```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import urllib.request
```

```
url = "https://api.covid19india.org/states_daily.json"
url
```

```
'https://api.covid19india.org/states_daily.json'
```

```
urllib.request.urlretrieve(url,'data.json')
```

```
('data.json', <http.client.HTTPMessage at 0x7fae07f444d0>)
```

```
cd=pd.read_json('data.json')
cd
```

	states_daily
0	{'an': '0', 'ap': '1', 'ar': '0', 'as': '0', '...
1	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
2	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
3	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
4	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
...	...
1558	{'an': '2', 'ap': '1835', 'ar': '255', 'as': '...
1559	{'an': '0', 'ap': '16', 'ar': '0', 'as': '10',...
1560	{'an': '1', 'ap': '909', 'ar': '165', 'as': '7...
1561	{'an': '0', 'ap': '1543', 'ar': '249', 'as': '...
1562	{'an': '0', 'ap': '13', 'ar': '0', 'as': '10',...

1563 rows × 1 columns

Online csv file retreievel example

```
url ='https://www.stats.govt.nz/assets/Uploads/Annual-enterprise-survey
url
```

```
'https://www.stats.govt.nz/assets/Uploads/Annual-enterprise-survey/An
nual-enterprise-survey-2020-financial-year-provisional/Download-data/
annual-enterprise-survey-2020-financial-year-provisional-size-bands-c
```

```
urllib.request.urlretrieve(url,'sample.csv')
```

```
('sample.csv', <http.client.HTTPMessage at 0x7fae07854650>)
```

```
scsv= pd.read_csv('sample.csv')
scsv.head()
```

	year	industry_code_ANZSIC	industry_name_ANZSIC	rme_size_grp	var
0	2011	A	Agriculture, Forestry and Fishing	a_0	Activ
1	2011	A	Agriculture, Forestry and Fishing	a_0	emp
2	2011	A	Agriculture, Forestry and Fishing	a_0	S and
3	2011	A	Agriculture, Forestry and Fishing	a_0	gove fu grar su

```
import json
with open('data.json') as f :
   data= json.load(f)
data
```

```
{'states_daily': [{'an': '0',
   'ap': '1',
   'ar': '0',
   'as': '0',
   'br': '0',
   'ch': '0',
   'ct': '0',
   'date': '14-Mar-20',
   'dateymd': '2020-03-14',
         'dd': '0',
         'dl': '7',
         'dn': '0',
         'ga': '0',
         'gj': '0',
         'hp': '0',
         'hr': '14',
         'jh': '0',
         'jk': '2',
         'ka': '6',
         'kl': '19',
         'la': '0',
         'ld': '0',
         'mh': '14',
         'ml': '0',
         'mn': '0',
         'mp': '0',
         'mz': '0',
         'nl': '0',
         'or': '0',
         'pb': '1',
         'py': '0',
         'rj': '3',
         'sk': '0',
         'status': 'Confirmed',
         'tg': '1',
         'tn': '1',
         'tr': '0',
         'tt': '81',
         'un': '0',
         'up': '12',
         'ut': '0',
         'wb': '0'},
        {'an': '0',
         'ap': '0',
         'ar': '0',
         'as': '0',
         'br': '0',
         'ch': '0',
         'ct': '0',
         'date': '14-Mar-20',
         'dateymd': '2020-03-14',
         'dd': '0',
         'dl': '1',
         'dn': '0',
         'ga': '0',
         'gj': '0',
         'hp': '0',
         'hr': '0',
         'jh': '0',
```

```python
data = data['states_daily']
```

```
data
```

```
    [{'an': '0',
      'ap': '1',
      'ar': '0',
      'as': '0',
      'br': '0',
      'ch': '0',
      'ct': '0',
      'date': '14-Mar-20',
      'dateymd': '2020-03-14',
      'dd': '0',
      'dl': '7',
      'dn': '0',
      'ga': '0',
      'gj': '0',
      'hp': '0',
      'hr': '14',
      'jh': '0',
      'jk': '2',
      'ka': '6',
      'kl': '19',
      'la': '0',
      'ld': '0',
      'mh': '14',
      'ml': '0',
      'mn': '0',
      'mp': '0',
      'mz': '0',
      'nl': '0',
      'or': '0',
      'pb': '1',
      'py': '0',
      'rj': '3',
      'sk': '0',
      'status': 'Confirmed',
      'tg': '1',
      'tn': '1',
      'tr': '0',
      'tt': '81',
      'un': '0',
      'up': '12',
      'ut': '0',
      'wb': '0'},
     {'an': '0',
      'ap': '0',
      'ar': '0',
      'as': '0',
      'br': '0',
      'ch': '0',
      'ct': '0',
      'date': '14-Mar-20',
      'dateymd': '2020-03-14',
      'dd': '0',
'dl': '1',
'dn': '0',
'ga': '0',
'gj': '0',
'hp': '0',
'hr': '0',
'jh': '0',
'jk': '0'
```

```python
cd= pd.json_normalize(data)
cd
```

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	sk	stat
0	0	1	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Confirm
1	0	0	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Recover
2	0	0	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Deceas
3	0	0	0	0	0	0	0	15-Mar-20	2020-03-15	0	...	0	Confirm
4	0	0	0	0	0	0	0	15-Mar-20	2020-03-15	0	...	0	Recover
...	...	...	...	...	...	...	...	...	...	...	...	...	
1558	2	1835	255	857	38	1	114	15-Aug-21	2021-08-15	0	...	213	Recover
1559	0	16	0	10	0	0	1	15-Aug-21	2021-08-15	0	...	0	Deceas
1560	1	909	165	758	14	2	68	16-Aug-21	2021-08-16	0	...	20	Confirm
1561	0	1543	249	1014	42	3	224	16-Aug-21	2021-08-16	0	...	147	Recover
1562	0	13	0	10	0	0	1	16-Aug-21	2021-08-16	0	...	0	Deceas

1563 rows × 42 columns

```python
df=cd
```

```
df.date=pd.to_datetime(df.date)
df
```

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	sk	stat
0	0	1	0	0	0	0	0	2020-03-14	2020-03-14	0	...	0	Confirn
1	0	0	0	0	0	0	0	2020-03-14	2020-03-14	0	...	0	Recove
2	0	0	0	0	0	0	0	2020-03-14	2020-03-14	0	...	0	Deceas
3	0	0	0	0	0	0	0	2020-03-15	2020-03-15	0	...	0	Confirn
4	0	0	0	0	0	0	0	2020-03-15	2020-03-15	0	...	0	Recove
...	...	...	...	...	...	...	...	...	...	...	...	...	
1558	2	1835	255	857	38	1	114	2021-08-15	2021-08-15	0	...	213	Recove
1559	0	16	0	10	0	0	1	2021-08-15	2021-08-15	0	...	0	Deceas
1560	1	909	165	758	14	2	68	2021-08-16	2021-08-16	0	...	20	Confirn
1561	0	1543	249	1014	42	3	224	2021-08-16	2021-08-16	0	...	147	Recove
1562	0	13	0	10	0	0	1	2021-08-16	2021-08-16	0	...	0	Deceas

1563 rows × 42 columns

```
df=df[df['status']=='Confirmed']
df
```

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	sk	status
0	0	1	0	0	0	0	0	2020-03-14	2020-03-14	0	...	0	Confirme
3	0	0	0	0	0	0	0	2020-03-15	2020-03-15	0	...	0	Confirme
6	0	0	0	0	0	0	0	2020-03-16	2020-03-16	0	...	0	Confirme
9	0	0	0	0	0	0	0	2020-03-17	2020-03-17	0	...	0	Confirme
12	0	0	0	0	0	0	0	2020-03-18	2020-03-18	0	...	0	Confirme
...	...	...	...	...	...	...	...	...	...	...	...	...	..
1548	0	1859	180	935	43	12	98	2021-08-12	2021-08-12	0	...	100	Confirme
1551	0	1746	166	763	47	15	77	2021-08-13	2021-08-13	0	...	150	Confirme
1554	0	1535	161	755	39	4	83	2021-08-14	2021-08-14	0	...	129	Confirme
1557	0	1506	48	411	28	1	49	2021-08-15	2021-08-15	0	...	152	Confirme
1560	1	909	165	758	14	2	68	2021-08-16	2021-08-16	0	...	20	Confirme

521 rows × 42 columns

```
df.drop('status',axis=1,inplace = True)
df.head()
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: Sett
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
  errors=errors,
```

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	rj	sk	tg	tn	tr
0	0	1	0	0	0	0	0	2020-03-14	2020-03-14	0	...	3	0	1	1	0
3	0	0	0	0	0	0	0	2020-03-15	2020-03-15	0	...	1	0	2	0	0
6	0	0	0	0	0	0	0	2020-03-16	2020-03-16	0	...	0	0	1	0	0
9	0	0	0	0	0	0	0	2020-03-17	2020-03-17	0	...	0	0	1	0	0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2020-	2020-03-	[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
df.drop('dateymd',axis=1,inplace = True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: Sett
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
  errors=errors,
```

```
df.head()
```

	an	ap	ar	as	br	ch	ct	date	dd	dl	...	rj	sk	tg	tn	tr	tt
0	0	1	0	0	0	0	0	2020-03-14	0	7	...	3	0	1	1	0	81
3	0	0	0	0	0	0	0	2020-03-15	0	0	...	1	0	2	0	0	27
6	0	0	0	0	0	0	0	2020-03-16	0	0	...	0	0	1	0	0	15
9	0	0	0	0	0	0	0	2020-03-17	0	1	...	0	0	1	0	0	11

```python
df.columns
```

```
Index(['an', 'ap', 'ar', 'as', 'br', 'ch', 'ct', 'date', 'dd', 'dl', '
       'ga', 'gj', 'hp', 'hr', 'jh', 'jk', 'ka', 'kl', 'la', 'ld', 'mh
       'mn', 'mp', 'mz', 'nl', 'or', 'pb', 'py', 'rj', 'sk', 'tg', 'tr
       'tt', 'un', 'up', 'ut', 'wb'],
      dtype='object')
```

```python
df.set_index('date',inplace=True)
```

```python
df=df.apply(pd.to_numeric) #apply for every cols
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 521 entries, 2020-03-14 to 2021-08-16
Data columns (total 39 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   an      521 non-null    int64
 1   ap      521 non-null    int64
 2   ar      521 non-null    int64
 3   as      521 non-null    int64
 4   br      521 non-null    int64
 5   ch      521 non-null    int64
 6   ct      521 non-null    int64
 7   dd      521 non-null    int64
 8   dl      521 non-null    int64
 9   dn      521 non-null    int64
 10  ga      521 non-null    int64
 11  gj      521 non-null    int64
 12  hp      521 non-null    int64
 13  hr      521 non-null    int64
 14  jh      521 non-null    int64
 15  jk      521 non-null    int64
 16  ka      521 non-null    int64
 17  kl      521 non-null    int64
 18  la      521 non-null    int64
 19  ld      521 non-null    int64
 20  mh      521 non-null    int64
 21  ml      521 non-null    int64
 22  mn      521 non-null    int64
 23  mp      521 non-null    int64
 24  mz      521 non-null    int64
 25  nl      521 non-null    int64
 26  or      521 non-null    int64
 27  pb      521 non-null    int64
 28  py      521 non-null    int64
 29  rj      521 non-null    int64
 30  sk      521 non-null    int64
 31  tg      521 non-null    int64
 32  tn      521 non-null    int64
 33  tr      521 non-null    int64
 34  tt      521 non-null    int64
 35  un      521 non-null    int64
 36  up      521 non-null    int64
 37  ut      521 non-null    int64
 38  wb      521 non-null    int64
dtypes: int64(39)
memory usage: 162.8 KB
```

```
df2=df.tail(7)
df2
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	...	rj	sk	tg	tn
date															
2021-08-10	2	1461	233	929	44	8	112	0	52	1	...	11	110	494	1893
2021-08-11	0	1869	188	886	47	5	83	0	37	0	...	19	157	482	1964
2021-08-12	0	1859	180	935	43	12	98	0	49	1	...	17	100	453	1942
2021-08-13	0	1746	166	763	47	15	77	0	50	0	...	24	150	427	1933
2021-08-14	0	1535	161	755	39	4	83	0	50	0	...	14	129	420	1916
2021-															

```
df2.style
```

date	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

Data frame table styling

```
def color_red_negative(x) :
    color = 'red' if x <0 else 'blue'
    return 'color:' + color
```

```
df2.style.applymap(color_red_negative) #applymap for rows # for each ce
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```python
df2.style.highlight_max(color='red') #columnwise max
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```
df2.style.highlight_max(color='red',axis=1) #rowwise max , axis =1
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```
df2.style.highlight_max(color='green').highlight_min(color='red') #bot
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```python
df2.drop('tt',axis=1,inplace=True)
df2
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: Sett
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
  errors=errors,
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	...	py	rj	sk	tg	
date																
2021-08-10	2	1461	233	929	44	8	112	0	52	1	...	101	11	110	494	1
2021-08-11	0	1869	188	886	47	5	83	0	37	0	...	114	19	157	482	1
2021-08-12	0	1859	180	935	43	12	98	0	49	1	...	109	17	100	453	1
2021-08-13	0	1746	166	763	47	15	77	0	50	0	...	113	24	150	427	1
2021-08-14	0	1535	161	755	39	4	83	0	50	0	...	101	14	129	420	1
2021-08-15	0	1506	48	411	28	1	49	0	53	0	...	79	18	152	245	1

```python
def bold_max_value(x) :
    ismax=(x==x.max())
    return ['font-weight: bold' if y else '' for y in ismax]
```

```
df2.style.apply(bold_max_value)
```

date	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
2021-08-10 00:00:00	**2**	1461	**233**	929	44	8	**112**	**0**	52	1	**141**	21	**419**	23	34
2021-08-11 00:00:00	0	**1869**	188	886	**47**	5	83	**0**	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	**935**	43	12	98	**0**	49	1	88	17	354	16	**44**
2021-08-13 00:00:00	0	1746	166	763	**47**	**15**	77	**0**	50	0	67	23	333	**26**	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	**0**	50	0	88	**25**	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	**0**	**53**	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	**0**	27	**2**	62	14	276	22	35

```python
df2.style.apply(bold_max_value).highlight_max(color='red') #statewise
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	**2**	1461	**233**	929	44	8	**112**	**0**	52	1	**141**	21	**419**	23	34
2021-08-11 00:00:00	0	**1869**	188	886	**47**	5	83	**0**	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	**935**	43	12	98	**0**	49	1	88	17	354	16	**44**
2021-08-13 00:00:00	0	1746	166	763	**47**	**15**	77	**0**	50	0	67	23	333	**26**	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	**0**	50	0	88	**25**	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	**0**	**53**	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	**0**	27	**2**	62	14	276	22	35

```
df2.style.apply(bold_max_value).highlight_max(color='red',axis=1) #row
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	**2**	1461	**233**	929	44	8	**112**	**0**	52	1	**141**	21	**419**	23	34
2021-08-11 00:00:00	0	**1869**	188	886	**47**	5	83	**0**	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	**935**	43	12	98	**0**	49	1	88	17	354	16	**44**
2021-08-13 00:00:00	0	1746	166	763	**47**	**15**	77	**0**	50	0	67	23	333	**26**	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	**0**	50	0	88	**25**	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	**0**	**53**	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	**0**	27	**2**	62	14	276	22	35

```python
df2.style.background_gradient(cmap='Greens') #column_wise
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```python
df2.style.background_gradient(cmap='Reds',axis=1)
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```
df2.style.background_gradient(cmap='Reds',subset=['kl','ka','ap','dl']
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```python
df2.style.bar() #columnwise
```

date	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```
df2.style.bar(axis=1) #rowwise
```

date	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```
df2.style.bar(subset=['kl','ka','mh'])
```

	an	ap	ar	as	br	ch	ct	dd	dl	dn	ga	gj	hp	hr	jh
date															
2021-08-10 00:00:00	2	1461	233	929	44	8	112	0	52	1	141	21	419	23	34
2021-08-11 00:00:00	0	1869	188	886	47	5	83	0	37	0	103	16	374	16	14
2021-08-12 00:00:00	0	1859	180	935	43	12	98	0	49	1	88	17	354	16	44
2021-08-13 00:00:00	0	1746	166	763	47	15	77	0	50	0	67	23	333	26	32
2021-08-14 00:00:00	0	1535	161	755	39	4	83	0	50	0	88	25	284	14	28
2021-08-15 00:00:00	0	1506	48	411	28	1	49	0	53	0	75	16	182	22	27
2021-08-16 00:00:00	1	909	165	758	14	2	68	0	27	2	62	14	276	22	35

```python
df2[['kl','ka','mh']].style.bar()
```

	kl	ka	mh
date			
2021-08-10 00:00:00	21119	1338	5609
2021-08-11 00:00:00	23500	1826	5560
2021-08-12 00:00:00	21445	1857	6388
2021-08-13 00:00:00	20452	1669	6686
2021-08-14 00:00:00	19451	1632	5787

```python
df2[['kl','ka','mh']].style.bar(axis=1)
```

	kl	ka	mh
date			
2021-08-10 00:00:00	21119	1338	5609
2021-08-11 00:00:00	23500	1826	5560
2021-08-12 00:00:00	21445	1857	6388
2021-08-13 00:00:00	20452	1669	6686
2021-08-14 00:00:00	19451	1632	5787

```python
df2[['kl','hr','mh']].style.bar(subset=['kl'],color='red').bar(subset=
```

	kl	hr	mh
date			
2021-08-10 00:00:00	21119	23	5609
2021-08-11 00:00:00	23500	16	5560
2021-08-12 00:00:00	21445	16	6388
2021-08-13 00:00:00	20452	26	6686
2021-08-14 00:00:00	19451	14	5787

```python
x=np.random.normal(size=1000)
x
```

```
array([-6.45796143e-01,  5.33061022e-01,  1.11772294e+00, -2.79107374e
       -1.68896087e+00, -1.03554915e+00, -7.61460138e-01,  3.70915468e
        6.09154702e-01, -1.05599751e+00, -1.00761335e+00, -2.26041849e
        3.58477156e-01, -7.68739432e-03,  1.09356776e+00, -4.37733640e
       -7.25151913e-01,  3.86062262e-02,  2.86321063e+00,  1.95194182e
       -3.05090416e-01, -1.23011913e+00, -9.11323744e-01, -8.15470112e
       -6.67888421e-01,  2.67636423e-01, -1.28221037e+00, -1.41251082e
        1.23628889e-01, -1.97324576e+00, -9.32576877e-01,  9.92754702e
       -1.13515737e+00,  1.10885659e+00,  5.32515489e-02,  1.88697923e
       -3.76202059e-01,  1.29933124e+00,  1.66592102e-01, -5.99221966e
       -1.67216933e-01,  1.12284617e+00,  3.14733861e-01,  5.33980981e
       -1.53518998e+00, -1.40169314e+00, -5.46726168e-01,  1.36613321e
       -1.51070397e+00, -8.50353109e-01, -4.91364570e-03,  1.41384753e
        2.11908821e-01, -4.36990054e-01,  2.27967244e-01,  1.15229781e
        4.93675520e-01, -4.06220757e-01,  5.00182924e-01, -6.13800343e
        8.63601016e-01,  1.50869916e+00,  7.64553614e-01,  5.64572432e
       -1.01260083e+00,  9.37785390e-02, -1.57626505e+00,  1.03615614e
       -6.85115095e-01,  1.42156687e+00, -7.76237480e-01,  1.05825296e
        1.10358129e+00,  1.59629956e-01,  2.90972477e-01, -1.34712172e
       -1.18889106e+00, -1.99260772e-01, -1.06300335e+00,  5.78343330e
        9.08584018e-01,  5.53523635e-01,  4.89386167e-01,  8.13997967e
        3.46841730e-01, -1.38350684e+00,  3.25912038e-01,  2.16359108e
       -3.91604809e-01, -1.98539433e+00,  5.85279958e-01,  2.70801633e
        6.32076699e-01,  6.50420402e-01,  5.41023853e-02, -1.51273102e
       -2.04454050e-03, -6.23842458e-01,  4.91750834e-01,  9.97190037e
        7.24035349e-01,  2.12116841e-01,  6.97724455e-01,  7.23099163e
       -3.44103697e-01, -3.40343842e+00,  1.87765268e+00, -5.81833118e
       -2.30535187e-01,  1.35547233e+00,  7.63379955e-01,  2.94902103e
        1.22304635e+00, -4.68595357e-01,  5.23455412e-01,  8.72267336e
```

```
 9.10675605e-01, -3.12611527e-01, -1.74265659e+00,  1.48496286e
 2.94275348e-01, -9.27565572e-02, -1.13660623e+00,  1.87259397e
-5.55584486e-01,  8.05997199e-01, -1.88874423e+00, -1.31377105e
-5.01088579e-01,  1.06484877e+00,  3.88889151e-01, -1.14107342e
 4.22916442e-01,  8.54785733e-01,  8.80198407e-02, -5.81392029e
 1.92998294e+00, -6.09139618e-01,  1.67450407e-01, -5.76895701e
 3.88153685e-01,  1.57690929e-01, -2.47970500e+00, -4.65935401e
 1.54659318e+00,  1.24778549e+00,  5.35153378e-01, -3.42589460e
 2.75797533e-01, -8.70901999e-01, -2.04728362e+00, -1.54131438e
-1.03010236e-01, -2.62709915e-01, -2.92062837e-02,  7.61103834e
 1.38179805e+00,  1.75337085e+00,  1.76138230e+00,  5.49321917e
 1.30493510e-01, -1.26013737e+00,  6.35297797e-01, -3.01582121e
 5.56682149e-02,  1.24608310e+00, -5.77261194e-01, -5.91181966e
 1.66060008e+00, -7.32077073e-01,  7.41494996e-02, -1.16608792e
-1.34382646e+00, -2.11347722e+00,  3.36473686e-01, -4.97358214e
 1.44113418e+00,  6.80374599e-01,  4.49313642e-01,  6.35231530e
 8.44227603e-01, -4.85561878e-01,  1.30578348e+00,  6.20825550e
 1.12369732e+00,  2.10584301e-01,  9.71214573e-01,  1.45914512e
-1.44572382e+00, -1.58522692e+00,  1.20387280e+00, -2.53104802e
 5.63845769e-01, -1.15703552e+00,  7.38605584e-01, -2.74855521e
 6.24442107e-01, -3.81412911e-01, -1.41628065e+00,  1.42654555e
 5.94156733e-02, -8.65370408e-01,  1.00753215e+00,  4.55287298e
-5.50342058e-01, -1.09115242e-01,  2.02614902e+00, -5.43020480e
 1.77711473e+00, -3.94476159e-02, -1.79537463e+00, -3.64555631e
 1.51382804e+00, -4.69401948e-01, -2.91832809e-01, -6.19250285e
 1.15708928e-01, -1.00082771e+00,  5.60708717e-01, -4.58222496e
-1.95103980e-01, -7.46520567e-01,  5.21542391e-01, -1.01200886e
-2.61400293e+00,  3.81189529e-01, -9.54191312e-01, -1.21989837e
 8.40664597e-01,  1.48821757e+00,  1.81896294e+00, -7.17758201e
 2.85321268e-01, -1.76789271e+00,  9.80570941e-01, -1.91354463e
 7.03009413e-01,  3.17972664e-01,  1.18824728e+00,  6.44401345e
```

```python
d=sns.load_dataset('diamonds')
d
```

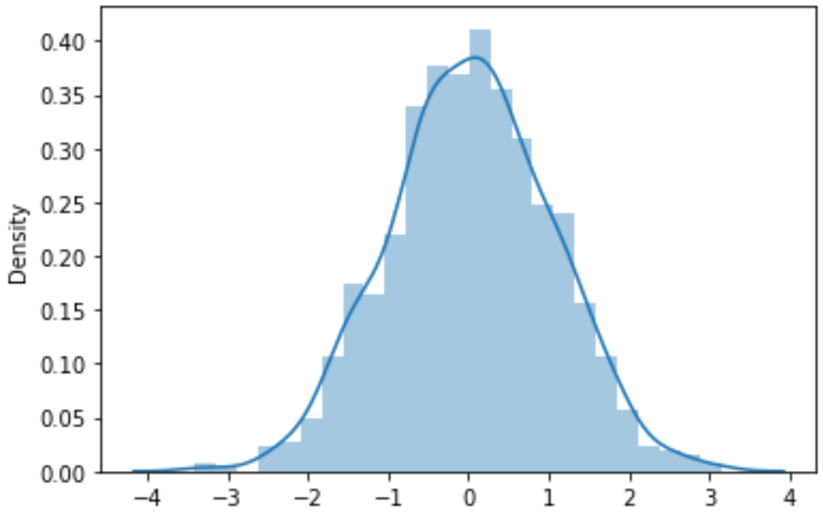
	carat	cut	color	clarity	depth	table	price	x	y	
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.4
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.3
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.3
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.6
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.7
...	...	...	...	...	...	...	...	...	...	..
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.5
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.6
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.5
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.7
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.6

53940 rows × 10 columns

dist plot

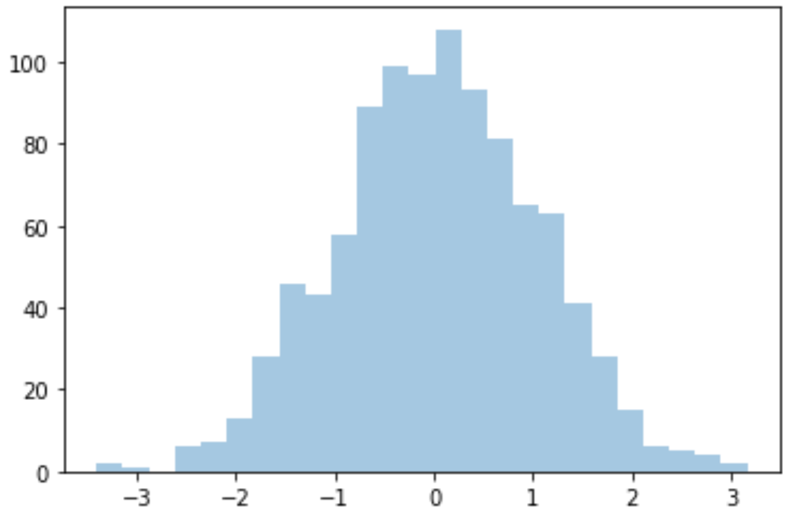
```
sns.distplot(x)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae0552a2d0>
```



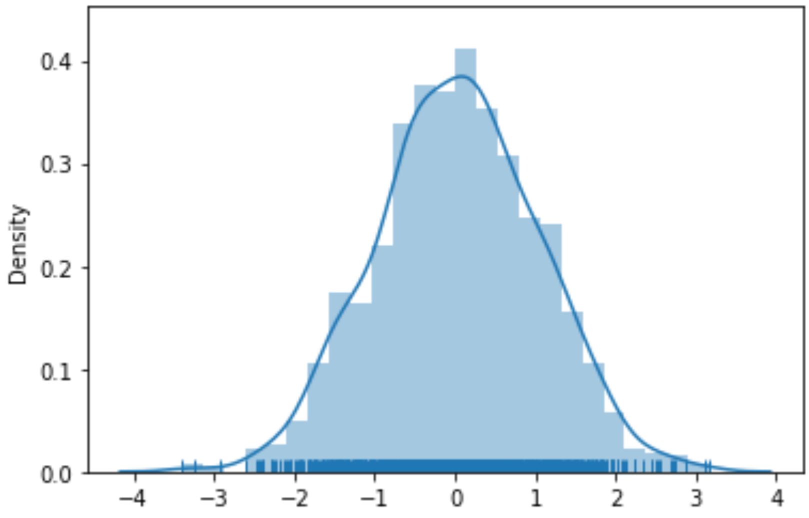
```
sns.distplot(x,kde=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae033cc650>
```



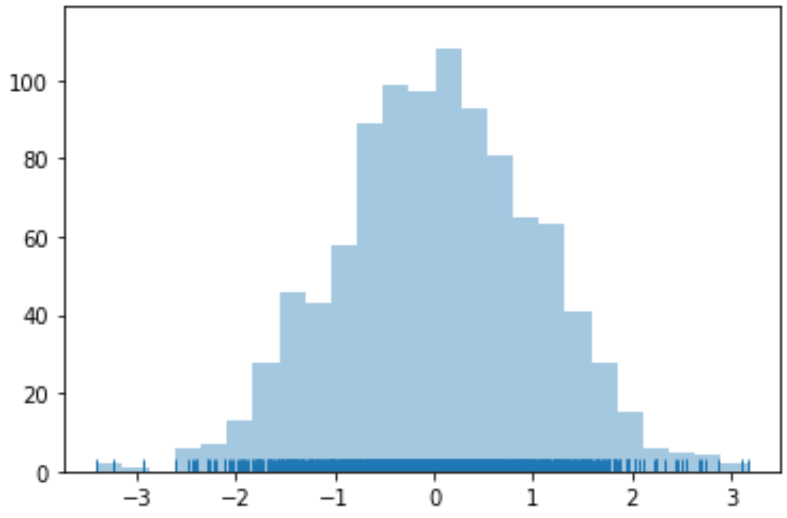
```
sns.distplot(x,rug=True) #observe rug below ,
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae02ede410>
```



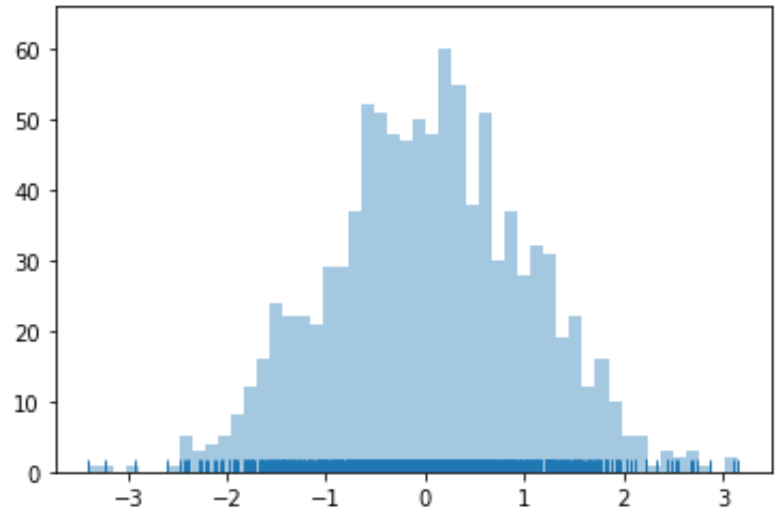
```
sns.distplot(x,kde=False,rug=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae00548a10>
```



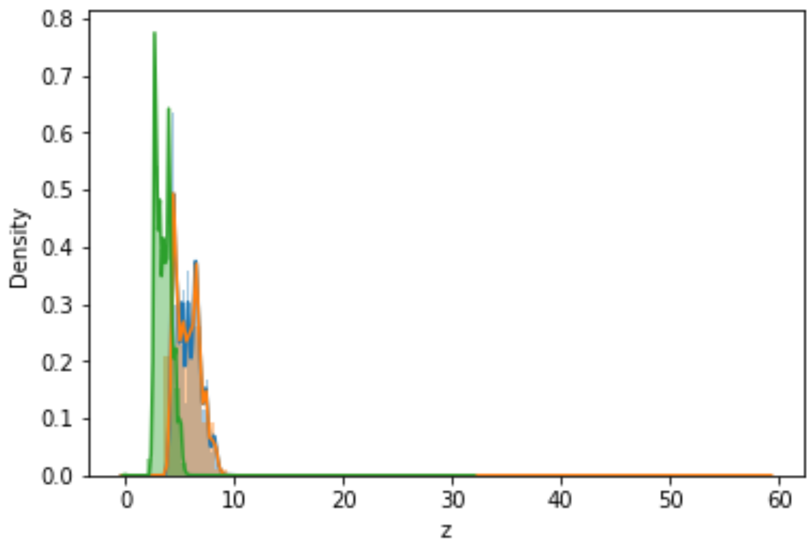
```
sns.distplot(x,kde=False,rug=True,bins=50) #bins
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae00532690>
```



```
sns.distplot(d.x,kde=True )
sns.distplot(d.y,kde=True )
sns.distplot(d.z,kde=True )
```

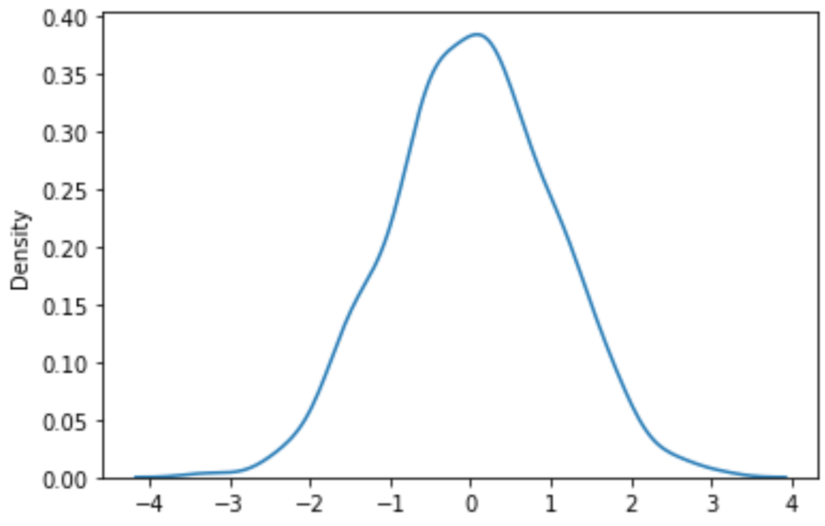
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fae0028e0d0>
```



kde plot

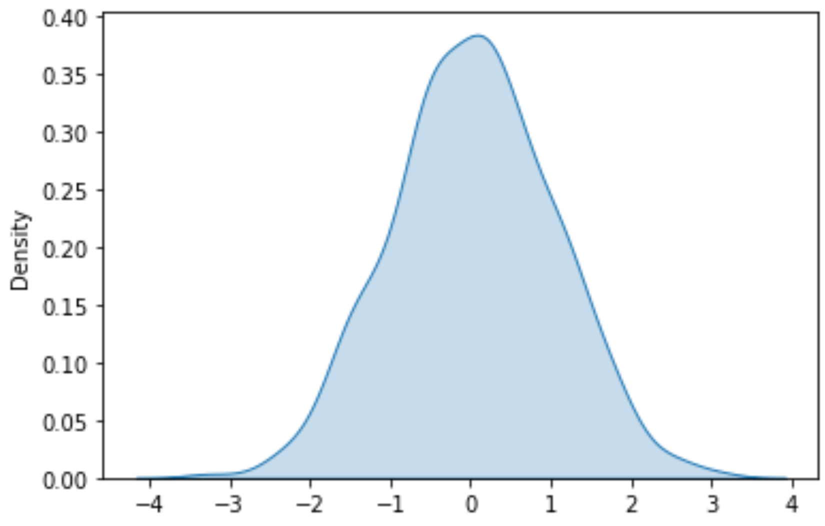
```
sns.kdeplot(x) #one line plot , no shade
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae000325d0>
```



```
sns.kdeplot(x,shade=True) # under curve
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae0009b090>
```

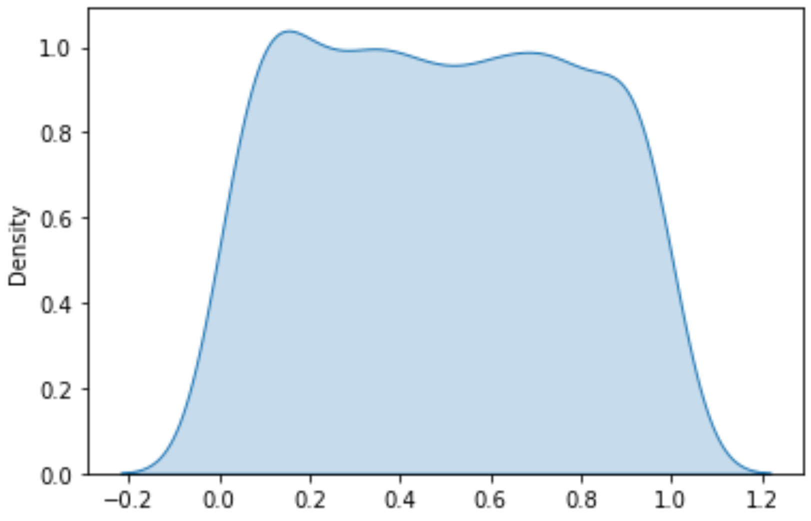


```
#super impose 2 plots together
y = np.random.uniform(size=1000)
y
```

```
array([1.92720952e-01, 4.60516047e-01, 6.99954373e-01, 6.51320683e-01,
       3.09724080e-01, 5.84727633e-01, 6.19135888e-01, 1.08908682e-01,
       8.86422435e-01, 5.24465660e-01, 1.64937072e-01, 6.46344596e-01,
       9.76349877e-01, 3.46973916e-01, 5.65978115e-02, 4.76703431e-01,
       4.64159977e-01, 5.37422179e-01, 1.21284741e-01, 5.05554499e-01,
       9.86405701e-01, 3.27868078e-01, 2.65695519e-01, 9.48464308e-01,
       9.09500972e-02, 2.45735317e-01, 2.39329792e-01, 2.32677447e-02,
       7.67535344e-01, 2.08673820e-01, 8.82995950e-01, 7.08567065e-01,
       8.76813195e-01. 2.19267120e-01. 3.54441656e-01. 4.89882455e-01.
```

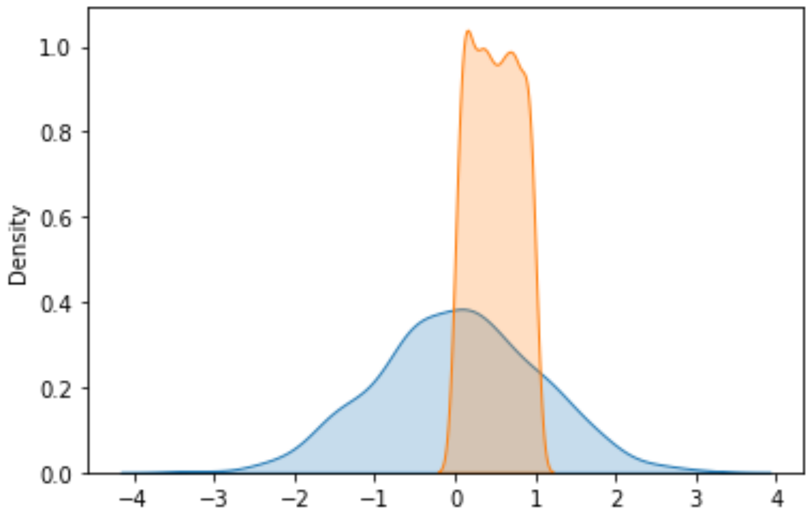
```
[illegible]
6.33184826e-01, 8.89640137e-01, 9.38804899e-01, 1.44105916e-01,
9.19302935e-01, 9.04211256e-01, 8.93625201e-01, 7.87039045e-01,
6.03777458e-01, 7.12763789e-02, 6.54172024e-01, 6.03425804e-01,
5.80346980e-01, 3.46397072e-01, 5.06732880e-01, 5.79359999e-01,
1.60792706e-01, 3.30158483e-01, 7.22212944e-01, 9.55284372e-01,
7.25298777e-01, 3.60494884e-01, 8.91231075e-01, 2.33868088e-02,
3.44435706e-01, 7.77994596e-01, 9.22262304e-01, 1.98630274e-01,
1.88093836e-01, 3.43155011e-01, 7.04777749e-01, 8.89098615e-01,
6.67965552e-01, 5.04529729e-01, 5.25141215e-01, 4.66785685e-01,
9.41069392e-01, 2.62822890e-02, 8.35291384e-01, 9.77664879e-02,
1.09395379e-01, 2.33453016e-01, 6.49436852e-01, 5.84173091e-01,
3.38673558e-01, 5.16244669e-01, 2.57841089e-01, 7.04308982e-01,
9.34234520e-01, 3.59795561e-01, 5.79386526e-01, 6.67919583e-01,
4.70339041e-01, 8.77432616e-01, 5.52671476e-01, 4.55194880e-02,
7.15852749e-01, 5.95245039e-01, 9.64662491e-01, 9.08711416e-01,
2.26455801e-01, 2.21261071e-01, 2.94096978e-01, 7.01122745e-01,
3.90689030e-01, 1.31510585e-01, 8.66106256e-01, 6.99305940e-01,
4.14361737e-01, 7.27104202e-01, 7.58098821e-01, 3.91773077e-01,
1.76417713e-01, 8.80355455e-01, 2.63364203e-01, 6.28531143e-03,
7.84423418e-01, 6.21292742e-01, 5.62642313e-01, 9.01117814e-01,
9.58892817e-01, 4.47832890e-01, 4.89338186e-01, 1.38582983e-01,
8.17037362e-01, 7.79430031e-01, 9.05429992e-01, 9.56197292e-01,
2.97664502e-02, 9.11257146e-01, 8.76058153e-01, 6.90462624e-01,
5.45025386e-01, 4.53755920e-01, 5.02511967e-01, 5.27086631e-01,
1.27687793e-01, 8.56088534e-01, 9.14200407e-01, 8.42622910e-02,
1.84313988e-01, 7.04945048e-02, 5.50541902e-01, 4.48051768e-01,
2.96847241e-02, 4.00331720e-01, 2.22347884e-01, 3.28923457e-01,
7.49984578e-01, 7.60124886e-01, 7.57879891e-02, 5.03875725e-01,
1.81397104e-01, 1.95983448e-01, 3.81044140e-01, 3.79469145e-01,
6.96369793e-01, 1.02504917e-01, 2.36963974e-02, 3.39266136e-01,
3.48754193e-01, 9.89151082e-01, 1.30676835e-01, 6.21404180e-01,
9.00680404e-01, 2.48025200e-01, 1.91808429e-01, 9.95846651e-01,
1.20935813e-01, 3.39237179e-01, 4.97804021e-01, 2.53679757e-01,
6.05994787e-01, 5.37923504e-01, 8.05558157e-01, 5.10234906e-01,
1.92456964e-01, 7.26726721e-01, 4.98275025e-01, 9.84898397e-01,
1.52533028e-01, 3.21026450e-01, 5.23979277e-01, 8.06948356e-01,
6.58027465e-02, 8.63442751e-01, 3.97715510e-01, 5.57418831e-01,
8.75522044e-01, 9.66797472e-01, 5.43664011e-01, 6.36862981e-01,
8.97894764e-01, 7.94182156e-01, 7.82027723e-01, 4.92709626e-01,
1.48787478e-01, 2.54962100e-01, 9.44602141e-01, 9.45968401e-01,
8.52689293e-03, 2.98284067e-01, 7.09280228e-01, 7.76556793e-01,
4.42458476e-01, 3.34684737e-01, 3.94064325e-01, 4.52644495e-01,
4.19831253e-01, 4.89332932e-02, 5.12153354e-01, 7.65436378e-01,
6.48353673e-01, 7.52140647e-01, 1.07370482e-01, 8.39049739e-01,
7.59270942e-03, 2.89194624e-01, 2.88754248e-01, 5.60090252e-01,
8.99526076e-01, 8.95644053e-02, 1.42006773e-01, 8.95487986e-01,
3.51217995e-01, 2.63191330e-01, 2.08294686e-01, 3.92075675e-01,
2.06903373e-01, 7.42804354e-01, 9.22859145e-01, 8.41822681e-01,
7.32535117e-01, 8.73167564e-02, 2.26357447e-01, 5.91874107e-01,
1.05385719e-01, 5.63958295e-01, 9.89069428e-01, 7.97372336e-01,
1.89216204e-01, 4.14577378e-01, 2.90786670e-01, 6.39800059e-01,
sns.kdeplot(y,shade=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadfffa0590>
```



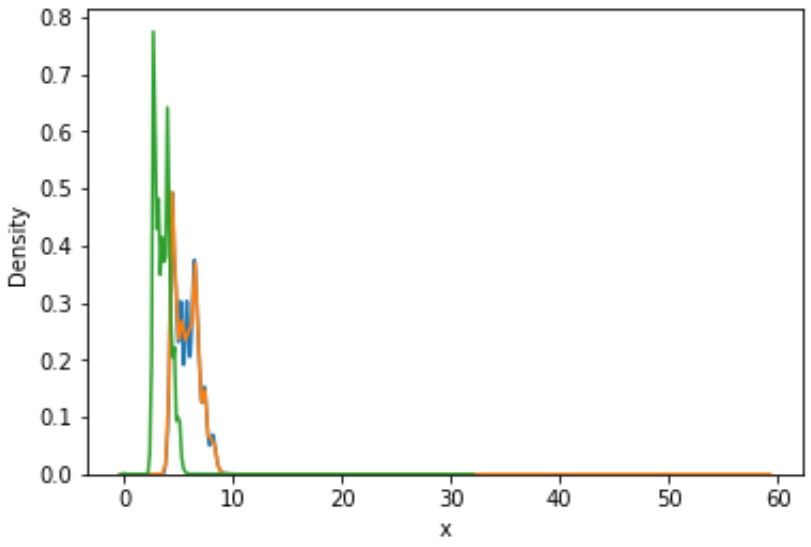
```
sns.kdeplot(x,shade=True)
sns.kdeplot(y,shade=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadfffb4a50>
```



```
sns.kdeplot(d.x) #blue
sns.kdeplot(d.y) #orange
sns.kdeplot(d.z) #green
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadfffb4e90>
```



```
x
```

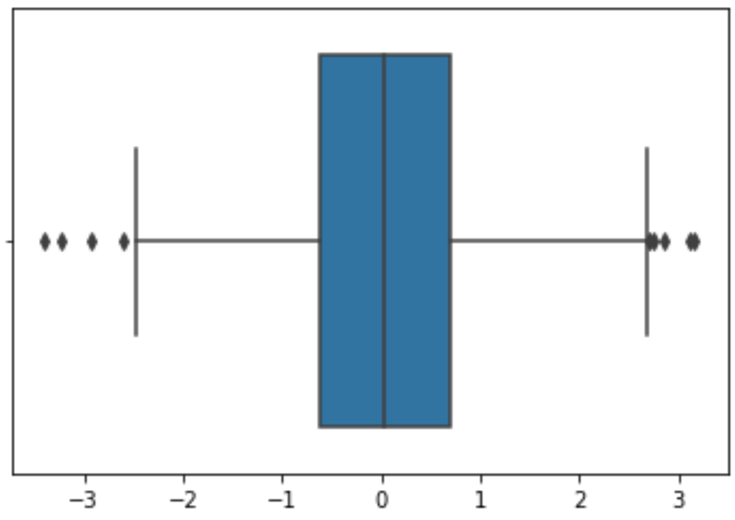
```
array([-6.45796143e-01,  5.33061022e-01,  1.11772294e+00, -2.79107374e
       -1.68896087e+00, -1.03554915e+00, -7.61460138e-01,  3.70915468e
        6.09154702e-01, -1.05599751e+00, -1.00761335e+00, -2.26041849e
        3.58477156e-01, -7.68739432e-03,  1.09356776e+00, -4.37733640e
       -7.25151913e-01,  3.86062262e-02,  2.86321063e+00,  1.95194182e
       -3.05090416e-01, -1.23011913e+00, -9.11323744e-01, -8.15470112e
       -6.67888421e-01,  2.67636423e-01, -1.28221037e+00, -1.41251082e
        1.23628889e-01, -1.97324576e+00, -9.32576877e-01,  9.92754702e
       -1.13515737e+00,  1.10885659e+00,  5.32515489e-02,  1.88697923e
       -3.76202059e-01,  1.29933124e+00,  1.66592102e-01, -5.99221966e
       -1.67216933e-01,  1.12284617e+00,  3.14733861e-01,  5.33980981e
       -1.53518998e+00, -1.40169314e+00, -5.46726168e-01,  1.36613321e
       -1.51070397e+00, -8.50353109e-01, -4.91364570e-03,  1.41384753e
        2.11908821e-01, -4.36990054e-01,  2.27967244e-01,  1.15229781e
        4.93675520e-01, -4.06220757e-01,  5.00182924e-01, -6.13800343e
        8.63601016e-01,  1.50869916e+00,  7.64553614e-01,  5.64572432e
       -1.01260083e+00,  9.37785390e-02, -1.57626505e+00,  1.03615614e
       -6.85115095e-01,  1.42156687e+00, -7.76237480e-01,  1.05825296e
        1.10358129e+00,  1.59629956e-01,  2.90972477e-01, -1.34712172e
       -1.18889106e+00, -1.99260772e-01, -1.06300335e+00,  5.78343330e
        9.08584018e-01,  5.53523635e-01,  4.89386167e-01,  8.13997967e
        3.46841730e-01, -1.38350684e+00,  3.25912038e-01,  2.16359108e
       -3.91604809e-01, -1.98539433e+00,  5.85279958e-01,  2.70801633e
        6.32076699e-01,  6.50420402e-01,  5.41023853e-02, -1.51273102e
       -2.04454050e-03, -6.23842458e-01,  4.91750834e-01,  9.97190037e
        7.24035349e-01,  2.12116841e-01,  6.97724455e-01,  7.23099163e
       -3.44103697e-01, -3.40343842e+00,  1.87765268e+00, -5.81833118e
       -2.30535187e-01,  1.35547233e+00,  7.63379955e-01,  2.94902103e
```

```
         1.22304635e+00, -4.68595357e-01,  5.23455412e-01,  8.72267336e
         9.10675605e-01, -3.12611527e-01, -1.74265659e+00,  1.48496286e
         2.94275348e-01, -9.27565572e-02, -1.13660623e+00,  1.87259397e
        -5.55584486e-01,  8.05997199e-01, -1.88874423e+00, -1.31377105e
        -5.01088579e-01,  1.06484877e+00,  3.88889151e-01, -1.14107342e
         4.22916442e-01,  8.54785733e-01,  8.80198407e-02, -5.81392029e
         1.92998294e+00, -6.09139618e-01,  1.67450407e-01, -5.76895701e
         3.88153685e-01,  1.57690929e-01, -2.47970500e+00, -4.65935401e
         1.54659318e+00,  1.24778549e+00,  5.35153378e-01, -3.42589460e
         2.75797533e-01, -8.70901999e-01, -2.04728362e+00, -1.54131438e
        -1.03010236e-01, -2.62709915e-01, -2.92062837e-02,  7.61103834e
         1.38179805e+00,  1.75337085e+00,  1.76138230e+00,  5.49321917e
         1.30493510e-01, -1.26013737e+00,  6.35297797e-01, -3.01582121e
         5.56682149e-02,  1.24608310e+00, -5.77261194e-01, -5.91181966e
         1.66060008e+00, -7.32077073e-01,  7.41494996e-02, -1.16608792e
        -1.34382646e+00, -2.11347722e+00,  3.36473686e-01, -4.97358214e
         1.44113418e+00,  6.80374599e-01,  4.49313642e-01,  6.35231530e
         8.44227603e-01, -4.85561878e-01,  1.30578348e+00,  6.20825550e
         1.12369732e+00,  2.10584301e-01,  9.71214573e-01,  1.45914512e
        -1.44572382e+00, -1.58522692e+00,  1.20387280e+00, -2.53104802e
         5.63845769e-01, -1.15703552e+00,  7.38605584e-01, -2.74855521e
         6.24442107e-01, -3.81412911e-01, -1.41628065e+00,  1.42654555e
         5.94156733e-02, -8.65370408e-01,  1.00753215e+00,  4.55287298e
        -5.50342058e-01, -1.09115242e-01,  2.02614902e+00, -5.43020480e
         1.77711473e+00, -3.94476159e-02, -1.79537463e+00, -3.64555631e
         1.51382804e+00, -4.69401948e-01, -2.91832809e-01, -6.19250285e
         1.15708928e-01, -1.00082771e+00,  5.60708717e-01, -4.58222496e
        -1.95103980e-01, -7.46520567e-01,  5.21542391e-01, -1.01200886e
        -2.61400293e+00,  3.81189529e-01, -9.54191312e-01, -1.21989837e
         8.40664597e-01,  1.48821757e+00,  1.81896294e+00, -7.17758201e
         2.85321268e-01, -1.76789271e+00,  9.80570941e-01, -1.91354463e
         7.03009413e-01, -3.17972664e-01,  1.18824728e+00,  6.44401345e
```

