

EDA Basics

July 20, 2022

0.1 Deleting rows and columns from a dataframe

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

```
[2]: # reading an excel file
orders =pd.read_excel('Superstore_Sales.xls')
```

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

```
[3]:   Row ID  Order ID Order Date Order Priority  Order Quantity    Sales \
0      1      3 2010-10-13          Low           6    261.5400
1      2      6 2012-02-20  Not Specified           2      6.9300
2      3     32 2011-07-15          High          26   2808.0800
3      4     32 2011-07-15          High          24   1761.4000
4      5     32 2011-07-15          High          23    160.2335
```

```
   Discount  Ship Mode  Profit  Unit Price  ...  Customer Name \
0    0.04    Regular Air -213.250    38.94  ...  Muhammed MacIntyre
1    0.01    Regular Air  -4.640     2.08  ...      Ruben Dartt
2    0.07    Regular Air 1054.820   107.53  ...    Liz Pelletier
3    0.09  Delivery Truck -1748.560    70.89  ...    Liz Pelletier
4    0.04    Regular Air  -85.129     7.99  ...    Liz Pelletier
```

```
   Province  Region Customer Segment Product Category \
0  Nunavut  Nunavut   Small Business  Office Supplies
1  Alberta   West      Corporate  Office Supplies
2  Alberta   West      Corporate    Furniture
3  Alberta   West      Corporate    Furniture
4  Alberta   West      Corporate    Technology
```

```
   Product Sub-Category \
0    Storage & Organization
1  Scissors, Rulers and Trimmers
2      Office Furnishings
```

```

3           Tables
4   Telephones and Communication

```

```

                                Product Name Product Container \
0   Eldon Base for stackable storage shelf, platinum      Large Box
1   Kleencut® Forged Office Shears by Acme United ...      Small Pack
2   Tenex Contemporary Contur Chairmats for Low an...      Medium Box
3                                   KI Conference Tables      Jumbo Box
4                                   Bell Sonecor JB700 Caller ID      Medium Box

```

```

Product Base Margin  Ship Date
0                   0.80 2010-10-20
1                   0.55 2012-02-21
2                   0.65 2011-07-17
3                   0.72 2011-07-16
4                   0.60 2011-07-17

```

[5 rows x 21 columns]

```

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

```

```

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

```

```

[5]:   Order ID Order Date Order Priority  Order Quantity    Sales  Discount \
0         3 2010-10-13           Low             6    261.5400     0.04
1         6 2012-02-20  Not Specified             2     6.9300     0.01
2        32 2011-07-15           High            26   2808.0800     0.07
3        32 2011-07-15           High            24   1761.4000     0.09
4        32 2011-07-15           High            23    160.2335     0.04

```

```

                Ship Mode    Profit  Unit Price  Shipping Cost    Customer Name \
0   Regular Air   -213.250      38.94       35.00  Muhammed MacIntyre
1   Regular Air    -4.640       2.08        2.56    Ruben Dartt
2   Regular Air  1054.820     107.53        5.81    Liz Pelletier
3  Delivery Truck -1748.560      70.89       89.30    Liz Pelletier
4   Regular Air   -85.129       7.99        5.03    Liz Pelletier

```

```

Province  Region Customer Segment Product Category \
0  Nunavut  Nunavut   Small Business  Office Supplies
1  Alberta   West      Corporate    Office Supplies
2  Alberta   West      Corporate    Furniture
3  Alberta   West      Corporate    Furniture
4  Alberta   West      Corporate    Technology

```

```

Product Sub-Category \

```

```

0      Storage & Organization
1  Scissors, Rulers and Trimmers
2      Office Furnishings
3      Tables
4  Telephones and Communication

```

```

                                Product Name Product Container \
0  Eldon Base for stackable storage shelf, platinum      Large Box
1  Kleencut® Forged Office Shears by Acme United ...      Small Pack
2  Tenex Contemporary Contur Chairmats for Low an...      Medium Box
3                                KI Conference Tables      Jumbo Box
4                                Bell Sonecor JB700 Caller ID      Medium Box

```

```

Product Base Margin  Ship Date
0                    0.80 2010-10-20
1                    0.55 2012-02-21
2                    0.65 2011-07-17
3                    0.72 2011-07-16
4                    0.60 2011-07-17

```

```

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

```

```

[7]: orders.head()

```

```

[7]:   Order ID  Order Quantity    Sales  Discount  Ship Mode  Profit \
0         3           6  261.5400    0.04  Regular Air  -213.250
1         6           2   6.9300    0.01  Regular Air   -4.640
2        32          26 2808.0800    0.07  Regular Air 1054.820
3        32          24 1761.4000    0.09  Delivery Truck -1748.560
4        32          23  160.2335    0.04  Regular Air  -85.129

```

```

Unit Price  Shipping Cost    Customer Name Province  Region \
0    38.94      35.00  Muhammed MacIntyre  Nunavut  Nunavut
1     2.08       2.56    Ruben Dartt  Alberta  West
2   107.53       5.81    Liz Pelletier  Alberta  West
3    70.89      89.30    Liz Pelletier  Alberta  West
4     7.99       5.03    Liz Pelletier  Alberta  West

```

```

Customer Segment Product Category    Product Sub-Category \
0  Small Business  Office Supplies    Storage & Organization
1    Corporate  Office Supplies  Scissors, Rulers and Trimmers
2    Corporate    Furniture    Office Furnishings
3    Corporate    Furniture    Tables
4    Corporate  Technology  Telephones and Communication

```

```

                                Product Name Product Container \

```

| | Product | Base Margin | Ship Date |
|---|---------|-------------|------------|
| 0 | | 0.80 | 2010-10-20 |
| 1 | | 0.55 | 2012-02-21 |
| 2 | | 0.65 | 2011-07-17 |
| 3 | | 0.72 | 2011-07-16 |
| 4 | | 0.60 | 2011-07-17 |

```
[9]: orders.head()
```

| | Unit Price | Shipping Cost | Customer Name | Province | Region \ |
|---|------------|---------------|--------------------|--------------|----------|
| 0 | 38.94 | 35.00 | Muhammed MacIntyre | Nunavut | Nunavut |
| 2 | 107.53 | 5.81 | Liz Pelletier | Alberta | West |
| 3 | 70.89 | 89.30 | Liz Pelletier | Alberta | West |
| 4 | 7.99 | 5.03 | Liz Pelletier | Alberta | West |
| 5 | 8.46 | 8.99 | Liz Pelletier | Saskatchewan | Prarie |

| | Product Name | Product Container |
|---|---|-------------------|
| 0 | Eldon Base for stackable storage shelf, platinum | Large Box |
| 2 | Tenex Contemporary Contur Chairmats for Low an... | Medium Box |
| 3 | KI Conference Tables | Jumbo Box |
| 4 | Bell Sonacor JB700 Caller ID | Medium Box |
| 5 | Imation 3.5 IBM Diskettes, 10/Box | Small Pack |

4

```

0          0.80 2010-10-20
2          0.65 2011-07-17
3          0.72 2011-07-16
4          0.60 2011-07-17
5          0.79 2011-07-16

```

```

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

```

```

[11]: orders.head()

```

```

[11]:   Order ID  Order Quantity    Sales  Discount  Ship Mode  Profit \
0         3             6  261.5400      0.04  Regular Air -213.250
4        32            23  160.2335      0.04  Regular Air  -85.129
5        32            15  140.5600      0.04  Regular Air -128.380
6        35            30  288.5600      0.03  Regular Air   60.720
7        35            14 1892.8480      0.01  Regular Air   48.987

      Unit Price  Shipping Cost    Customer Name    Province  Region \
0       38.94       35.00  Muhammed MacIntyre    Nunavut  Nunavut
4        7.99        5.03    Liz Pelletier      Alberta    West
5        8.46        8.99    Liz Pelletier  Saskatchewan  Prarie
6        9.11        2.25   Julie Creighton  British Columbia  West
7       155.99        8.99   Julie Creighton  British Columbia  West

      Customer Segment  Product Category    Product Sub-Category \
0   Small Business  Office Supplies    Storage & Organization
4      Corporate    Technology  Telephones and Communication
5      Corporate    Technology    Computer Peripherals
6      Corporate  Office Supplies    Pens & Art Supplies
7      Corporate    Technology  Telephones and Communication

                                Product Name  Product Container \
0  Eldon Base for stackable storage shelf, platinum    Large Box
4                                Bell Sonecor JB700 Caller ID    Medium Box
5                                Imation 3.5 IBM Diskettes, 10/Box    Small Pack
6    Dixon Ticonderoga Core-Lock Colored Pencils    Wrap Bag
7                                CF 688    Small Box

      Product Base Margin  Ship Date
0          0.80 2010-10-20
4          0.60 2011-07-17
5          0.79 2011-07-16
6          0.52 2011-10-23
7          0.58 2011-10-24

```

1 Duplicate Values

```
[12]: raw_data = {
        "city":_
        ↪["Faridabad","Delhi","Faridabad","Noida","Noida","Faridabad","Noida","Delhi","Delhi"],
        "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
```

```
[12]:
```

| | 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 |

```
[13]: #check for duplicate data
df.duplicated()

# So, thereisno duplicate row in a dataframe
```

```
[13]: 0    False
      1    False
      2    False
      3    False
      4    False
      5    False
      6    False
      7    False
      8    False
      dtype: bool
```

```
[14]: #check for duplicate rows in city
df.duplicated(['city'])

#the first occurrences of data also treated as False
```

```
[14]: 0    False
      1    False
```

```
2    True
3    False
4    True
5    True
6    True
7    True
8    True
dtype: bool
```

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

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

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

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

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

```
[17]: 0    False
      1    False
      2     True
      3    False
      4    False
      5    False
      6     True
```

```

7     True
8     False
dtype: bool

```

```

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

```

```

[18]:
   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

```

```

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

```

```

[19]:
   city rank  score1  score2
0  Faridabad  1st     44     67
1    Delhi  2nd     48     63
3    Noida  2nd     41     70

```

```

[20]: df.drop_duplicates(['city','rank'])
      # all the duplicate rows are dropped on the basis of combination of city and
      ↪rank column

```

```

[20]:
   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

```

```

[21]: orders.describe()

```

```

[21]:
   Order ID  Order Quantity  Sales  Discount  Profit \
count  8396.000000      8396.000000  8396.000000  8396.000000  8396.000000
mean    29975.878394        25.574678  1775.967653    0.049669   181.332342
std    17254.682794        14.481362  3585.621358    0.031822  1196.642082
min         3.000000         1.000000    2.240000    0.000000 -14140.701600
25%    15033.750000        13.000000   143.242500    0.020000   -83.307500
50%    29860.500000        26.000000   449.295000    0.050000   -1.495000

```


| | | | | | |
|-----|--------------|-----------|--------------|----------|--------------|
| 75% | 44609.000000 | 38.000000 | 1705.432500 | 0.080000 | 162.707000 |
| max | 59973.000000 | 50.000000 | 89061.050000 | 0.250000 | 27220.690000 |

| | Unit Price | Shipping Cost | Product Base Margin |
|-------|-------------|---------------|---------------------|
| count | 8396.000000 | 8396.000000 | 8333.000000 |
| mean | 89.356685 | 12.831511 | 0.512467 |
| std | 290.404559 | 17.246422 | 0.135585 |
| min | 0.990000 | 0.490000 | 0.350000 |
| 25% | 6.480000 | 3.300000 | 0.380000 |
| 50% | 20.990000 | 6.070000 | 0.520000 |
| 75% | 85.990000 | 13.990000 | 0.590000 |
| max | 6783.020000 | 164.730000 | 0.850000 |

1.1 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 = $Q3 + (1.5 * IQR)$

```
[22]: data = pd.read_csv('house_prices.csv')
```

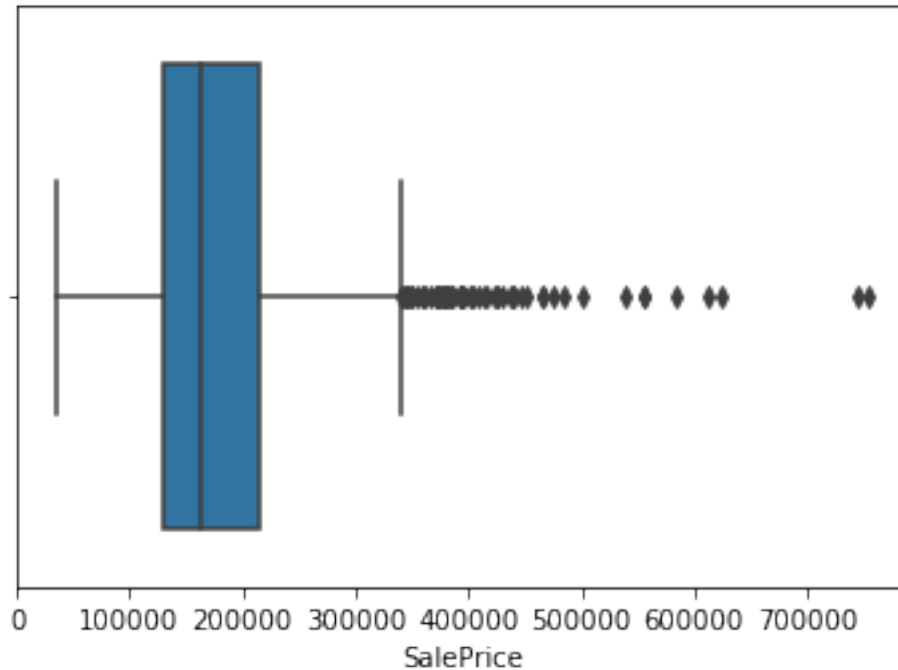
```
[23]: data['SalePrice'].describe()
```

```
[23]: count      1460.000000
      mean      180921.195890
      std       79442.502883
      min       34900.000000
      25%      129975.000000
      50%      163000.000000
      75%      214000.000000
      max       755000.000000
      Name: SalePrice, dtype: float64
```

```
[24]: import seaborn as sns
```

```
[25]: sns.boxplot(x=data['SalePrice'])
```

```
[25]: <AxesSubplot:xlabel='SalePrice'>
```



```
[26]: # Checking the shape of the data
data.shape
```

```
[26]: (1460, 81)
```

```
[27]: # Let's calculate first quartile, third quartile and inter quartile range

first_quartile = data['SalePrice'].quantile(.25)
third_quartile = data['SalePrice'].quantile(.75)
IQR = third_quartile - first_quartile
```

```
[28]: new_boundary = third_quartile + 3 * IQR
```

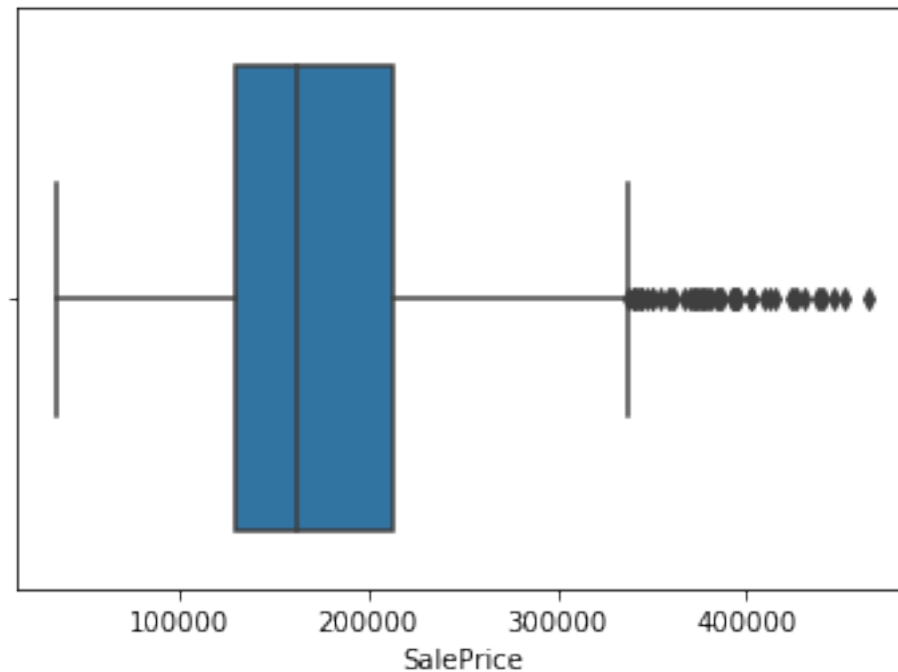
```
[29]: # dropped the outliers data
new_data = data.drop(data[data['SalePrice'] > new_boundary].index, axis = 0,
↳ inplace=False)
```

```
[30]: # 12 rows are dropped
new_data.shape
```

```
[30]: (1448, 81)
```

```
[31]: sns.boxplot(x=new_data['SalePrice'])
```

```
[31]: <AxesSubplot: xlabel='SalePrice'>
```



