In [3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```python
data=pd.read_csv("C:\\Users\\priya\\Downloads\\diamonds.csv.zip")
```

In [8]:
```python
data.head()
```

Out[8]:

	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 [9]:
```python
data.shape
```

Out[9]:
```
(53940, 11)
```

In [10]:
```python
data.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
```

In [12]:
```python
data=data.drop(["Unnamed: 0"],axis=1)
data.describe()
```

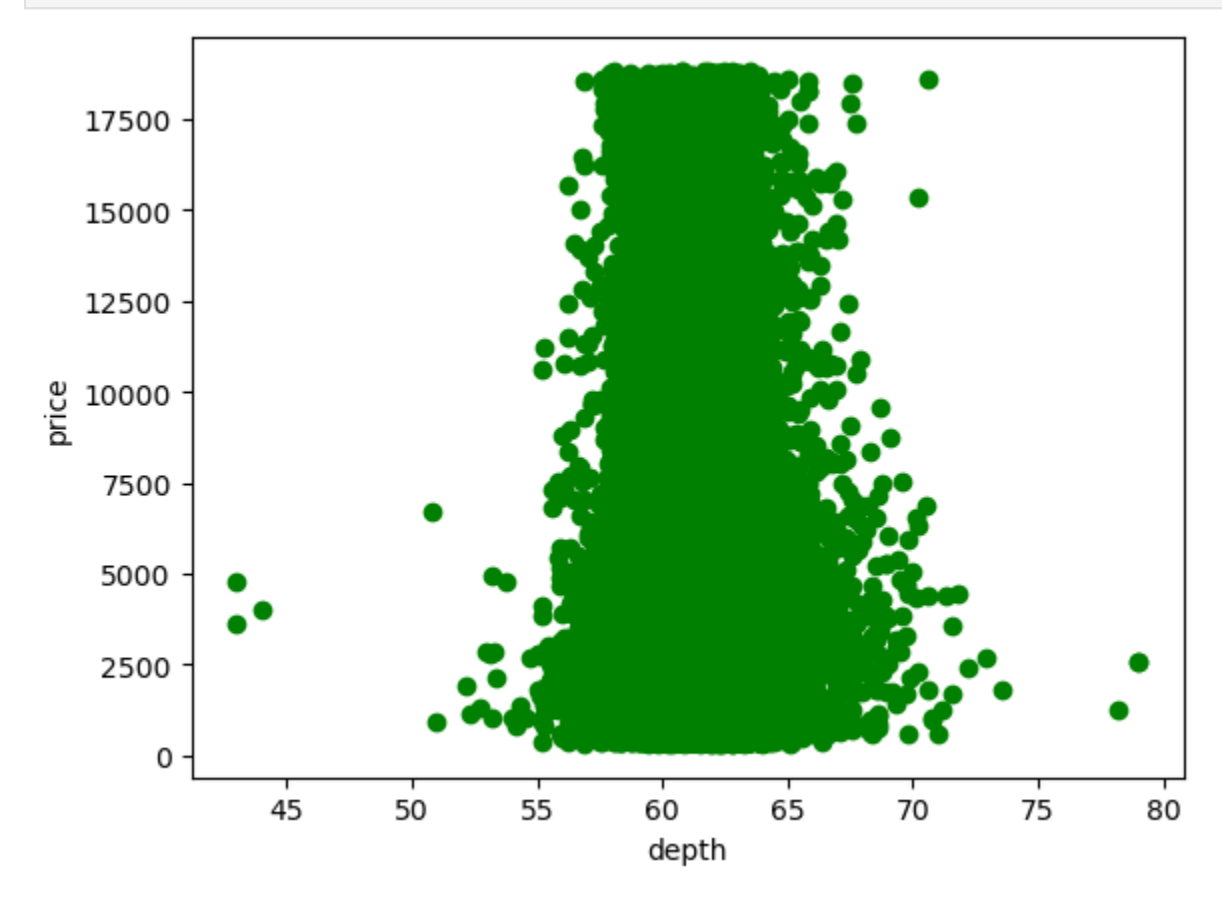
Out[12]:

	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

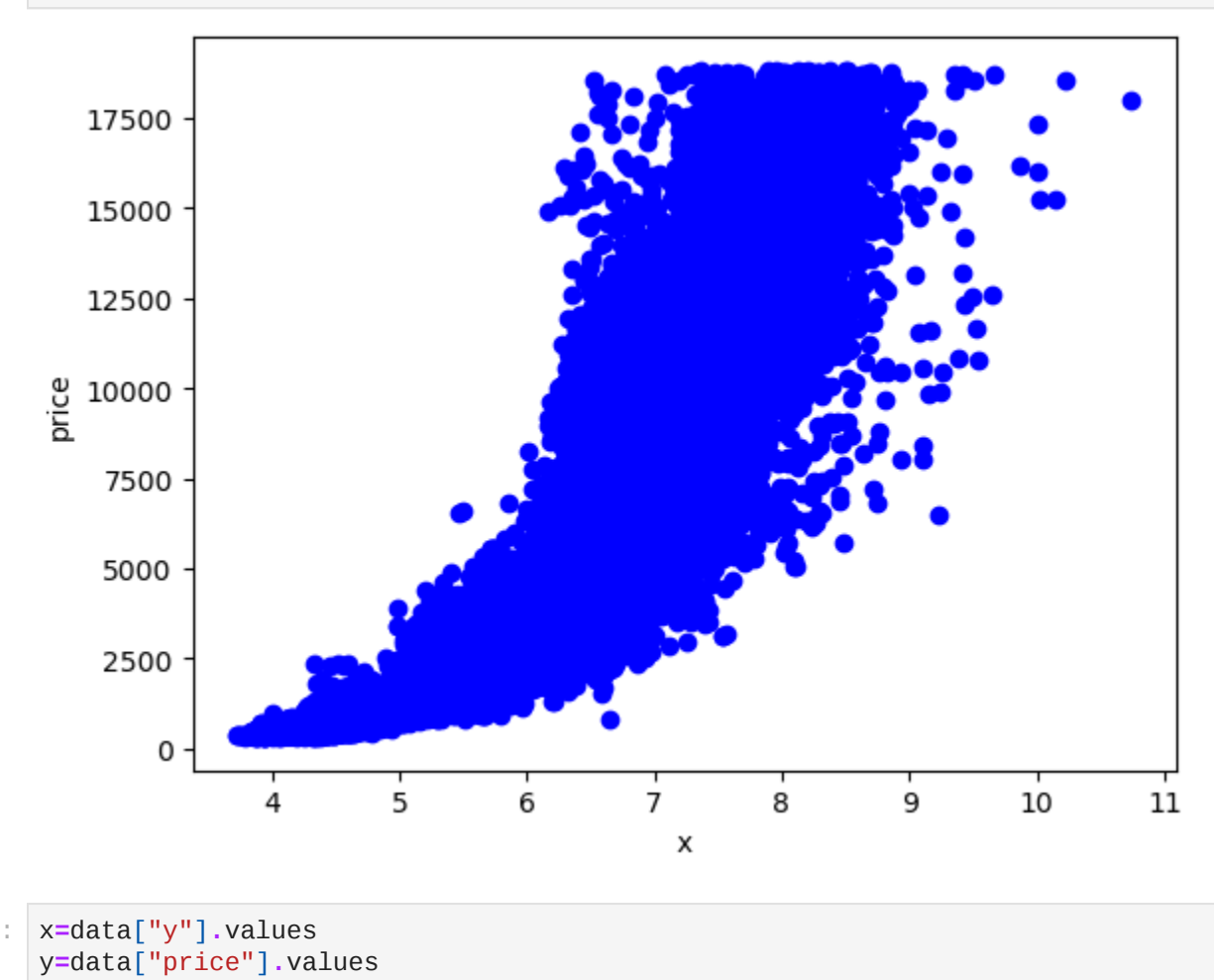
In [14]:
```python
#dropping dimentionless diamonds
data=data.drop(data[data["x"]==0].index)
data=data.drop(data[data["y"]==0].index)
data=data.drop(data[data["z"]==0].index)
data.shape
```

Out[14]:
```
(53920, 10)
```