box plot

```
sns.boxplot(x)
```

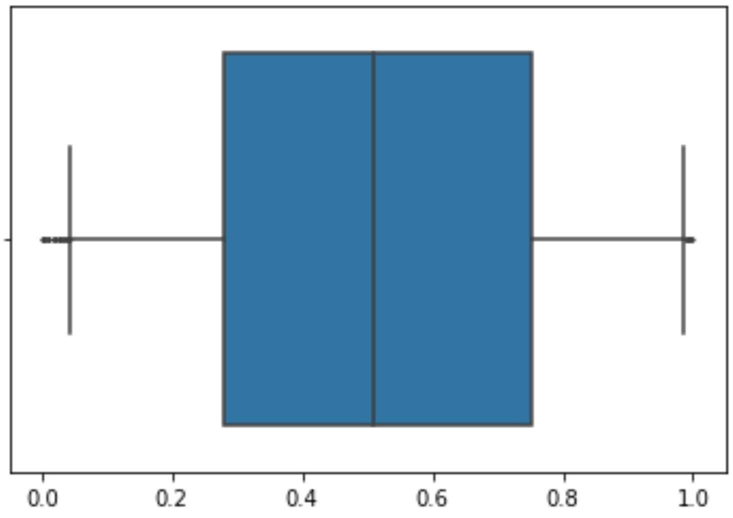
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fadffdb2b50>
```



```
x=np.random.uniform(size=1000)
x
```

```
array([4.10473693e-01, 5.79792550e-01, 7.87323651e-01, 7.71162949e-01,
       1.22592620e-01, 9.25707737e-01, 5.66721020e-01, 1.35692374e-01,
       7.34108440e-01, 3.91250033e-01, 2.64315209e-01, 9.59868191e-01,
       7.32746691e-01, 2.61484694e-01, 9.87178298e-01, 3.48123499e-01,
       1.97820435e-02, 4.88121742e-01, 8.92001132e-01, 4.03228265e-01,
       1.13045488e-01, 6.62793521e-01, 9.16026254e-01, 3.36152720e-01,
       3.99297778e-01, 9.79689526e-01, 3.41420832e-01, 6.74667625e-03,
       4.29616811e-03, 6.19127397e-01, 4.38472632e-01, 6.69146569e-01,
       6.95537168e-01, 1.61197107e-01, 1.03186039e-01, 5.64095927e-01,
       6.99592048e-01, 6.87077987e-01, 9.51134601e-02, 9.90442699e-01,
       2.33326368e-01, 4.31417752e-02, 4.00067508e-01, 3.22096376e-01,
       2.16828892e-02, 6.95613438e-02, 3.73178201e-01, 2.51073953e-01,
       5.82455723e-01, 3.11945122e-02, 4.23654630e-01, 3.18400654e-01,
       1.53060328e-02, 5.34820343e-01, 7.82034089e-01, 5.53027815e-01,
       4.04846268e-01, 6.08224324e-01, 4.32090906e-01, 4.12490299e-02,
       1.02461430e-01, 8.97631605e-01, 2.14669170e-02, 7.37240827e-01,
       6.14899167e-02, 4.15095963e-01, 5.11402927e-01, 3.56636909e-01,
       2.10039199e-01, 3.88788289e-01, 3.50277675e-01, 4.39402726e-01,
       4.24646993e-01, 9.66567594e-01, 1.17128178e-01, 6.14038636e-01,
       7.48814219e-02, 5.19149015e-01, 5.10038956e-02, 8.56068959e-01,
       3.78478145e-01, 5.24102941e-01, 6.71589306e-01, 4.64299189e-01,
       7.36762598e-01, 7.79292276e-01, 3.33444042e-01, 6.51064771e-01,
       7.51274781e-01, 7.20839749e-02, 8.18027222e-02, 8.48609494e-01,
       4.07193462e-01, 1.10709070e-02, 2.43287545e-01, 1.00238994e-01,
       5.24094321e-01, 8.37554217e-01, 9.93049453e-01, 5.77036559e-02,
       1.65827266e-01, 2.94118054e-01, 2.92889658e-01, 7.48525878e-01,
       3.57639683e-01, 5.88482446e-01, 5.56125836e-01, 6.04540968e-01,
3.37659085e-01, 3.88482446e-01, 3.36125836e-01, 6.04340968e-01,
8.83906688e-01, 9.52565477e-01, 5.99384885e-02, 6.31514102e-01,
7.86748352e-01, 1.18189754e-03, 8.83884711e-01, 5.71103556e-01,
1.38350166e-01, 7.55508724e-01, 7.32059294e-01, 4.68144669e-01,
4.05978828e-01, 9.11094024e-01, 3.07976582e-01, 3.17970046e-01,
7.51915613e-01, 1.30836256e-01, 9.83173770e-01, 9.84200247e-01,
3.79018319e-01, 3.96351205e-01, 8.67975544e-01, 3.96430197e-01,
3.22031772e-01, 2.73828075e-01, 4.23347168e-01, 6.33456932e-01,
2.53183509e-01, 4.76810188e-01, 9.51862278e-01, 4.32208992e-01,
9.44355566e-01, 9.70582078e-04, 6.48960286e-01, 5.78863573e-01,
5.71349077e-01, 5.55331217e-01, 6.90167960e-01, 5.83786151e-01,
3.78249124e-01, 2.56232900e-01, 2.41881819e-01, 3.81595724e-02,
2.42343293e-01, 8.96214316e-02, 3.24727827e-01, 7.08886875e-01,
8.35110413e-01, 9.70018583e-01, 4.34886608e-02, 4.52270224e-01,
5.75438292e-01, 8.22165416e-01, 8.55797462e-01, 3.37801566e-02,
9.74699111e-01, 7.36243075e-01, 4.20722992e-01, 7.05903285e-01,
2.65872997e-01, 4.54560236e-01, 3.74699412e-01, 1.71196123e-01,
5.59349158e-01, 1.46765814e-01, 3.73672171e-01, 8.02606936e-01,
5.13037096e-01, 5.44284461e-02, 5.02956830e-03, 3.42896839e-01,
2.28477257e-01, 3.22170001e-01, 9.47694938e-01, 3.69826702e-01,
4.51807697e-01, 6.01969675e-01, 1.93562545e-02, 8.31088221e-02,
6.48486368e-01, 1.54306687e-01, 8.45081774e-01, 6.92230787e-01,
2.85453987e-01, 5.02782675e-01, 9.32204107e-03, 7.38418435e-01,
7.01582410e-01, 6.46581584e-01, 4.85203598e-01, 7.46522858e-02,
2.15441855e-01, 6.33560263e-01, 8.99097499e-01, 5.35781274e-01,
9.54585598e-01, 4.84645672e-02, 7.97492760e-01, 6.13568091e-01,
5.31767599e-01, 4.51151329e-01, 6.12458301e-01, 8.81632512e-01,
2.16775928e-01, 4.83909921e-01, 3.72737657e-01, 8.68680455e-01,
2.36664926e-01, 2.19440714e-01, 1.37749763e-01, 3.03315449e-01,
2.02002734e-01, 4.36417390e-01, 8.91129682e-01, 7.23185693e-01,
2.79708594e-01, 7.74481257e-01, 1.79237883e-01, 3.16970153e-01,
2.47241489e-01, 9.07072695e-01, 6.92141226e-01, 7.63259805e-02,
3.53248468e-01, 3.17586959e-01, 6.71917923e-01, 4.31647132e-01,
9.86157101e-01, 7.25579052e-01, 4.92169032e-01, 4.40194672e-01,
sns.boxplot(x,whis=0.5,fliersize=1,orient='v')
```

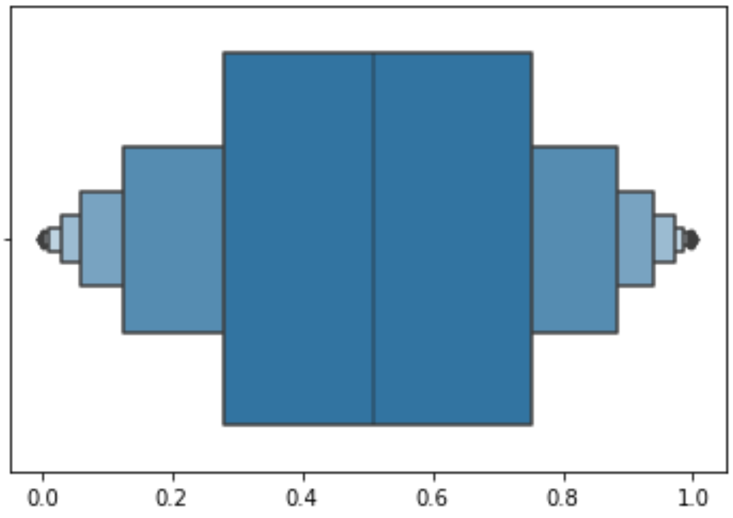
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326: UserWarn
  warnings.warn(single_var_warning.format("Vertical", "x"))
<matplotlib.axes._subplots.AxesSubplot at 0x7fadffcfb210>
```



Boxen Plot

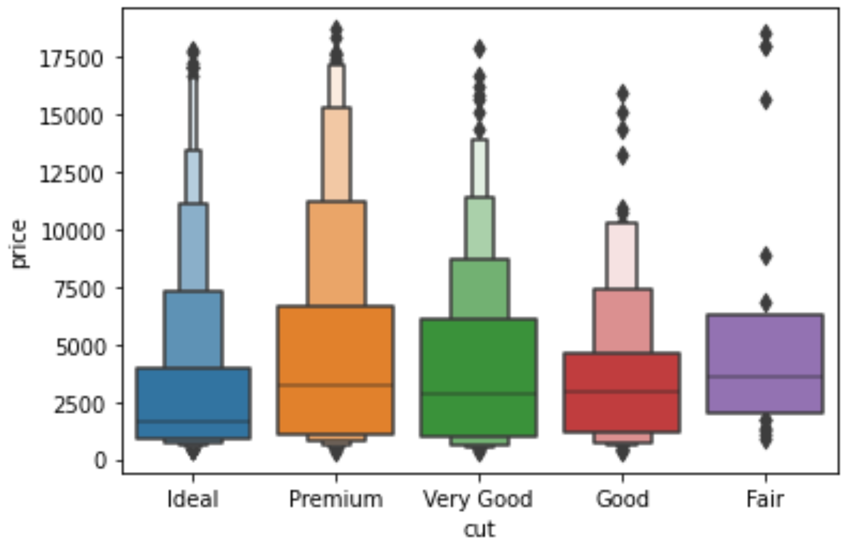
```
sns.boxenplot(x)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fadffd2a090>
```



```
sns.boxenplot(x = 'cut', y = 'price', data = d.sample(1000))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadffc7b210>
```



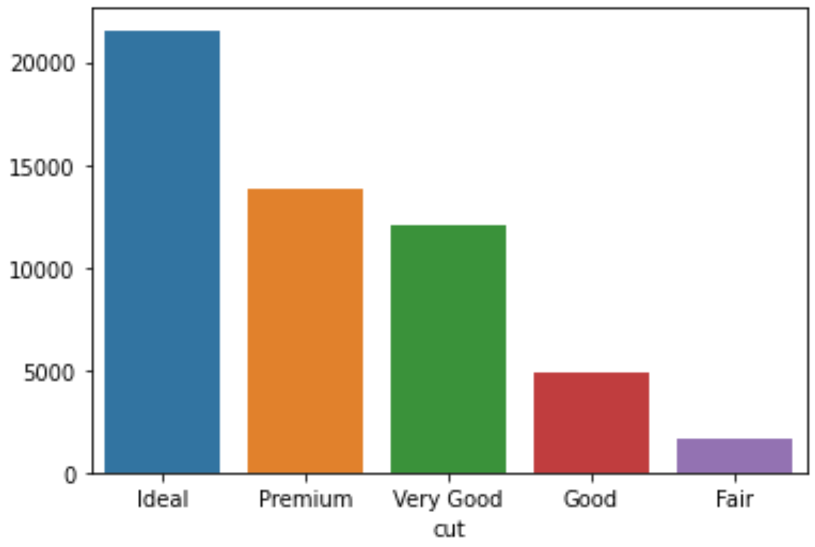
Bar Plot

```
c=d.groupby('cut')['cut'].count()
c
```

```
cut
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: cut, dtype: int64
```

```
sns.barplot(x=c.index,y=c.values)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadffc21850>
```



Joint Plot

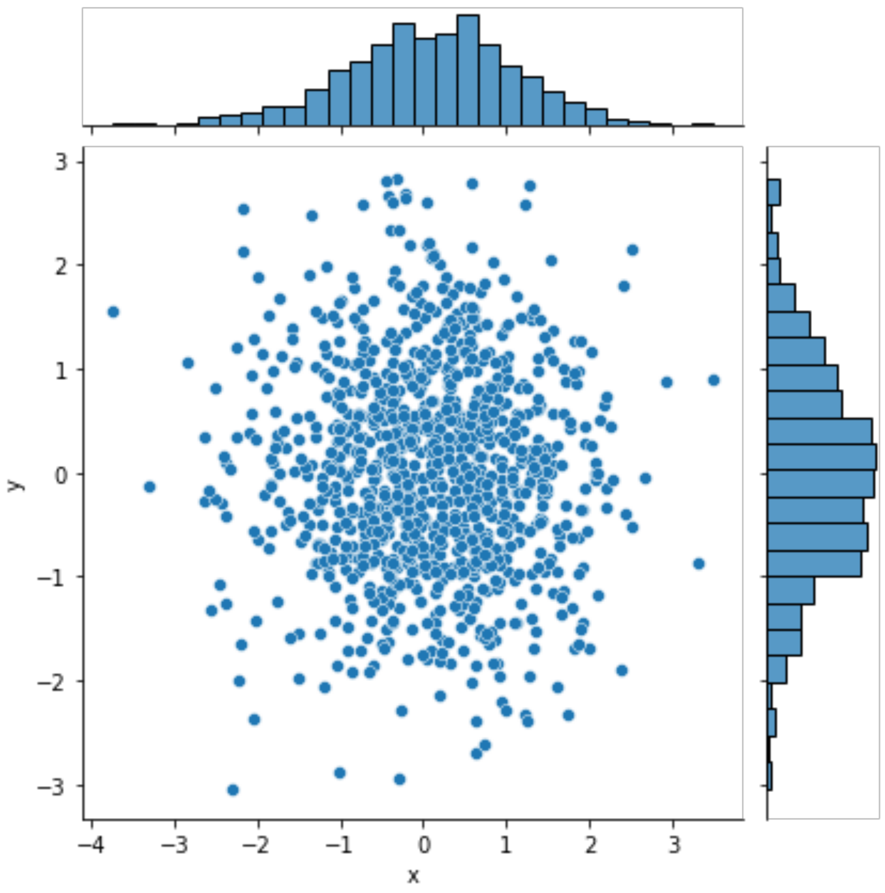
```
x=np.random.normal(size=1000)
y=np.random.normal(size=1000)
```

```
df=pd.DataFrame({'x': x, 'y':y})
df.head()
```

	x	y
0	0.485909	-0.094854
1	-0.419370	-0.548537
2	0.193044	0.539795
3	-0.306320	-1.303087
4	0.474361	-0.527140

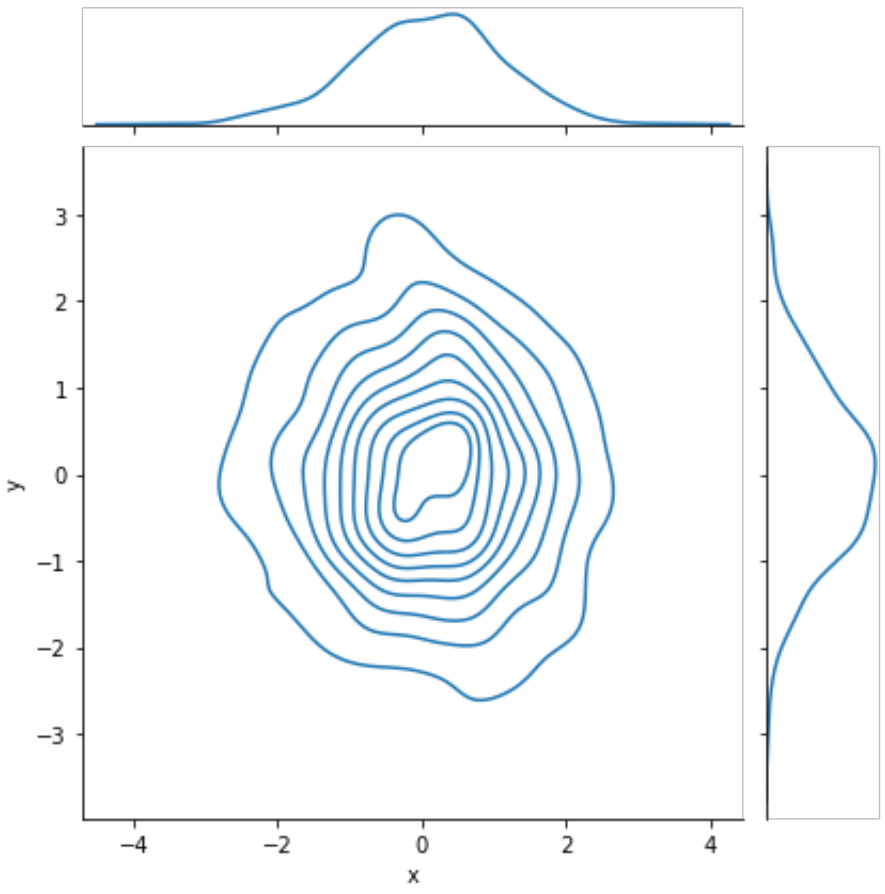
```
sns.jointplot(df.x,df.y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fae05a22150>
```



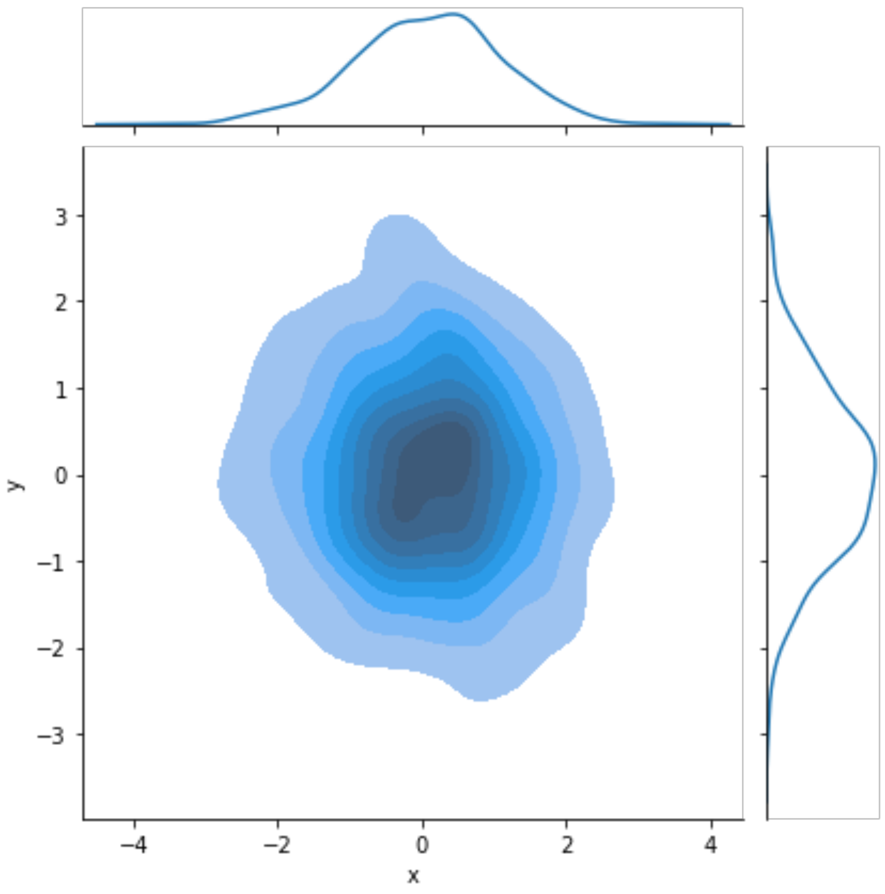
```
sns.jointplot('x','y',data=df,kind='kde',shade=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fadffbc9d50>
```



```
sns.jointplot('x','y',data=df,kind='kde',shade=True)
```

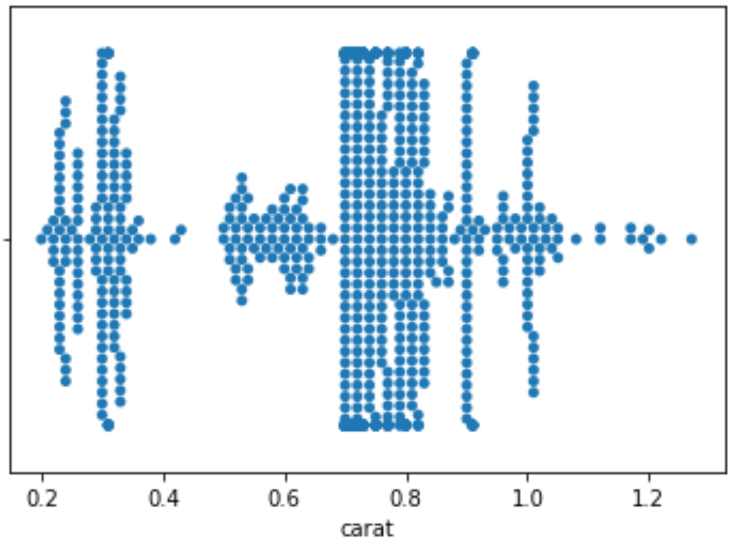
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fadff964550>
```



Swarm Plot

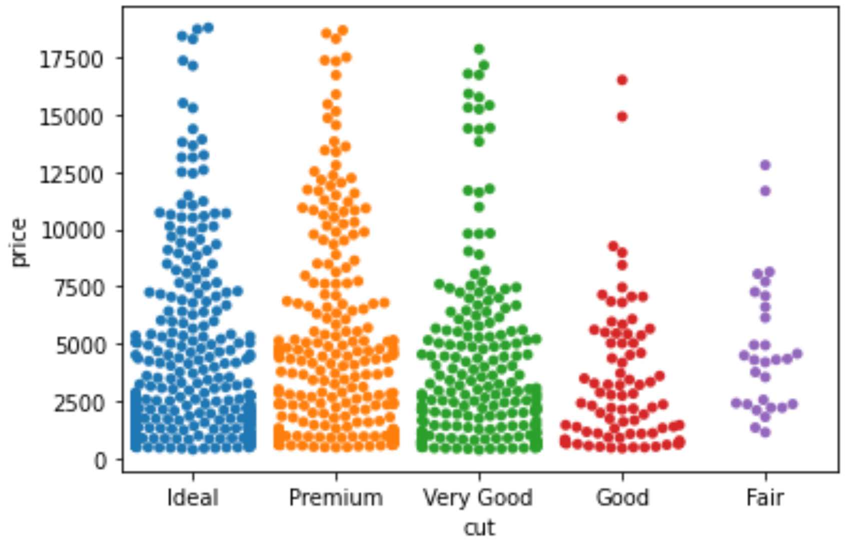
```
sns.swarmplot(d.head(1000).carat)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Futu
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fadff909ed0>
```



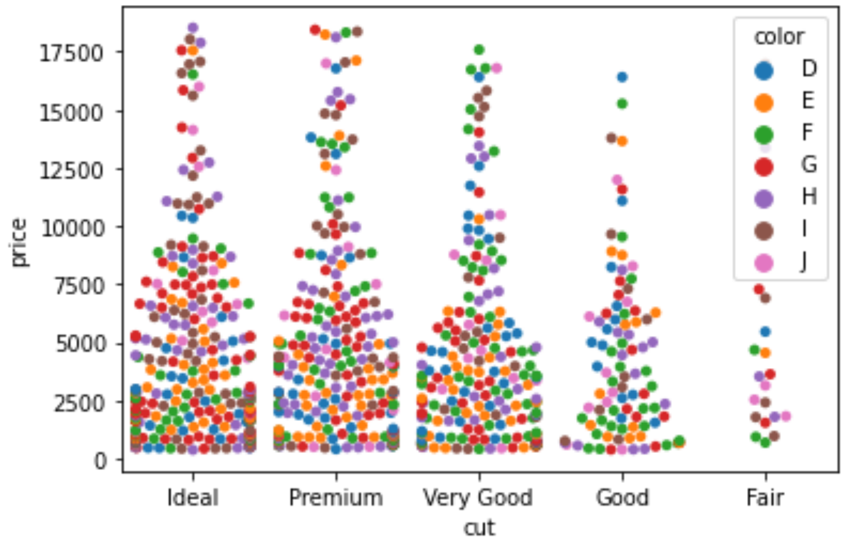
```
sns.swarmplot(x='cut',y='price',data=d.sample(1000))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fadff4b5990>
```



```
sns.swarmplot(x='cut',y='price',data=d.sample(1000),hue='color')
```

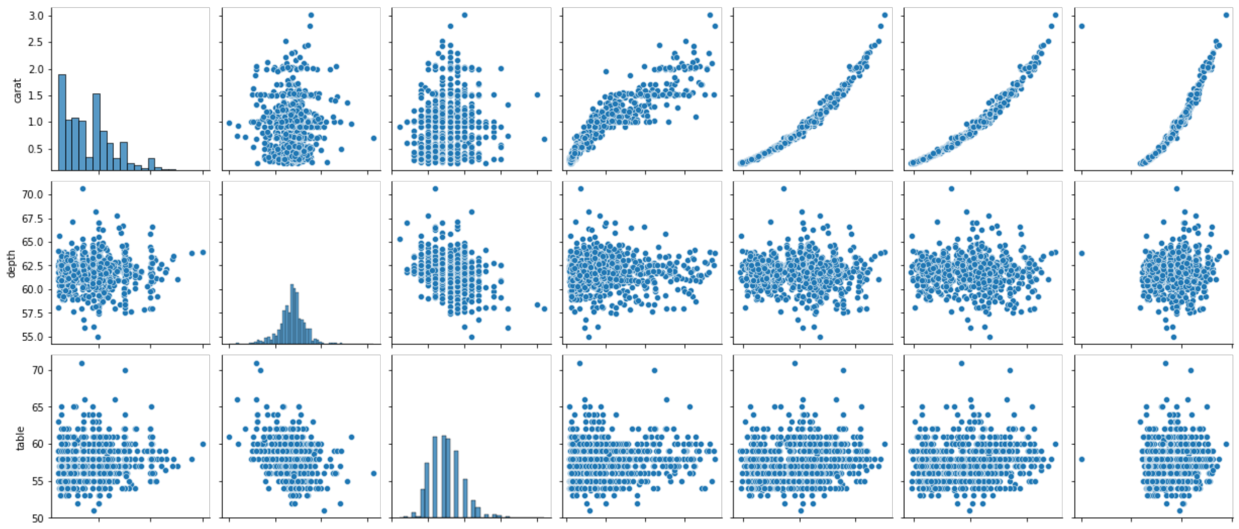
```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: Us
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fadff48e310>
```

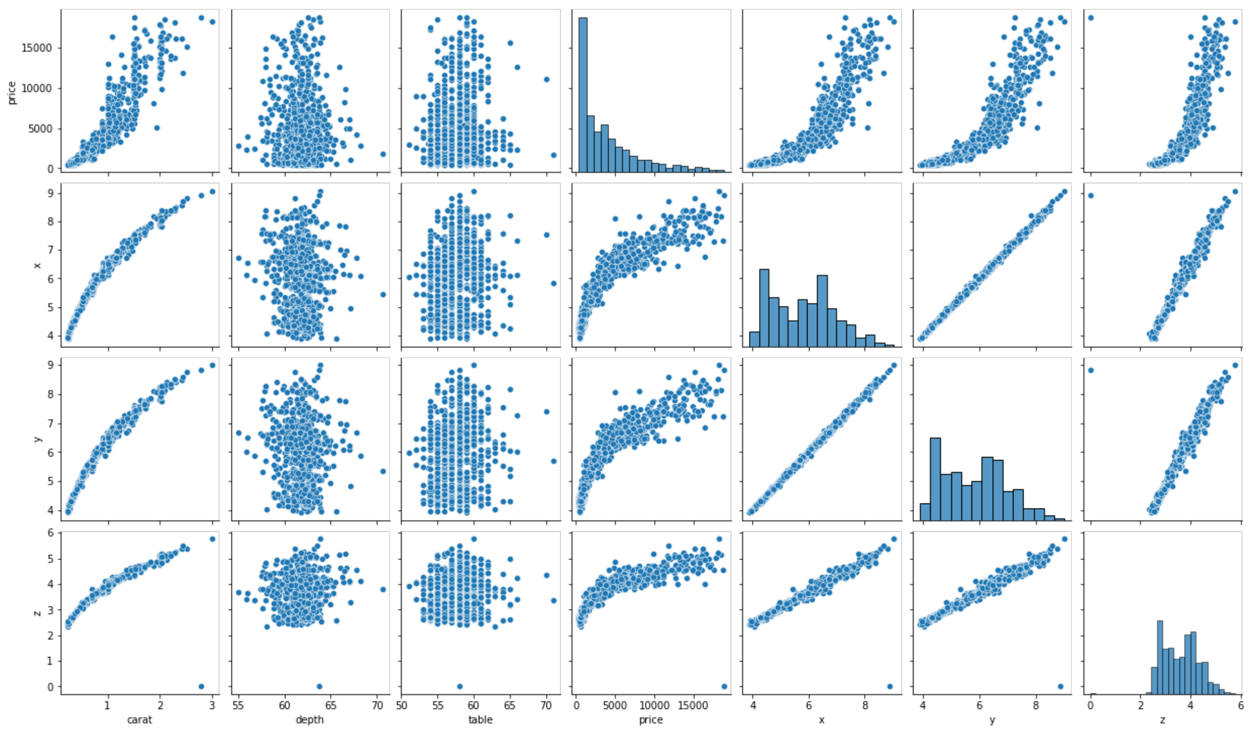


Pair Plot

```
sns.pairplot(d.sample(1000))
```

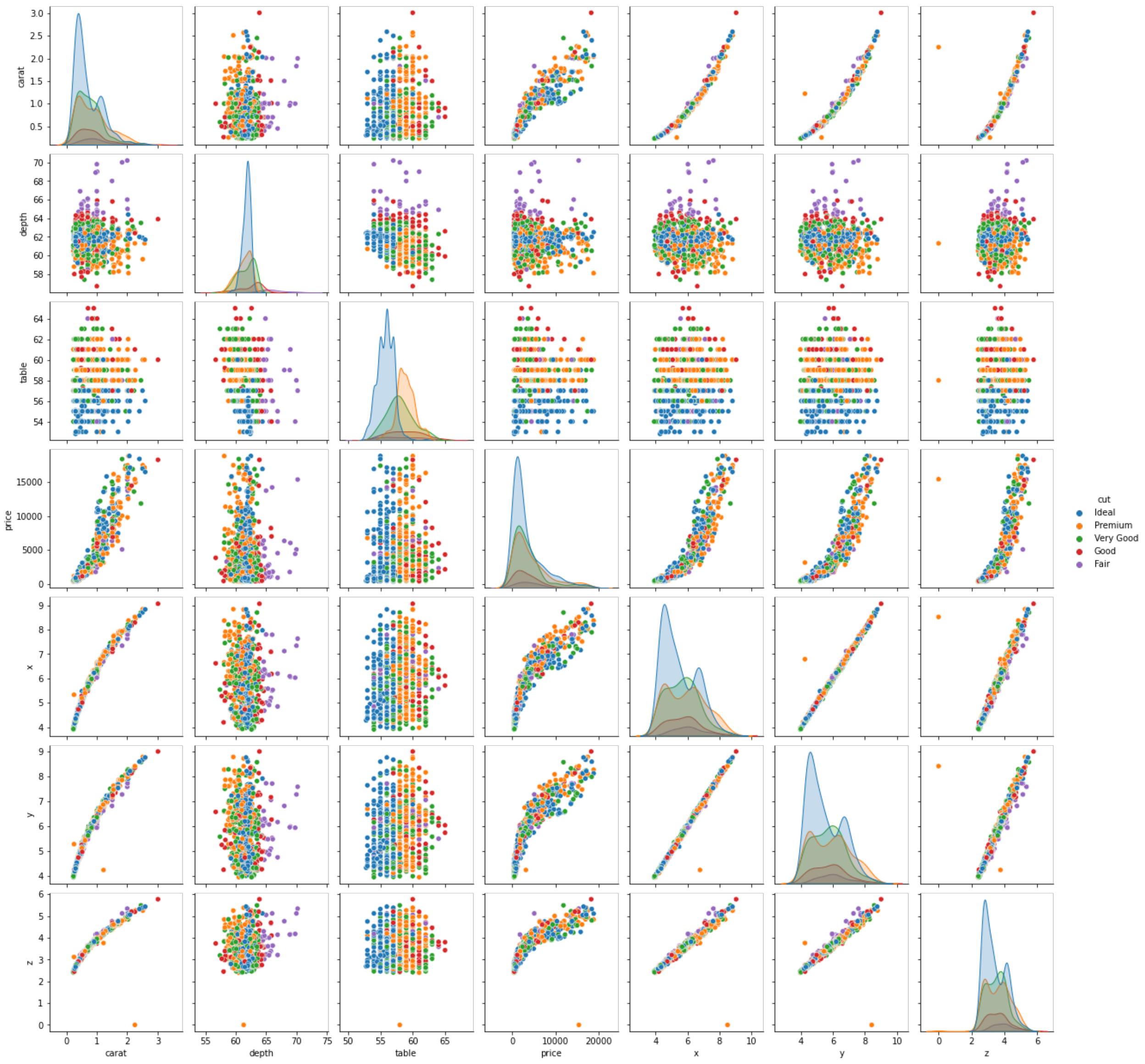
```
<seaborn.axisgrid.PairGrid at 0x7fadff3bafd0>
```





```
sns.pairplot(d.sample(1000),hue='cut')
```

```
<seaborn.axisgrid.PairGrid at 0x7fadfdda4090>
```

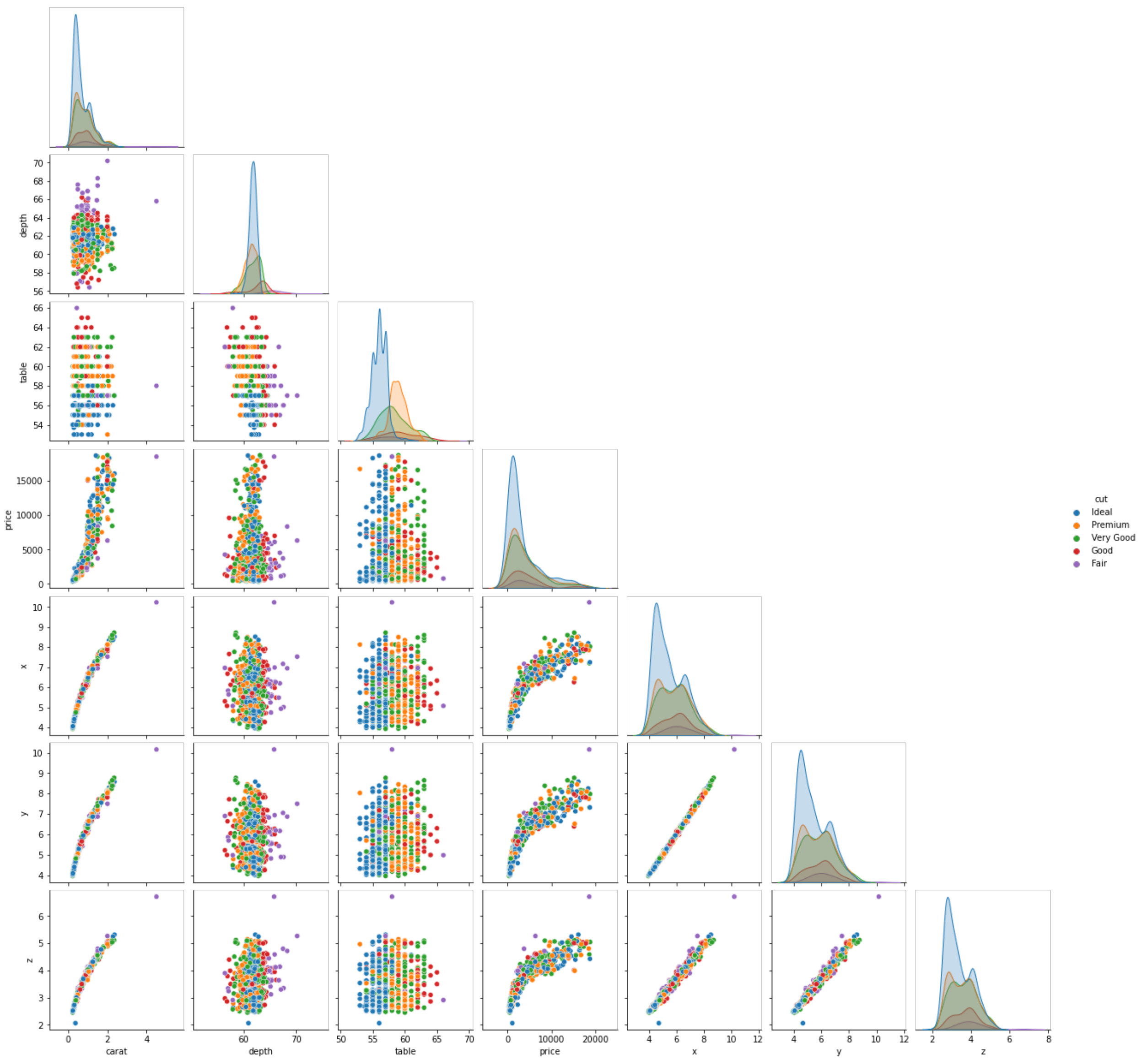


carat
depth
table
price
x
y
z
cut
Ideal
Premium
Very Good
Good
Fair

```
[illegible]
```

```
sns.pairplot(d.sample(1000),hue='cut',corner=True)
```

```
<seaborn.axisgrid.PairGrid at 0x7fadfc7fe350>
```



```python
x = np.random.normal(size = 1000)
x
```

```
array([-1.5408589 ,  0.31414744, -1.08454835,  0.54341336,  0.05841287
       -0.19381068, -0.15402474,  0.31299157,  0.44324955,  0.08122288
       -2.04985348, -0.22168664, -0.62487735,  1.39640519, -1.67823214
       -0.90548437,  1.05044179, -0.24304118, -0.38114778, -1.17426539
       -0.78123916, -0.53974043, -1.44900373,  0.96791077, -0.77634903
        0.50344983, -0.38247554, -0.19052075, -0.83202069, -0.71269474
        0.03946082,  0.62212585, -0.82654734, -0.5308725 ,  1.41967874
       -0.23845816, -1.1260212 , -1.08374572,  0.02362653, -2.37819398
       -0.92140196,  0.3511207 , -0.93504539,  1.204106  , -0.46278859
       -1.34255927,  1.43612726, -0.50176471,  0.04792627,  0.06436396
        0.12048998,  1.07453094,  1.37808676,  1.43876018,  0.55388844
       -0.24103904, -0.46482834,  1.19878822,  0.27366822,  0.38504734
       -1.27535121, -0.10960326, -1.03636951,  0.76068392, -1.68607403
       -0.85909036,  0.35718312, -2.29488383, -0.37387963,  0.62099369
        0.9661161 , -0.11905878,  0.59697153,  0.12666188, -0.70781358
        0.09203656, -0.28929562,  0.64529043,  0.9963595 , -0.180621
       -1.07645531, -0.81285739, -0.78994105,  0.9158719 ,  0.05402228
        1.45413072, -0.80288418, -2.42918531, -1.23105112, -0.56635535
        0.19350531,  0.5157937 ,  0.44657614, -0.80894103,  0.66536809
       -0.60593322,  1.95075983,  0.33298194,  1.18926907, -1.18976729
        1.78126274, -0.8804249 , -0.62299977, -1.19223383, -1.43917781
       -0.86023058, -0.3624674 ,  0.87574018,  0.10084246, -0.25325806
        1.24862026, -0.53863346,  0.55437207, -0.37969016, -0.39275166
        0.20018019,  1.82916255, -0.42554775,  0.05149978, -1.40849871
        0.36110905,  0.84181884,  0.67548338, -1.13150334,  1.12920479
        0.67420772,  1.05899444,  0.76034249, -0.5046482 ,  1.48159481
        0.41945773,  1.85156428, -0.34933514,  1.11143917,  0.35429724
       -0.01981235,  0.05702834,  0.81536858,  0.64081669, -0.19959129
       -0.42165024,  1.94704022, -1.02483609,  0.05782776,  1.19643802
       -0.53611427, -0.34856118,  1.09515151, -1.52266155,  0.86799633
       -0.59320628,  1.0902365 , -0.28066822, -1.49531286, -0.12707186
       -0.10582323,  0.74126045,  0.68099027,  0.19806501,  0.6854728
       -0.54702496, -0.44040621, -0.96206151,  1.77571973, -0.66234274
       -1.39793283,  0.3002495 , -0.66693897, -2.7748688 , -0.1547819
       -0.26987615, -0.48023806,  0.5361358 , -1.25725391,  0.91889902
        0.87609301,  1.8606888 ,  1.95380417, -1.14763285,  1.0020392
        0.48310954,  0.422956  ,  1.49744637, -1.23291409, -0.76692182
        0.41993437, -1.8849403 ,  0.97011516, -0.05453794, -0.55217346
       -0.67665474,  0.30568778, -0.7679713 , -0.66542954,  0.65724133
        0.50318147,  2.08145858,  0.92930261, -0.97723438, -0.30437803
       -0.37766873,  0.03304174,  1.62723823,  0.94800823,  0.51742982
       -2.25794487,  1.9962048 , -0.79499017, -0.17744284,  0.13728656
       -1.02836834,  0.24704924,  0.38143617, -0.55691057, -1.09423911
        0.17599412,  1.61903092,  0.23637054, -0.67332395,  0.29129584
        1.51528474,  1.17567963,  0.59819519,  0.45248986, -0.15076311
        0.5478045 , -0.58921236,  0.87622622,  0.6939038 , -1.91883456
       -0.45401721,  0.75834498,  0.38284632, -1.2547314 ,  0.09454121
        0.56736013, -0.4666509 , -0.58245923, -0.25210719, -1.04490143
        0.05870783,  0.93879845,  0.05800298, -1.15914985, -0.25897393
```

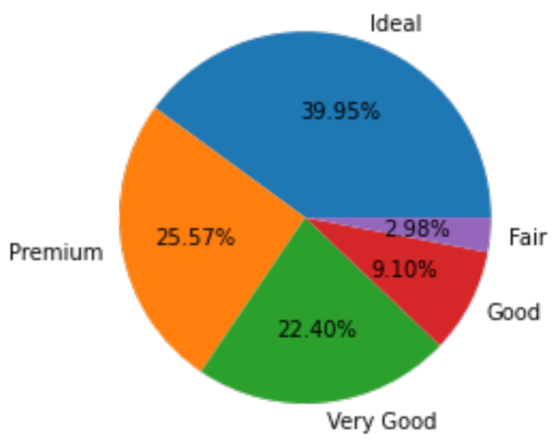
```
-0.73546025,  0.4399434 ,  0.57691282,  1.24260958,  0.692465
 0.37838903, -0.561859  ,  0.54537309,  1.04342445,  1.8133875
-0.41434894, -0.51614793,  1.33664763,  0.13786519,  0.3205777
-1.24011122, -0.31961426, -0.81388918, -0.71823451,  0.6629676
 1.15665267, -0.14233912, -1.07574514,  0.0962298 , -0.1886076
 0.89304573, -1.14729954, -1.26800228, -0.58623406, -0.3630975
 1.13934824,  1.09965935, -0.57654485, -0.11149861,  0.1335341
-1.92351461, -0.89195511,  0.6093047 , -0.4313865 ,  1.4783560
-0.38357394, -0.07246237,  0.23407139, -0.70949952, -1.0060309
 0.38889827,  1.00737495,  0.4183324 ,  0.18286599, -1.0293618
```

Pie Chart

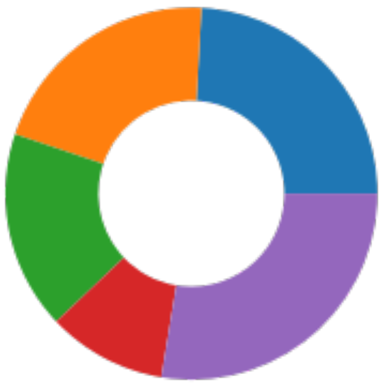
```
d.groupby('cut')['cut'].count()
```

```
cut
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: cut, dtype: int64
```

```
z_d = d.groupby('cut')['cut'].count()
plt.pie(z_d, labels = z_d.index, autopct = "%.2f%%")
plt.show()
```



```python
plt.pie(np.random.randint(0,10,5), wedgeprops = dict(width = 0.5))
plt.show()
```



```python
cmap = plt.get_cmap('Set1')
cmap
```

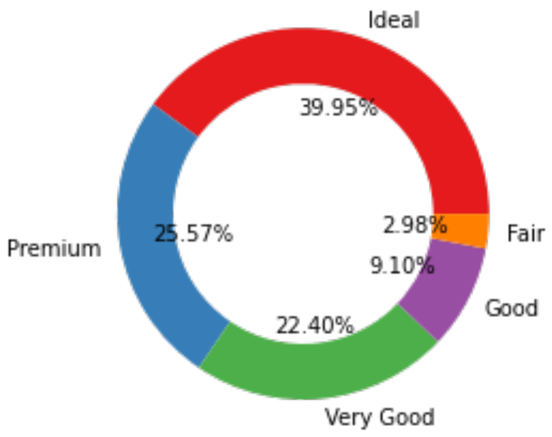
```
<matplotlib.colors.ListedColormap at 0x7fae234e8910>
```

```python
mycolor = cmap(np.arange(10))
mycolor
```

```
array([[0.89411765, 0.10196078, 0.10980392, 1.        ],
       [0.21568627, 0.49411765, 0.72156863, 1.        ],
       [0.30196078, 0.68627451, 0.29019608, 1.        ],
       [0.59607843, 0.30588235, 0.63921569, 1.        ],
       [1.        , 0.49803922, 0.        , 1.        ],
       [1.        , 1.        , 0.2       , 1.        ],
       [0.65098039, 0.3372549 , 0.15686275, 1.        ],
       [0.96862745, 0.50588235, 0.74901961, 1.        ],
       [0.6       , 0.6       , 0.6       , 1.        ],
       [0.6       , 0.6       , 0.6       , 1.        ]])
```

```
plt.pie(np.random.randint(0,10,5), wedgeprops = dict(width = 0.3), col
plt.show()
```

```
plt.pie(z_d, labels = z_d.index, autopct = "%.2f%%", wedgeprops = dict
plt.show()
```


Ideal
Premium
Fair
Good
Very Good