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

```
df = pd.read_csv("/content/diamonds.csv")
df.head()
```

	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

```
df.columns
```

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

```
df.shape
```

```
(53940, 11)
```

```
df.isnull().sum() # No Missing Values
```

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

```
df.duplicated().sum() # No Duplicated Values.
```

```
0
```

```
df.drop("Unnamed: 0",axis=1,inplace=True) #UnWanted Column
```

```
numericals = df.select_dtypes(include=['float64', 'int64'])
q1 = numericals.quantile(0.25)
q3 = numericals.quantile(0.75)
IQR = q3-q1
lower = q1 - 1.5*(IQR)
higher = q3 + 1.5*(IQR)
outliers = df[((numericals < lower) | (numericals > higher)).any(axis=1)]
print("Total Outliers in this dataset : ",outliers.shape[0])
```

```
Total Outliers in this dataset :  6416
```

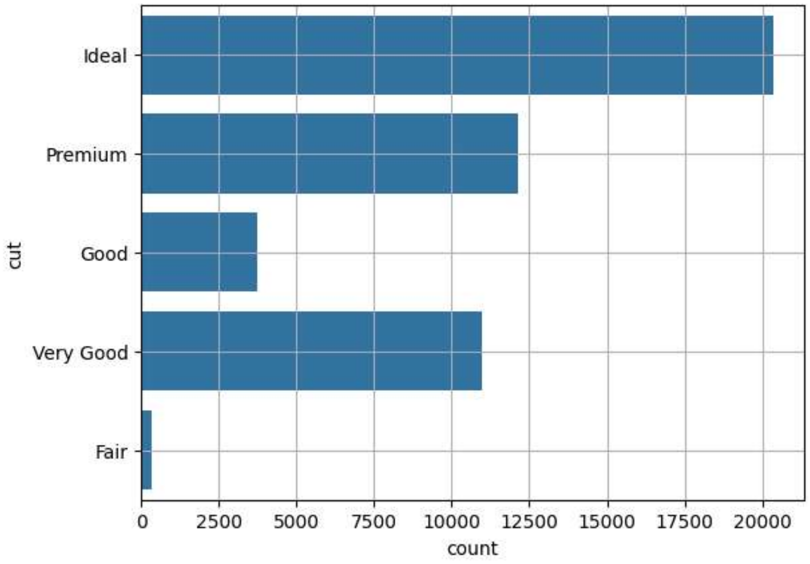
```
df_cleaned = df[~((numericals < lower) | (numericals > higher)).any(axis=1)]
df_cleaned.shape # After Removing Outliers
```

```
(47524, 10)
```

```
df["cut"].unique()
```

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

```
sns.countplot(df_cleaned["cut"])
plt.grid(True)
plt.show()
```



1. In this dataset, most of the diamonds have an Ideal cut.

2. if the cut grade of a diamond is "Ideal," it will generally be more expensive and have superior brilliance compared to diamonds with lower cut grades.

```
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import OrdinalEncoder
order = ['Fair', 'Good', 'Very Good', 'Premium', 'Ideal'] # Ranking Order From Lowest to highest
encoder = OrdinalEncoder(categories=[order])
df_cleaned["Cut"]=encoder.fit_transform(df_cleaned[['cut']])
df_cleaned.head()
```

	carat	cut	color	clarity	depth	table	price	x	y	z	Cut
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	4.0
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	3.0
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	3.0
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	1.0
5	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48	2.0

```
df_cleaned.drop('cut',axis=1,inplace=True)
```

```
df_cleaned["color"].unique()
```

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

Make Note :

D: Colorless (highest quality, most valuable)

E: Colorless

F: Colorless to Near Colorless

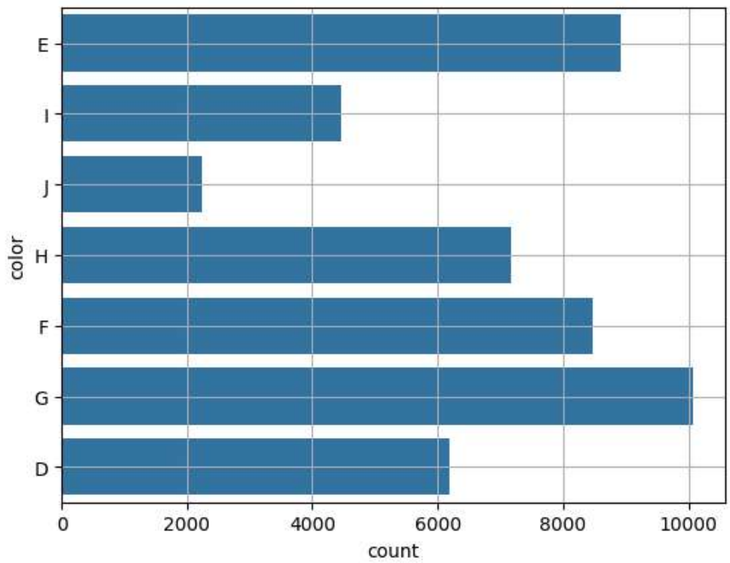
G: Near Colorless

H: Near Colorless

I: Near Colorless to Faint Yellow (lower in the near colorless range)

J: Faint Yellow (lower in the near colorless range)

```
sns.countplot(df_cleaned["color"])
plt.grid(True)
plt.show()
```



In this dataset, the majority of diamonds are graded as 'G', indicating they are near colorless. This suggests that these diamonds are less expensive compared to higher-grade colorless diamonds (D, E, F), yet they still maintain good quality and brilliance..

```
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import OrdinalEncoder
order = ['J', 'I', 'H', 'G', 'F', 'E', 'D'] # Ranking Order From Lowest to highest
encoder = OrdinalEncoder(categories=[order])
df_cleaned["Color"]=encoder.fit_transform(df_cleaned[['color']])
df_cleaned.drop('color',axis=1,inplace=True)
df_cleaned.head()
```

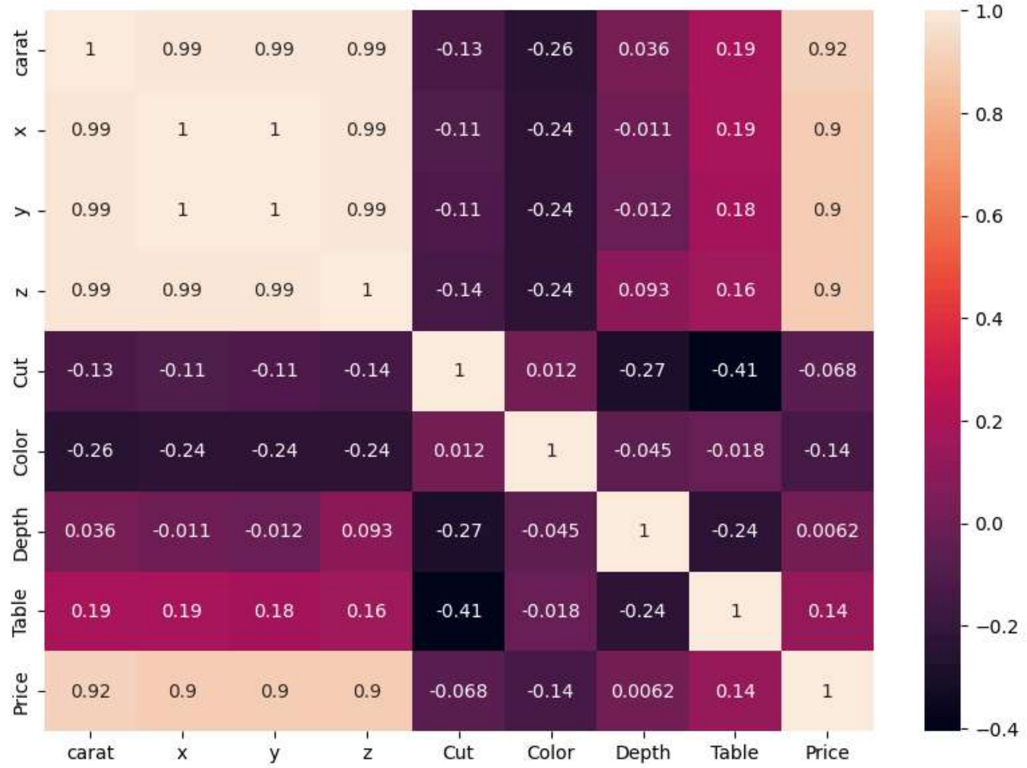
	carat	clarity	depth	table	price	x	y	z	Cut	Color
0	0.23	SI2	61.5	55.0	326	3.95	3.98	2.43	4.0	5.0
1	0.21	SI1	59.8	61.0	326	3.89	3.84	2.31	3.0	5.0
3	0.29	VS2	62.4	58.0	334	4.20	4.23	2.63	3.0	1.0
4	0.31	SI2	63.3	58.0	335	4.34	4.35	2.75	1.0	0.0
5	0.24	VVS2	62.8	57.0	336	3.94	3.96	2.48	2.0	0.0

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaling_columns=df_cleaned[['depth','table','price']]
df_cleaned[['Depth','Table','Price']] = scaler.fit_transform(scaling_columns) #Scaling down the data.
```

```
df_cleaned.drop(['depth','table','price'],axis=1,inplace=True)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(df_cleaned.select_dtypes(include=['float64', 'int64']).corr(),annot=True)
```

```
<Axes: >
```



```
df_cleaned.describe()
```

	carat	x	y	z	Cut	Color	Depth	Table	Price
count	47524.000000	47524.000000	47524.000000	47524.000000	47524.000000	47524.000000	4.752400e+04	4.752400e+04	4.752400e+04
mean	0.708700	5.546656	5.551478	3.428376	3.018222	3.464376	-8.247104e-16	-1.997486e-16	-3.827517e-17
std	0.371104	0.979906	0.973990	0.606158	1.018196	1.683839	1.000011e+00	1.000011e+00	1.000011e+00
min	0.200000	3.730000	3.680000	1.410000	0.000000	0.000000	-2.700528e+00	-2.602081e+00	-1.012363e+00
25%	0.380000	4.640000	4.650000	2.860000	2.000000	2.000000	-6.207727e-01	-6.233822e-01	-8.072475e-01
50%	0.600000	5.440000	5.450000	3.360000	3.000000	3.000000	1.026205e-01	-1.287076e-01	-3.813779e-01
75%	1.010000	6.410000	6.410000	3.980000	4.000000	5.000000	6.451654e-01	8.606416e-01	5.551899e-01
max	2.000000	8.280000	8.270000	5.300000	4.000000	6.000000	2.634497e+00	3.086677e+00	3.190690e+00

```
Start coding or generate with AI.
```