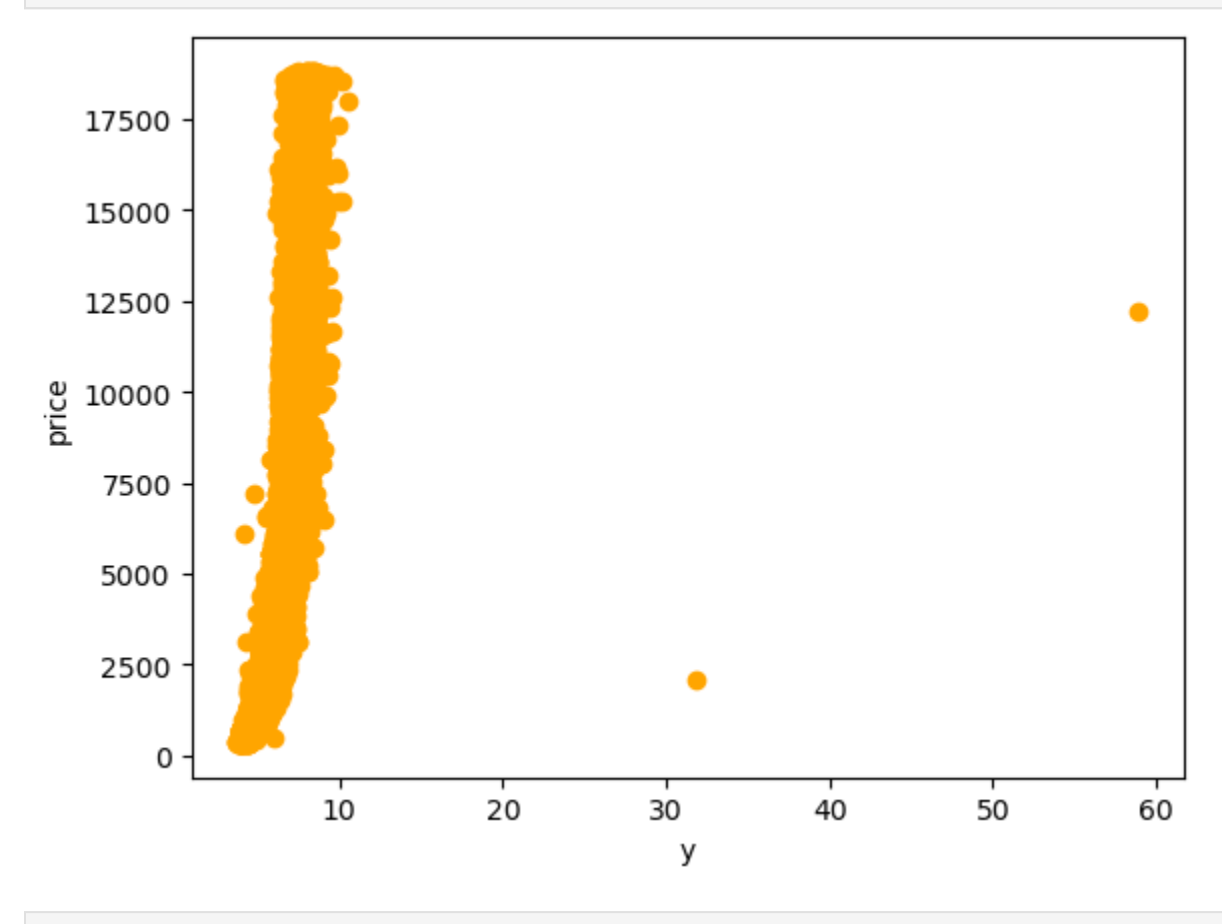
In [15]:
```python
#visualization
import matplotlib.pyplot as plt
x=data["depth"].values
y=data["price"].values
plt.scatter(x,y,color='green')
plt.xlabel("depth")
plt.ylabel("price")
plt.show()
```



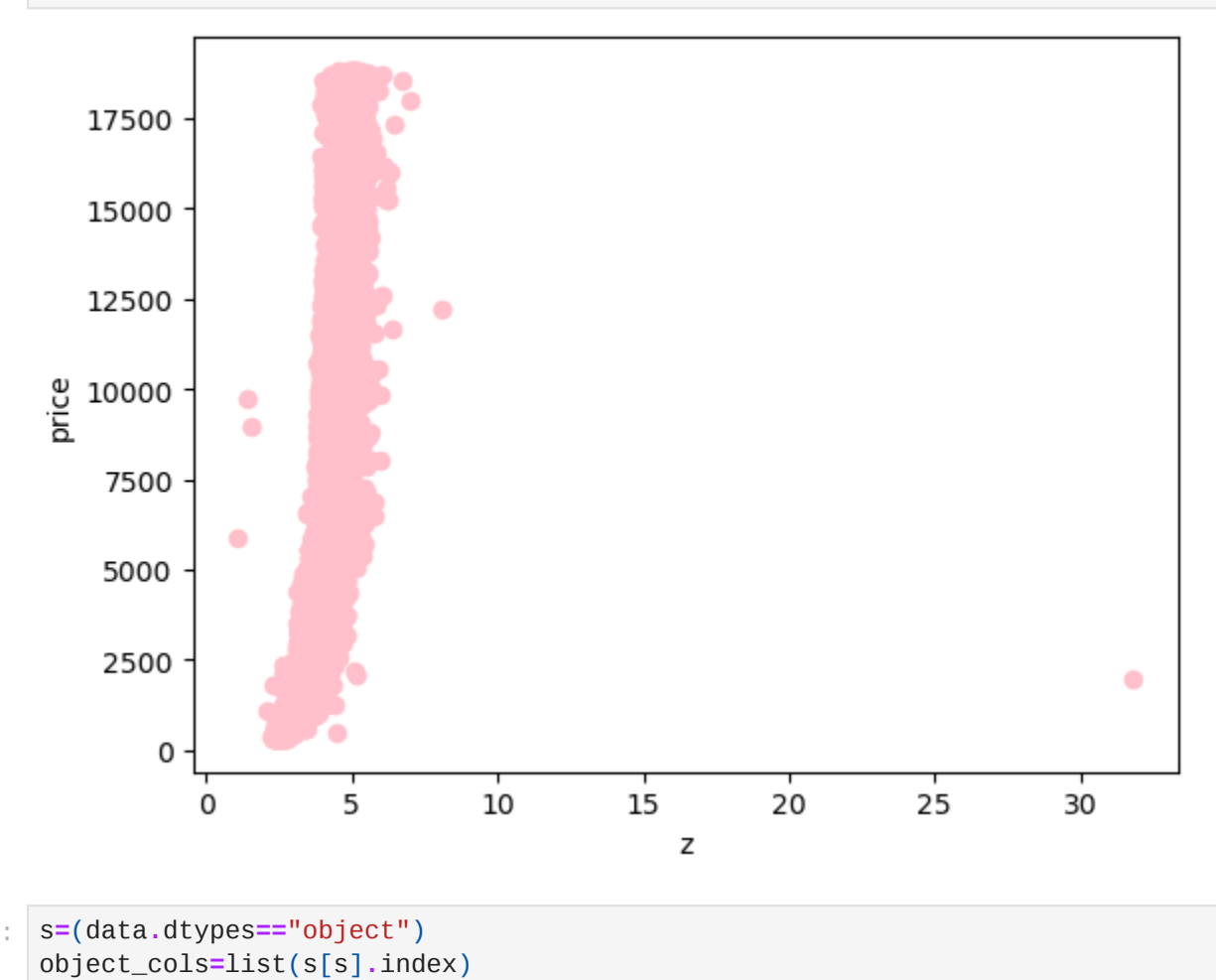
In [17]:
```python
x=data["x"].values
y=data["price"].values
plt.scatter(x,y,color='blue')
plt.xlabel("x")
plt.ylabel("price")
plt.show()
```



In [19]:
```python
x=data["y"].values
y=data["price"].values
plt.scatter(x,y,color='orange')
plt.xlabel("y")
plt.ylabel("price")
plt.show()
```



In [20]:
```python
x=data["z"].values
y=data["price"].values
plt.scatter(x,y,color='pink')
plt.xlabel("z")
plt.ylabel("price")
plt.show()
```



In [21]:
```python
s=(data.dtypes=="object")
object_cols=list(s[s].index)
print("Categorical variables:")
print(object_cols)
```

```
Categorical variables:
['cut', 'color', 'clarity']
```

In [22]:
```python
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
```

In [25]:
```python
label_data = data.copy()
#encoding
label_encoder=LabelEncoder()
for col in object_cols:
    label_data[col] = label_encoder.fit_transform(label_data[col])
label_data.head()
```

Out[25]:

	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

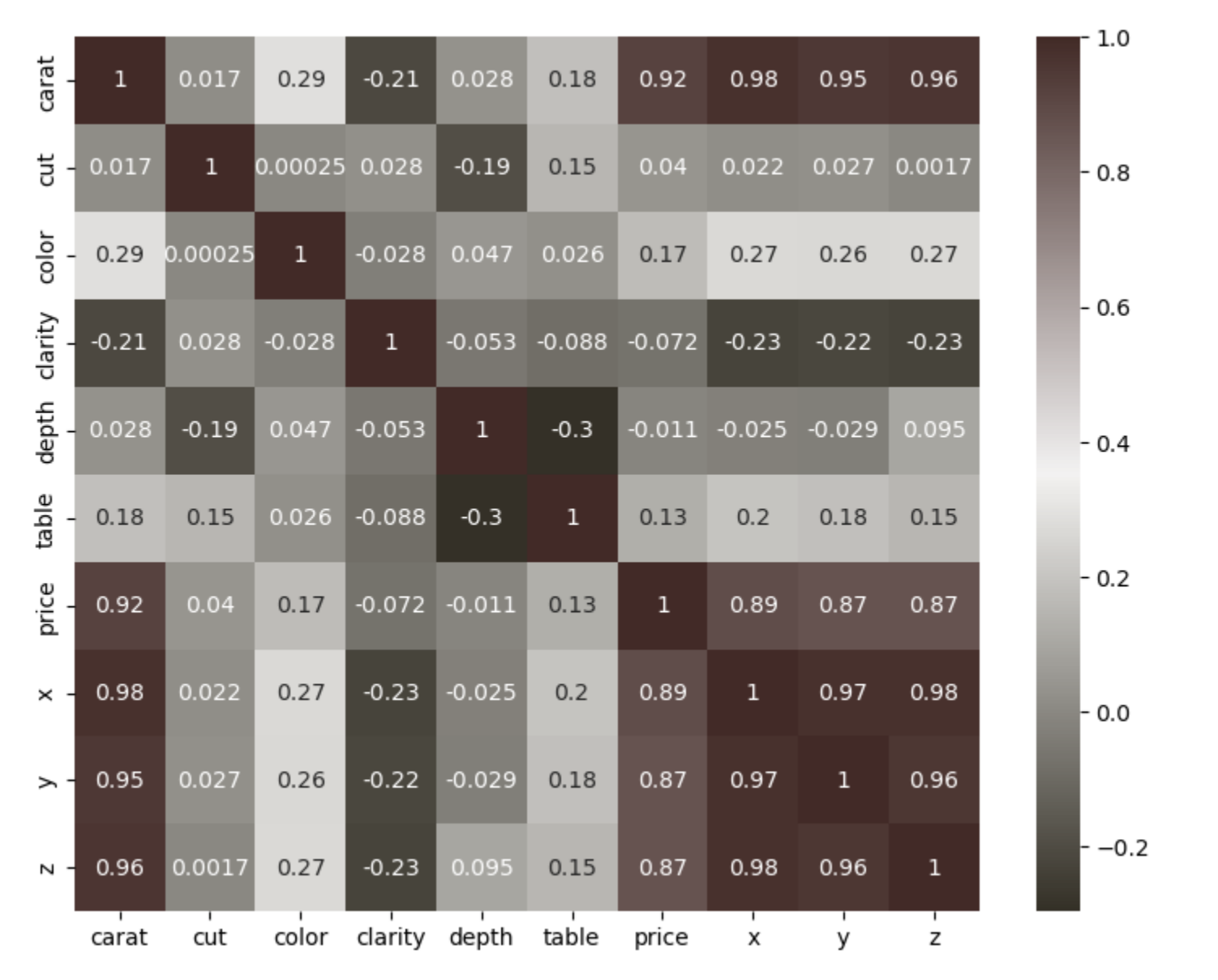
In [26]:
```python
data.describe()
```

Out[26]:

	carat	depth	table	price	x	y	z
count	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000
mean	0.797698	61.749514	57.456834	3930.993231	5.731627	5.734887	3.540046
std	0.473795	1.432331	2.234064	3987.280446	1.119423	1.140126	0.702530
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.680000	1.070000
25%	0.400000	61.000000	56.000000	949.000000	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	5323.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

In [31]:
```python
import seaborn as sns
cmap = sns.diverging_palette(70,20,s=30, l=20, n=6,as_cmap=True)
corrmat= label_data.corr()
f, ax = plt.subplots(figsize=(9,7))
sns.heatmap(corrmat,cmap=cmap,annot=True, )
```

Out[31]:
```
<Axes: >
```



In [33]:
```python
#defining x,y
from sklearn.model_selection import train_test_split
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=0.30,random_state=5)
```

In [34]:
```python
print(x_train)
```

```
       carat  cut  color  clarity  depth  table     x     y     z
35298   0.30    4      5        4   60.8   59.0  4.30  4.32  2.62
12842   1.06    2      2        3   61.8   56.0  6.56  6.51  4.04
47470   0.76    2      6        2   61.9   53.0  5.85  5.92  3.64
25691   0.40    4      1        3   62.1   60.0  4.65  4.69  2.90
17214   1.01    3      3        4   60.9   58.0  6.47  6.43  3.93
...      ...  ...    ...      ...    ...    ...   ...   ...   ...
5524    1.01    1      3        3   63.7   58.0  6.31  6.35  4.03
35831   0.50    2      6        5   62.5   53.0  5.09  5.12  3.19
20472   1.59    2      5        3   60.5   55.1  7.55  7.62  4.59
18647   1.03    2      2        4   59.0   55.0  6.62  6.67  3.92
35700   0.30    3      0        5   62.5   59.0  4.28  4.23  2.66

[37744 rows x 9 columns]
```

In [35]:
```python
print(y_train)
```

```
35298     473
12842    5354
47470    1859
25691     644
17214    6882
         ...
5524     3852
35831     915
20472    8814
18647    7613
35700     911
Name: price, Length: 37744, dtype: int64
```

In [38]:
```python
from sklearn.linear_model import LinearRegression
mymodel = LinearRegression()
```

In [39]:
```python
mymodel.fit(x_train,y_train)
```

Out[39]:
```
▾ LinearRegression
LinearRegression()
```

In [40]:

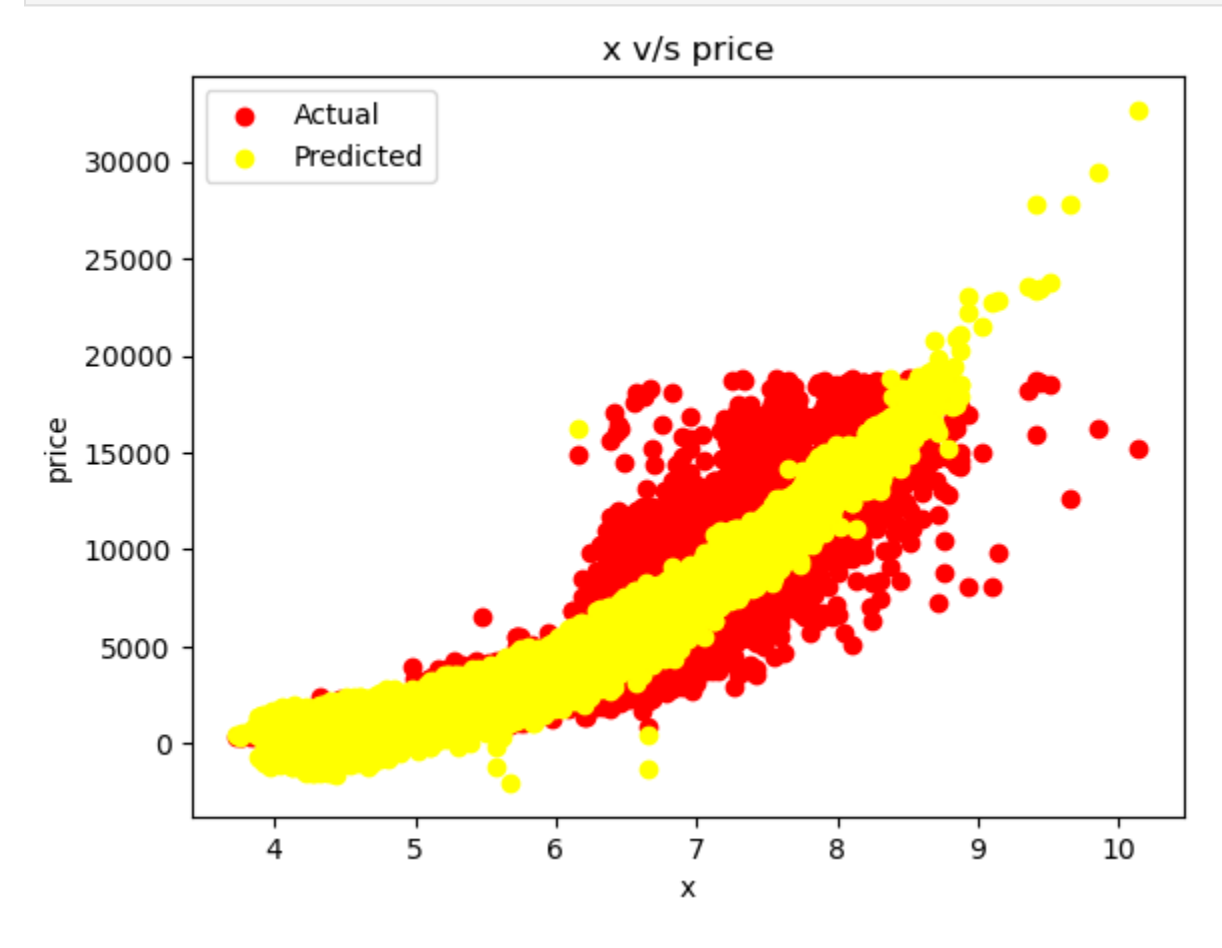
In [41]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
import numpy as np
```

In [46]:
```python
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_test)-x_test.shape[1]-1))
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: 0.8874584893915494
Adjusted R^2: 0.8873958348328783
MAE: 852.9738753256918
MSE: 1775827.4344662614
RMSE: 1332.6017538883332
```

In [81]:
```python
pred = mymodel.predict(x_test)
```

In [82]:
```python
import matplotlib.pyplot as plt
plt.scatter(x_test['x'],y_test,color="red")
plt.scatter(x_test['x'],pred,color="yellow")
plt.legend(["Actual","Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x v/s price')
plt.show()
```



In [61]:
```python
from sklearn.tree import DecisionTreeRegressor
decisiontreemodel=DecisionTreeRegressor()
decisiontreemodel.fit(x_train,y_train)
```

Out[61]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [66]:
```python
treeoutput=decisiontreemodel.predict(x_test)
print(treeoutput[:5])
```

```
[  829. 10546.  3587.  2463.  1586.]
```

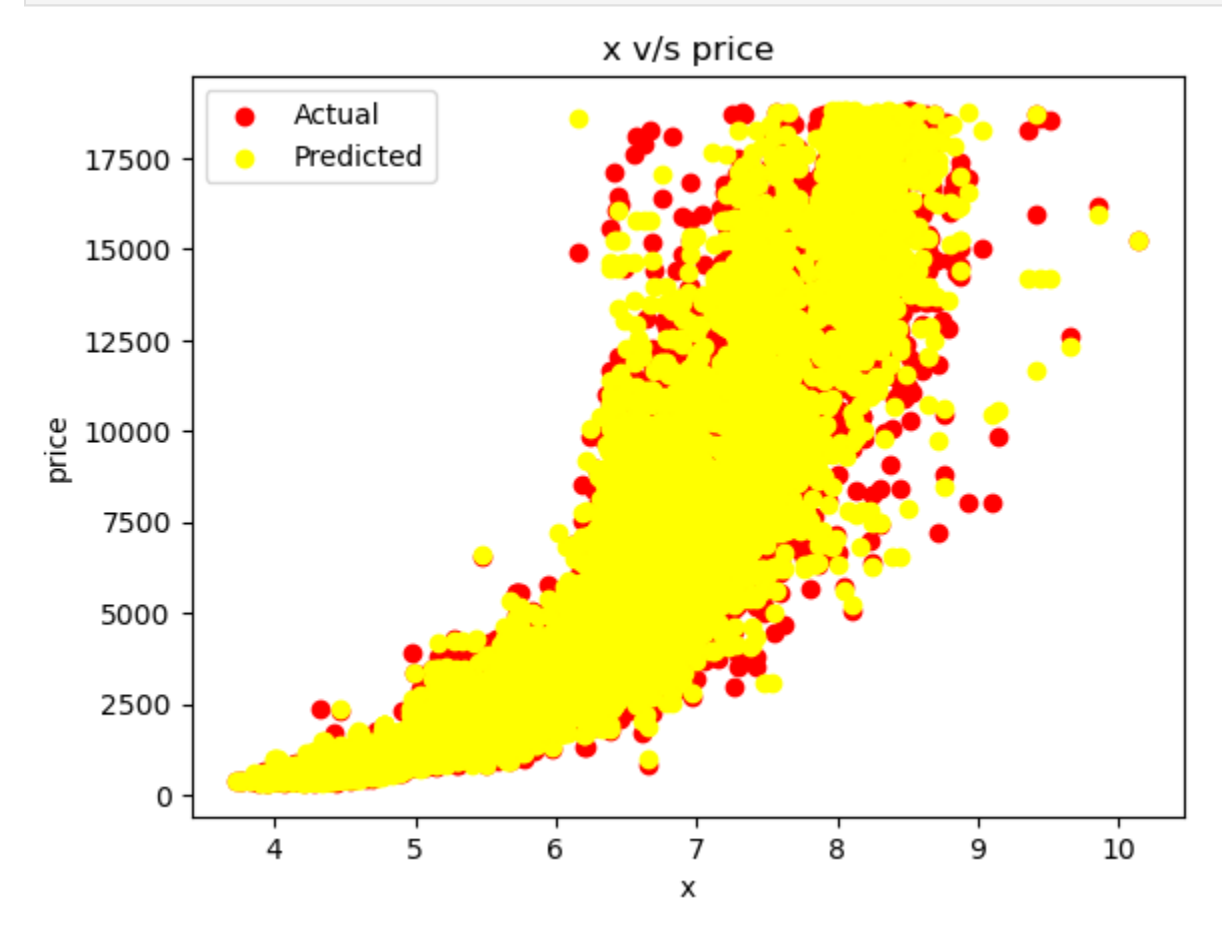
In [67]:
```python
print(y_test[:5])
```

```
33535      829
22491    10521
460       2817
51847     2423
46426     1769
Name: price, dtype: int64
```

In [69]:
```python
print("R^2:",metrics.r2_score(y_test, treeoutput))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, treeoutput))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
print("MAE:",metrics.mean_absolute_error(y_test, treeoutput))
print("MSE:",metrics.mean_squared_error(y_test, treeoutput))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, treeoutput)))
```

```
R^2: 0.9648325934500817
Adjusted R^2: 0.9648130149112378
MAE: 362.2040677546983
MSE: 554917.4256923838
RMSE: 744.9277989794607
```

In [71]:
```python
import matplotlib.pyplot as plt
plt.scatter(x_test['x'],y_test,color='red')
plt.scatter(x_test['x'],treeoutput,color="yellow")
plt.legend(["Actual", "Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x v/s price')
plt.show()
```



In [72]:
```python
from sklearn.ensemble import RandomForestRegressor
```

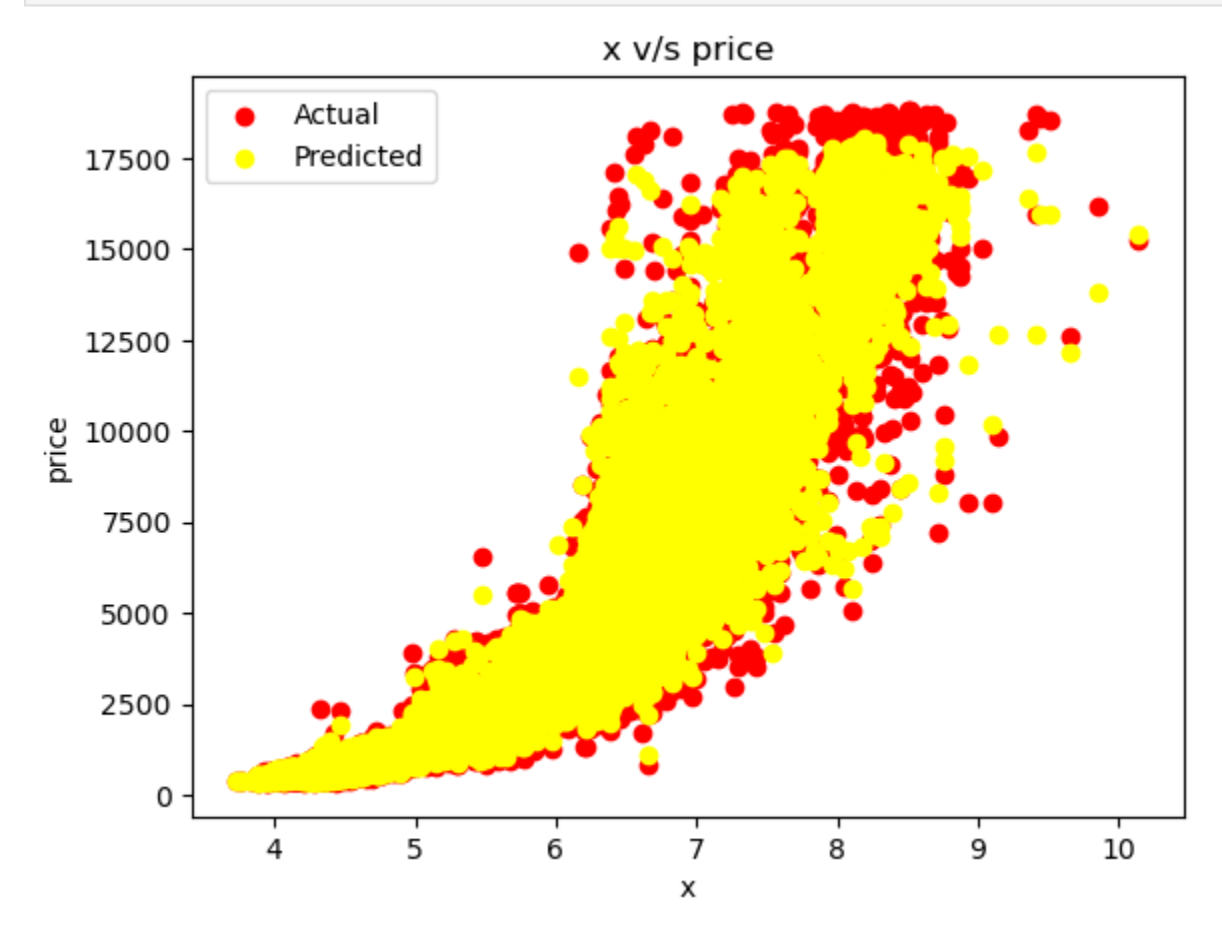
In [74]:
```python
forestmodel=RandomForestRegressor()
```

In [75]:
```python
forestmodel.fit(x_train,y_train)
```

Out[75]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [76]:
```python
forestoutput=forestmodel.predict(x_test)
```

In [77]:
```python
plt.scatter(x_test['x'],y_test,color="red")
plt.scatter(x_test['x'],forestoutput,color="yellow")
plt.legend(["Actual","Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x v/s price')
plt.show()
```



In [79]:
```python
#model evaluation
print("R^2:",metrics.r2_score(y_test, forestoutput))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, forestoutput))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
print("MAE:",metrics.mean_absolute_error(y_test, forestoutput))
print("MSE:",metrics.mean_squared_error(y_test, forestoutput))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, forestoutput)))
```

```
R^2: 0.9810424435588266
Adjusted R^2: 0.9810318894323903
MAE: 272.37833364096133
MSE: 299137.16846937634
RMSE: 546.9343365243915
```

In [ ]: