1. Deleting rows and columns from a dataframe

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
# reading an excel file
orders=pd.read_excel('C:/Users/raviy/Downloads/Superstore_Sales.xls ')
```

In [5]:
```
#viewing the top 5 records
orders.head()
```

Out[5]:

	Row ID	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	...
0	1	3	2010-10-13	Low	6	261.5400	0.04	Regular Air	-213.2500	38.94	...
1	49	293	2012-10-01	High	49	10123.0200	0.07	Delivery Truck	457.8100	208.16	...
2	50	293	2012-10-01	High	27	244.5700	0.01	Regular Air	46.7075	8.69	...
3	80	483	2011-07-10	High	30	4965.7595	0.08	Regular Air	1198.9710	195.99	...
4	85	515	2010-08-28	Not Specified	19	394.2700	0.08	Regular Air	30.9400	21.78	...

5 rows × 21 columns

In [5]:
```
# Dropping the Row ID column (axis=1 for column)
orders.drop('Row ID',axis=1,inplace=True)
```

In [6]:
```
#viewing the top 5 records after dropping the Row ID column
orders.head()
```

Out[6]:

	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost
0	3	2010-10-13	Low	6	261.5400	0.04	Regular Air	-213.2500	38.94	35.00
1	293	2012-10-01	High	49	10123.0200	0.07	Delivery Truck	457.8100	208.16	68.02
2	293	2012-10-01	High	27	244.5700	0.01	Regular Air	46.7075	8.69	2.99
3	483	2011-07-10	High	30	4965.7595	0.08	Regular Air	1198.9710	195.99	3.99
4	515	2010-08-28	Not Specified	19	394.2700	0.08	Regular Air	30.9400	21.78	5.94

In [7]:
```
# Dropping the multiple columns together
orders.drop(['Order Date','Order Priority'],axis=1,inplace=True)
```

In [8]:
```
orders.head()
```

Out[8]:

	Order ID	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Pro
0	3	6	261.5400	0.04	Regular Air	-213.2500	38.94	35.00	Muhammed MacIntyre	Nu
1	293	49	10123.0200	0.07	Delivery Truck	457.8100	208.16	68.02	Barry French	Nu
2	293	27	244.5700	0.01	Regular Air	46.7075	8.69	2.99	Barry French	Nu
3	483	30	4965.7595	0.08	Regular Air	1198.9710	195.99	3.99	Clay Rozendal	Nu
4	515	19	394.2700	0.08	Regular Air	30.9400	21.78	5.94	Carlos Soltero	Nu

In [9]:
```
# Drop Rows (axis=0 for rows)
orders.drop(1,axis=0,inplace=True)
```

In [10]:
```
orders.head()
```

Out[10]:

	Order ID	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Prov
0	3	6	261.5400	0.04	Regular Air	-213.2500	38.94	35.00	Muhammed MacIntyre	Nun
2	293	27	244.5700	0.01	Regular Air	46.7075	8.69	2.99	Barry French	Nun
3	483	30	4965.7595	0.08	Regular Air	1198.9710	195.99	3.99	Clay Rozendal	Nun
4	515	19	394.2700	0.08	Regular Air	30.9400	21.78	5.94	Carlos Soltero	Nun
5	515	21	146.6900	0.05	Regular Air	4.4300	6.64	4.95	Carlos Soltero	Nun

In [11]:
```
# Dropping Multiple Rows
orders.drop([2,3],axis =0,inplace=True)
```

In [12]:
```
orders.head()
```

Out[12]:

	Order ID	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Province
0	3	6	261.54	0.04	Regular Air	-213.2500	38.94	35.00	Muhammed MacIntyre	Nunavut
4	515	19	394.27	0.08	Regular Air	30.9400	21.78	5.94	Carlos Soltero	Nunavut
5	515	21	146.69	0.05	Regular Air	4.4300	6.64	4.95	Carlos Soltero	Nunavut
6	613	12	93.54	0.03	Regular Air	-54.0385	7.30	7.72	Carl Jackson	Nunavut
7	613	22	905.08	0.09	Regular Air	127.7000	42.76	6.22	Carl Jackson	Nunavut

Duplicate Values

In [13]:
```
raw_data = {
"city":["Faridabad","Delhi","Faridabad","Noida","Noida","Faridabad","Noida","D
"rank": ["1st","2nd","1st","2nd","1st","2nd","1st","2nd","1st"],
"score1":[44,48,39,41,38,44,38,53,61],
"score2":[67,63,55,70,64,77,45,66,72]
}
df=pd.DataFrame(raw_data,columns=["city","rank","score1","score2"])
df
```

Out[13]:

	city	rank	score1	score2
0	Faridabad	1st	44	67
1	Delhi	2nd	48	63
2	Faridabad	1st	39	55
3	Noida	2nd	41	70
4	Noida	1st	38	64
5	Faridabad	2nd	44	77
6	Noida	1st	38	45
7	Delhi	2nd	53	66
8	Delhi	1st	61	72

In [14]:
```
#check for duplicate data
df.duplicated()
# So, thereisno duplicate row in a dataframe
```

Out[14]:
```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
dtype: bool
```

In [15]:
```
#check for duplicate rows in city
df.duplicated(['city'])
#the first occurrences of data also treated as False
```

Out[15]:
```
0    False
1    False
2     True
3    False
4     True
5     True
6     True
7     True
8     True
dtype: bool
```

In [16]:
```
df.duplicated(['rank'])
#the first occurrences of data also treated as False
```

Out[16]:
```
0    False
1    False
2     True
3     True
4     True
5     True
6     True
7     True
8     True
dtype: bool
```

In [17]:
```
df.duplicated(['rank'],keep='last')
# the last occurances is treated as True, when we classify keep as last
```

Out[17]:
```
0     True
1     True
2     True
3     True
4     True
5     True
6     True
7    False
8    False
dtype: bool
```

In [18]:
```
#checking duplicate values on the basis of combinations
df.duplicated(['city','rank'])
```

Out[18]:
```
0    False
1    False
2     True
3    False
4    False
5    False
6     True
7     True
8    False
dtype: bool
```

In [19]:
```
df.drop_duplicates()
# No rows are dropped because there is no duplicate rows
```

Out[19]:

	city	rank	score1	score2
0	Faridabad	1st	44	67
1	Delhi	2nd	48	63
2	Faridabad	1st	39	55
3	Noida	2nd	41	70
4	Noida	1st	38	64
5	Faridabad	2nd	44	77
6	Noida	1st	38	45
7	Delhi	2nd	53	66
8	Delhi	1st	61	72

In [20]:
```
df.drop_duplicates(['city'])
#all the duplicate cities are dropped
```

Out[20]:

	city	rank	score1	score2
0	Faridabad	1st	44	67
1	Delhi	2nd	48	63
3	Noida	2nd	41	70

In [21]:
```
df.drop_duplicates(['city','rank'])
# all the duplicate rows are dropped on the basis of combination of city and r
```

Out[21]:

	city	rank	score1	score2
0	Faridabad	1st	44	67
1	Delhi	2nd	48	63
3	Noida	2nd	41	70
4	Noida	1st	38	64
5	Faridabad	2nd	44	77
8	Delhi	1st	61	72

In [22]:
```
orders.describe()
```

Out[22]:

	Order ID	Order Quantity	Sales	Discount	Profit	Unit Price	Sh
count	8396.000000	8396.000000	8396.000000	8396.000000	8396.000000	8396.000000	8396.
mean	29975.759409	25.568247	1774.686455	0.049670	181.046269	89.329013	12.
min	3.000000	1.000000	2.240000	0.000000	-14140.701600	0.990000	0.
25%	15033.750000	13.000000	143.067500	0.020000	-83.352500	6.480000	3.
50%	29860.500000	26.000000	449.295000	0.050000	-1.540000	20.990000	6.
75%	44609.000000	38.000000	1705.432500	0.080000	162.301500	85.990000	13.
max	59973.000000	50.000000	89061.050000	0.250000	27220.690000	6783.020000	164.
std	17254.888135	14.481312	3584.325298	0.031823	1196.810857	290.399696	17.

What is an outlier ?

- A data point which is significantly far away from other data points
- IQR (Inter Quartile Range)
- IQR = Q3 - Q1
- Lower Boundary = Q1 - (1.5 * IQR)
- Upper Boundary = Q3 + (1.5 * IQR)

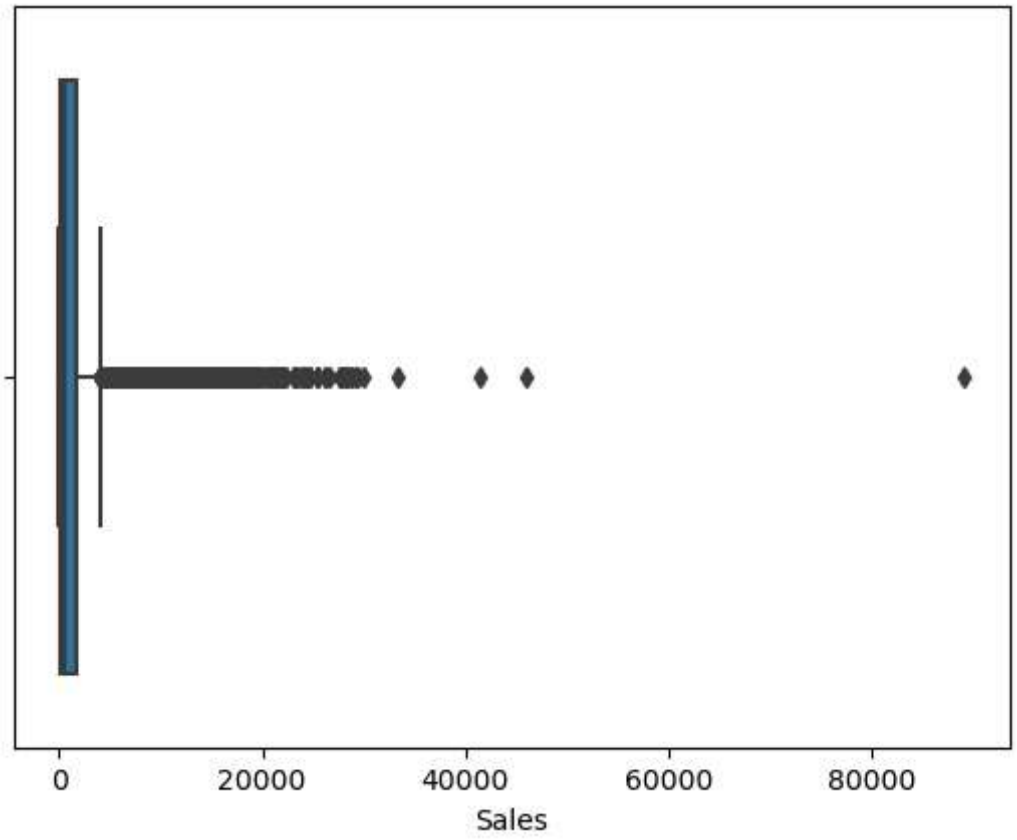
In [23]:
```
orders['Sales'].describe()
```

Out[23]:
```
count     8396.000000
mean      1774.686455
std       3584.325298
min          2.240000
25%        143.067500
50%        449.295000
75%       1705.432500
max      89061.050000
Name: Sales, dtype: float64
```

In [2]:
```
import seaborn as sns
```

In [26]:
```
sns.boxplot(x=orders['Sales'])
```

Out[26]: <Axes: xlabel='Sales'>



In [6]:
```
# Checking the shapeof the data
orders.shape
```

Out[6]: (8399, 21)

In [7]:
```
# Let's calculate first quartile,third_quartile and inter_quartile range
first_quartile = orders['Sales'].quantile(.25)
third_quartile = orders['Sales'].quantile(.75)
IQR = third_quartile - first_quartile
```

In [8]:
```
new_boundary = third_quartile + 3 * IQR
```

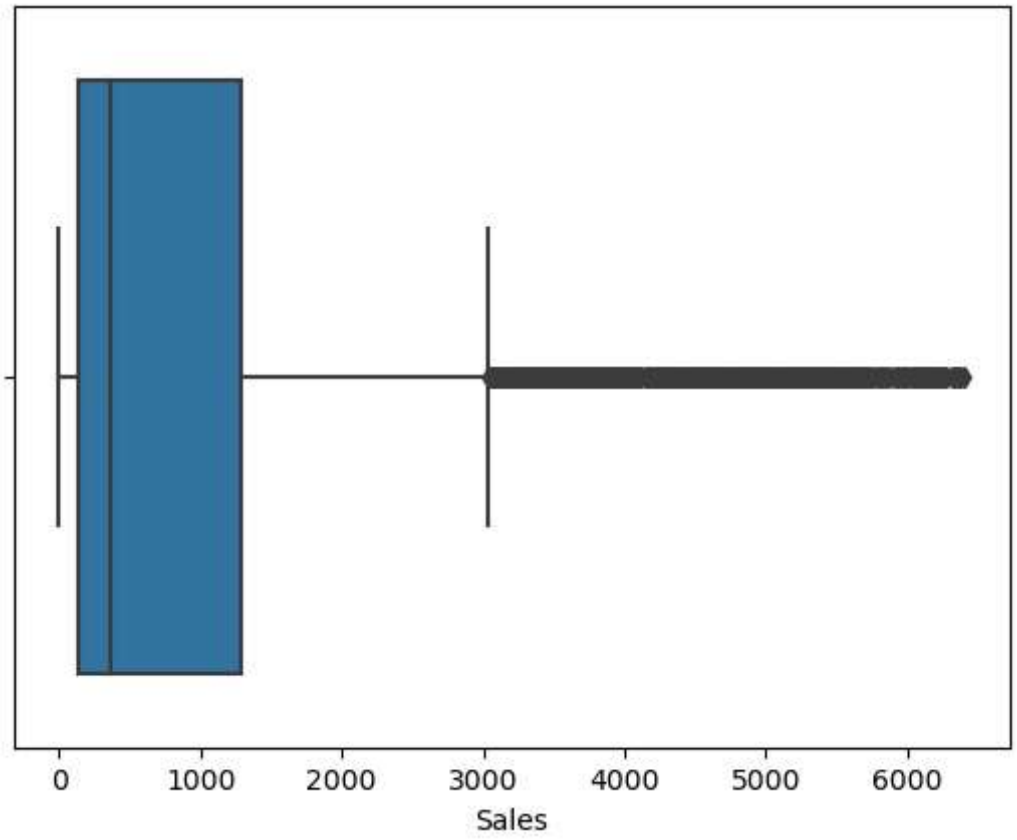
In [11]:
```
#dropped the outliers data
new_data = orders.drop(orders[orders['Sales']>new_boundary].index, axis = 0,
inplace=False)
```

In [12]:
```
# 12 rows are dropped
new_data.shape
```

Out[12]: (7817, 21)

In [15]:
```
sns.boxplot(x=new_data['Sales'])
```

Out[15]:
```
<Axes: xlabel='Sales'>
```



1.2 Outliers detection using IQR

In [17]:
```
# height of a persons
raw ={'name':['mohan','maria','deepak','kunal','piyush','avinash','lisa','smit
              'height':[1.2,2.3,4.9,5.1,5.2,5.4,5.5,5.5,5.6,5.6,5.8,5.9,6,6.1,
}
df =pd.DataFrame(raw)
```

In [18]:
```
df
```

Out[18]:

	name	height
0	mohan	1.2
1	maria	2.3
2	deepak	4.9
3	kunal	5.1
4	piyush	5.2
5	avinash	5.4
6	lisa	5.5
7	smita	5.5
8	tanu	5.6
9	khusboo	5.6
10	nishant	5.8
11	johnson	5.9
12	donald	6.0
13	rakesh	6.1
14	pritvi	6.2

In [19]:
```
df.describe()
```

Out[19]:

	height
count	15.000000
mean	5.086667
std	1.418181
min	1.200000
25%	5.150000
50%	5.500000
75%	5.850000
max	6.200000

In [20]:
```
# Calculating the Q1 ,Q3
Q1 = df.height.quantile(.25)
Q3 = df.height.quantile(.75)
Q1 , Q3
```

Out[20]:
```
(5.15, 5.85)
```

In [21]:
```
#Calculating IQR
IQR = Q3 - Q1
IQR
```

Out[21]:
```
0.6999999999999993
```

In [22]:
```
# Calculating IQR
lower_limit = Q1 - 1.5 * (IQR)
upper_limit = Q3 + 1.5 * (IQR)
lower_limit,upper_limit
```

Out[22]:
```
(4.100000000000001, 6.899999999999999)
```

In [23]:
```
#outliers
df[(df.height<lower_limit)|(df.height>upper_limit)]
```

Out[23]:

	name	height
0	mohan	1.2
1	maria	2.3

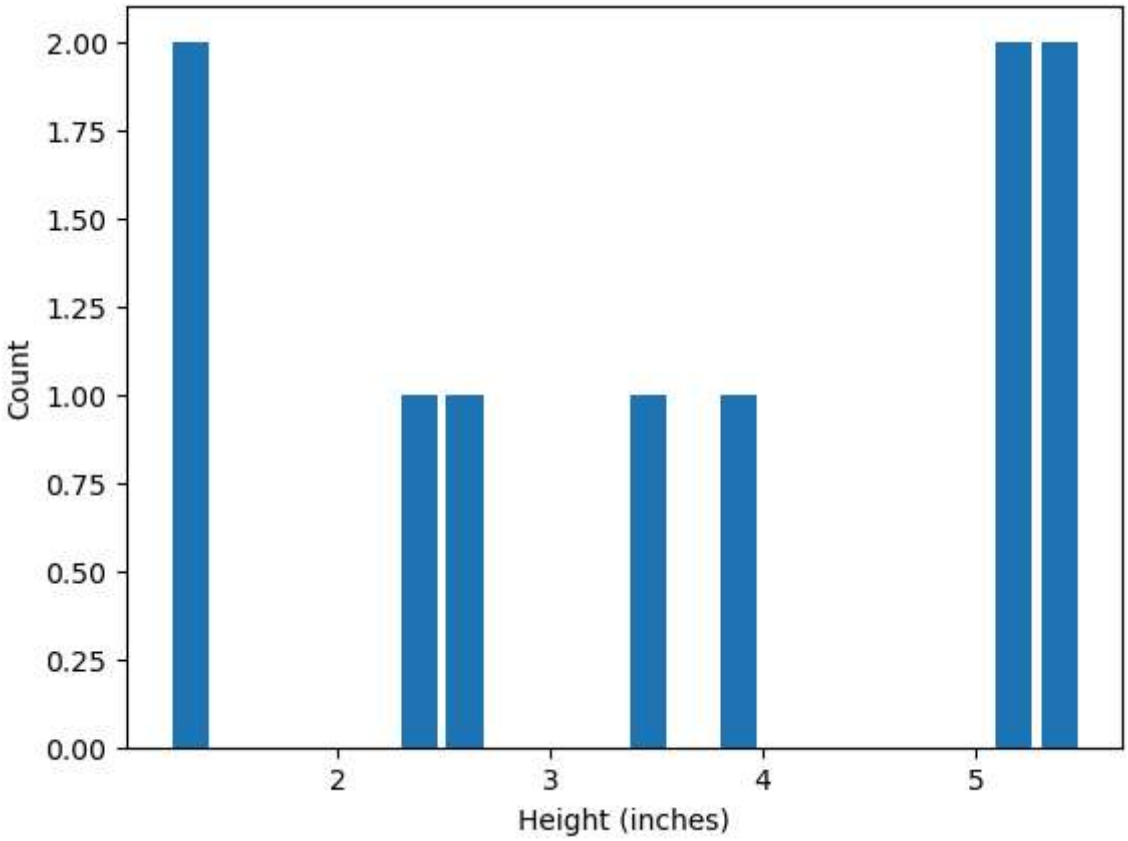
In [24]:
```
# data with no outliers
df_no_outliers = df[(df.height>lower_limit) & (df.height < upper_limit) ]
```

In [25]:
```
df_no_outliers
```

Out[25]:

	name	height
2	deepak	4.9
3	kunal	5.1
4	piyush	5.2
5	avinash	5.4
6	lisa	5.5
7	smita	5.5
8	tanu	5.6
9	khusboo	5.6
10	nishant	5.8
11	johnson	5.9
12	donald	6.0
13	rakesh	6.1
14	pritvi	6.2

In [40]:
```
plt.hist(df.height,bins=20,rwidth=0.8)
plt.xlabel('Height (inches)')
plt.ylabel('Count')
plt.show()
```



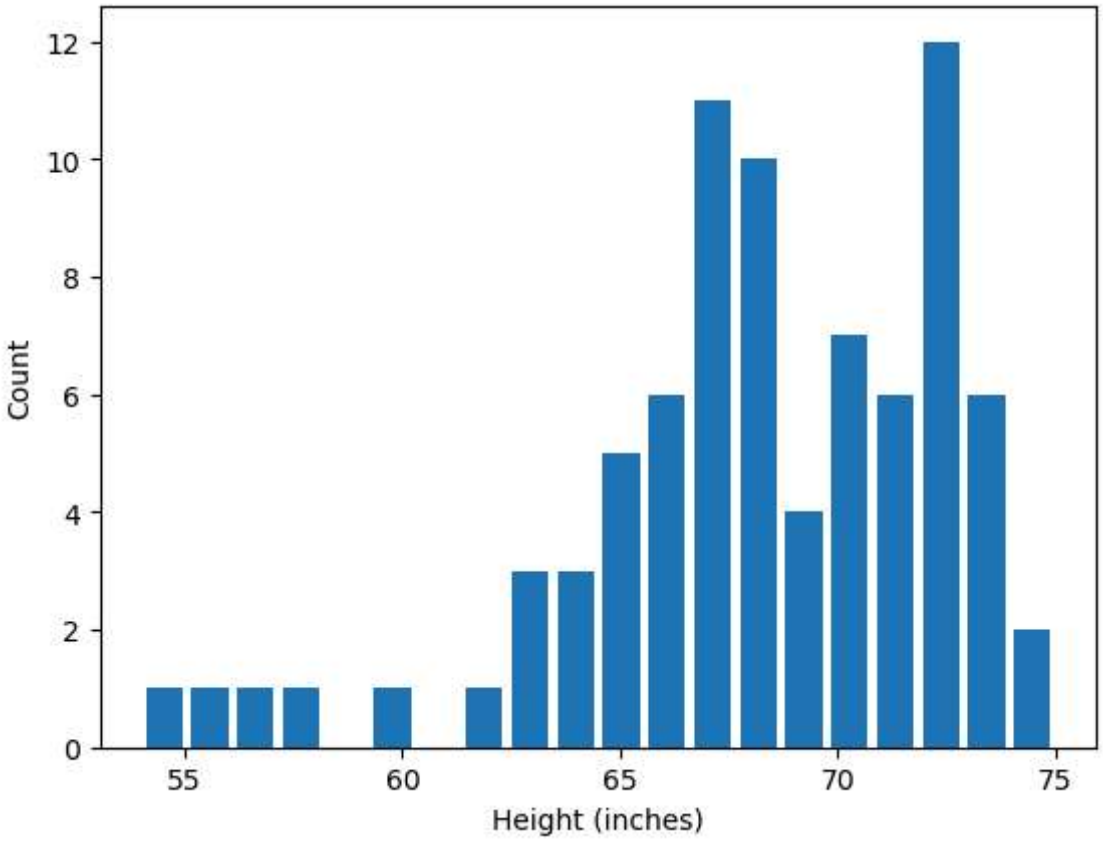
In [4]:
```
df =pd.read_csv("C:/Users/raviy/Downloads/height.csv")
```

In [5]:
```
df.sample(5)
```

Out[5]:

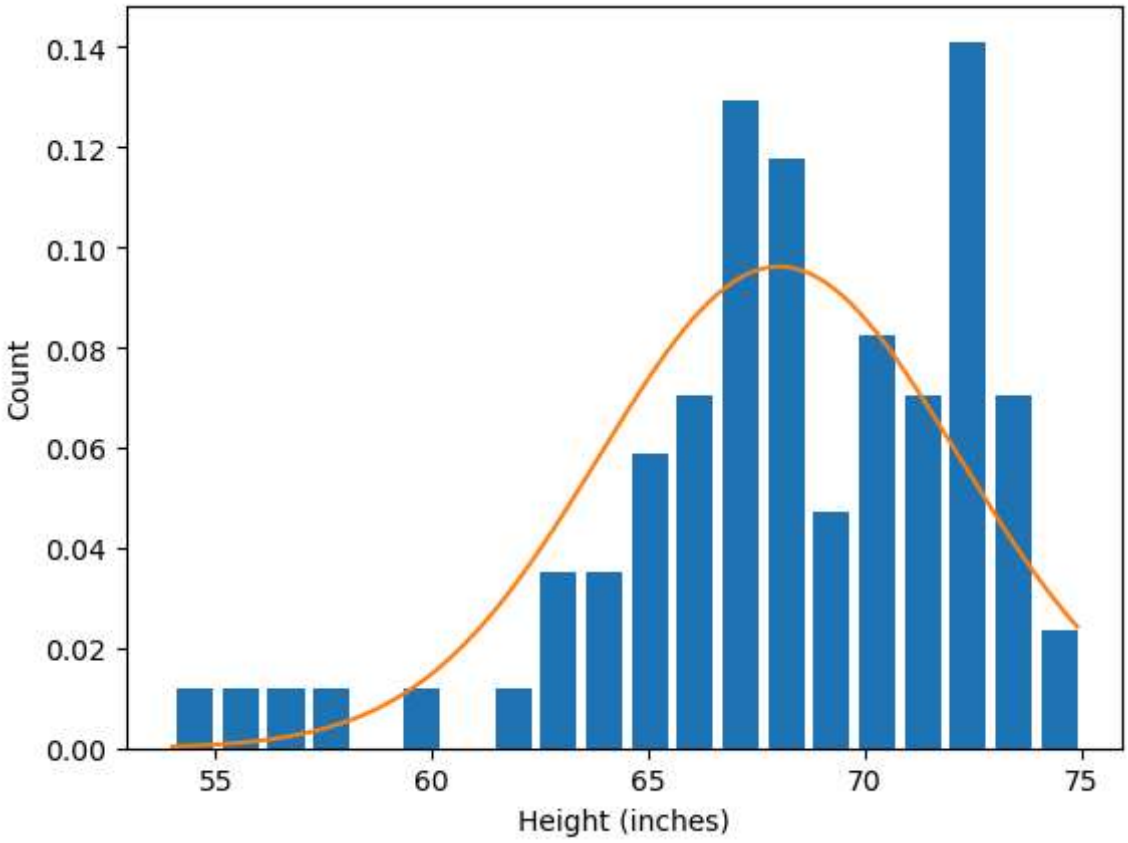
	gender	height	weight
48	0	66	135
75	0	70	200
7	1	64	125
61	0	73	220
14	1	66	125

In [6]:
```
plt.hist(df.height,bins=20,rwidth=0.8)
plt.xlabel('Height (inches)')
plt.ylabel('Count')
plt.show()
```



In [7]:
```
from scipy.stats import norm
import numpy as np
plt.hist(df.height,bins=20,rwidth=0.8,density=True)
plt.xlabel('Height (inches)')
plt.ylabel('Count')
rng= np.arange(df.height.min(),df.height.max(),0.1)
plt.plot(rng,norm.pdf(rng,df.height.mean(),df.height.std()))
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x19c3599ca60>]
```



In [8]:
```
#Calculating Mean
df.height.mean()
```

Out[8]:
```
68.01234567901234
```

In [9]:
```
#Calculating Standard Deviation
df.height.std()
```

Out[9]:
```
4.148776407449834
```

In [10]:
```
#Calculating upper_limit
upper_limit = df.height.mean() + 3 * df.height.std()
upper_limit
```

Out[10]:
```
80.45867490136185
```

In [11]:
```
#Calculating lower_limit
lower_limit = df.height.mean() - 3 * df.height.std()
lower_limit
```

Out[11]:
```
55.56601645666284
```

In [12]:
```
# checking the outliers
df[(df.height>upper_limit )| (df.height <lower_limit)]
```

Out[12]:

	gender	height	weight
5	1	54	120

In [13]:
```
df_no_outlier_std_dev = df[(df.height<upper_limit ) & (df.height >lower_limit)]
df_no_outlier_std_dev.shape
```

Out[13]:
```
(80, 3)
```

In [14]:
```
df.shape[0] - df_no_outlier_std_dev.shape[0]
```

Out[14]:
```
1
```

1.3 Outlier detection and removal using Z score

Z score indicates how many standard deviation away a datapoint is.

For example in our case mean is 66.37 and the standard deviation is 3.84
If a value of data point is 77.91, then Z score for that is 3 because it is 3 standard

deviation away (77.91 = 66.37 + 3 * 3.84)

Score is refer to data point below

In [15]:
```
df['zscore'] = (df.height - df.height.mean())/df.height.std()
```

In [16]:
```
df.head(5)
```

Out[16]:

	gender	height	weight	zscore
0	0	72	155	0.961164
1	0	67	145	-0.244011
2	0	65	125	-0.726081
3	1	67	120	-0.244011
4	1	63	105	-1.208150

In [17]:
```
#outliers
df[(df['zscore']>3) | (df['zscore']<-3)]
```

Out[17]:

	gender	height	weight	zscore
5	1	54	120	-3.377465

In [18]:
```
#removing outliers as per Z score
df_no_outliers = df[(df.zscore <3) & (df.zscore > -3)]
df_no_outliers.head()
```

Out[18]:

	gender	height	weight	zscore
0	0	72	155	0.961164
1	0	67	145	-0.244011
2	0	65	125	-0.726081
3	1	67	120	-0.244011
4	1	63	105	-1.208150

In [19]:
```
#checkig the number of ouliers
df.shape[0] - df_no_outliers.shape[0]
```

Out[19]:
```
1
```

In [20]:
```
#Load the dataset
df=pd.read_csv('C:/Users/raviy/Downloads/netflix_movies_dataset.csv')
```

In [21]:
```
#preview of dataset
df.head()
```

Out[21]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
0	10000	Movie	Movie_0	Director_408	Actor_985, Actor_1133	United States	04/07/2014	2015	6.5
1	10001	Movie	Movie_1	Director_119	Actor_888, Actor_1289	United States	09/08/2016	2012	7.1
2	10002	Movie	Movie_2	Director_398	Actor_223, Actor_1016	United Kingdom	05/10/2014	2008	6.8
3	10003	Movie	Movie_3	Director_235	Actor_538, Actor_1984	Canada	12/24/2020	2011	4.2
4	10004	Movie	Movie_4	Director_231	Actor_798, Actor_1085	India	02/26/2019	2015	7.1

In [22]:
```
df['genres']
```

Out[22]:
```
0         Romance
1           Drama
2        Thriller
3          Comedy
4          Comedy
           ...
15995     Romance
15996      Action
15997      Action
15998      Action
15999       Drama
Name: genres, Length: 16000, dtype: object
```

In [23]:
```
from collections import Counter
```

In [24]:
```
Counter(df.genres)
```

Out[24]:
```
Counter({'Romance': 3225,
         'Drama': 3184,
         'Thriller': 3151,
         'Comedy': 3192,
         'Action': 3248})
```

In [26]:
```
# to get the frequency
df.genres.value_counts()
```

Out[26]:
```
genres
Action      3248
Romance     3225
Comedy      3192
Drama       3184
Thriller    3151
Name: count, dtype: int64
```

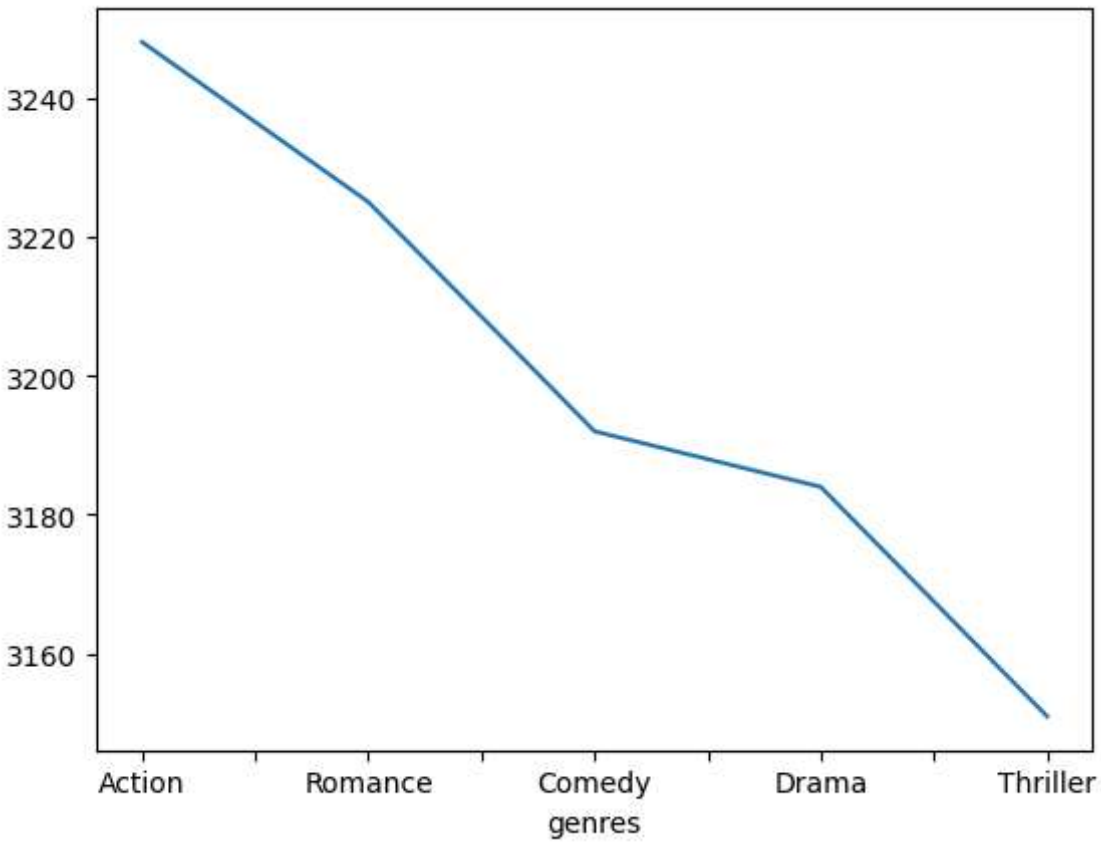
In [27]:
```
#checking the type
type(df.genres.value_counts())
```

Out[27]:
```
pandas.core.series.Series
```

In [29]:
```
gc = df['genres'].value_counts()
```

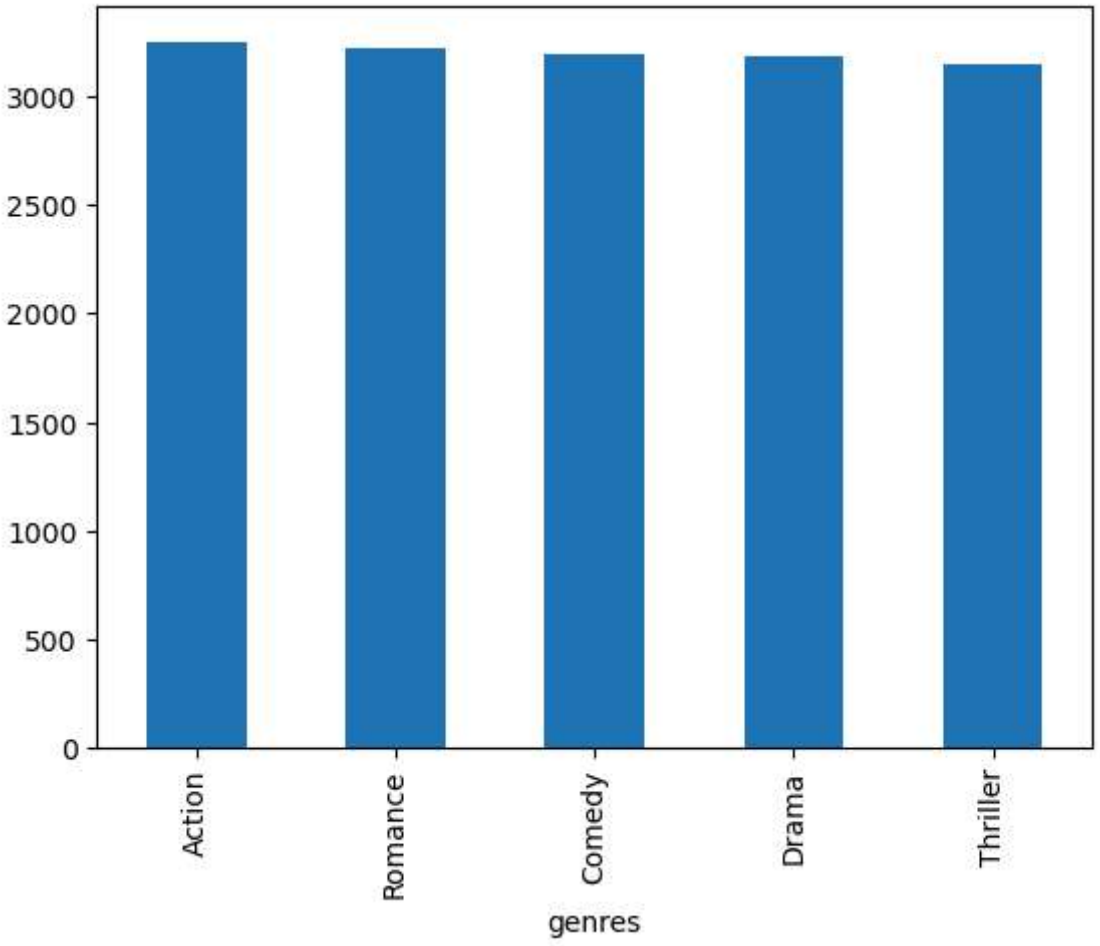
In [30]:
```
gc.plot()
```

Out[30]:
```
<Axes: xlabel='genres'>
```



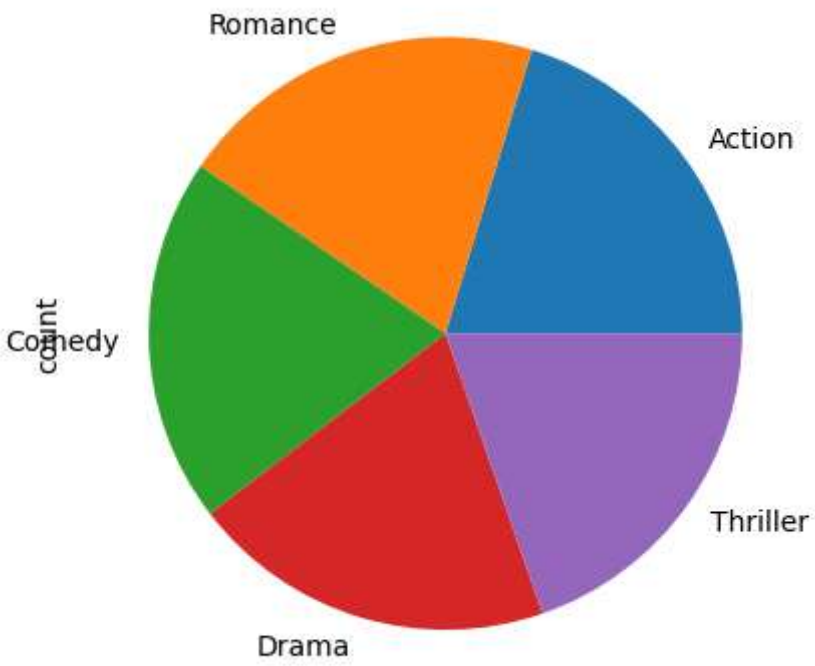
In [31]:
```
#Visualizing through a bar chart
gc.plot(kind='bar')
```

Out[31]: <Axes: xlabel='genres'>



In [32]:
```
#Visualizing through pie chart
gc.plot(kind='pie')
```

Out[32]: <Axes: ylabel='count'>



In [33]:
```
titanic_df = pd.read_csv('C:/Users/raviy/Downloads/titanic.csv')
```

In [34]:
```
titanic_df.info()
```

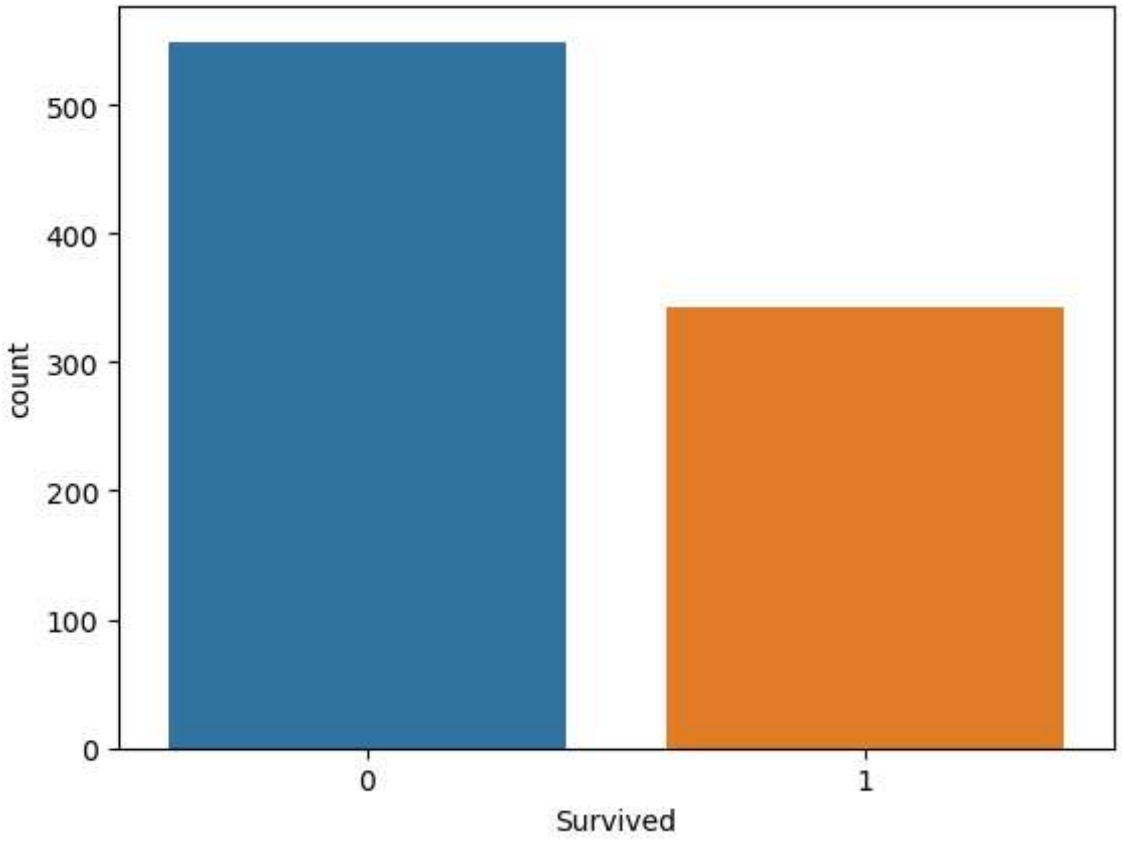
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [35]:
```
#checking top 5 recordsof the dataset
titanic_df.head()
```

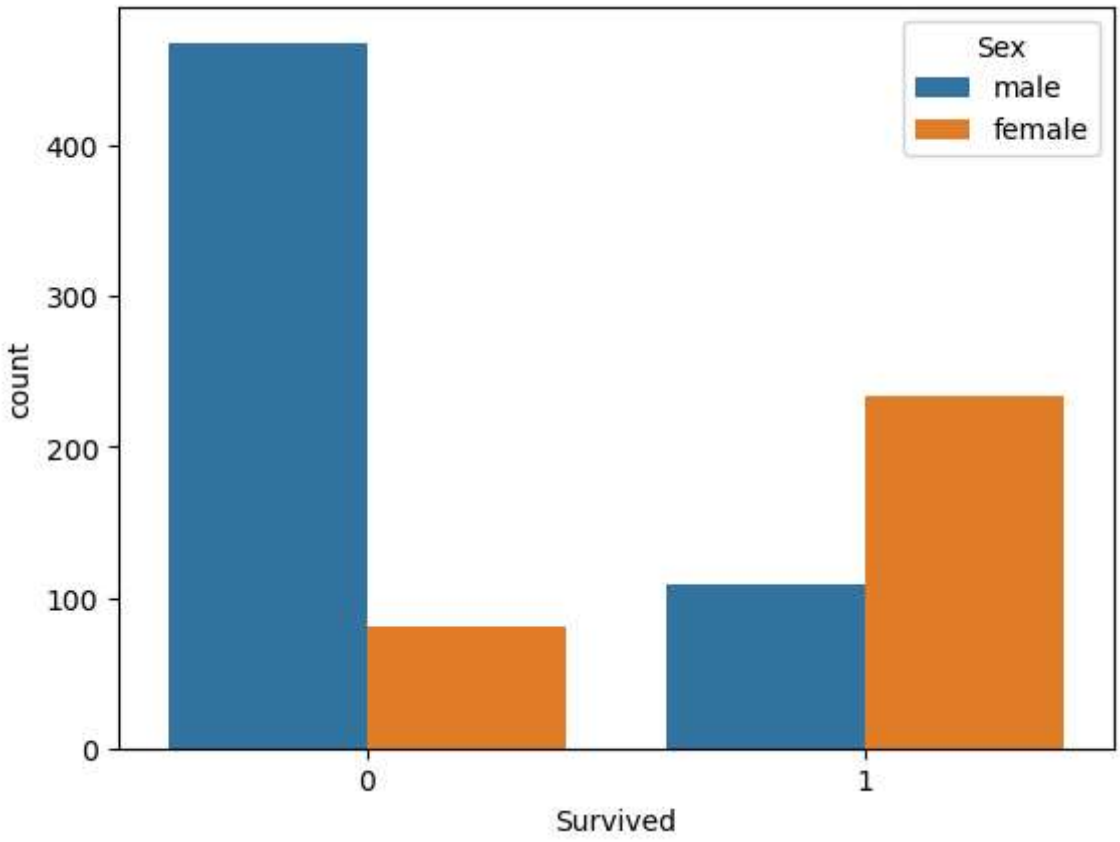
Out[35]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

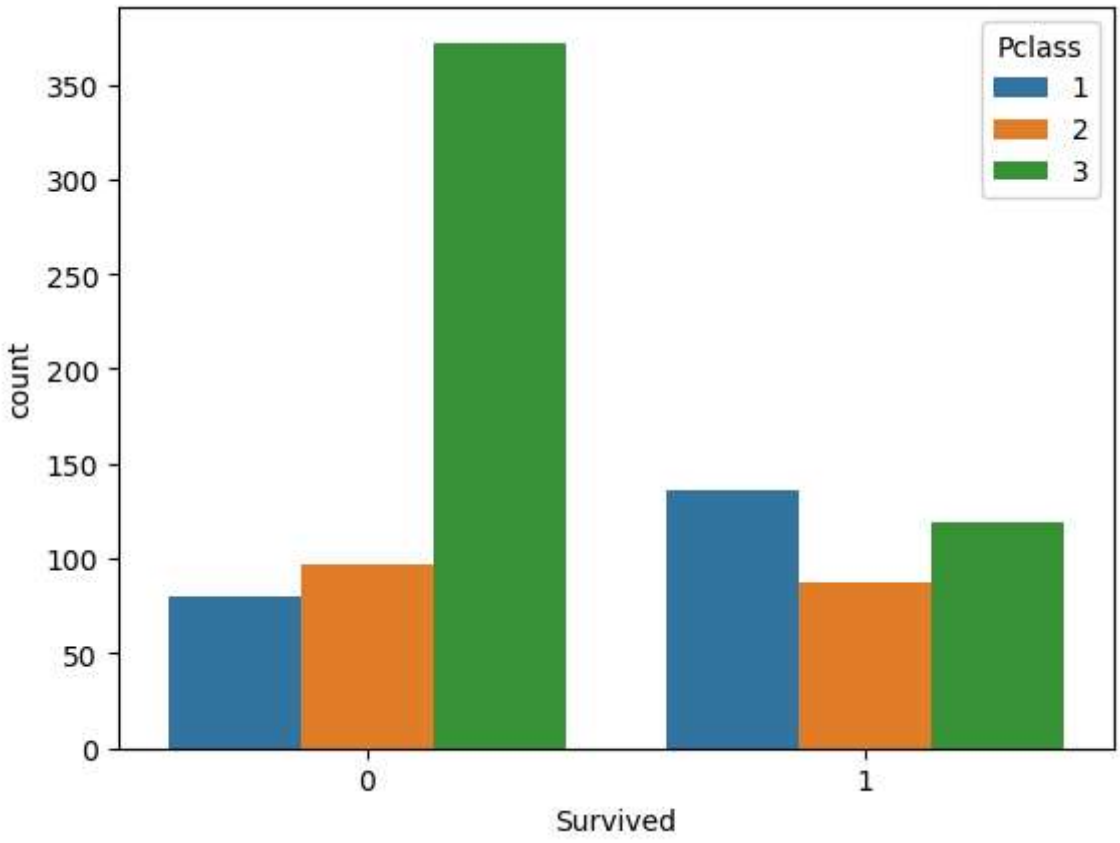
In [36]:
```
#Let's visualize the survivors count
sns.countplot(x=titanic_df.Survived)
plt.show()
```



In [37]:
```
# Lets check how many males and females were survived
sns.countplot(x=titanic_df.Survived,hue=titanic_df.Sex)
plt.show()
```



In [38]:
```
 # Let's distinguish the data by passenger class
sns.countplot(x=titanic_df.Survived,hue=titanic_df.Pclass)
plt.show()
```



1.4.1 Numerical Data

In [39]:
```
tips_df = pd.read_csv('C:/Users/raviy/Downloads/tips.csv')
```

In [40]:
```
tips_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object 
 3   smoker            244 non-null    object 
 4   day               244 non-null    object 
 5   time              244 non-null    object 
 6   size              244 non-null    int64  
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object 
 9   CC Number         244 non-null    int64  
 10  Payment ID        244 non-null    object 
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

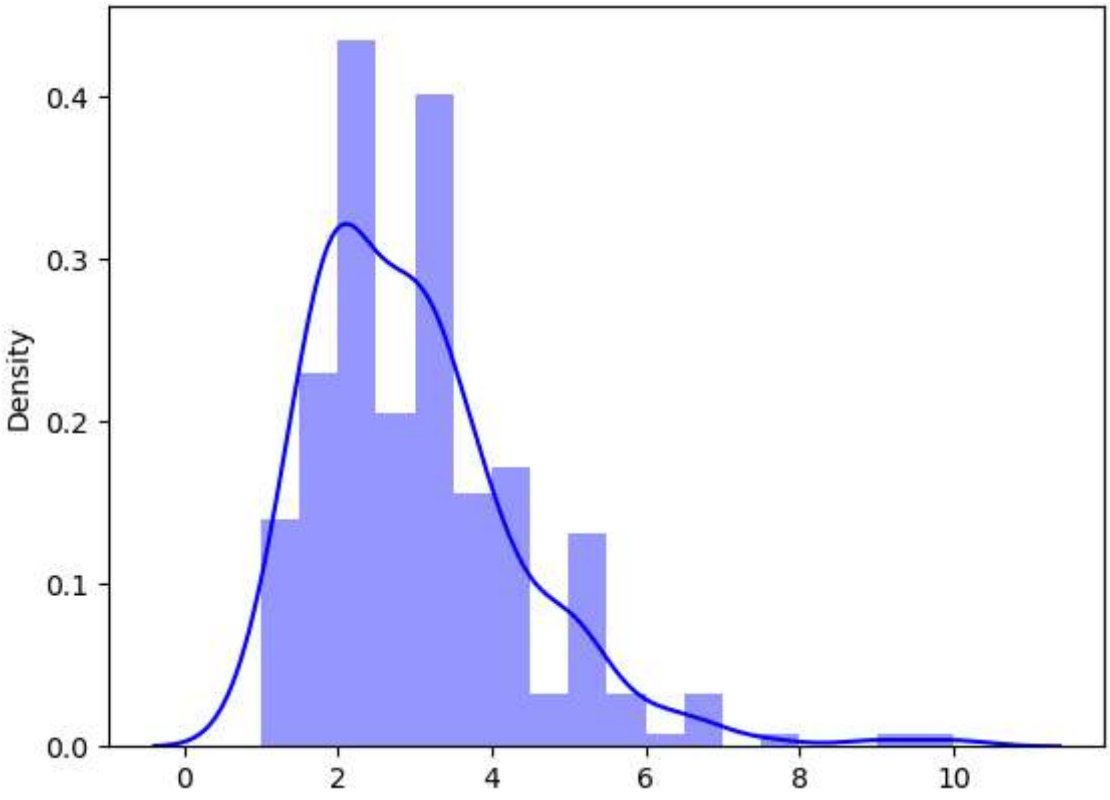
In [41]:
```
#checking top 5 recordsof the dataset
tips_df.head()
```

Out[41]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032516
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807137
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181211
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613764
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273261

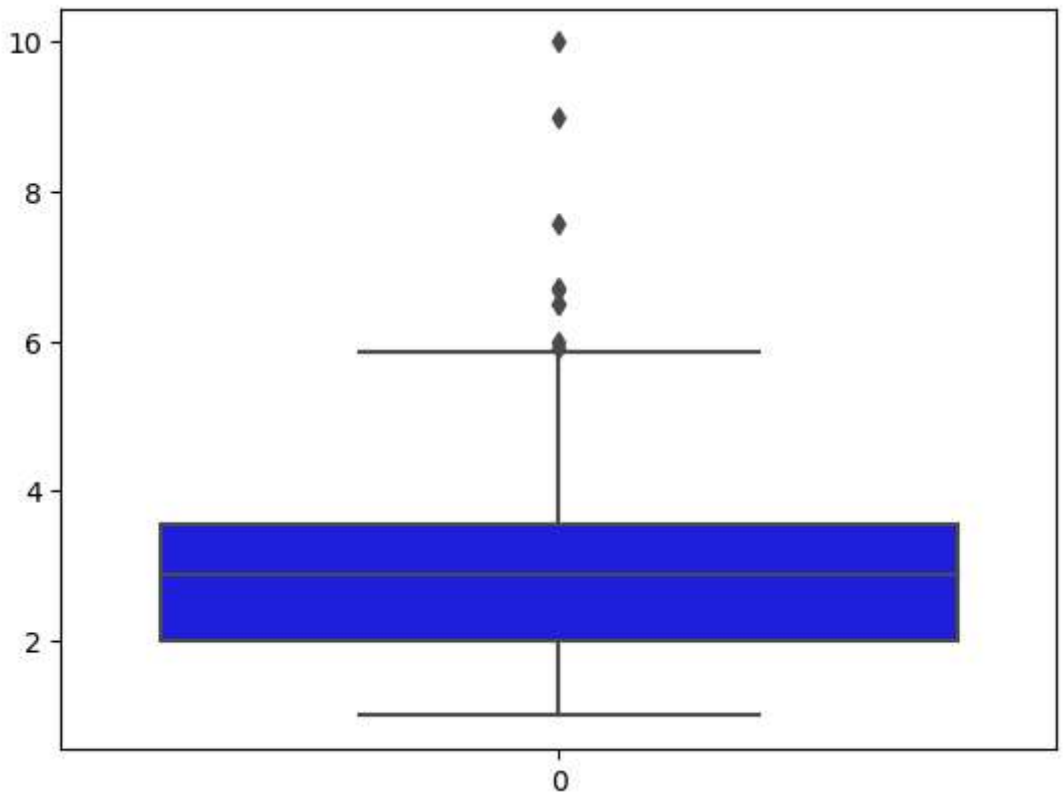
In [43]:
```
#distribution of tips
sns.distplot(x=tips_df.tip, hist=True, kde=True, color='b')
```

Out[43]: <Axes: ylabel='Density'>



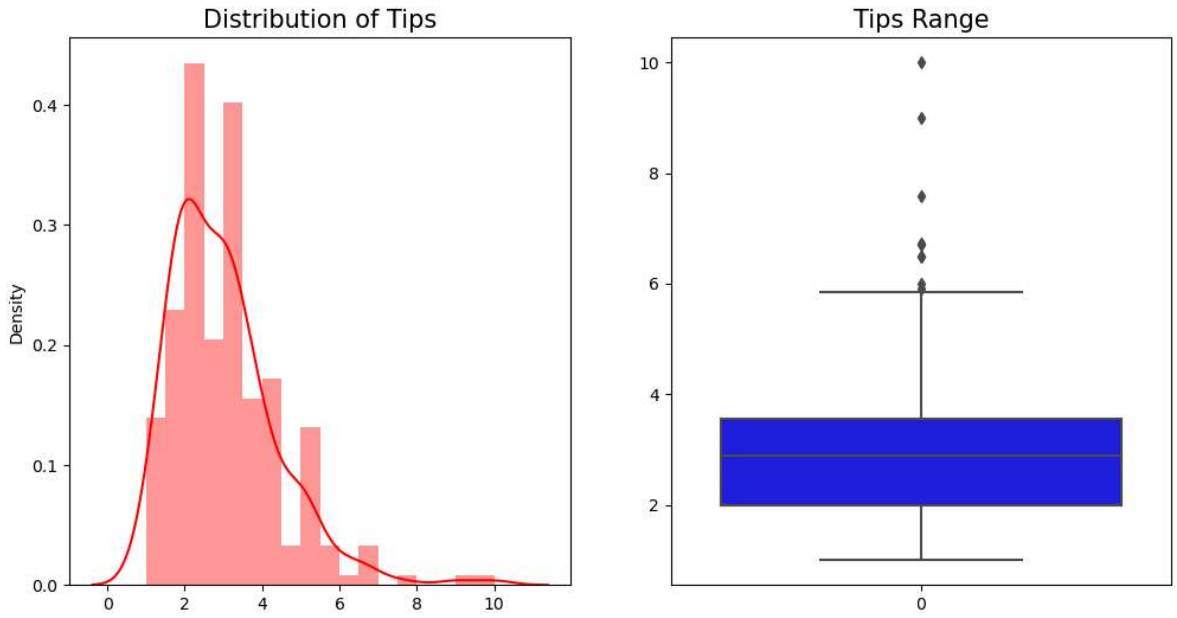
In [44]:
```
# to find out the range
sns.boxplot(tips_df.tip, color = 'b')
```

Out[44]: <Axes: >



In [45]:
```
fig, axes = plt.subplots(1,2, figsize =(12,6))
#distribution of tips
sns.distplot(x=tips_df.tip, hist=True, kde=True, color='r',ax=axes[0])
# to find out the range
sns.boxplot(tips_df.tip, color = 'b',ax=axes[1])
axes[0].set_title("Distribution of Tips",fontsize=15)
axes[1].set_title("Tips Range",fontsize=15)
```

Out[45]: Text(0.5, 1.0, 'Tips Range')

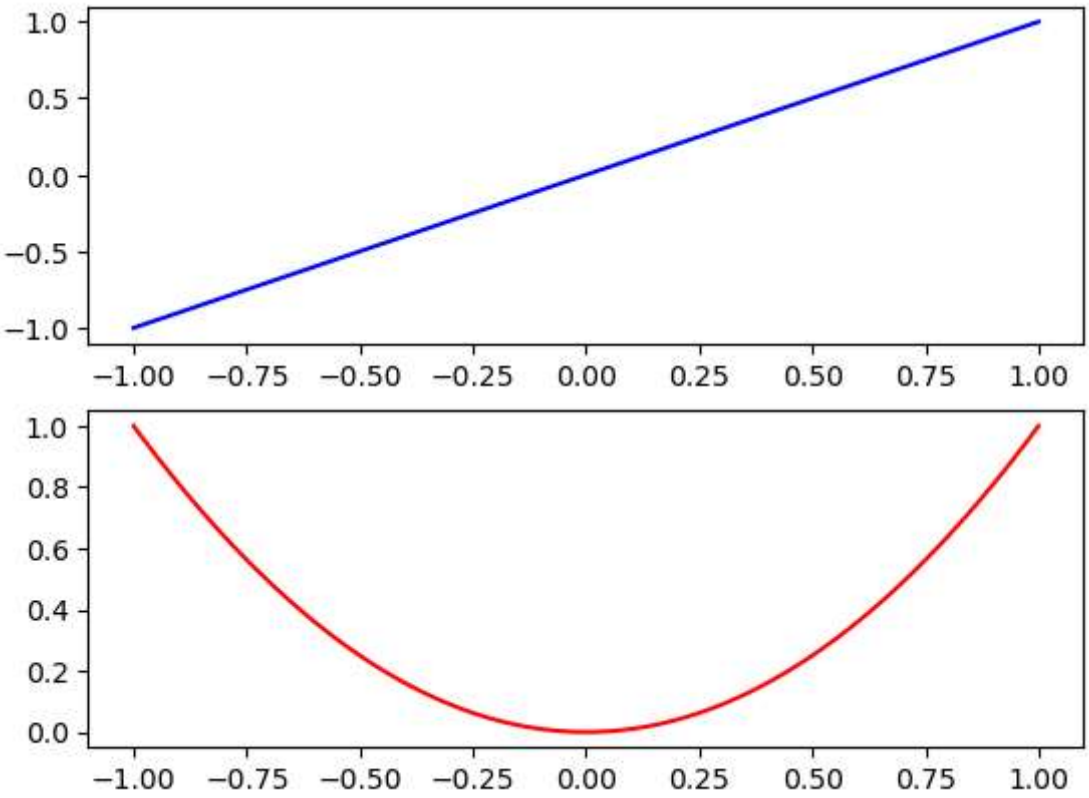


1.5 Subplots

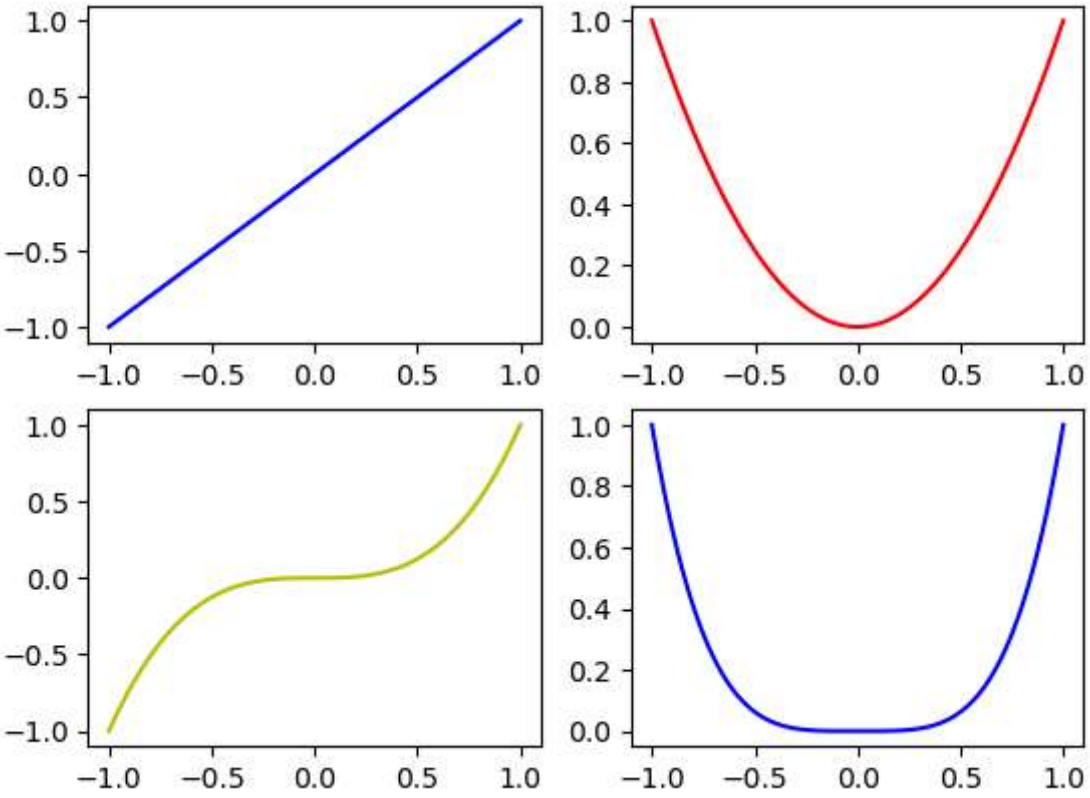
In [46]:
```
x = np.linspace(-1,1,101)
y1 = x
y2 = x**2
y3 = x**3
y4 = x**4
```

In [47]:
```
fig,ax = plt.subplots(2,1)
ax[0].plot(x,y1,color ='b')
ax[1].plot(x,y2, color='r')
```

Out[47]:
```
[<matplotlib.lines.Line2D at 0x19c42b6f310>]
```



In [48]:
```
fig,ax = plt.subplots(2,2)
ax[0,0].plot(x,y1, color ='b')
ax[0,1].plot(x,y2, color='r')
ax[1,0].plot(x,y3, color='y')
ax[1,1].plot(x,y4, color='b')
fig.show()
```



In [50]:
```
#Load the dataset
diamonds = pd.read_csv('C:/Users/raviy/Downloads/diamonds.csv')
```

In [51]:
```
diamonds.head()
```

Out[51]:

	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 [52]:
```
#checking the shape
diamonds.shape
```

Out[52]:
```
(53940, 11)
```

In [53]:
```
#filter the value where clarity is in {'SI1','VI2'}
diamond =diamonds[diamonds.clarity.isin(['SI1','VS2'])]
```

In [54]:
```
#checkig the shape after filtering
diamond.shape
```

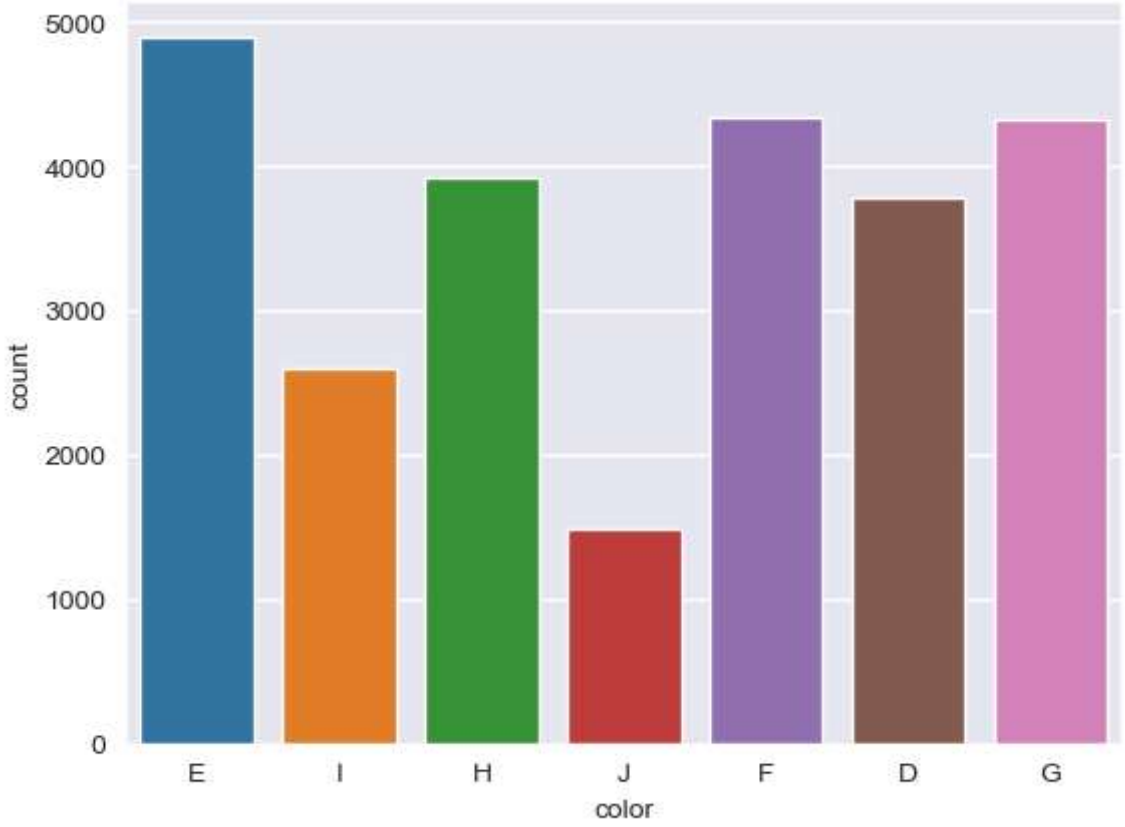
Out[54]:
```
(25323, 11)
```

Countplots

In [55]:
```
sns.set_style('darkgrid')
```

In [56]:
```
sns.countplot(x='color',data=diamond)
```

Out[56]:
```
<Axes: xlabel='color', ylabel='count'>
```

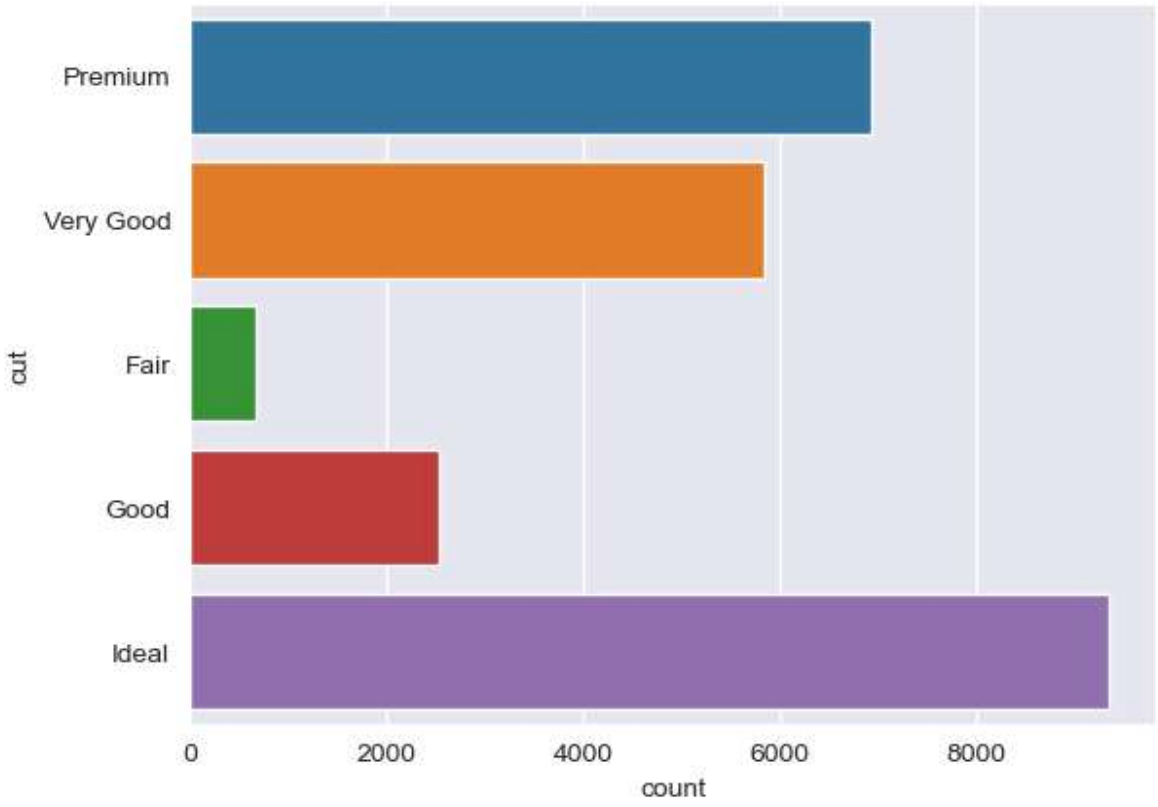


In [57]:
```
#checking the value counts of color
diamond.color.value_counts()
```

Out[57]:
```
color
E    4896
F    4332
G    4323
H    3918
D    3780
I    2593
J    1481
Name: count, dtype: int64
```

In [58]:
```
# use y to change the orientation instead of x
sns.countplot(y='cut',data=diamond)
```

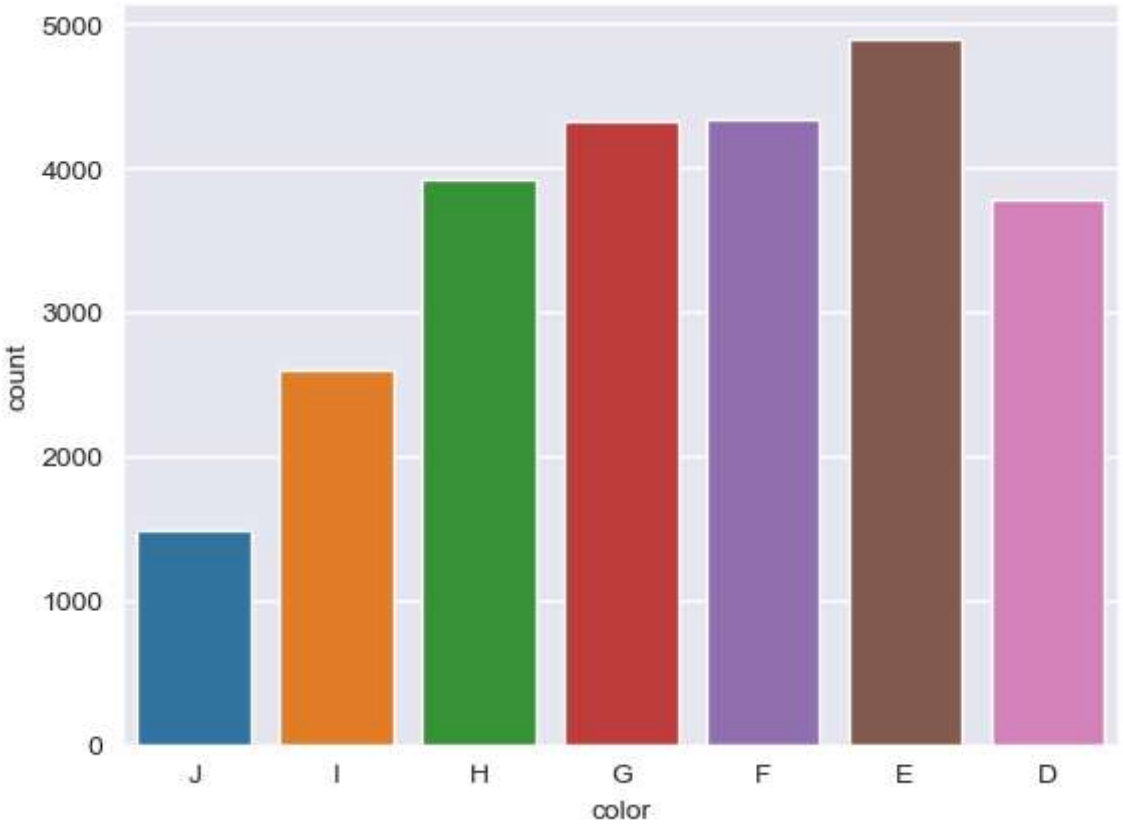
Out[58]:
```
<Axes: xlabel='count', ylabel='cut'>
```



In [59]:
```
#checking the datatypes
diamond.dtypes
```

Out[59]:
```
Unnamed: 0      int64
carat         float64
cut            object
color          object
clarity        object
depth         float64
table         float64
price           int64
x             float64
y             float64
z             float64
dtype: object
```

In [60]:
```
#Providing the order
color_order = ['J','I','H','G','F','E','D']
sns.countplot(x='color',data=diamond,order=color_order);
```



1.6 Order Ascending or Descending

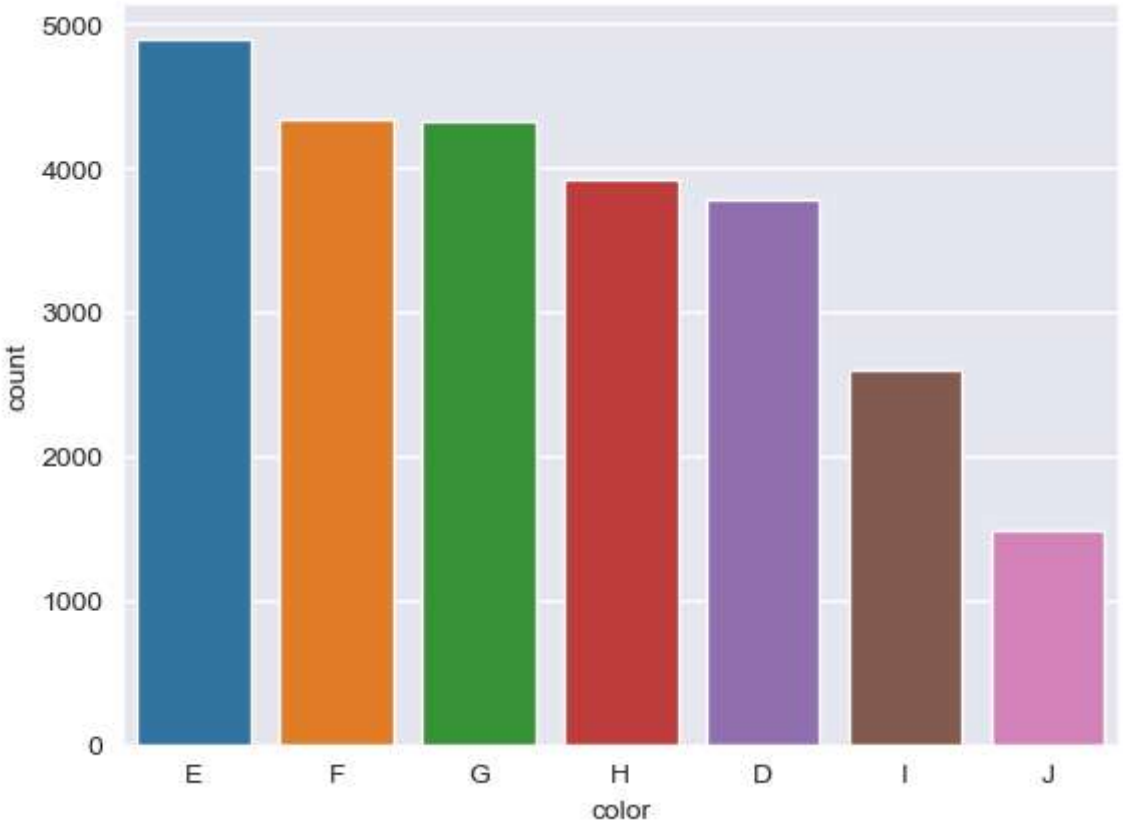
In [61]:
```
#checking the value counts
diamond.color.value_counts()
```

Out[61]:
```
color
E    4896
F    4332
G    4323
H    3918
D    3780
I    2593
J    1481
Name: count, dtype: int64
```

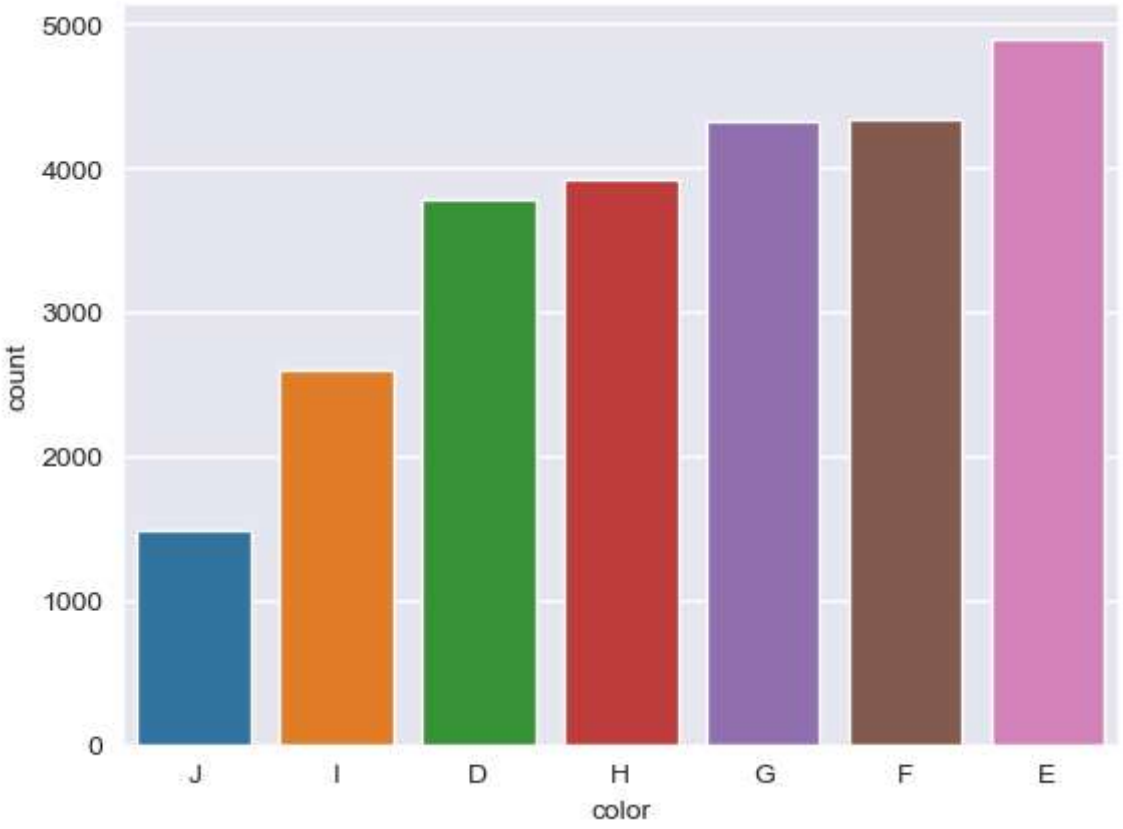
In [62]:
```
#checking the Index
diamond.color.value_counts().index
```

Out[62]:
```
Index(['E', 'F', 'G', 'H', 'D', 'I', 'J'], dtype='object', name='color')
```

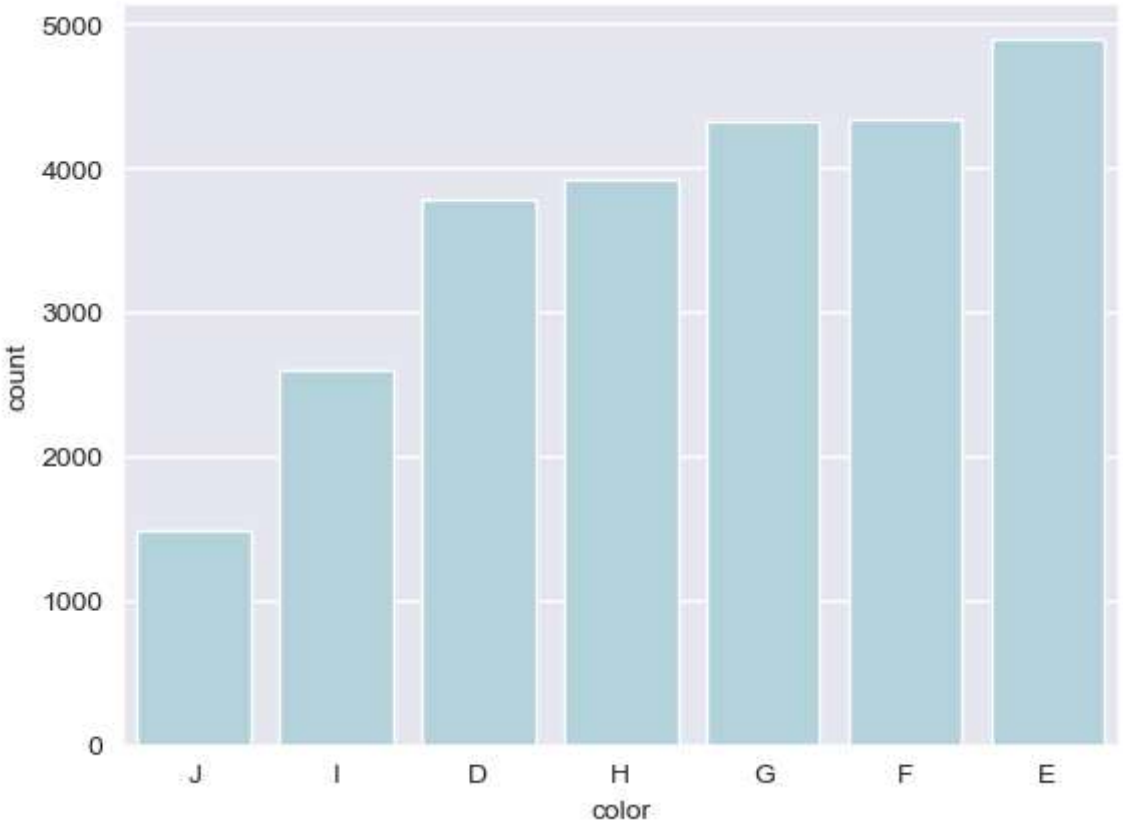
In [63]:
```
#plotting as per index (plotting in descending order)
sns.countplot(x='color', data=diamond, order=diamond.color.
              value_counts().index);
```



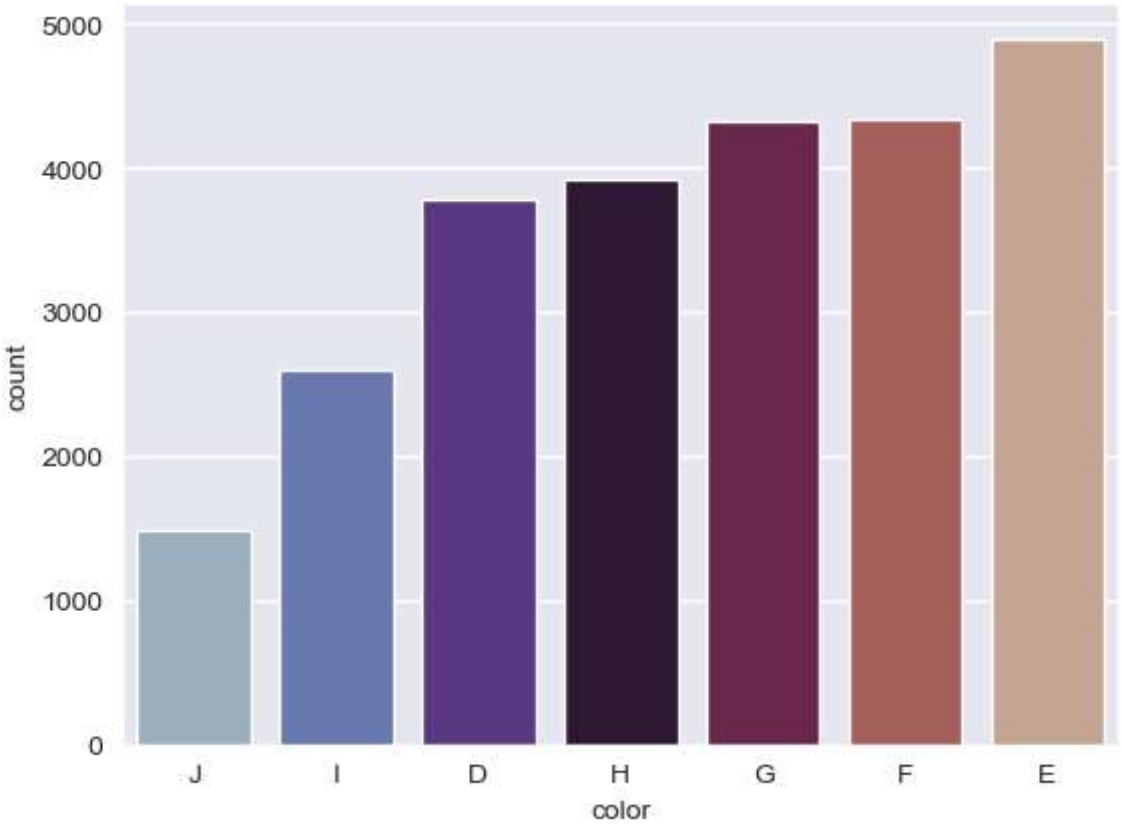
In [64]:
```
#plotting in ascending order
sns.countplot(x='color', data=diamond, order=diamond.color.value_counts().
              index[::-1]);
```



In [66]:
```
# setting the color to Black
sns.countplot(x='color', data=diamond, order=diamond.color.value_counts().
index[::-1],color='lightblue');
```

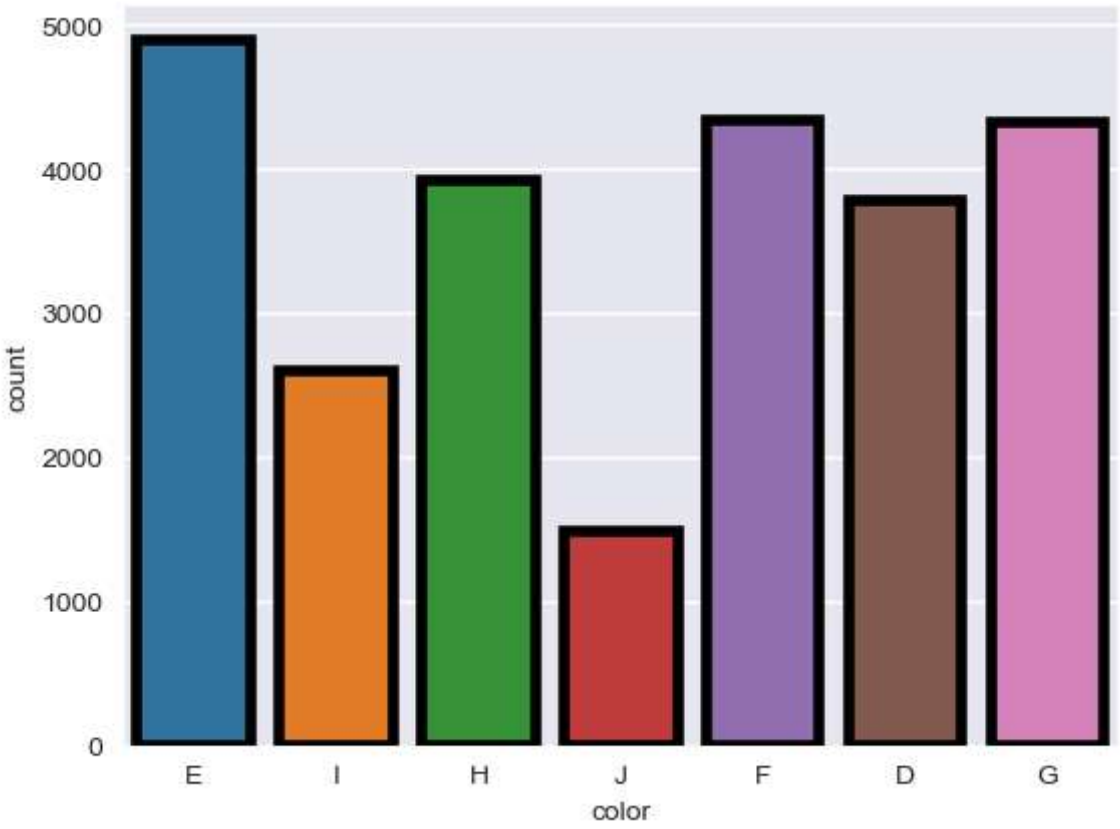


In [67]:
```
# setting the color to Black
sns.countplot(x='color', data=diamond, order=diamond.color.value_counts().
index[::-1],palette='twilight');
```



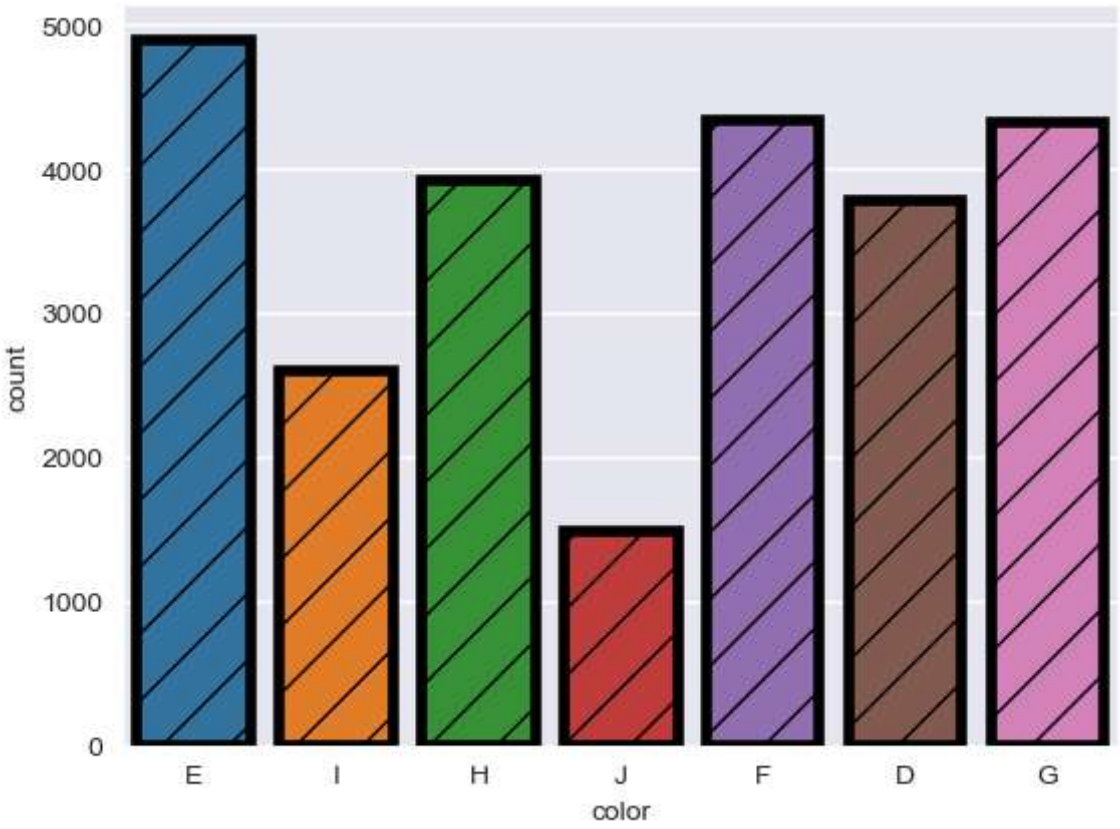
In [68]:
```
#Line width and edge color
sns.countplot(x='color',data=diamond, lw=4, ec='black')
```

Out[68]: <Axes: xlabel='color', ylabel='count'>



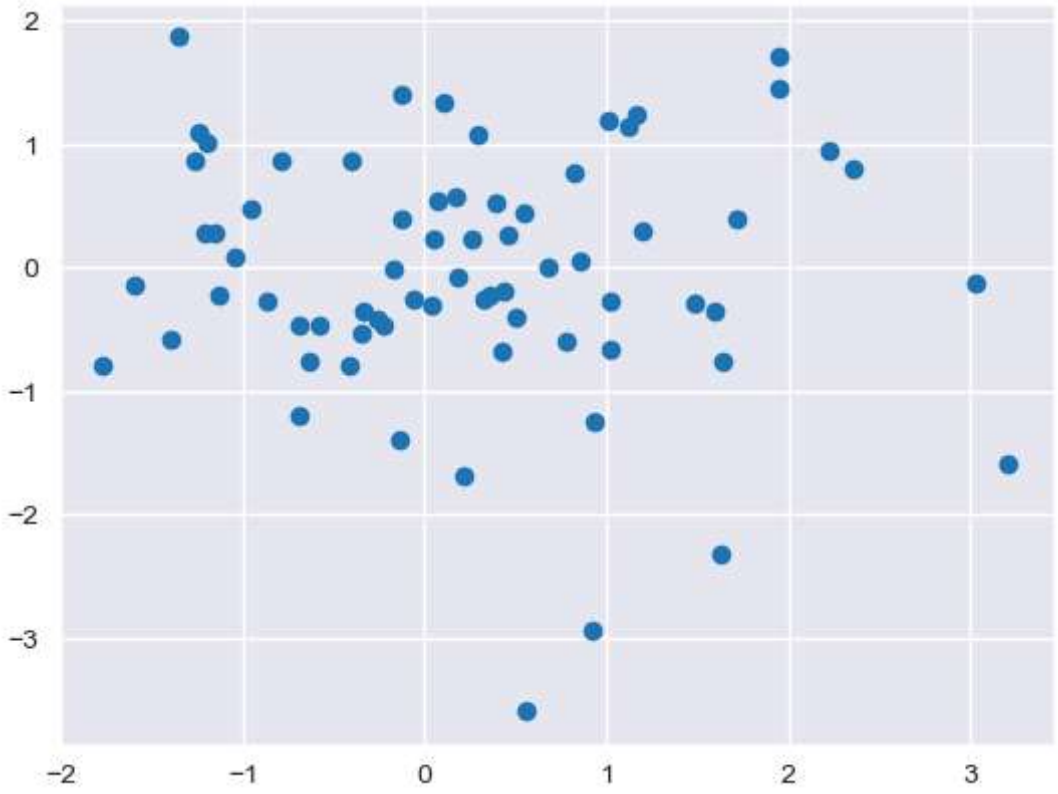
In [69]:
```
#hatching
sns.countplot(x='color',data=diamond, lw=4, ec='black',hatch='/')
```

Out[69]: <Axes: xlabel='color', ylabel='count'>

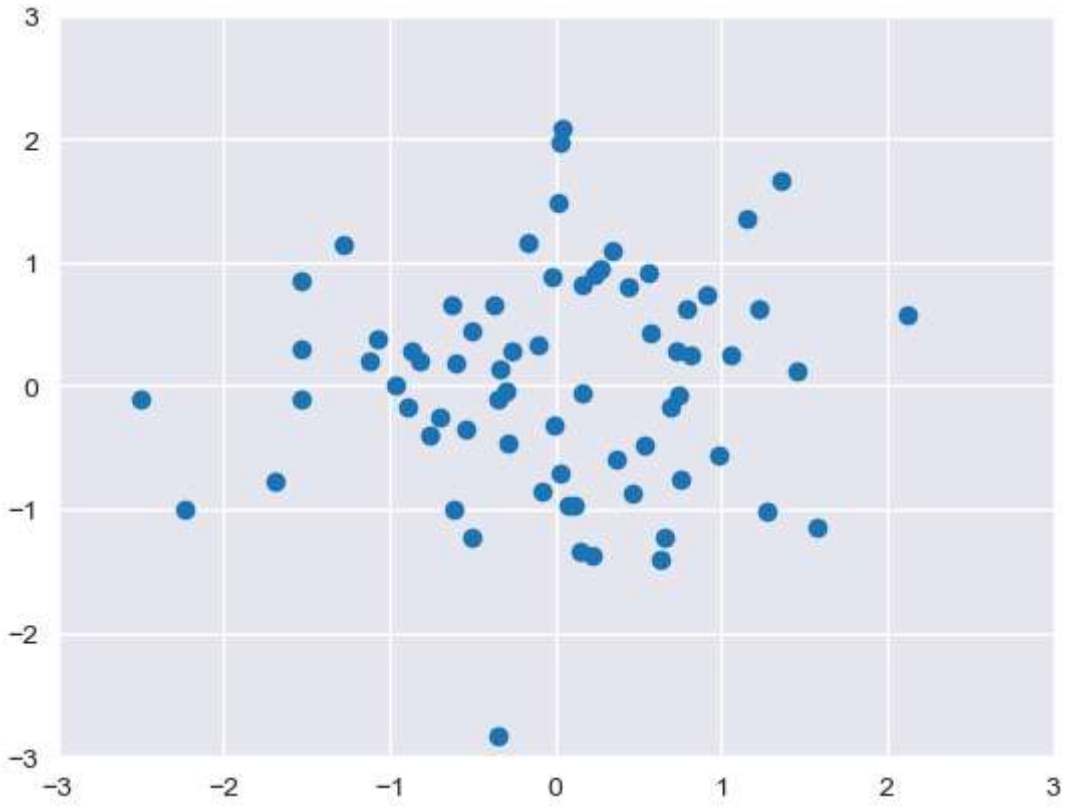


1.7 Scatter plot

In [70]:
```
y1 = np.random.randn(70)
y2 = np.random.randn(70)
# Both y1 and y2 should be of same size
plt.scatter(y1,y2)
plt.show()
```

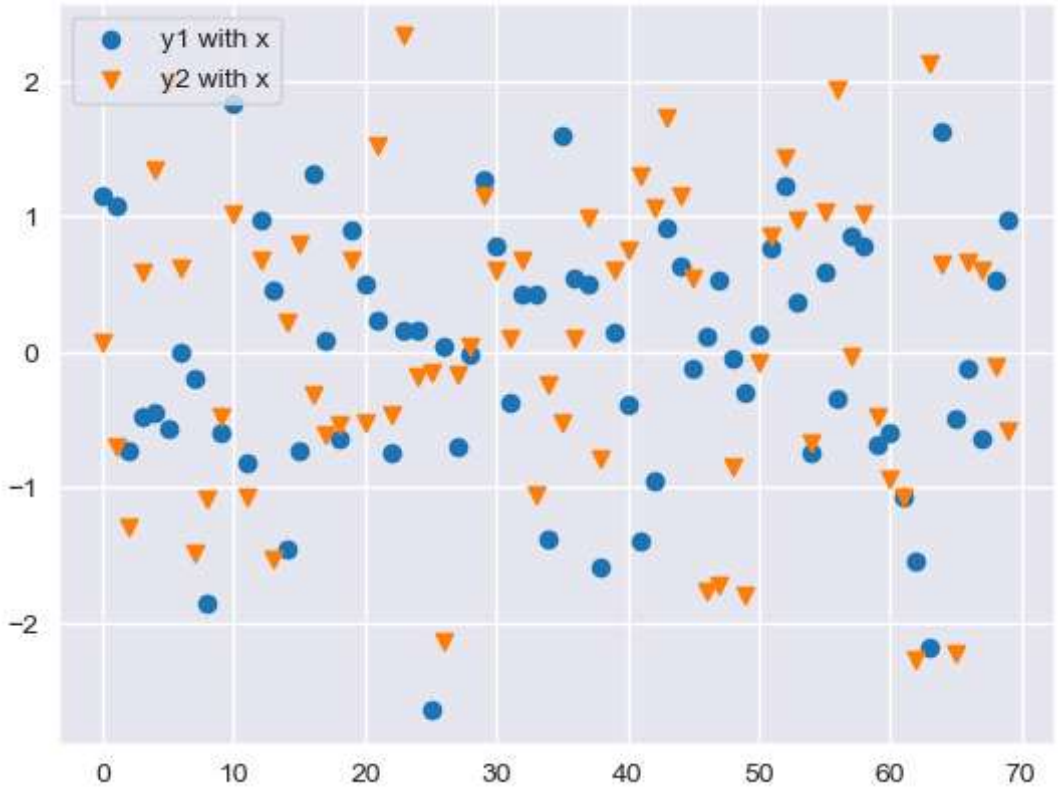


In [71]:
```
## adjust axis, if required
y1 = np.random.randn(70)
y2 = np.random.randn(70)
plt.scatter(y1,y2)
plt.axis([-3,3,-3,3]) #plt.axis([Xmin,Xmax,Ymin,Ymax])
plt.show()
```



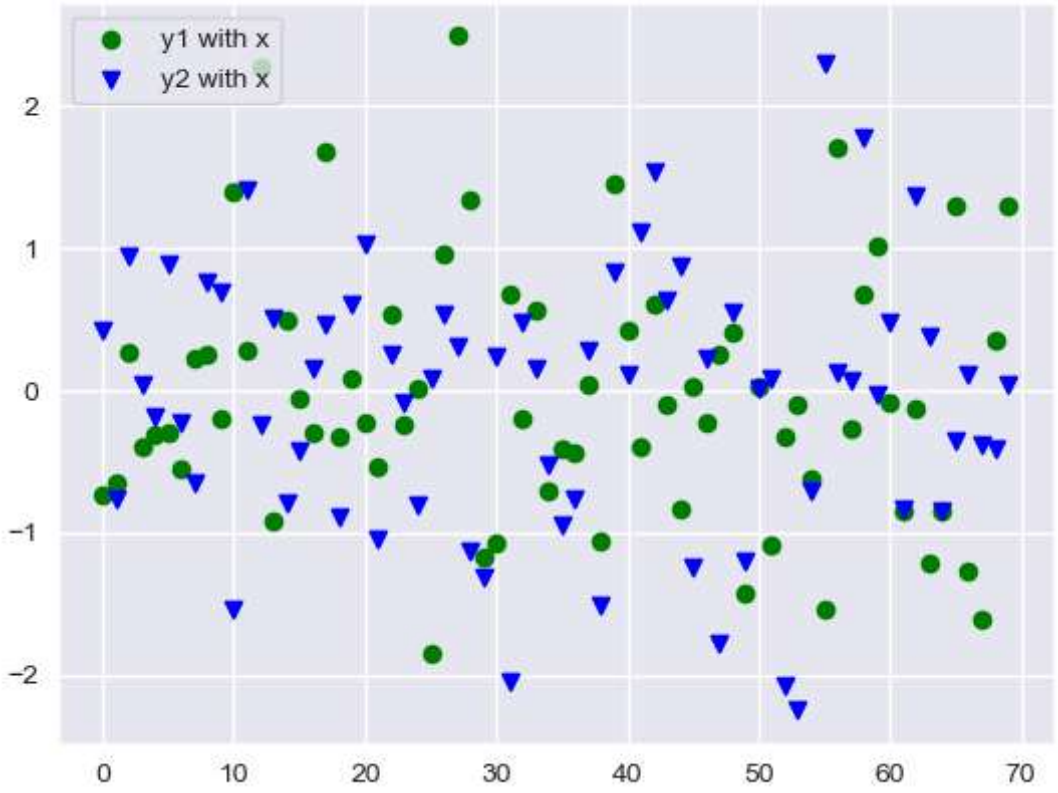
In [72]:
```
# using the scatter plot with comman axis
x = np.arange(70)
y1 = np.random.randn(70)
y2 = np.random.randn(70)
plt.scatter(x,y1, marker='o', label ='y1 with x')
plt.scatter(x,y2, marker='v', label ='y2 with x')
plt.legend(loc='upper left')
```

Out[72]: <matplotlib.legend.Legend at 0x19c442b2e90>



In [73]:
```
#assigning the colors
x = np.arange(70)
y1 = np.random.randn(70)
y2 = np.random.randn(70)
plt.scatter(x,y1, marker='o', label ='y1 with x',color='g')
plt.scatter(x,y2, marker='v', label ='y2 with x',color='b')
plt.legend(loc='upper left')
```

Out[73]: <matplotlib.legend.Legend at 0x19c442d0130>



1.8 Line Plot

In [4]:
```
# extract data from various Internet sources into a pandas DataFrame
!pip install pandas-datareader
import pandas_datareader as pdr
```

```
Collecting pandas-datareader
  Downloading pandas_datareader-0.10.0-py3-none-any.whl (109 kB)
     ---------------------------------- 109.5/109.5 kB 910.3 kB/s eta 0:
00:00
Requirement already satisfied: pandas>=0.23 in c:\users\raviy\anaconda3\li
b\site-packages (from pandas-datareader) (2.3.3)
Requirement already satisfied: lxml in c:\users\raviy\anaconda3\lib\site-p
ackages (from pandas-datareader) (4.9.1)
Requirement already satisfied: requests>=2.19.0 in c:\users\raviy\anaconda
3\lib\site-packages (from pandas-datareader) (2.28.1)
Requirement already satisfied: numpy>=1.22.4 in c:\users\raviy\anaconda3\l
ib\site-packages (from pandas>=0.23->pandas-datareader) (1.26.4)
Requirement already satisfied: pytz>=2020.1 in c:\users\raviy\anaconda3\li
b\site-packages (from pandas>=0.23->pandas-datareader) (2022.7)
Requirement already satisfied: tzdata>=2022.7 in c:\users\raviy\anaconda3
\lib\site-packages (from pandas>=0.23->pandas-datareader) (2025.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\raviy\an
aconda3\lib\site-packages (from pandas>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\raviy\anaconda3\li
```

In [5]:
```
#extracting Data
stocks = ['GOOG', 'AMZN']
data = pdr.get_data_yahoo(stocks, start = '2022-01-01')['Close']
data.head()
```

```
---------------------------------------------------------------------------
RemoteDataError                           Traceback (most recent call last)
Cell In[5], line 3
      1 #extracting Data
      2 stocks = ['GOOG', 'AMZN']
----> 3 data = pdr.get_data_yahoo(stocks, start = '2022-01-01')['Close']
      4 data.head()

File ~\anaconda3\lib\site-packages\pandas_datareader\data.py:80, in get_data
_yahoo(*args, **kwargs)
     79 def get_data_yahoo(*args, **kwargs):
---> 80     return YahooDailyReader(*args, **kwargs).read()

File ~\anaconda3\lib\site-packages\pandas_datareader\base.py:258, in _DailyB
aseReader.read(self)
    256     df = self._dl_mult_symbols(self.symbols.index)
    257 else:
--> 258     df = self._dl_mult_symbols(self.symbols)
    259 return df

File ~\anaconda3\lib\site-packages\pandas_datareader\base.py:277, in _DailyB
aseReader._dl_mult_symbols(self, symbols)
    275 if len(passed) == 0:
    276     msg = "No data fetched using {0!r}"
--> 277     raise RemoteDataError(msg.format(self.__class__.__name__))
    278 try:
    279     if len(stocks) > 0 and len(failed) > 0 and len(passed) > 0:

RemoteDataError: No data fetched using 'YahooDailyReader'
```

In [3]:
```
#Loading the Dataset
tips = pd.read_csv('C:/Users/raviy/Downloads/tips.csv')
```

In [7]:
```
#checking the head of data
tips.head()
```

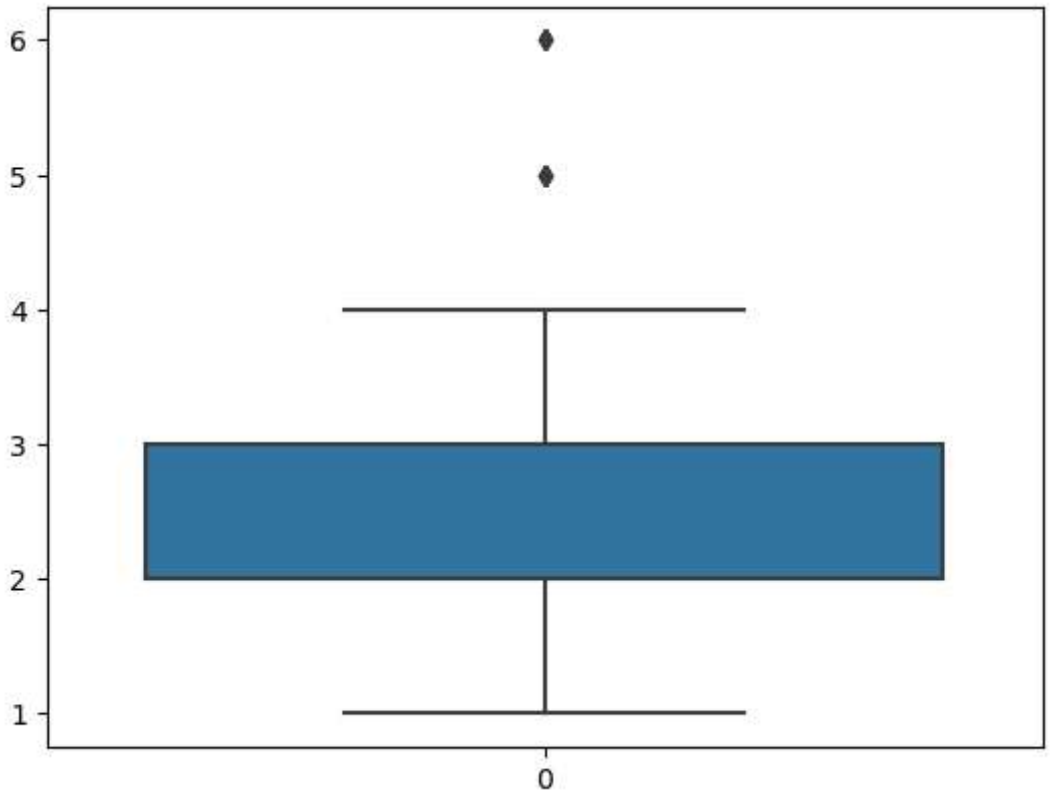
Out[7]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032516
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807137
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181211
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613764
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273261

1.9 Boxplot

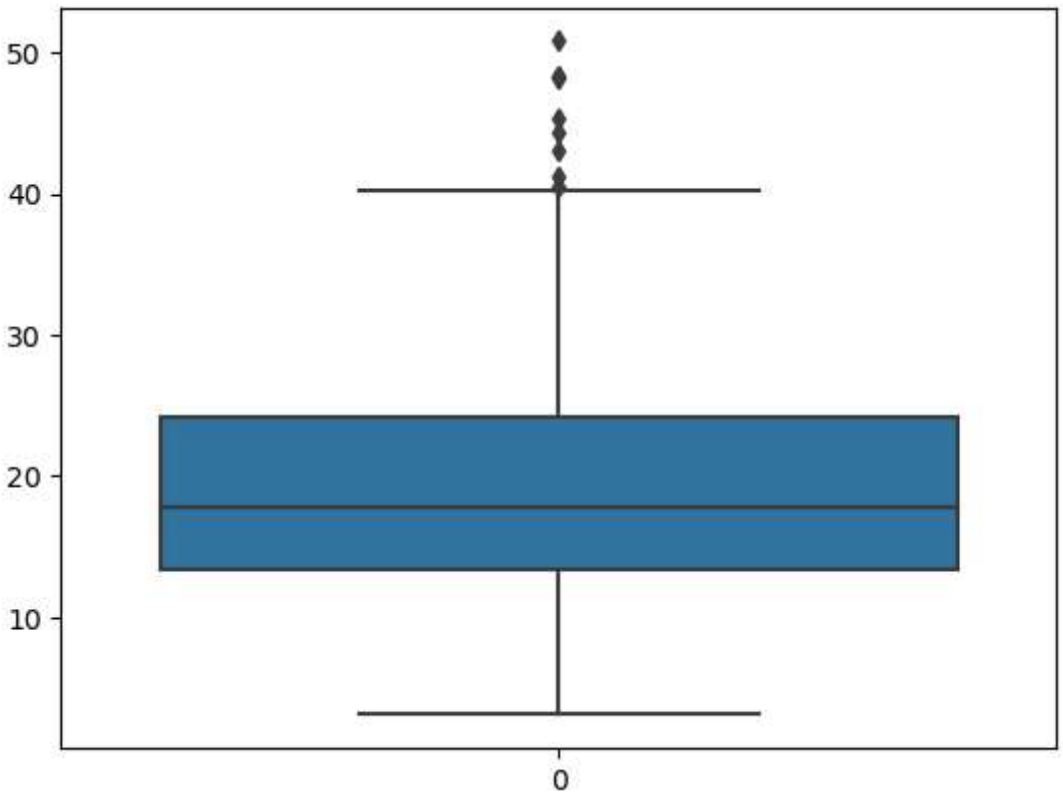
In [8]:
```
sns.boxplot(tips['size'])
```

Out[8]: <Axes: >



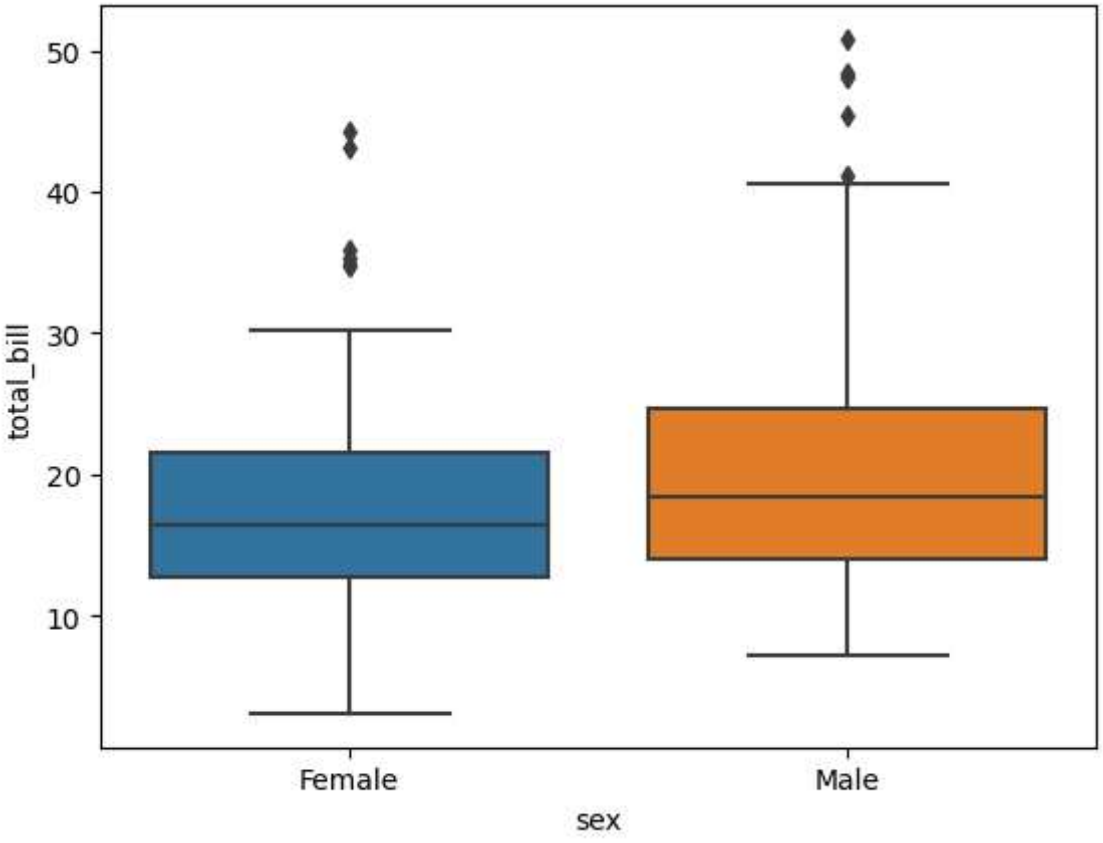
In [9]:
```
sns.boxplot(tips['total_bill'])
```

Out[9]: <Axes: >



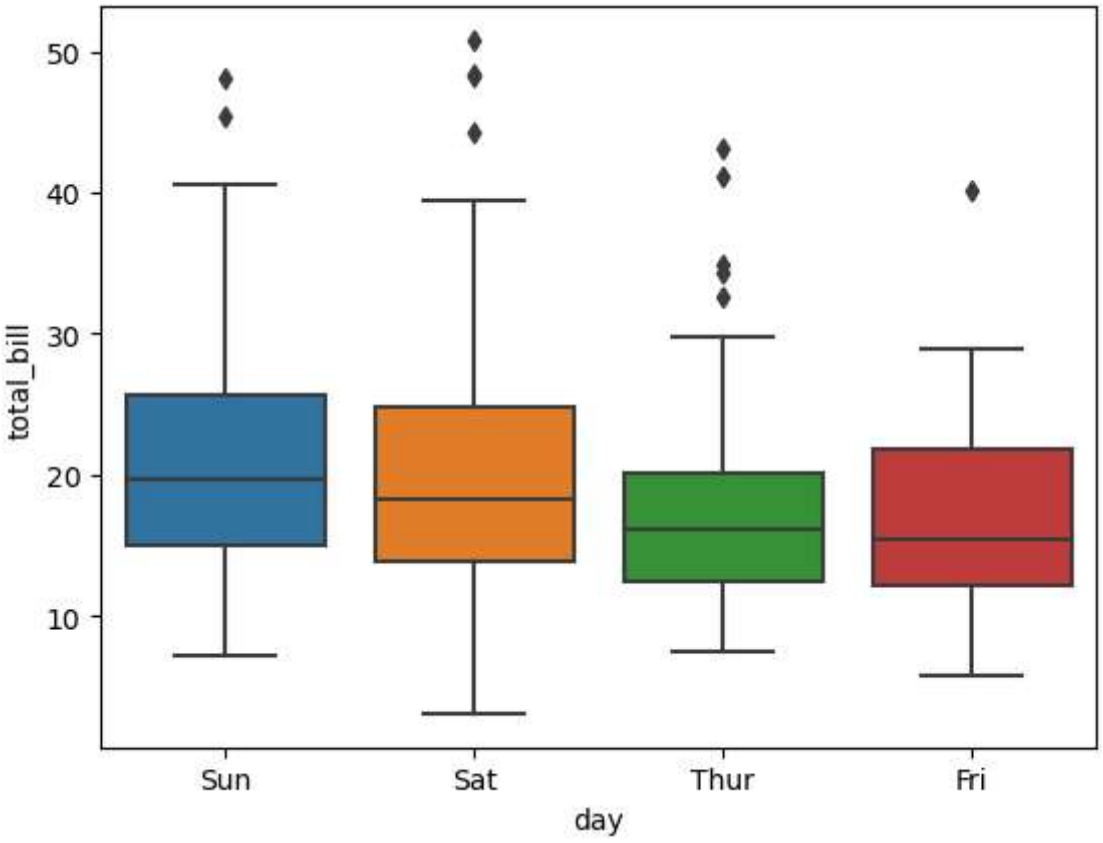
In [10]:
```
sns.boxplot(x='sex', y ='total_bill', data=tips)
```

Out[10]: <Axes: xlabel='sex', ylabel='total_bill'>



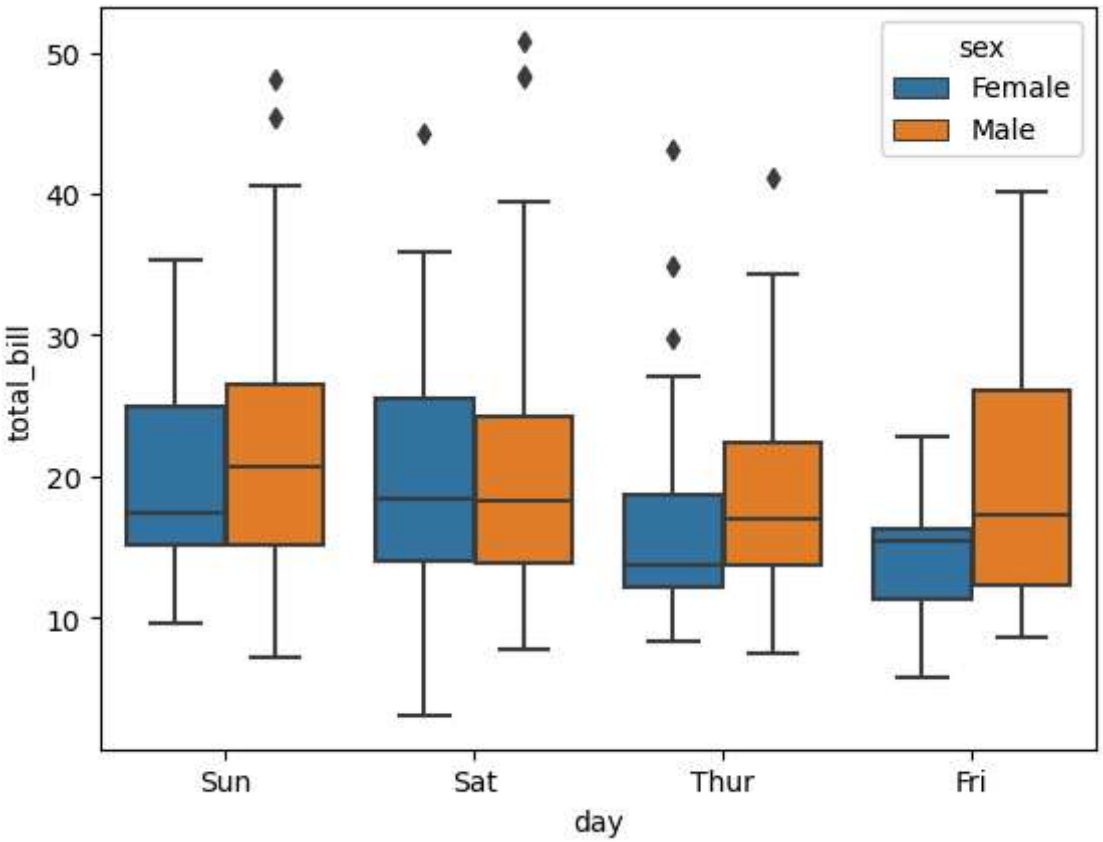
In [11]:
```
sns.boxplot(x='day', y ='total_bill', data=tips)
```

Out[11]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



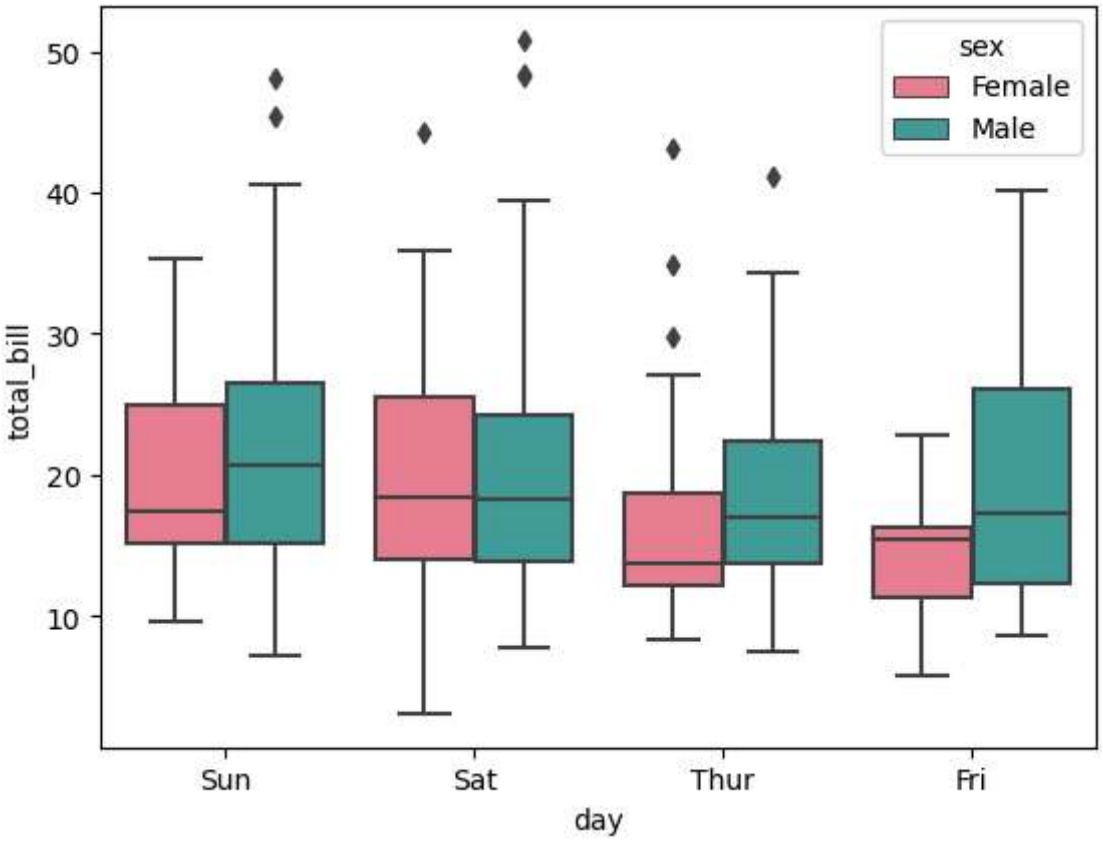
In [12]:
```
sns.boxplot(x='day', y ='total_bill', data=tips,hue='sex')
```

Out[12]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



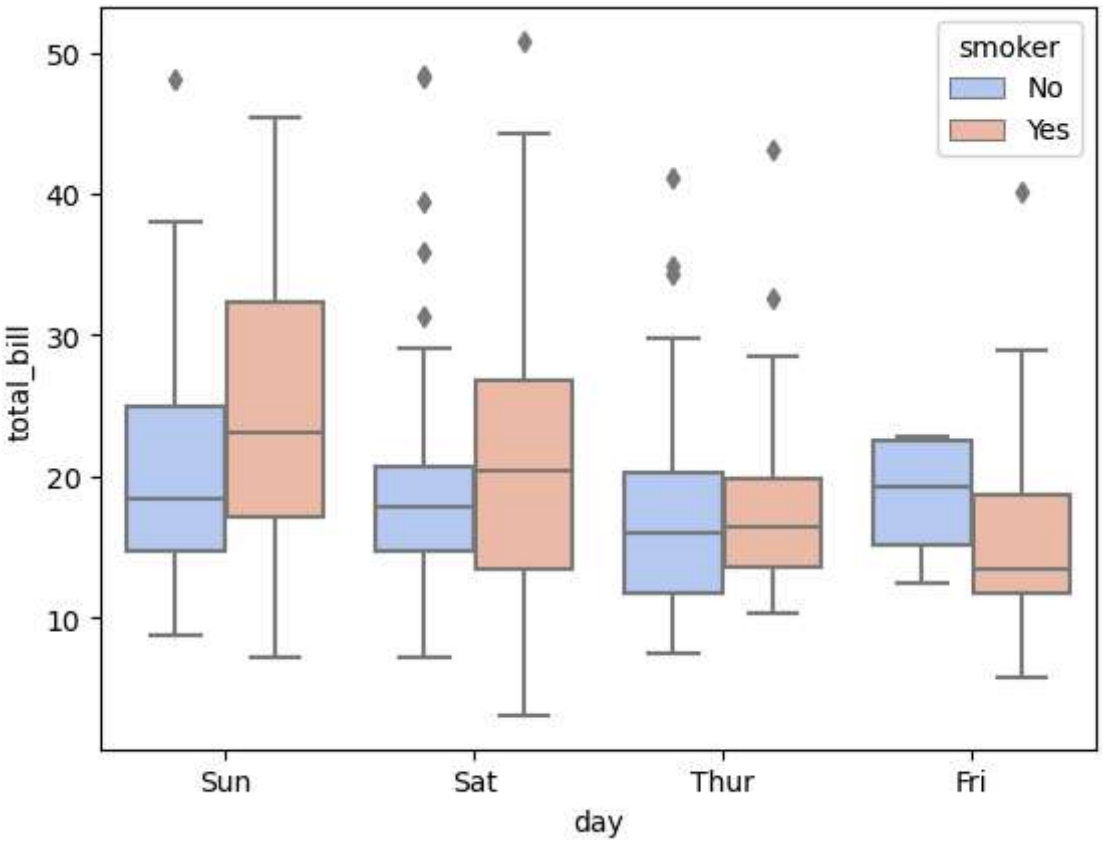
In [13]:
```
sns.boxplot(x='day', y ='total_bill', data=tips,hue='sex',palette ='husl')
```

Out[13]: <Axes: xlabel='day', ylabel='total_bill'>



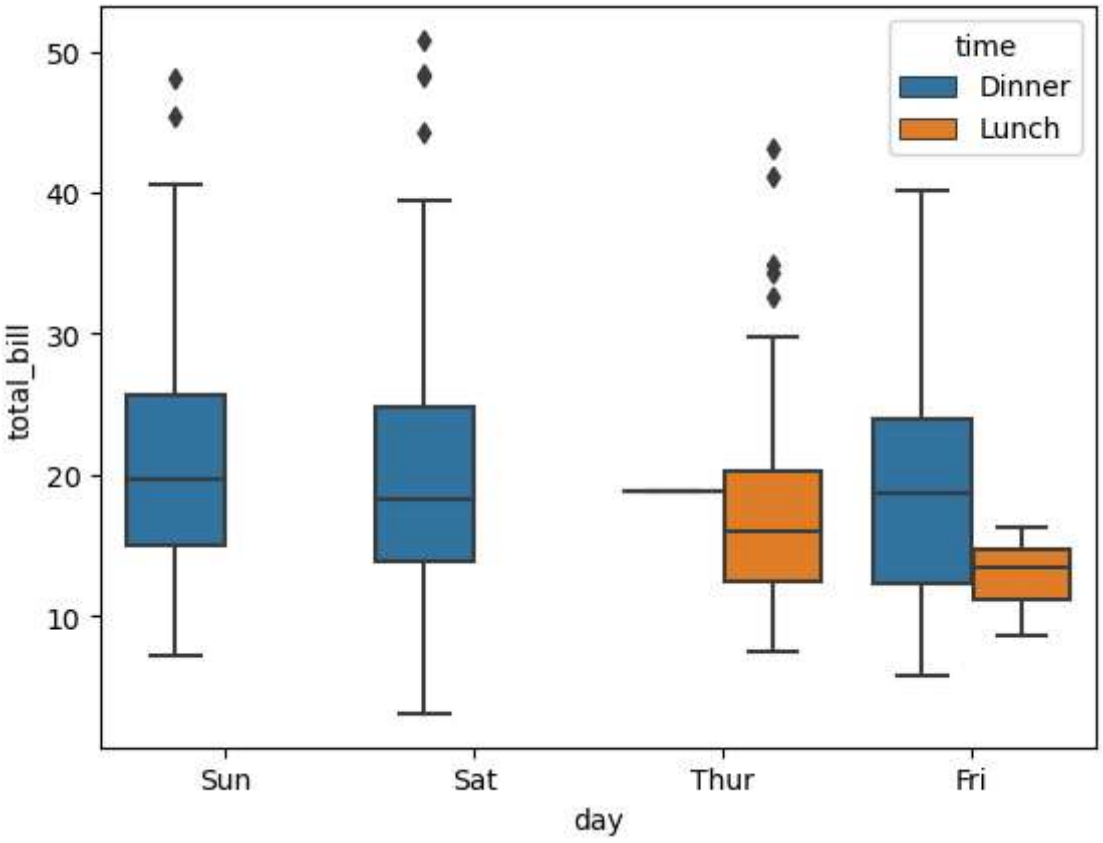
In [14]:
```
sns.boxplot(x='day', y ='total_bill', data=tips,hue='smoker',palette='coolwarm
```

Out[14]: <Axes: xlabel='day', ylabel='total_bill'>



In [15]:
```
sns.boxplot(x='day', y ='total_bill', data=tips,hue='time')
```

Out[15]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



1.10 Joint Distribution

jointplot shows scatter plot in the center and distribution plots on the sides to give full insight.

jointplot is actually a combination of 3 plots, not one.

Center plot → Scatter plot

Top plot,Right plot → Histogram / Bar plot

In [16]:
```
iris = pd.read_csv('C:/Users/raviy/Downloads/Iris.csv')
```

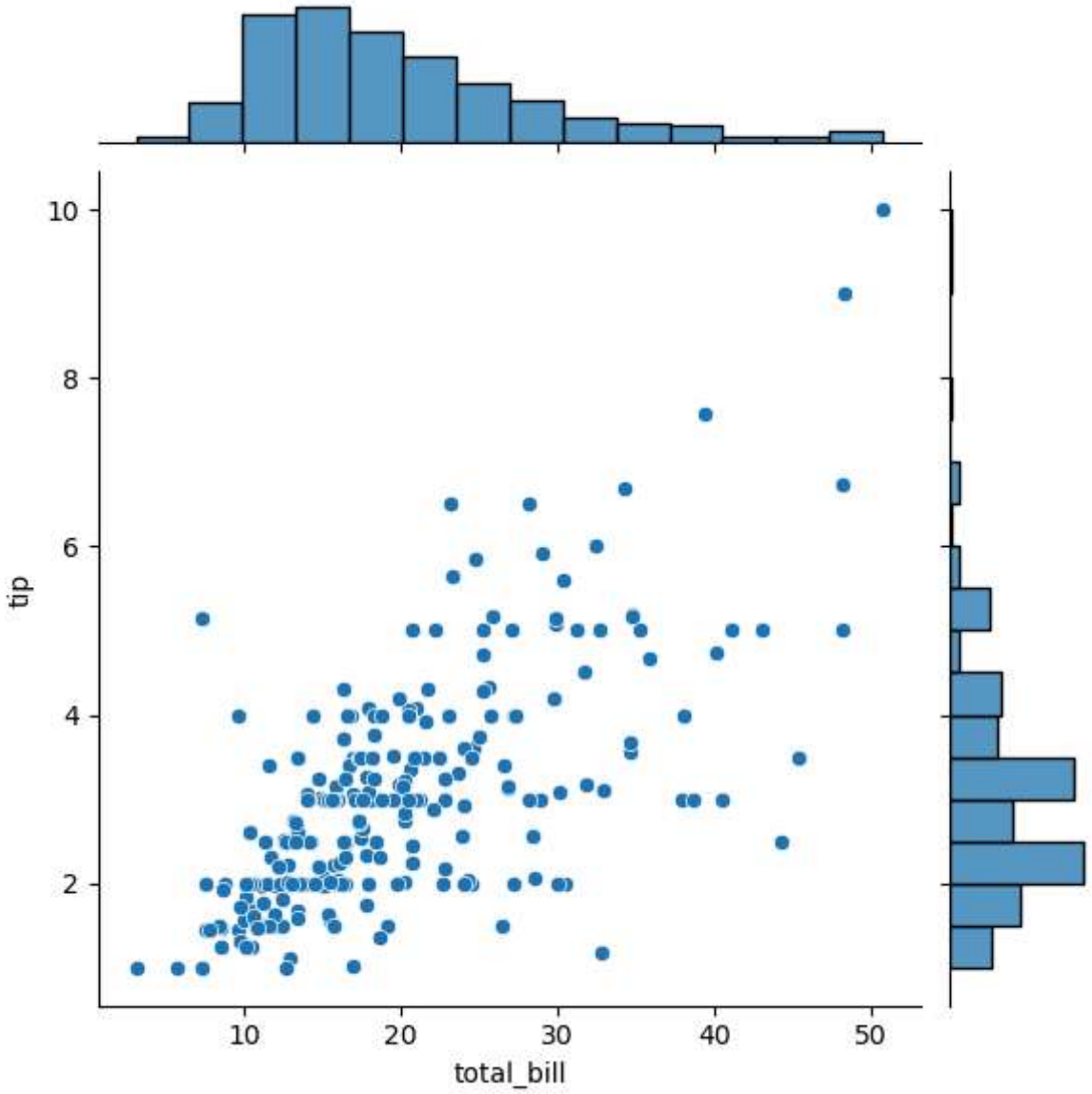
In [17]:
```
iris.head()
```

Out[17]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

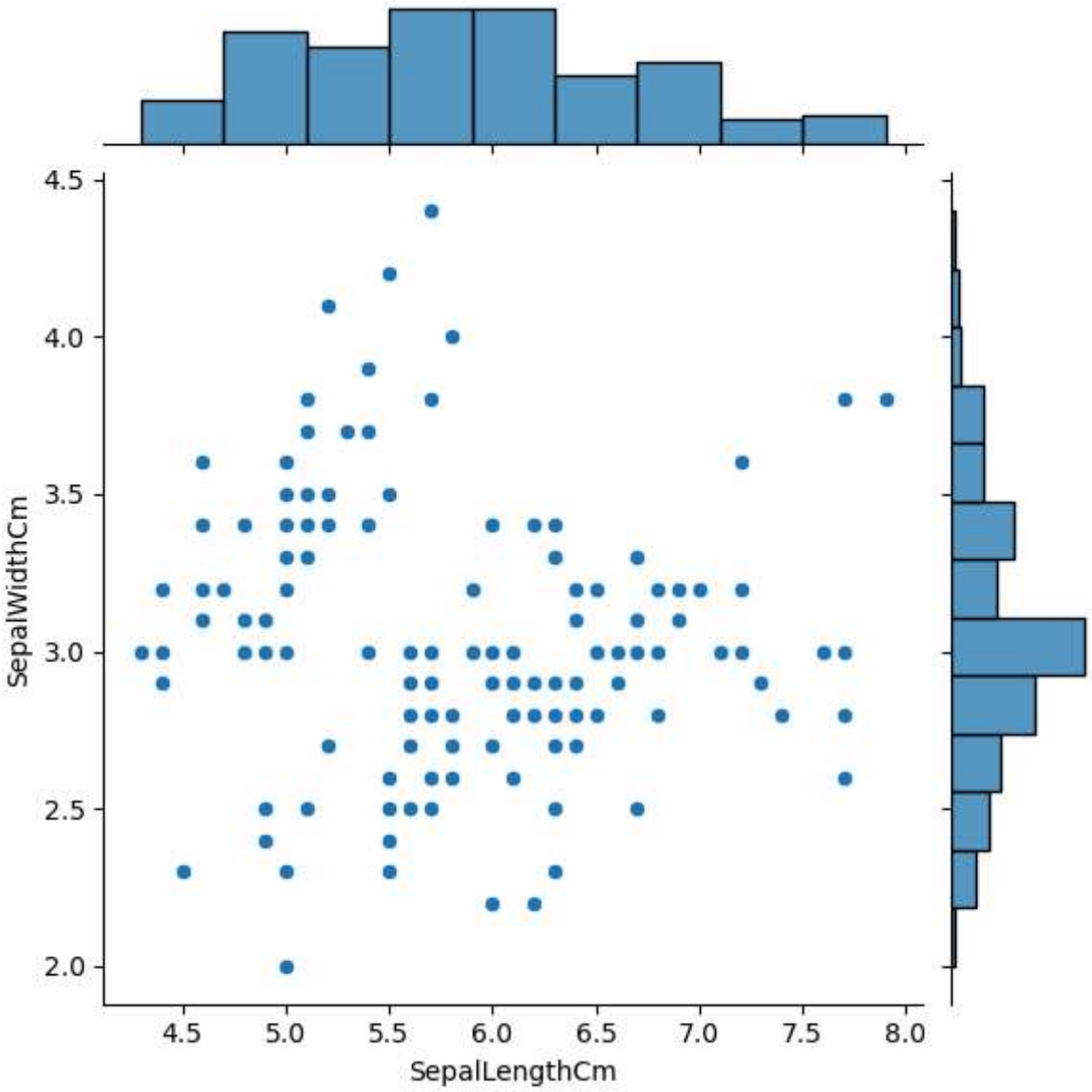
In [18]:
```
sns.jointplot(x='total_bill', y='tip', data=tips)
```

Out[18]:
```
<seaborn.axisgrid.JointGrid at 0x14aed551240>
```



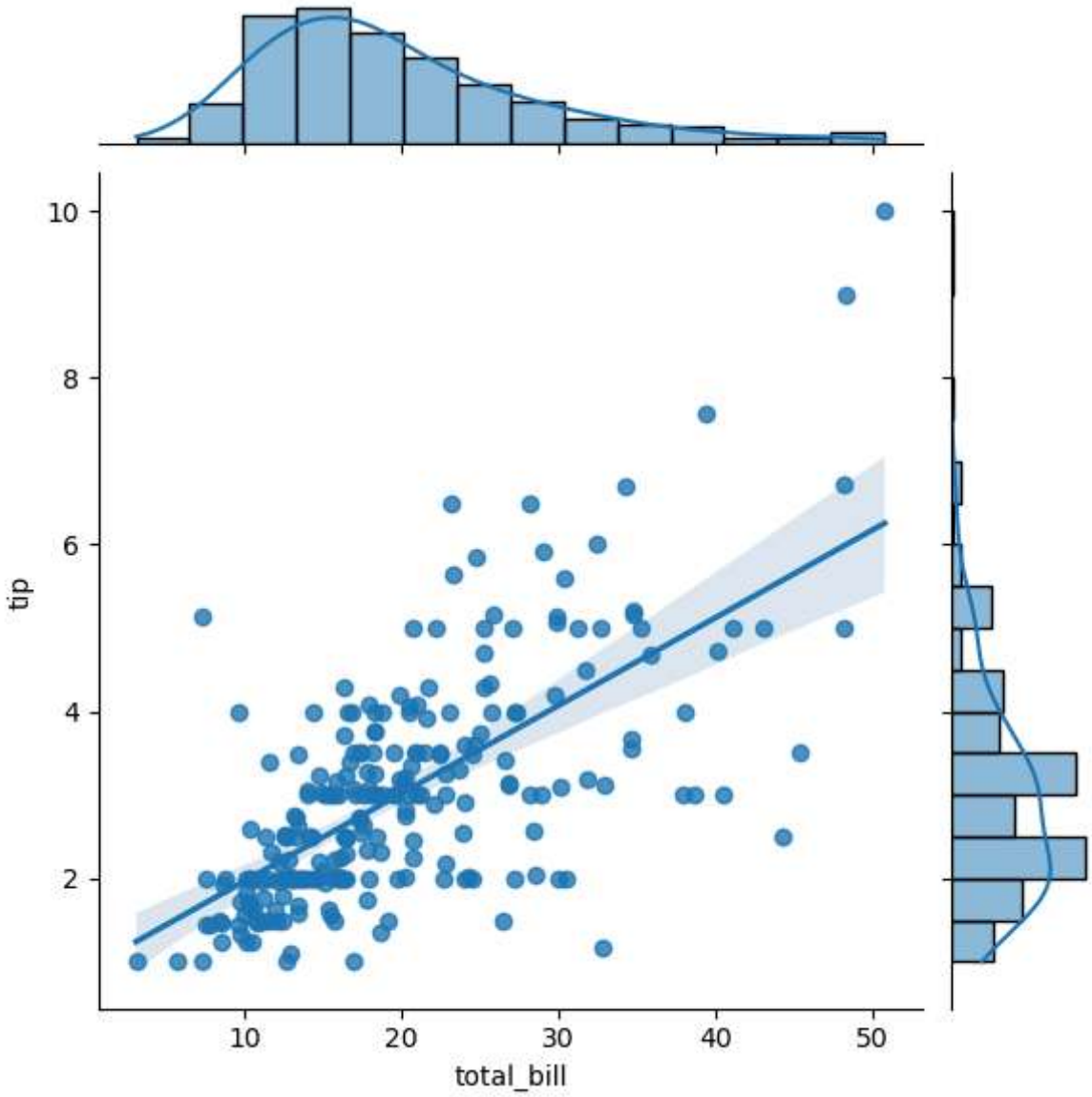
In [19]:
```
sns.jointplot(x='SepalLengthCm', y='SepalWidthCm', data=iris)
```

Out[19]:
```
<seaborn.axisgrid.JointGrid at 0x14aefc2e5c0>
```



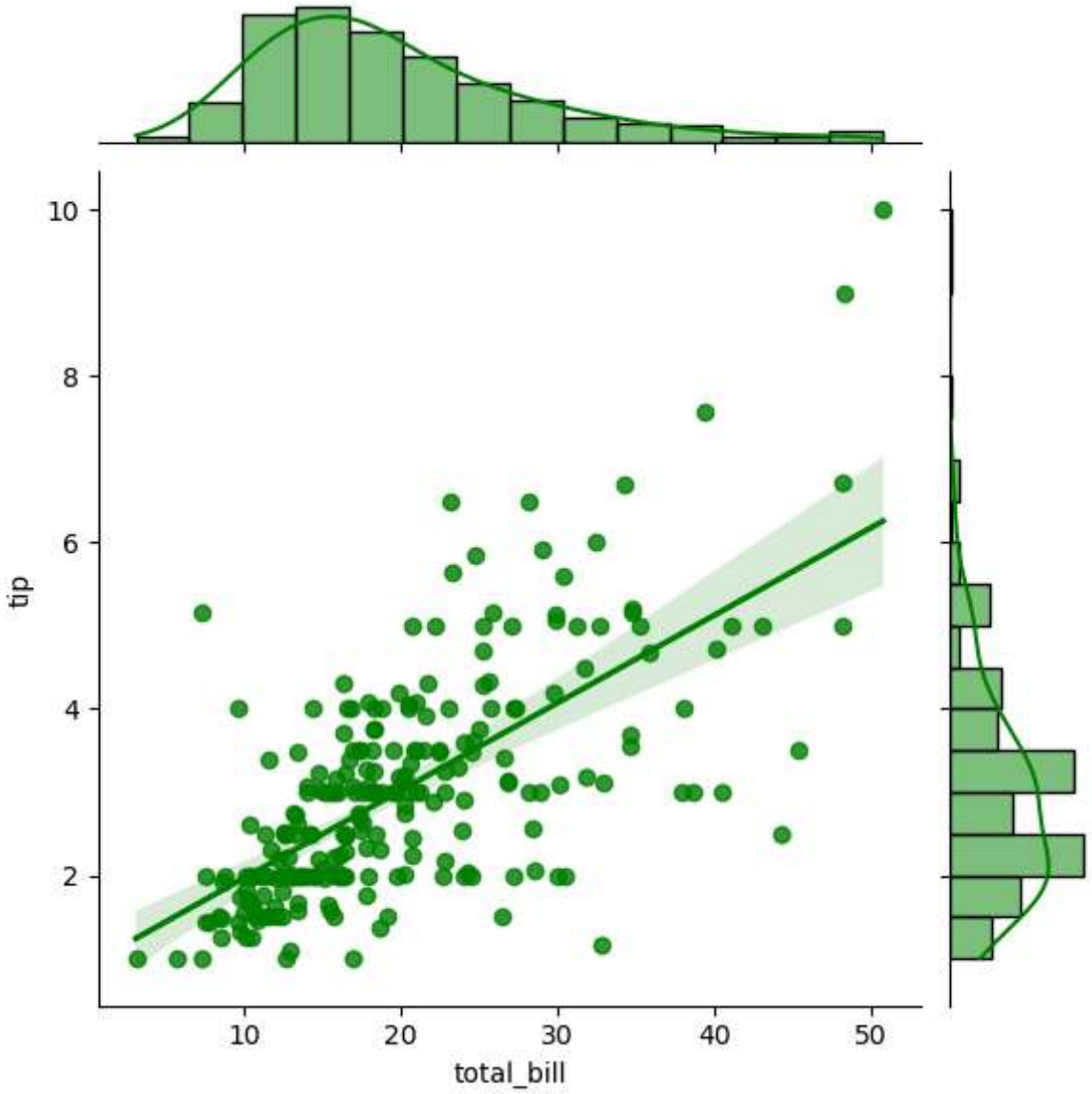
In [20]:
```
# adding regression lines
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg')
```

Out[20]: <seaborn.axisgrid.JointGrid at 0x14af00d03d0>



In [21]:
```
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg',color='green')
```

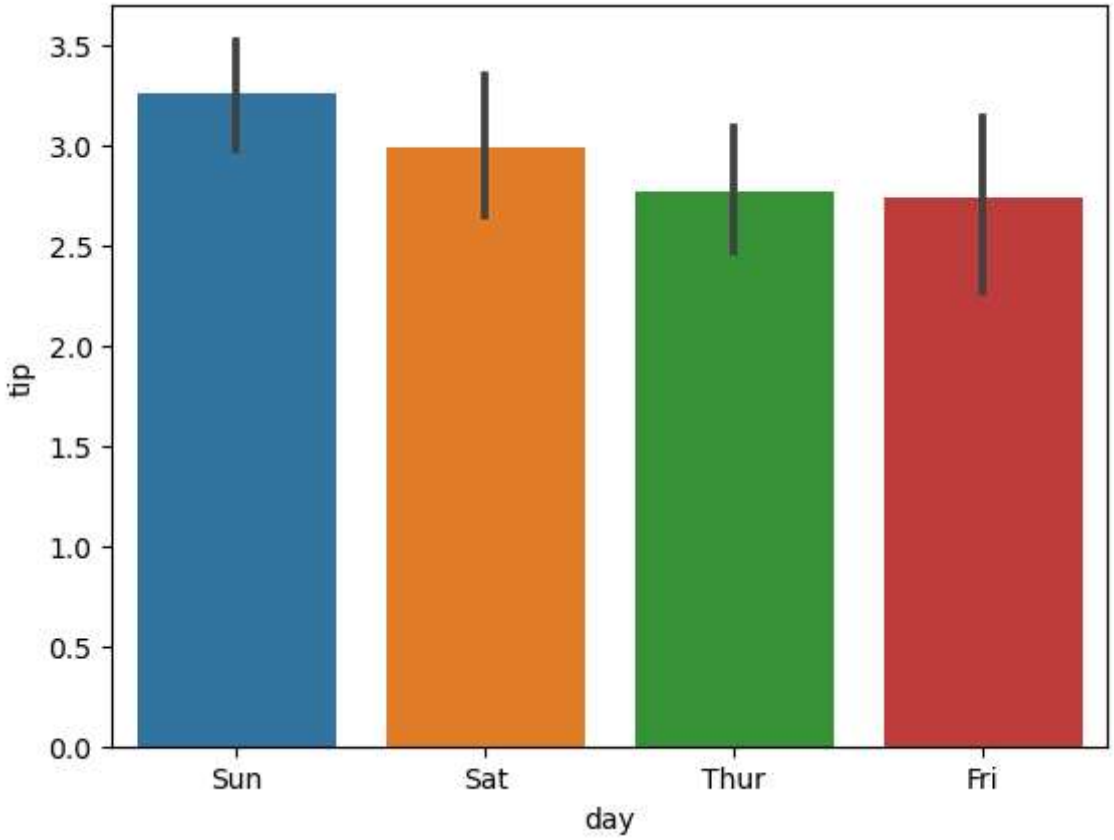
Out[21]:
```
<seaborn.axisgrid.JointGrid at 0x14af36a0520>
```



1.11 Bar Plots

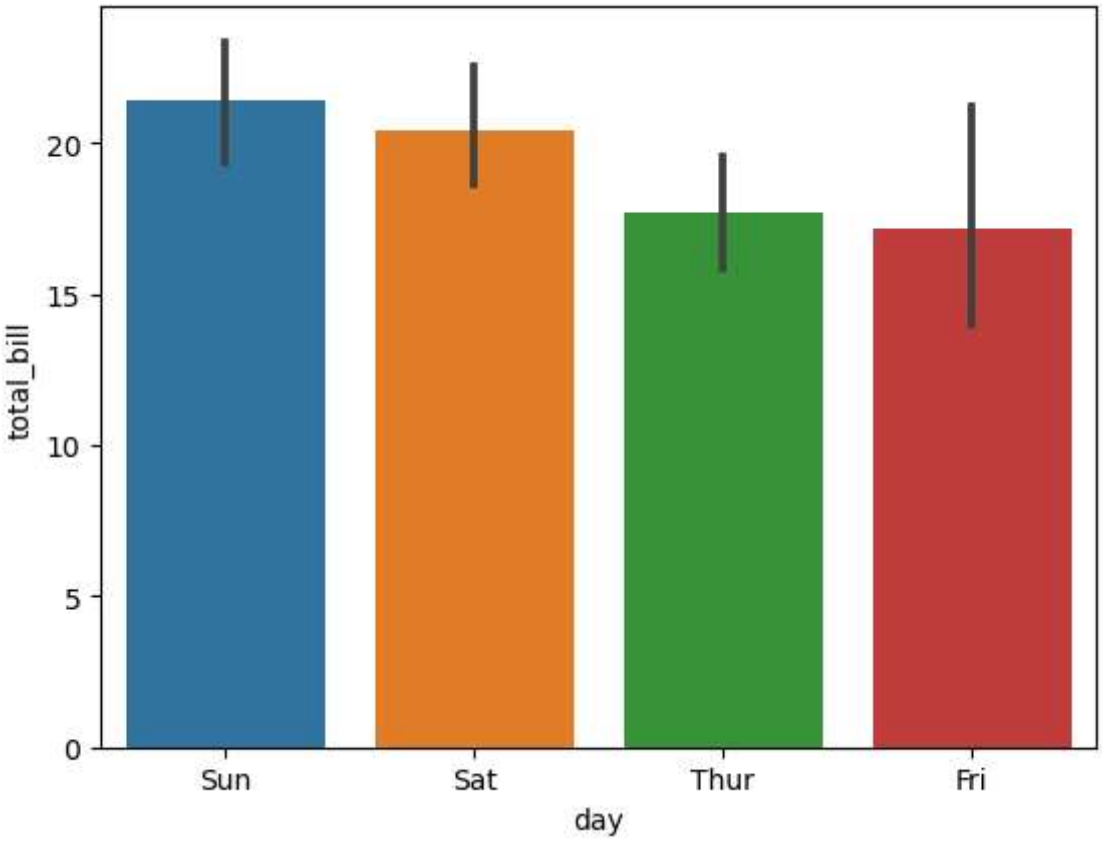
In [4]:
```
sns.barplot(x='day', y ='tip',data=tips)
```

Out[4]:
```
<Axes: xlabel='day', ylabel='tip'>
```



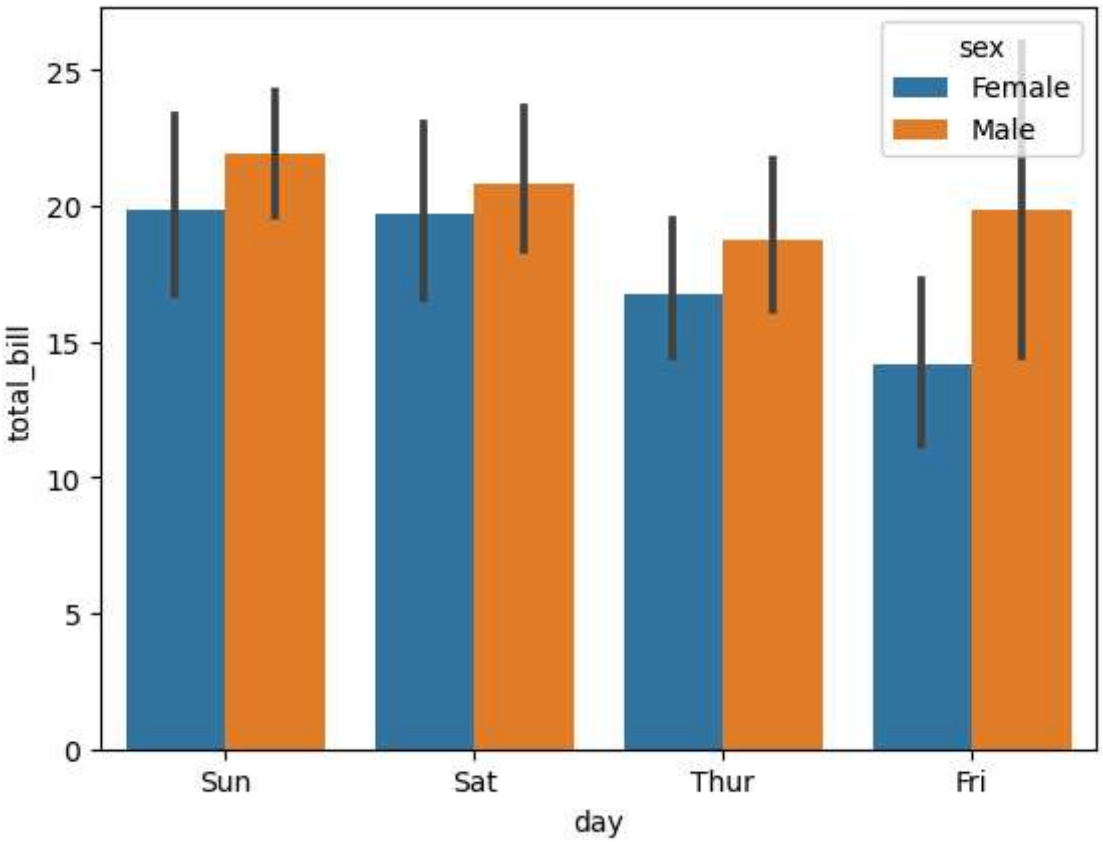
In [5]:
```
sns.barplot(x='day', y ='total_bill',data=tips)
```

Out[5]: <Axes: xlabel='day', ylabel='total_bill'>



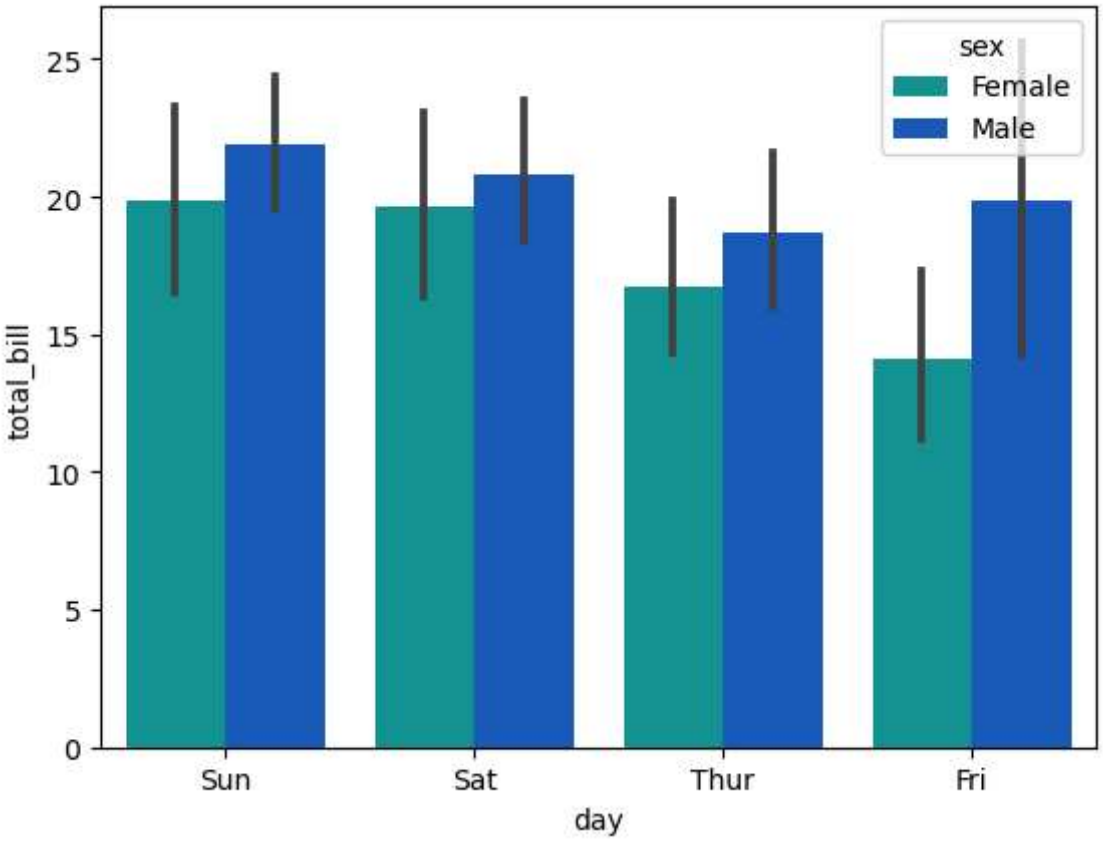
In [6]:
```
sns.barplot(x='day', y ='total_bill',data=tips,hue='sex')
```

Out[6]: <Axes: xlabel='day', ylabel='total_bill'>



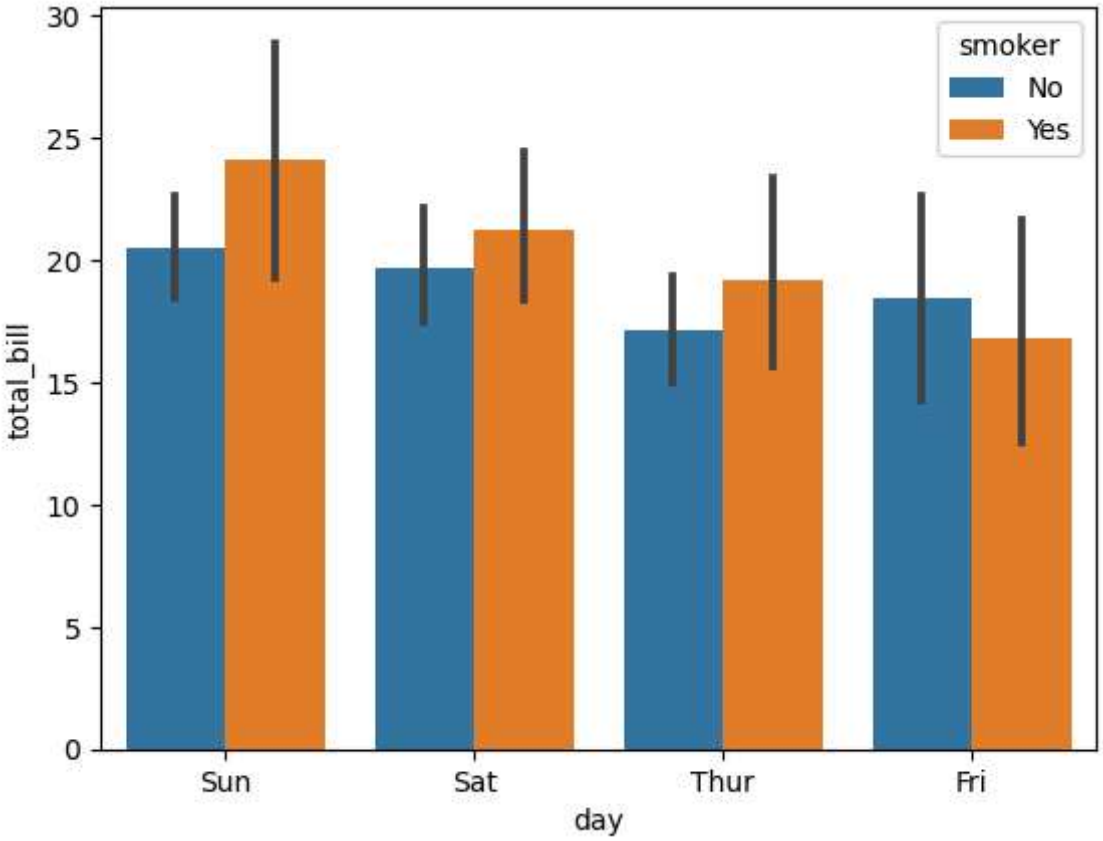
In [7]:
```
sns.barplot(x='day', y ='total_bill',data=tips,hue='sex', palette='winter_r')
```

Out[7]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



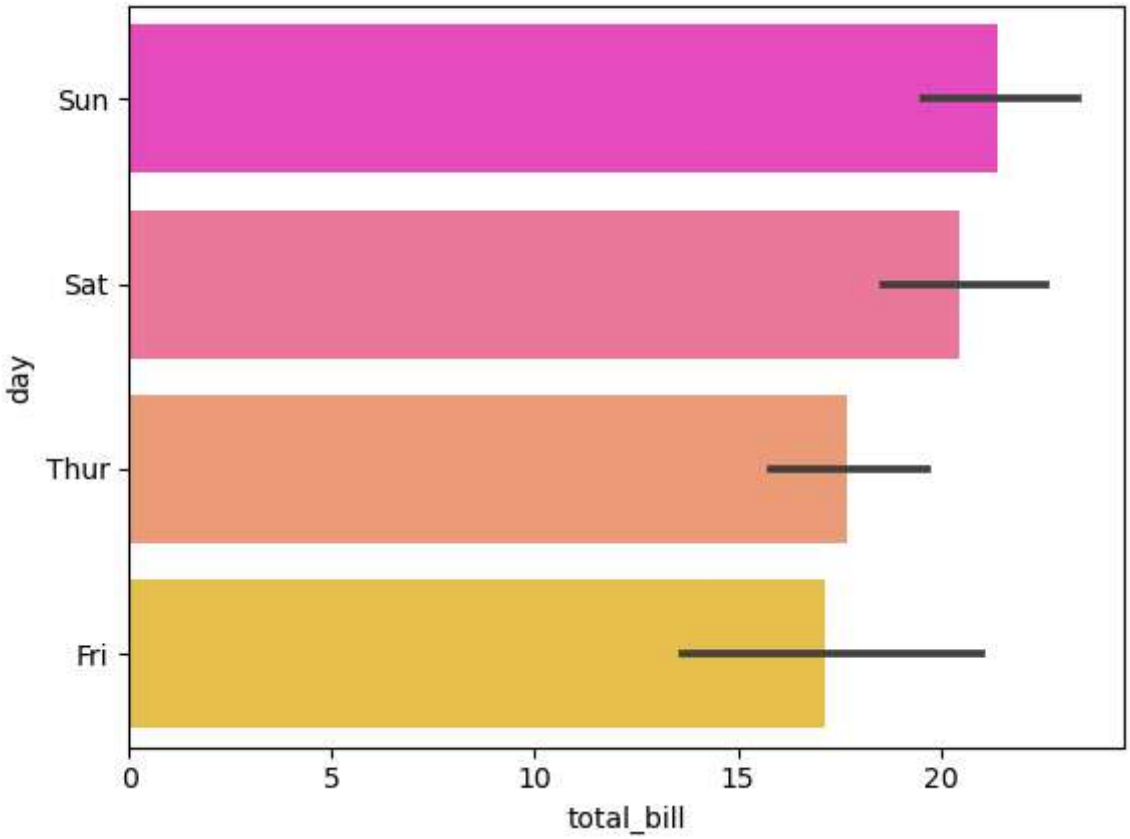
In [8]:
```
sns.barplot(x='day', y ='total_bill',data=tips,hue='smoker')
```

Out[8]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



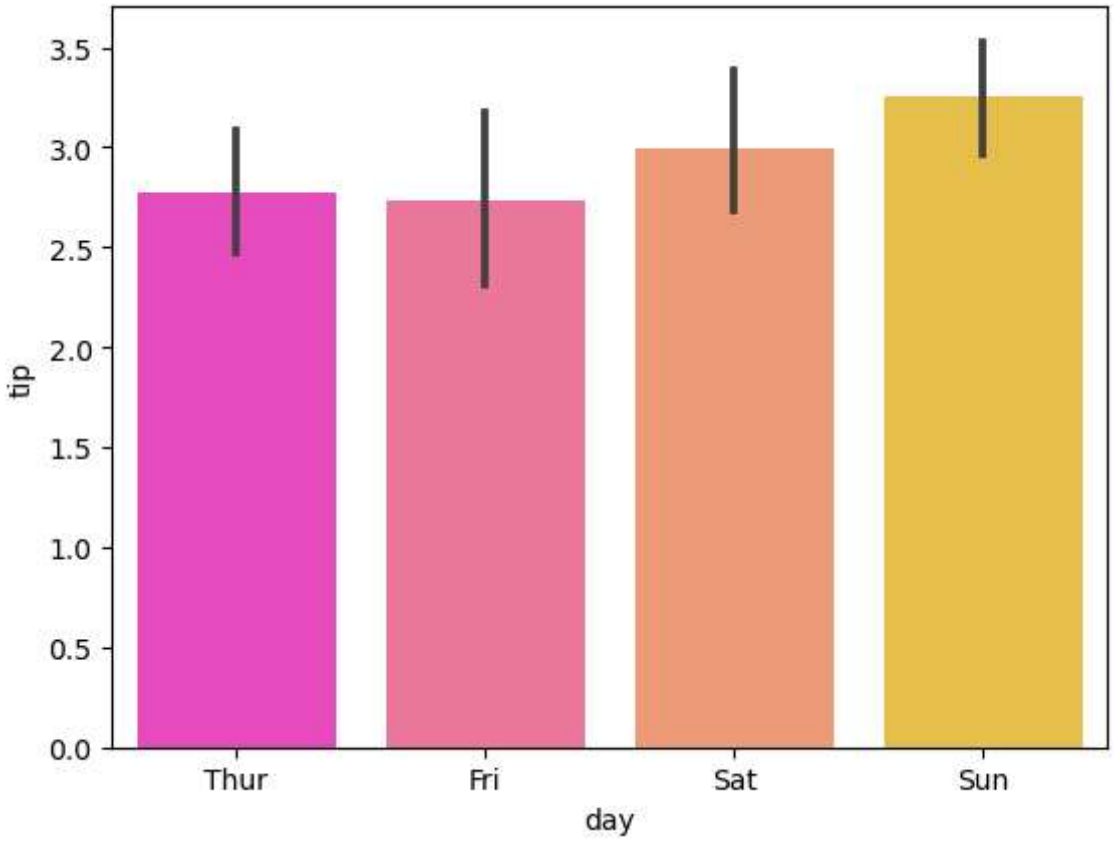
In [9]:
```
sns.barplot(x='total_bill', y ='day' , data = tips, palette ='spring')
```

Out[9]:
```
<Axes: xlabel='total_bill', ylabel='day'>
```



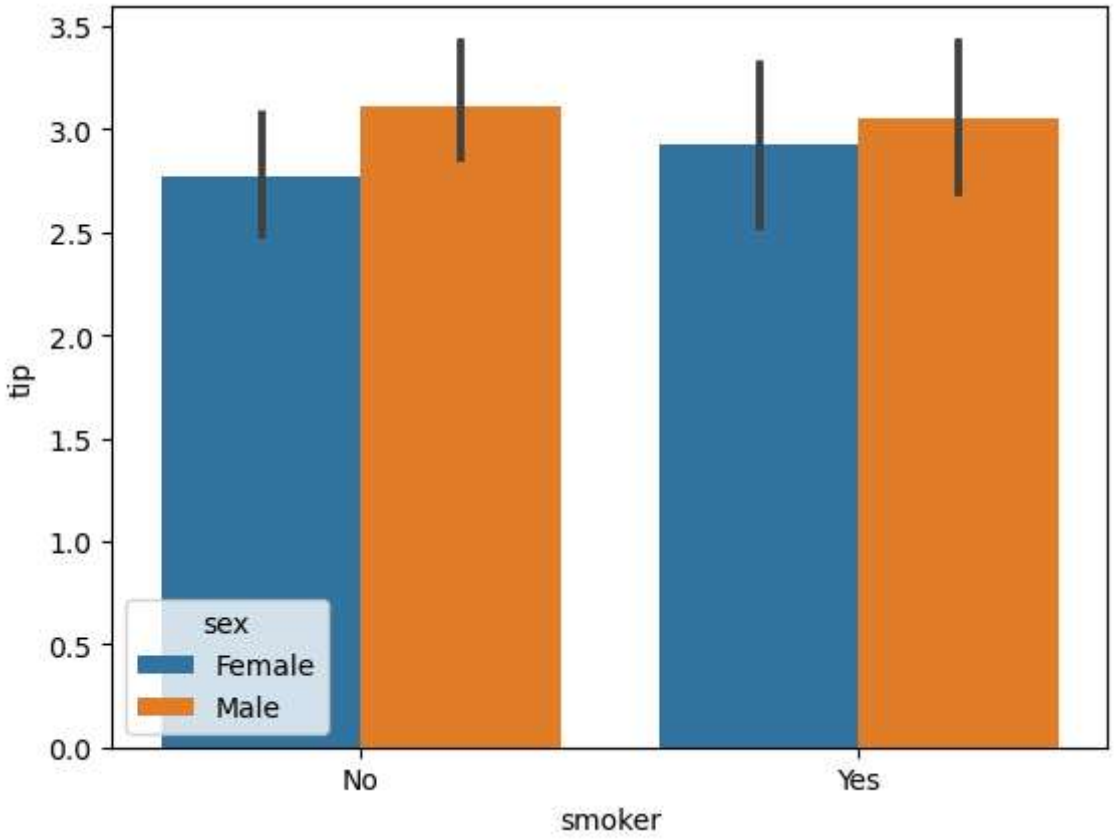
In [10]:
```
sns.barplot(x='day', y ='tip' , data = tips, palette
            ='spring',order=['Thur','Fri','Sat','Sun'])
```

Out[10]: <Axes: xlabel='day', ylabel='tip'>



In [11]:
```
sns.barplot(x='smoker', y ='tip' , data = tips,hue='sex')
```

Out[11]: <Axes: xlabel='smoker', ylabel='tip'>



In [ ]:
```

```