In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
df= pd.read_csv("C:\\Users\\DELL\\Downloads\\archive (1)\\diamonds.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

In [4]:

```
df.tail()
```

Out[4]:

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

In [5]:

```
df.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 [6]:
```
df.describe()
```

Out[6]:

	carat	depth	table	price	x	y	z
count	53940.000000	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	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

In [7]:
```
df.shape
```

Out[7]:
```
(53940, 10)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

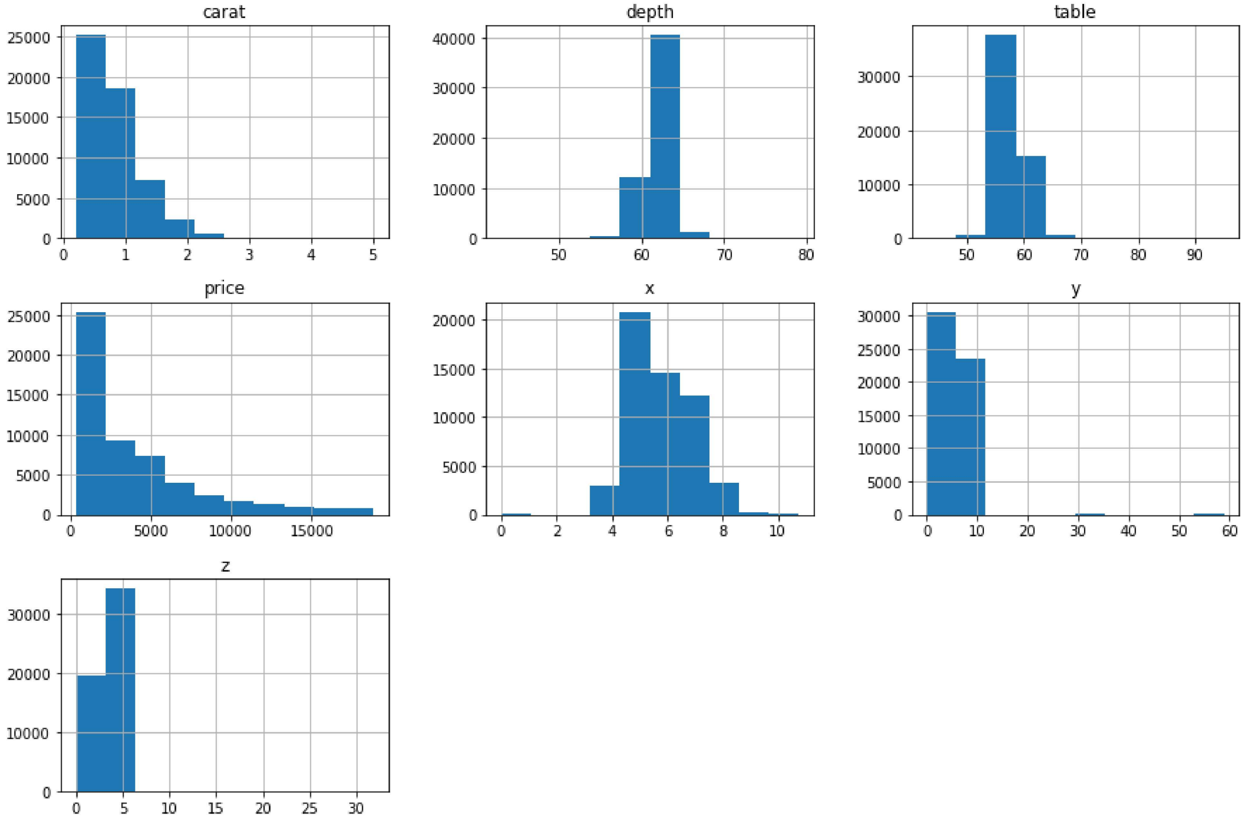
EDA

In [9]:
```
num_features = df.select_dtypes(include =['int64','float64'])
print(num_features.columns)
```

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

In [10]:
```
num_features.hist(figsize =(15,10))
```

Out[10]:
```
array([[<AxesSubplot:title={'center':'carat'}>,
        <AxesSubplot:title={'center':'depth'}>,
        <AxesSubplot:title={'center':'table'}>],
       [<AxesSubplot:title={'center':'price'}>,
        <AxesSubplot:title={'center':'x'}>,
        <AxesSubplot:title={'center':'y'}>],
       [<AxesSubplot:title={'center':'z'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



In [11]:
```
cat_features = df.select_dtypes(include=['object'])
print(cat_features.columns)
```

```
Index(['cut', 'color', 'clarity'], dtype='object')
```

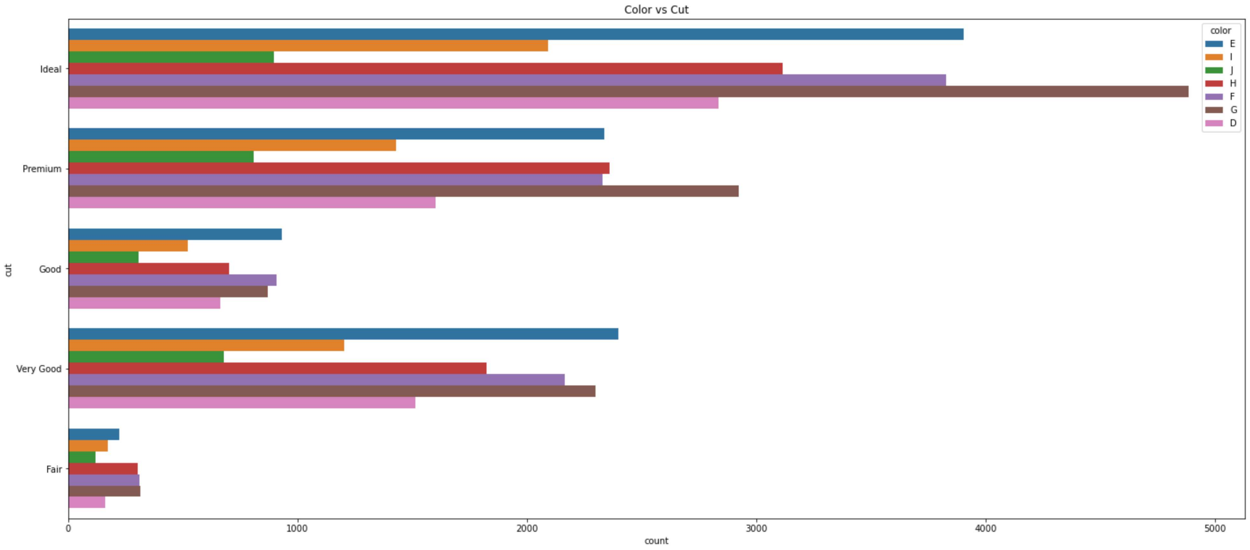
In [12]:

```
plt.figure(figsize=(24, 48))

plt.subplot(411)
sns.countplot(y='cut', hue='color', data = cat_features)
plt.title('Color vs Cut')
```

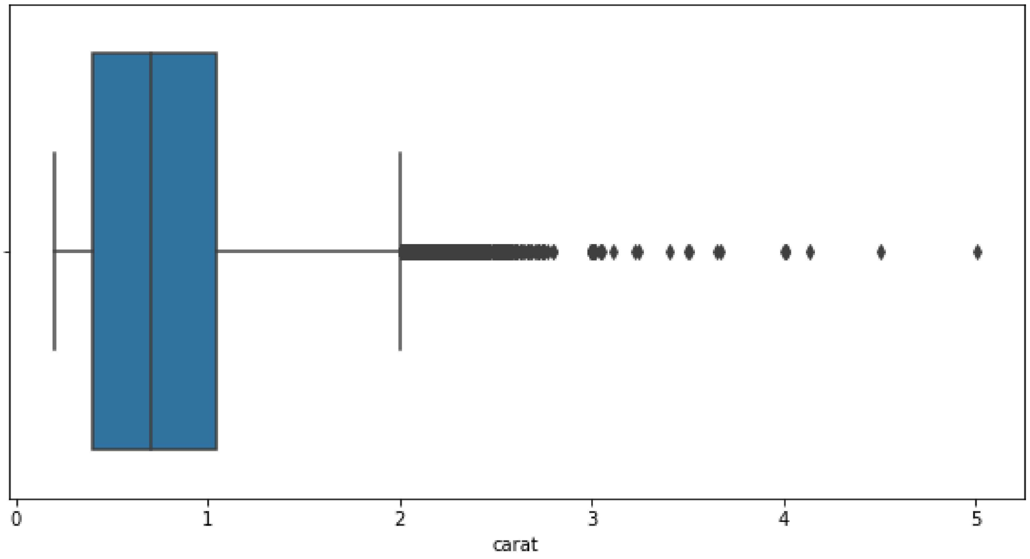
Out[12]:

```
Text(0.5, 1.0, 'Color vs Cut')
```



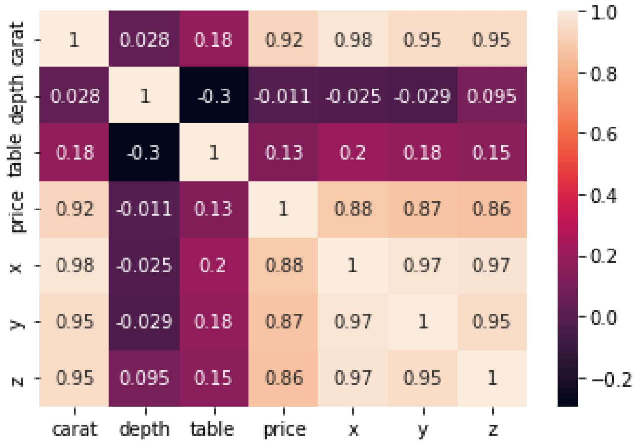
In [13]:

```
for num_var in num_features:
    plt.figure(figsize=(10,5))
    sns.boxplot(x=df[num_var])
```

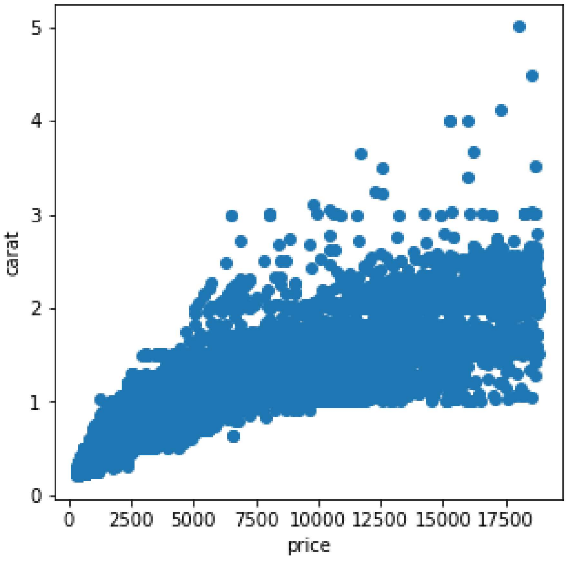


In [14]:
```
corr=df.corr()
sns.heatmap(corr,annot=True)
```

Out[14]:
```
<AxesSubplot:>
```



In [15]:
```
for num_var in num_features:
    plt.figure(figsize=(5,5))
    plt.xlabel('price')
    plt.ylabel(num_var)
    x=df['price']
    y=df[num_var]
    plt.scatter(x,y)
```



Feature Transformation

In [16]:
```
df_cat = pd.get_dummies(cat_features,drop_first=True)
df_cat.head()
```

Out[16]:

	cut_Good	cut_Ideal	cut_Premium	cut_Very Good	color_E	color_F	color_G	color_H	color_I	color_J	clarity_IF
0	0	1	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	1	0	0
4	1	0	0	0	0	0	0	0	0	1	0

In [17]:
```
new_df = pd.concat([num_features, df_cat], axis=1)
new_df.head()
```

Out[17]:

	carat	depth	table	price	x	y	z	cut_Good	cut_Ideal	cut_Premium	...	color_H	color_I	color_J
0	0.23	61.5	55.0	326	3.95	3.98	2.43	0	1	0	...	0	0	0
1	0.21	59.8	61.0	326	3.89	3.84	2.31	0	0	1	...	0	0	0
2	0.23	56.9	65.0	327	4.05	4.07	2.31	1	0	0	...	0	0	0
3	0.29	62.4	58.0	334	4.20	4.23	2.63	0	0	1	...	0	1	0
4	0.31	63.3	58.0	335	4.34	4.35	2.75	1	0	0	...	0	0	1

5 rows × 24 columns

In [18]:
```
X = new_df.drop(columns = ['price'],axis = 1)
X.head()
```

Out[18]:

	carat	depth	table	x	y	z	cut_Good	cut_Ideal	cut_Premium	cut_Very Good	...	color_H	color_I	colo
0	0.23	61.5	55.0	3.95	3.98	2.43	0	1	0	0	...	0	0	
1	0.21	59.8	61.0	3.89	3.84	2.31	0	0	1	0	...	0	0	
2	0.23	56.9	65.0	4.05	4.07	2.31	1	0	0	0	...	0	0	
3	0.29	62.4	58.0	4.20	4.23	2.63	0	0	1	0	...	0	1	
4	0.31	63.3	58.0	4.34	4.35	2.75	1	0	0	0	...	0	0	

5 rows × 23 columns

In [19]:
```
y =new_df['price']
y.head()
```

Out[19]:
```
0    326
1    326
2    327
3    334
4    335
Name: price, dtype: int64
```

Splitting

In [21]:

```
# split into train and test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=100
```

In [22]:

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

```
(37758, 23) (37758,)
(16182, 23) (16182,)
```

In [23]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [24]:

```
scaler.fit_transform(X_train)
scaler.transform(X_test)
```

Out[24]:

```
array([[0.07692308, 0.51388889, 0.26923077, ..., 1.        , 0.        ,
        0.        ],
       [0.1995842 , 0.51388889, 0.23076923, ..., 0.        , 0.        ,
        0.        ],
       [0.06444906, 0.56111111, 0.28846154, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.08523909, 0.50833333, 0.23076923, ..., 0.        , 0.        ,
        0.        ],
       [0.04365904, 0.525     , 0.25      , ..., 0.        , 0.        ,
        0.        ],
       [0.16216216, 0.51666667, 0.44230769, ..., 0.        , 0.        ,
        0.        ]])
```

Training the Model

In [25]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [26]:

```
model_rf = RandomForestRegressor()
```

In [27]:

```
model_rf.fit(X_train,y_train)
```

Out[27]:

```
RandomForestRegressor()
```

In [28]:

```
y_pred_rf = model_rf.predict(X_test)
```

In [30]:

```
from sklearn import metrics
```

In [31]:

```
metrics.r2_score(y_test,y_pred_rf)
```

Out[31]:

```
0.9738458454322558
```

In [ ]: