Libraries Imported

> NumPy is a general-purpose array-processing package.
>
> Pandas library provides easy-to-use data structures and data analysis tools for Python.
>
> Matplotlib is a Python 2D plotting library.
>
> Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn import preprocessing
warnings.filterwarnings('ignore')
```

In [5]:

```
data = pd.read_csv("C:\\Users\\admin\\Downloads\\adult.csv")
```

In [13]:

```
data.shape
```

Out[13]:

```
(32561, 15)
```

Subset : column label or sequence of labels, optional

```
    Only consider certain columns for identifying duplicates, by default use all
of the columns
```

Keep : {‘first’, ‘last’, False}, default ‘first’

```
    first : Markduplicates as True except for the first occurrence.
    last : Mark duplicates as True except for the last occurrence.
    False : Mark all duplicates as True
```

In [9]:

```
bool_series = data.duplicated(subset=None, keep = False)
```

In [10]:

```
data.duplicated(subset=None, keep = False).value_counts()
```

Out[10]:

```
False    32514
True        47
dtype: int64
```

In [11]:

```
bool_series = data.duplicated(subset=None, keep = False)
```

In [12]:

```
data_unique = data[~bool_series]
```

In [17]:

```
print("Berfore Removing Duplicates " , data.shape)
print("After Removing Duplicates " , data_unique.shape)
```

```
Berfore Removing Duplicates  (32561, 15)
After Removing Duplicates  (32514, 15)
```

Concatenation of Datasets

In [19]:

```
dataset1 = pd.read_csv("C:\\Users\\admin\\Downloads\\WorldCupMatches.csv")
dataset1.shape
```

Out[19]:

```
(852, 20)
```

In [20]:

```
dataset2 = pd.read_csv("C:\\Users\\admin\\Downloads\\WorldCupMatches.csv")
dataset2.shape
```

Out[20]:

```
(852, 20)
```

In [21]:

```
combined_csv = pd.concat([dataset1,dataset2])
```

In [23]:

```
combined_csv.to_csv("combined_csv.csv",index=False , encoding="utf-8-sig")
combined_csv.shape
```

Out[23]:

```
(1704, 20)
```

Data Viewing using Matplotlib and Seaborn Libraries

Hist function in in matplot can take multiple inputs such as :

> Orientation
>
> Color
>
> Column Name
>
> Density
>
> Bin

In [24]:

```
data = pd.read_csv("C:\\Users\\admin\\Downloads\\diamonds.csv")
```

In [25]:

```
data
```

Out[25]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
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

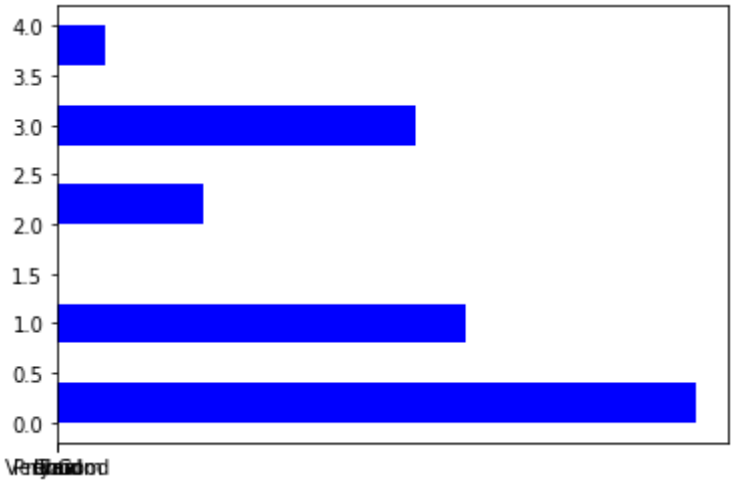
53940 rows × 11 columns

In [29]:

```
plt.hist(data['cut'], color='blue' , orientation='horizontal')
```

Out[29]:

```
(array([21551.,     0., 13791.,     0.,     0.,  4906.,     0., 12082.,
            0.,  1610.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <a list of 10 Patch objects>)
```

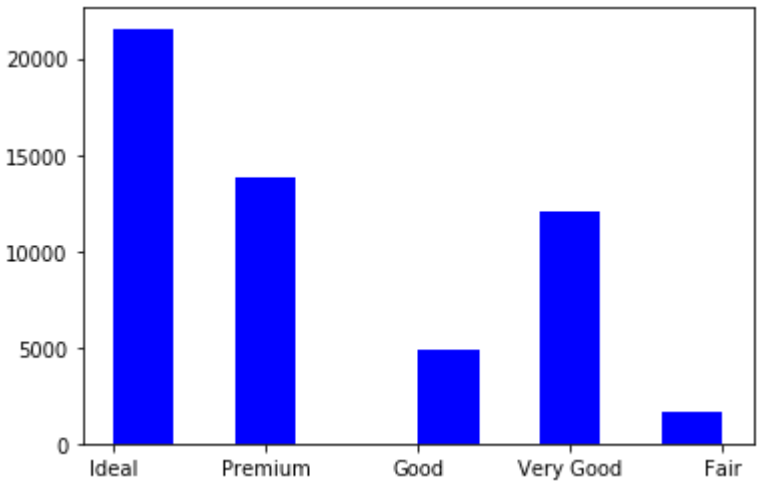


In [28]:

```
plt.hist(data['cut'], color='blue' , orientation='vertical')
```

Out[28]:

```
(array([21551.,     0., 13791.,     0.,     0.,  4906.,     0., 12082.,
            0.,  1610.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <a list of 10 Patch objects>)
```

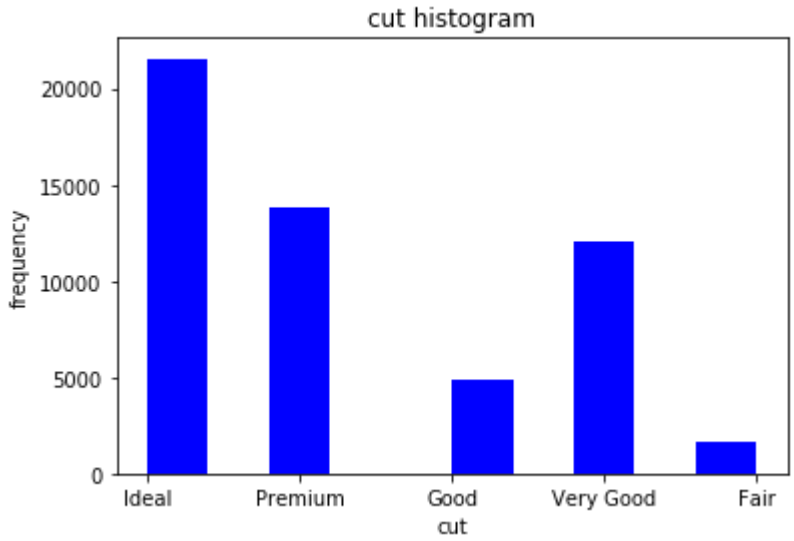


In [30]:

```
plt.hist(data['cut'],color='blue',orientation='vertical')
plt.title('cut histogram')
plt.xlabel('cut')
plt.ylabel('frequency')
cut=('ideal','Premium','Good','Very good','fair')
index=np.arrange(len(cut))
plt.xticks(index,cut,rotation=90)
plt.show()
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-30-fbf65d46d844> in <module>
      4 plt.ylabel('frequency')
      5 cut=('ideal','Premium','Good','Very good','fair')
----> 6 index=np.arrange(len(cut))
      7 plt.xticks(index,cut,rotation=90)
      8 plt.show()

AttributeError: module 'numpy' has no attribute 'arrange'
```



In [31]:

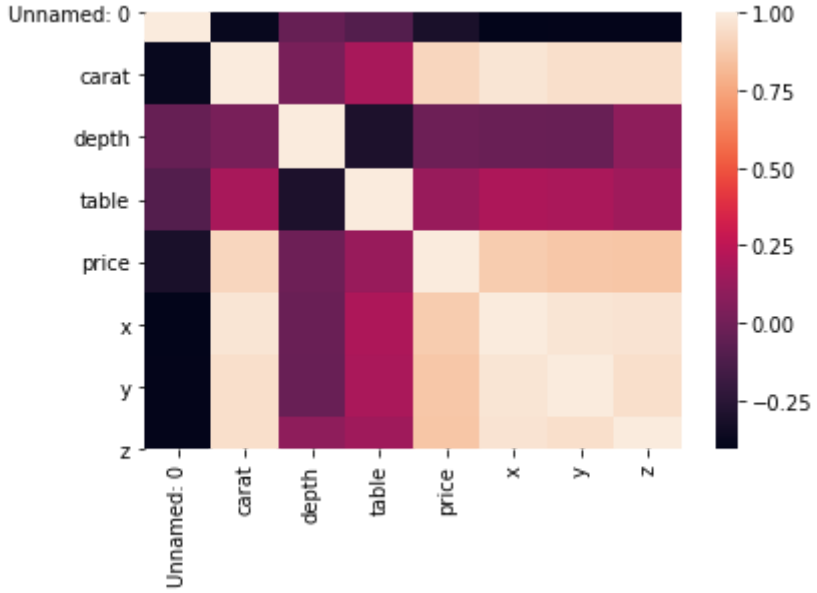
```
data.corr()
```

Out[31]:

	Unnamed: 0	carat	depth	table	price	x	y	
Unnamed: 0	1.000000	-0.377983	-0.034800	-0.100830	-0.306873	-0.405440	-0.395843	-0.399
carat	-0.377983	1.000000	0.028224	0.181618	0.921591	0.975094	0.951722	0.953
depth	-0.034800	0.028224	1.000000	-0.295779	-0.010647	-0.025289	-0.029341	0.094
table	-0.100830	0.181618	-0.295779	1.000000	0.127134	0.195344	0.183760	0.150
price	-0.306873	0.921591	-0.010647	0.127134	1.000000	0.884435	0.865421	0.861
x	-0.405440	0.975094	-0.025289	0.195344	0.884435	1.000000	0.974701	0.970
y	-0.395843	0.951722	-0.029341	0.183760	0.865421	0.974701	1.000000	0.952
z	-0.399208	0.953387	0.094924	0.150929	0.861249	0.970772	0.952006	1.000

In [39]:

```
sns.heatmap(data.corr(method="pearson"))
plt.show()
plt.savefig("heatmap_pearson.jpg")
plt.show()
```



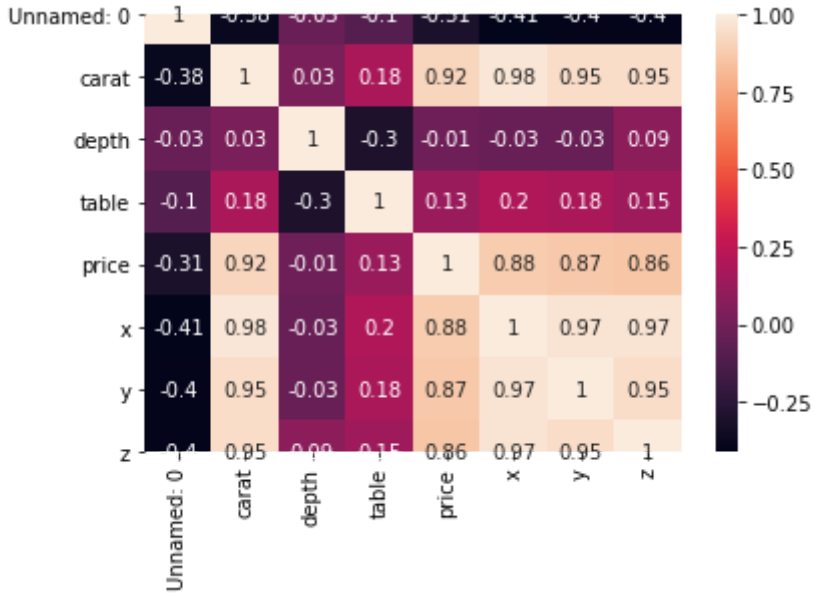
<Figure size 432x288 with 0 Axes>

In [40]:

```
correlation_matrix=data.corr().round(2)
sns.heatmap(data=correlation_matrix , annot=True)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16939280fc8>
```



In [41]:

```
data["cut"]=data["cut"].str.lower()
data.head(10)
```

Out[41]:

	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
5	6	0.24	very good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	7	0.24	very good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	8	0.26	very good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	9	0.22	fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	10	0.23	very good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

In [42]:

```
data["cut"]=data["cut"].str.title()
data.head(10)
```

Out[42]:

	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
5	6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

In [43]:

```
data
```

Out[43]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
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

53940 rows × 11 columns

In [45]:

```
price_val=data['price']
price_val
```

Out[45]:

```
0         326
1         326
2         327
3         334
4         335
         ...
53935    2757
53936    2757
53937    2757
53938    2757
53939    2757
Name: price, Length: 53940, dtype: int64
```

In [47]:

```
normalized_price = (price_val - price_val.min())/(price_val.max()-price_val.min())
data['normalized_price']=normalized_price
data.sort_values(by='price')
```

Out[47]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z	norm
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	
...	...	...	...	...	...	...	...	...	...	...	...	
27745	27746	2.00	Very Good	H	SI1	62.8	57.0	18803	7.95	8.00	5.01	
27746	27747	2.07	Ideal	G	SI2	62.5	55.0	18804	8.20	8.13	5.11	
27747	27748	1.51	Ideal	G	IF	61.7	55.0	18806	7.37	7.41	4.56	
27748	27749	2.00	Very Good	G	SI1	63.5	56.0	18818	7.90	7.97	5.04	
27749	27750	2.29	Premium	I	VS2	60.8	60.0	18823	8.50	8.47	5.16	

53940 rows × 12 columns

In [ ]:

```

```