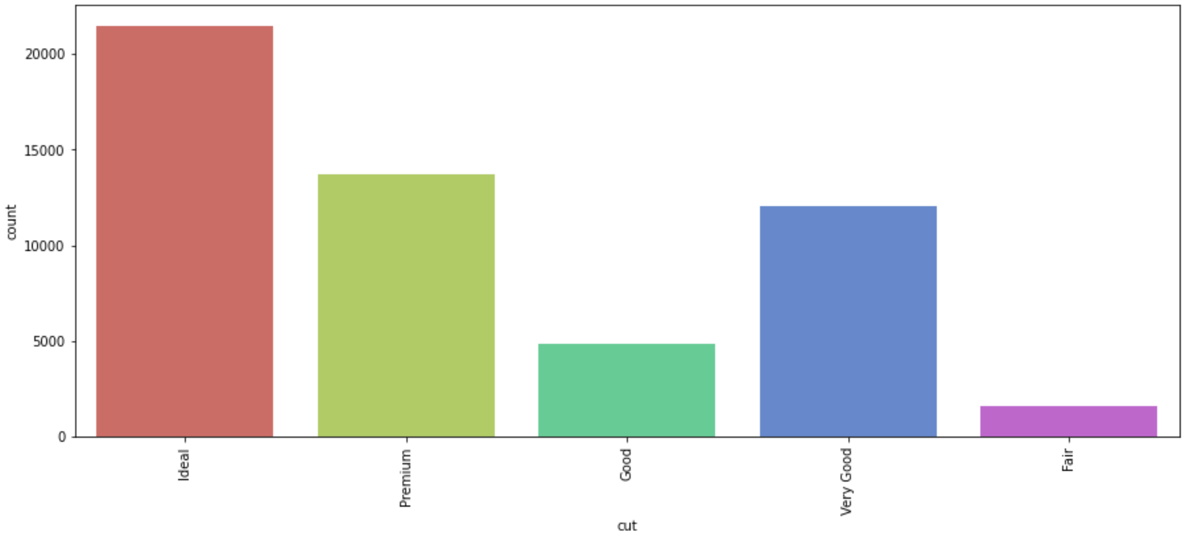
```
diamond_data['cut'].value_counts()
```

Out[22]:

```
Ideal        21488
Premium      13748
Very Good    12069
Good          4891
Fair          1598
Name: cut, dtype: int64
```

In [23]:

```
plt.figure(figsize=(15,6))
sns.countplot(diamond_data['cut'], data = diamond_data,
                  palette='hls')
plt.xticks(rotation = 90)
plt.show()
```

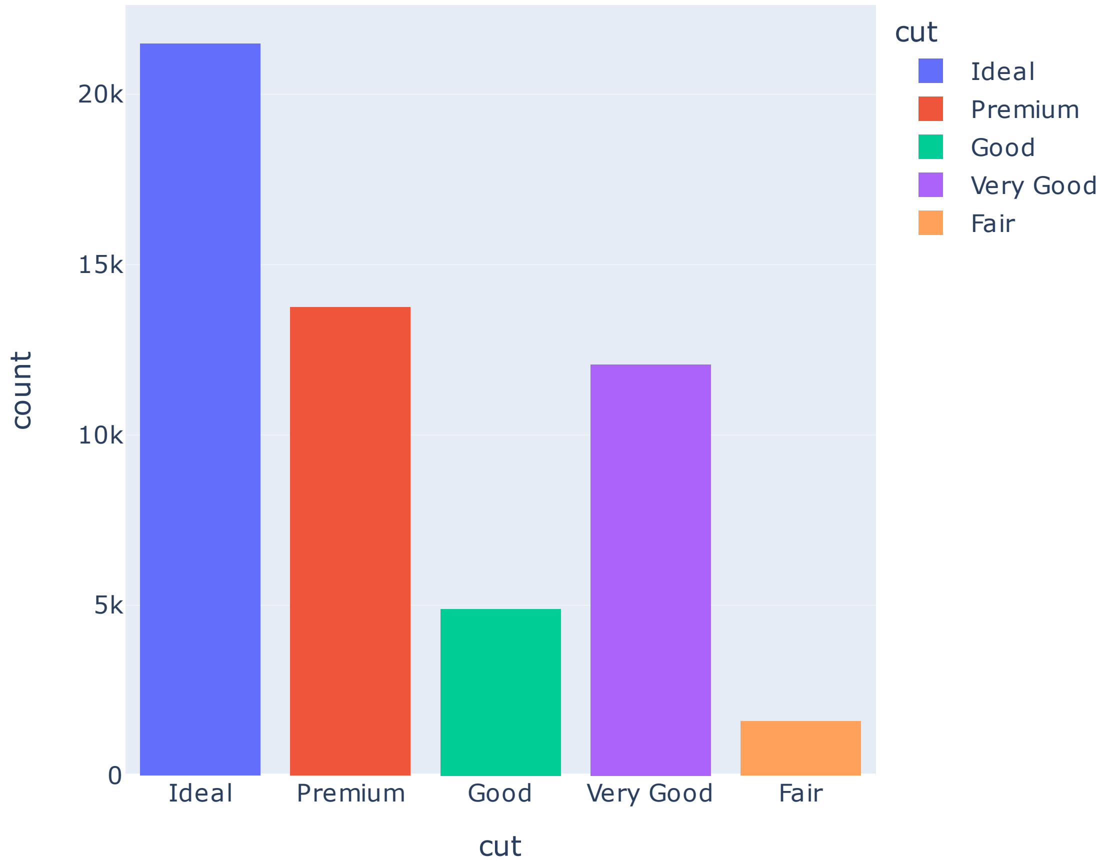


In [24]:

```
import plotly.express as px
```

In [25]:

```
fig1 = px.histogram(diamond_data, x = 'cut', color = 'cut')
fig1.show()
```



In [26]:

```
diamond_data['color'].unique()
```

Out[26]:

```
array(['E', 'I', 'J', 'H', 'F', 'G', 'D'], dtype=object)
```

In [27]:

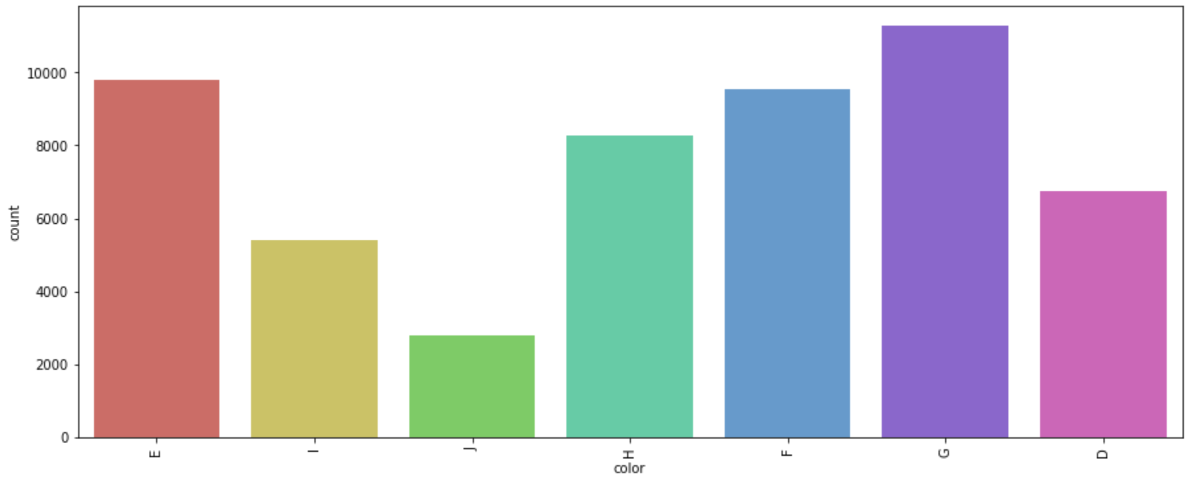
```
diamond_data['color'].value_counts()
```

Out[27]:

```
G    11262
E     9776
F     9520
H     8272
D     6755
I     5407
J     2802
Name: color, dtype: int64
```

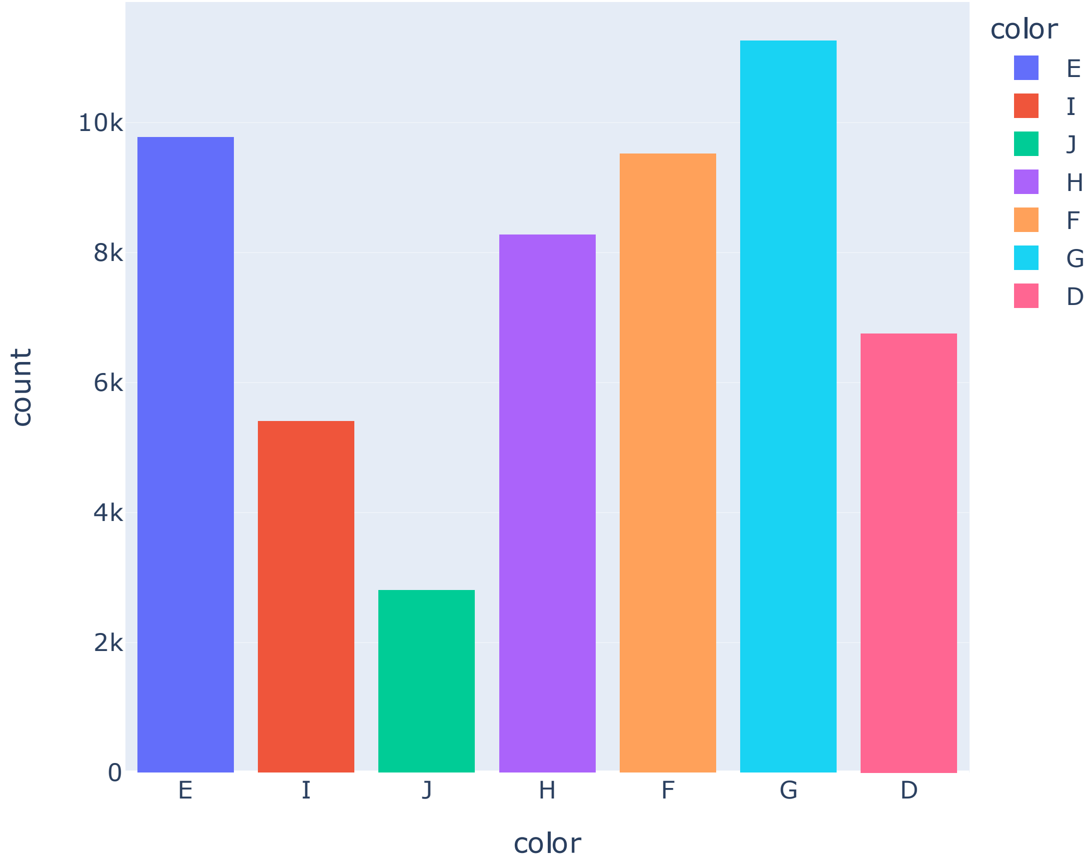
In [28]:

```
plt.figure(figsize=(15,6))
sns.countplot(diamond_data['color'], data = diamond_data,
                  palette='hls')
plt.xticks(rotation = 90)
plt.show()
```



In [29]:

```
fig2 = px.histogram(diamond_data, x = 'color', color = 'color')
fig2.show()
```



In [30]:

```
diamond_data['clarity'].unique()
```

Out[30]:

```
array(['SI2', 'SI1', 'VS1', 'VS2', 'VVS2', 'VVS1', 'I1', 'IF'],
      dtype=object)
```

In [31]:

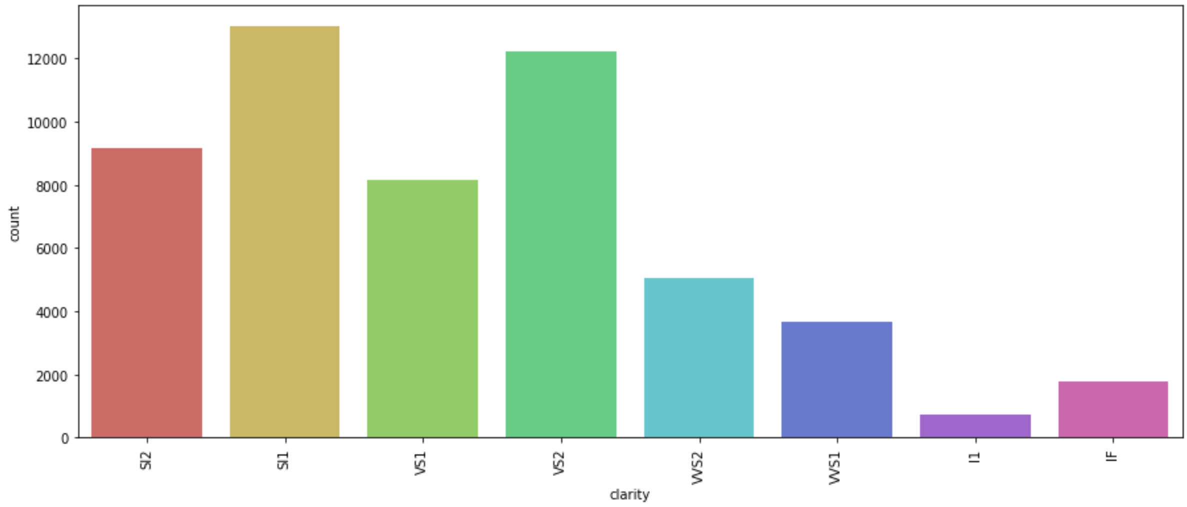
```
diamond_data['clarity'].value_counts()
```

Out[31]:

```
SI1     13032
VS2     12229
SI2      9150
VS1      8156
VVS2     5056
VVS1     3647
IF       1784
I1        740
Name: clarity, dtype: int64
```

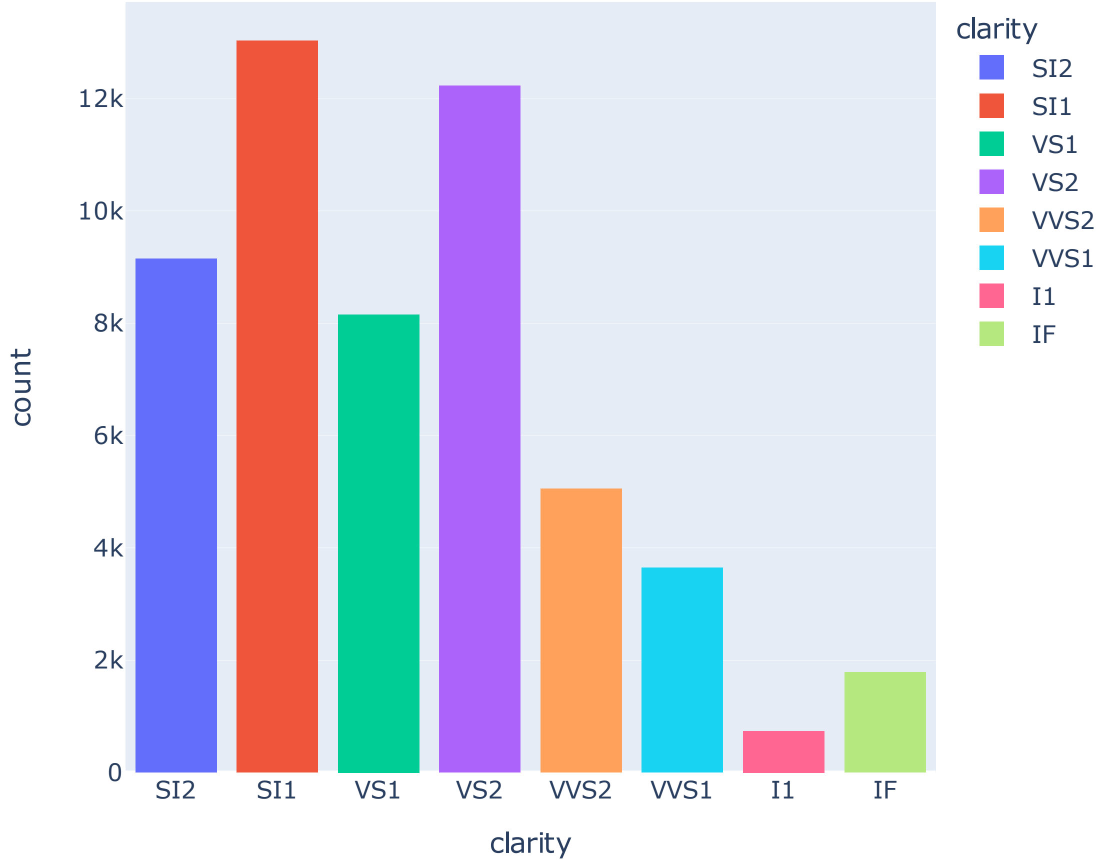
In [32]:

```
plt.figure(figsize=(15,6))
sns.countplot(diamond_data['clarity'], data = diamond_data,
                  palette='hls')
plt.xticks(rotation = 90)
plt.show()
```



In [34]:

```
fig3 = px.histogram(diamond_data, x = 'clarity', color = 'clarity')
fig3.show()
```



In [35]:

```
diamond_data = diamond_data.drop(diamond_data[diamond_data["x"]==0].index)
diamond_data = diamond_data.drop(diamond_data[diamond_data["y"]==0].index)
diamond_data = diamond_data.drop(diamond_data[diamond_data["z"]==0].index)
```

In [36]:

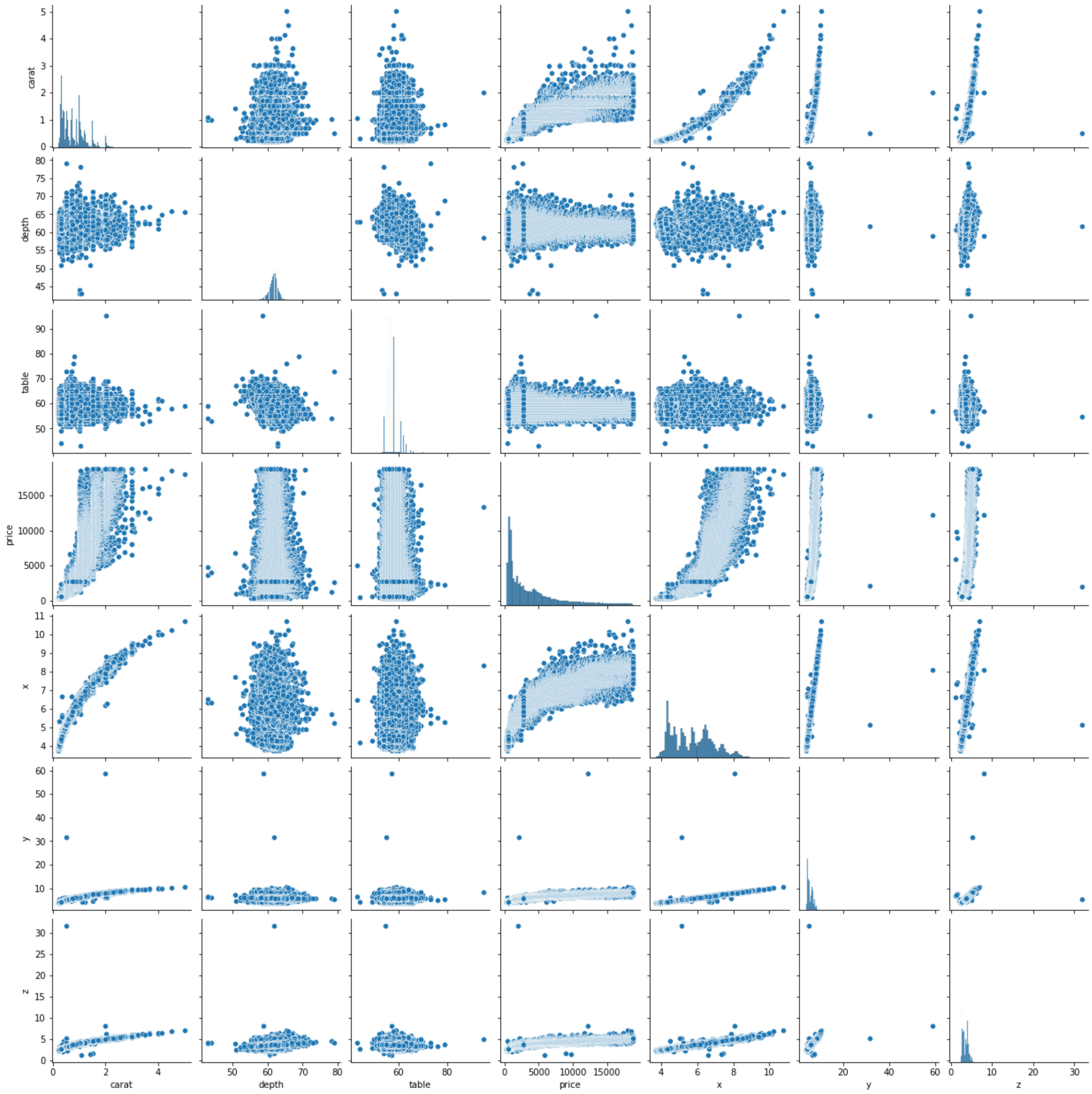
```
diamond_data.shape
```

Out[36]:

```
(53775, 10)
```

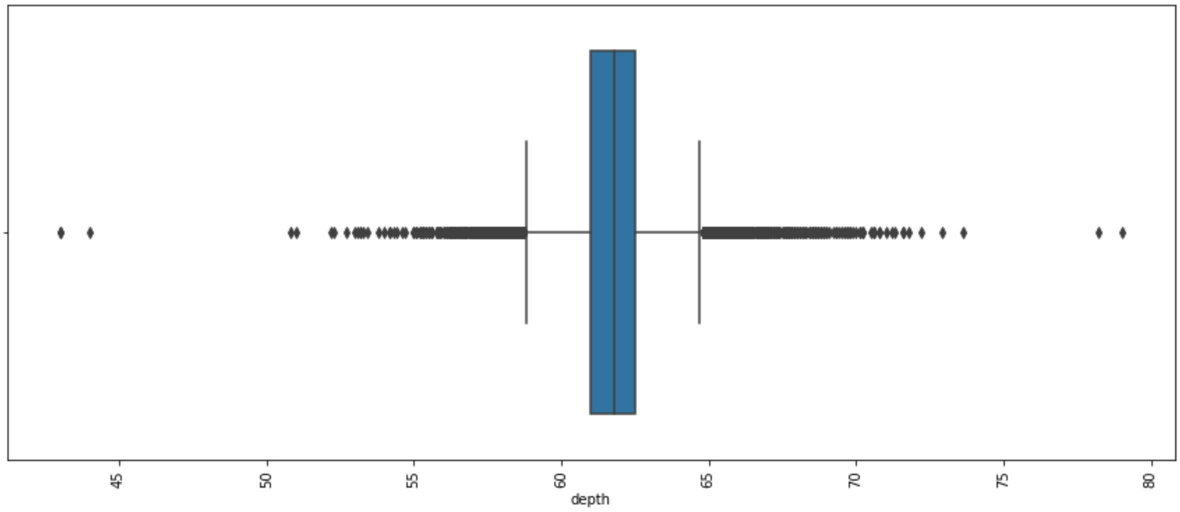
In [37]:

```
sns.pairplot(diamond_data)
plt.show()
```



In [38]:

```
plt.figure(figsize=(15,6))
sns.boxplot(diamond_data['depth'])
plt.xticks(rotation = 90)
plt.show()
```



In [39]:

```
diamond_depth = diamond_data['depth']
Q3 = diamond_depth.quantile(0.75)
Q1 = diamond_depth.quantile(0.45)
IQR = Q3-Q1
lower_limit = Q1 -(1.5*IQR)
upper_limit = Q3 +(1.5*IQR)
depth_outliers = diamond_depth[(diamond_depth <lower_limit) | (diamond_depth >upper_
depth_outliers
```

Out[39]:

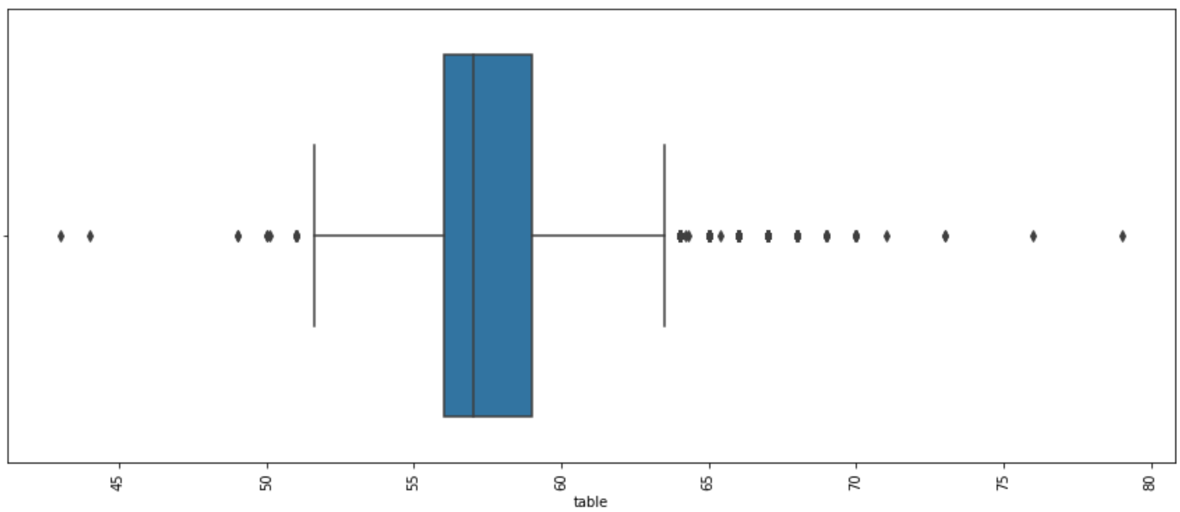
```
1        59.8
2        56.9
8        65.1
9        59.4
10       64.0
         ...
53918    59.3
53927    58.1
53930    60.5
53931    59.8
53932    60.5
Name: depth, Length: 11924, dtype: float64
```

In [40]:

```
diamond_data = diamond_data[(diamond_data["depth"]<75)&(diamond_data["depth"]>45)]
```

In [49]:

```
plt.figure(figsize=(15,6))
sns.boxplot(diamond_data['table'])
plt.xticks(rotation = 90)
plt.show()
```



In [41]:

```
diamond_table = diamond_data['table']
Q3 = diamond_table.quantile(0.80)
Q1 = diamond_table.quantile(0.40)
IQR = Q3-Q1
lower_limit = Q1 -(1.5*IQR)
upper_limit = Q3 +(1.5*IQR)
table_outliers = diamond_table[(diamond_table <lower_limit) | (diamond_table >upper_
table_outliers
```

Out[41]:

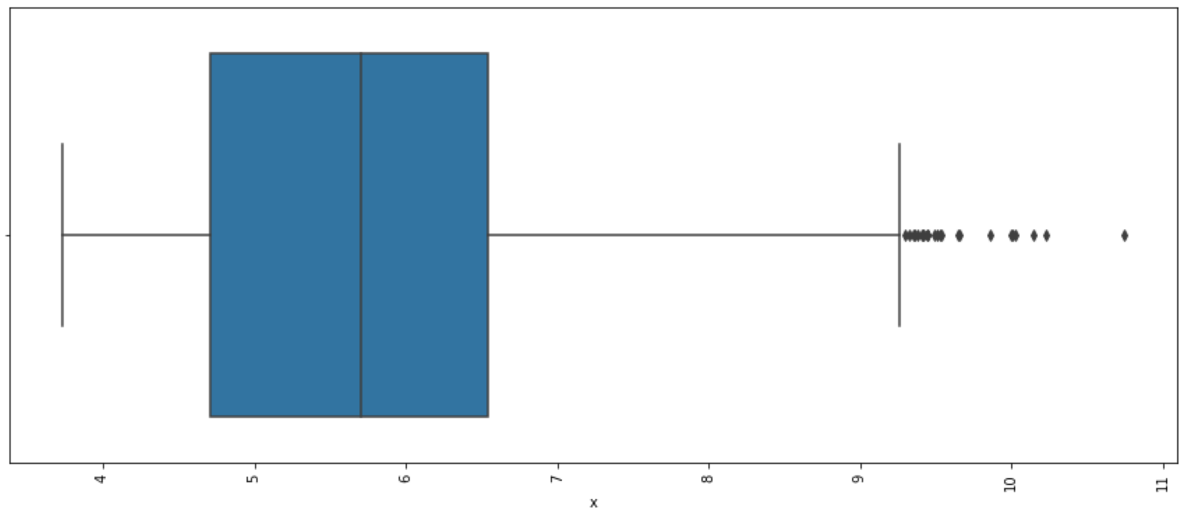
```
2        65.0
43       63.0
91       69.0
115      53.0
122      63.0
         ...
53825    63.0
53828    63.0
53840    63.0
53881    53.0
53897    63.0
Name: table, Length: 1966, dtype: float64
```

In [42]:

```
diamond_data = diamond_data[(diamond_data["table"]<80)&(diamond_data["table"]>40)]
```

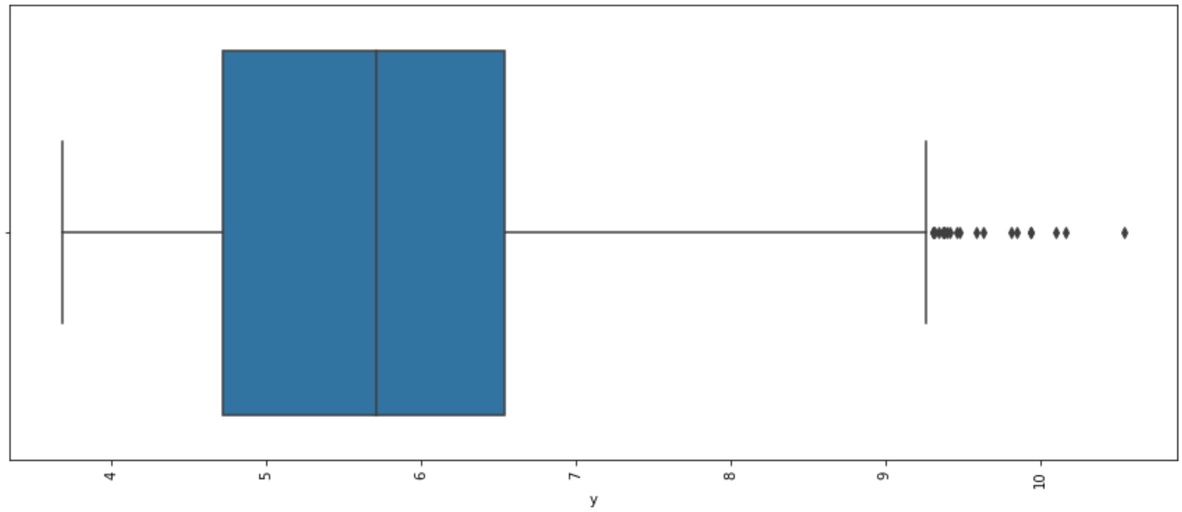
In [50]:

```
plt.figure(figsize=(15,6))
sns.boxplot(diamond_data['x'])
plt.xticks(rotation = 90)
plt.show()
```



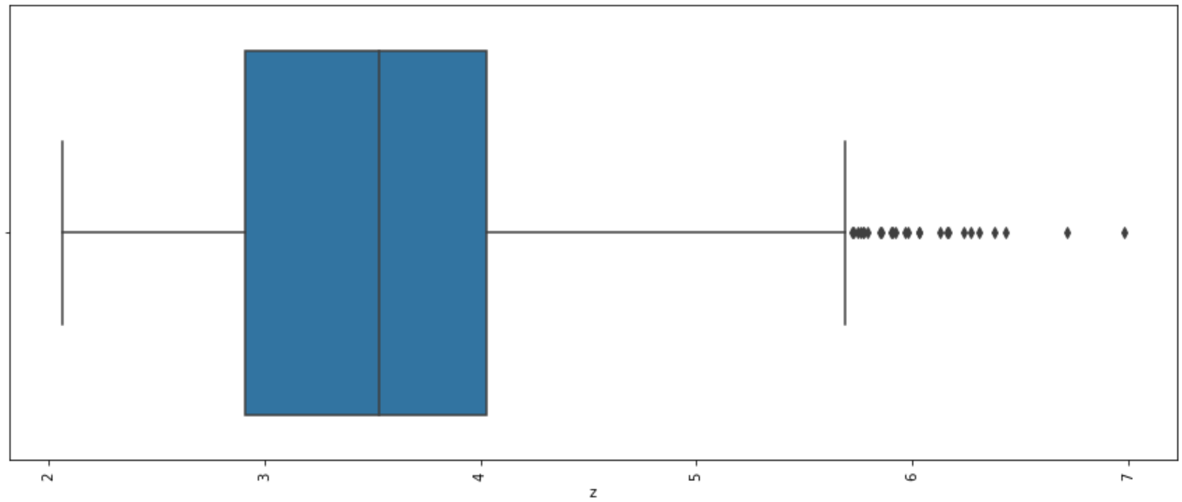
In [51]:

```
plt.figure(figsize=(15,6))
sns.boxplot(diamond_data['y'])
plt.xticks(rotation = 90)
plt.show()
```



In [52]:

```
plt.figure(figsize=(15,6))
sns.boxplot(diamond_data['z'])
plt.xticks(rotation = 90)
plt.show()
```



In [43]:

```
diamond_data = diamond_data[(diamond_data["x"]<30)]
diamond_data = diamond_data[(diamond_data["y"]<30)]
diamond_data = diamond_data[(diamond_data["z"]<30)&(diamond_data["z"]>2)]
```

In [44]:

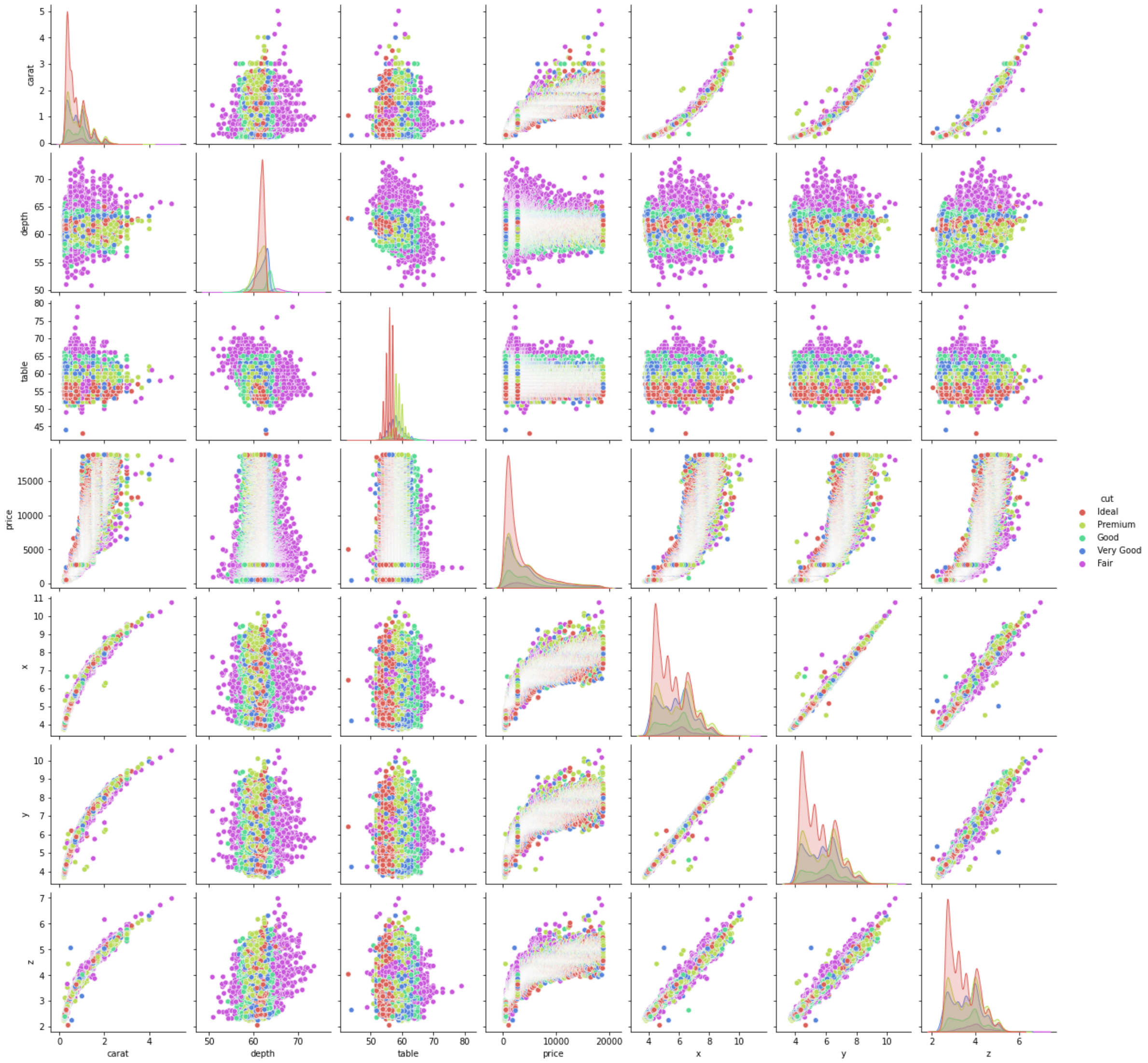
```
diamond_data.shape
```

Out[44]:

(53763, 10)

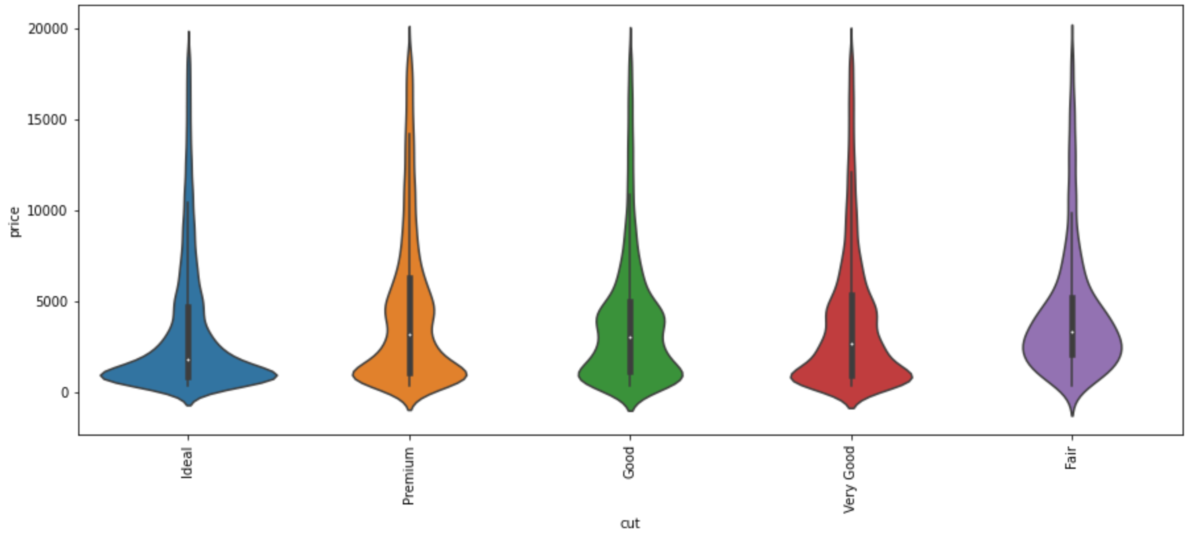
In [48]:

```
sns.pairplot(diamond_data, hue= "cut", palette = 'hls')
plt.show()
```



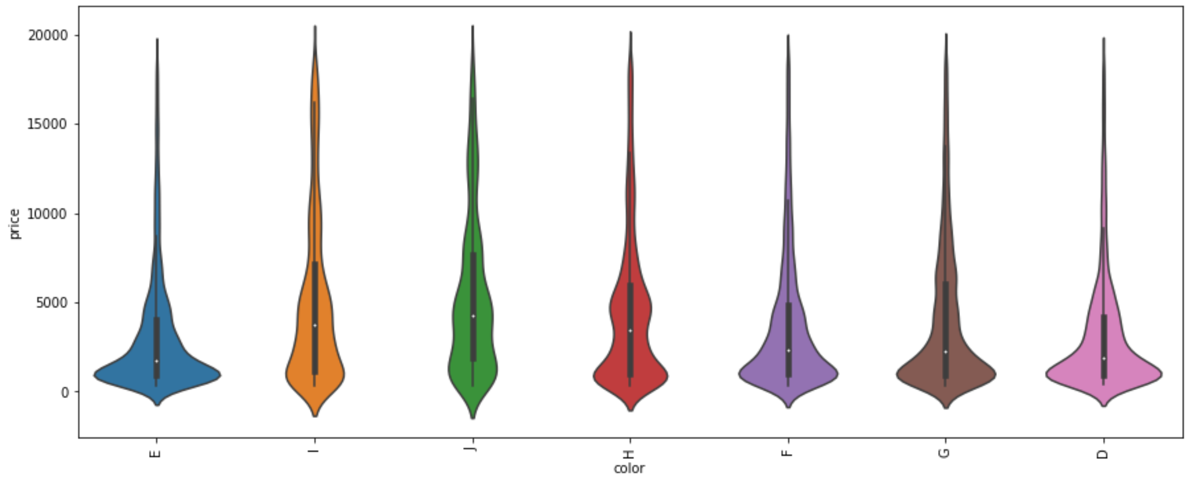
In [54]:

```
plt.figure(figsize=(15,6))
sns.violinplot(x = 'cut' , y = 'price' , data = diamond_data)
plt.xticks(rotation = 90)
plt.show()
```



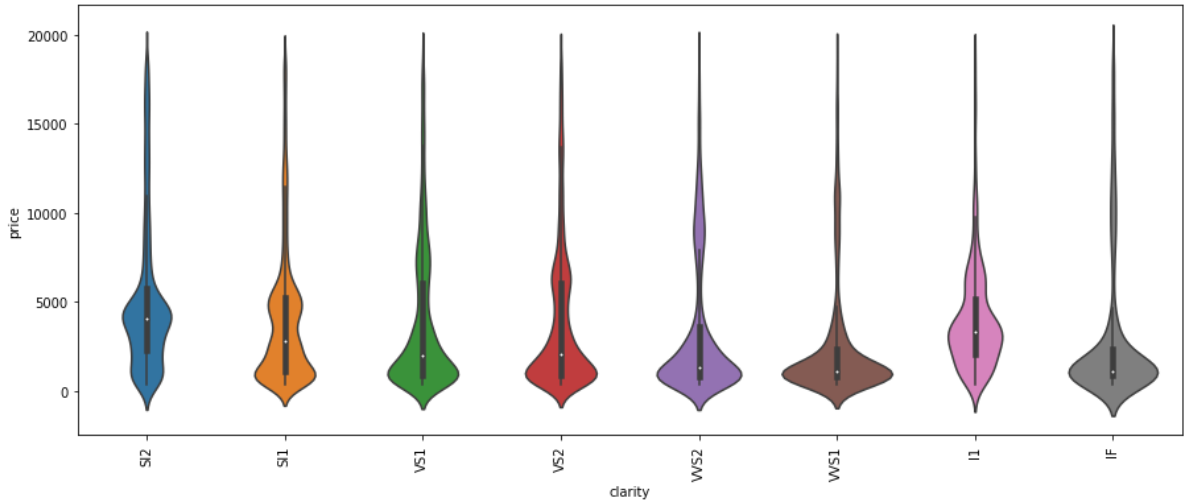
In [55]:

```
plt.figure(figsize=(15,6))
sns.violinplot(x = 'color' , y = 'price' , data = diamond_data)
plt.xticks(rotation = 90)
plt.show()
```



In [56]:

```
plt.figure(figsize=(15,6))
sns.violinplot(x = 'clarity' , y = 'price' , data = diamond_data)
plt.xticks(rotation = 90)
plt.show()
```



In [57]:

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
```

In [59]:

```
s = (diamond_data.dtypes =="object")
object_cols = list(s[s].index)
```

In [60]:

```
label_data = diamond_data.copy()
label_encoder = LabelEncoder()
for col in object_cols:
    label_data[col] = label_encoder.fit_transform(label_data[col])
```

In [61]:

```
label_data.head()
```

Out[61]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	2	1	3	61.5	55.0	326	3.95	3.98	2.43
1	0.21	3	1	2	59.8	61.0	326	3.89	3.84	2.31
2	0.23	1	1	4	56.9	65.0	327	4.05	4.07	2.31
3	0.29	3	5	5	62.4	58.0	334	4.20	4.23	2.63
4	0.31	1	6	3	63.3	58.0	335	4.34	4.35	2.75

In [62]:

```
diamond_data.describe()
```

Out[62]:

	carat	depth	table	price	x	y
count	53763.000000	53763.000000	53763.000000	53763.000000	53763.000000	53763.000000
mean	0.797460	61.748781	57.457207	3930.785336	5.731405	5.733299
std	0.473136	1.419309	2.226311	3985.807738	1.118563	1.110473
min	0.200000	50.800000	43.000000	326.000000	3.730000	3.680000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000
75%	1.040000	62.500000	59.000000	5324.000000	6.540000	6.540000
max	5.010000	73.600000	79.000000	18823.000000	10.740000	10.540000

In [63]:

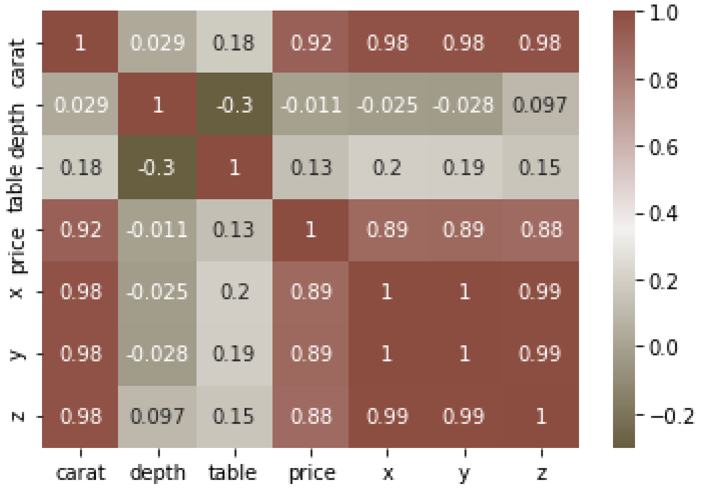
```
diamond_data.corr()
```

Out[63]:

	carat	depth	table	price	x	y	z
carat	1.000000	0.028718	0.181098	0.921546	0.977855	0.976940	0.977069
depth	0.028718	1.000000	-0.301988	-0.010766	-0.024679	-0.027810	0.097230
table	0.181098	-0.301988	1.000000	0.126425	0.195530	0.189325	0.154900
price	0.921546	-0.010766	0.126425	1.000000	0.887118	0.888708	0.882504
x	0.977855	-0.024679	0.195530	0.887118	1.000000	0.998654	0.991666
y	0.976940	-0.027810	0.189325	0.888708	0.998654	1.000000	0.991330
z	0.977069	0.097230	0.154900	0.882504	0.991666	0.991330	1.000000

In [65]:

```
cmap = sns.diverging_palette(70,20,s=50, l=40, n=6,as_cmap=True)
sns.heatmap(diamond_data.corr(), cmap=cmap,annot=True, )
plt.show()
```



In [66]:

```
X= label_data.drop(["price"],axis =1)
y= label_data["price"]
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=7)
```

In [67]:

```
pipeline_lr=Pipeline([("scalar1",StandardScaler()),
                     ("lr_classifier",LinearRegression())])

pipeline_dt=Pipeline([("scalar2",StandardScaler()),
                     ("dt_classifier",DecisionTreeRegressor())])

pipeline_rf=Pipeline([("scalar3",StandardScaler()),
                     ("rf_classifier",RandomForestRegressor())])

pipeline_kn=Pipeline([("scalar4",StandardScaler()),
                     ("rf_classifier",KNeighborsRegressor())])

pipeline_xgb=Pipeline([("scalar5",StandardScaler()),
                     ("rf_classifier",XGBRegressor())])

pipelines = [pipeline_lr, pipeline_dt, pipeline_rf, pipeline_kn,
             pipeline_xgb]
pipe_dict = {0: "LinearRegression", 1: "DecisionTree", 2: "RandomForest",3: "KNeigh

for pipe in pipelines:
    pipe.fit(X_train, y_train)
```

In [68]:

```
cv_results_rms = []
for i, model in enumerate(pipelines):
    cv_score = cross_val_score(model, X_train,y_train,scoring="neg_root_mean_square
    cv_results_rms.append(cv_score)
    print("%s: %f " % (pipe_dict[i], cv_score.mean()))
```

```
LinearRegression: -1333.321776
DecisionTree: -754.017952
RandomForest: -551.247491
KNeighbors: -828.946820
XGBRegressor: -551.226106
```

In [71]:

```
pred = pipeline_xgb.predict(X_test)
```

In [70]:

```
print("Training Accuracy :", pipeline_xgb.score(X_train, y_train))
print("Testing Accuracy :", pipeline_xgb.score(X_test, y_test))
```

```
Training Accuracy : 0.9911507531791393
Testing Accuracy : 0.9812464134865915
```

In [72]:

```
print("R^2:",metrics.r2_score(y_test, pred))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, pred))*(len(y_test)-1)/(len(y
print("MAE:",metrics.mean_absolute_error(y_test, pred))
print("MSE:",metrics.mean_squared_error(y_test, pred))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
R^2: 0.9812464134865915
Adjusted R^2: 0.9812338468661894
MAE: 281.2550541364262
MSE: 294422.2563174091
RMSE: 542.6069077310103
```