1.2 Outliers detection using IQR

```
[32]: # height of a persons
raw ={'name':
      ↳['mohan','maria','deepak','kunal','piyush','avinash','lisa','smita','tanu',
        ↳
        ↳'khusboo','nishant','johnson','donald','rakesh','pritvi','roy','ashish','abhishek','jassi',
          ↳'height':[1.2,2.3,4.9,5.1,5.2,5.4,5.5,5.5,5.6,5.6,5.8,5.9,6,6.1,6.2,6.5,7.
            ↳1,14.5,23.2,40.2]
      }
df =pd.DataFrame(raw)
```

```
[33]: df
```

```
[33]:
```

| | 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 |
| 15 | roy | 6.5 |
| 16 | ashish | 7.1 |
| 17 | abhishek | 14.5 |
| 18 | jassi | 23.2 |
| 19 | puneet | 40.2 |

```
[34]: df.describe()
```

```
[34]:          height
count  20.000000
mean    8.390000
std     8.782812
min     1.200000
25%     5.350000
50%     5.700000
75%     6.275000
max     40.200000
```

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

```
[35]: (5.3500000000000005, 6.275)
```

```
[36]: # Calculating IQR
IQR = Q3 - Q1
IQR
```

```
[36]: 0.9249999999999998
```

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

```
[37]: (3.9625000000000001, 7.6625)
```

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

```
[38]:      name  height
0    mohan    1.2
1    maria    2.3
17  abhishek   14.5
18    jassi   23.2
19  puneet   40.2
```

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

```
[40]: df_no_outliers
```

```
[40]:      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
15    roy    6.5
16  ashish    7.1
```

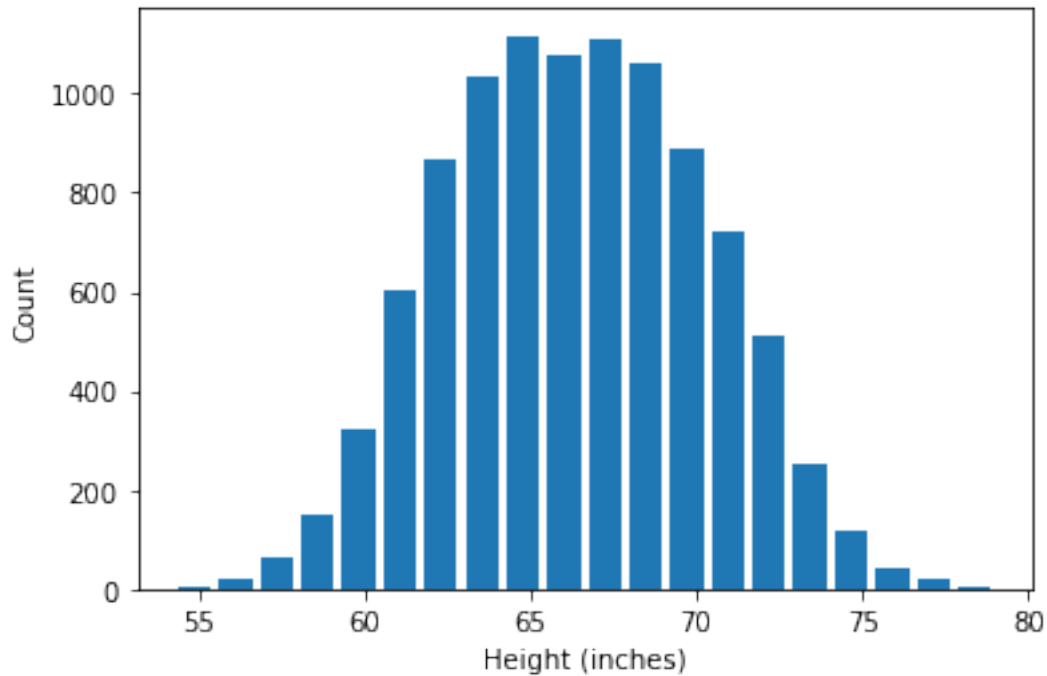
```
[ ]:
```

```
[41]: df =pd.read_csv("heights.csv")
```

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

```
[42]:      Gender      Height
3346    Male  67.880407
5767  Female  60.503327
8271  Female  65.632600
5869  Female  62.243849
3701    Male  69.481714
```

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



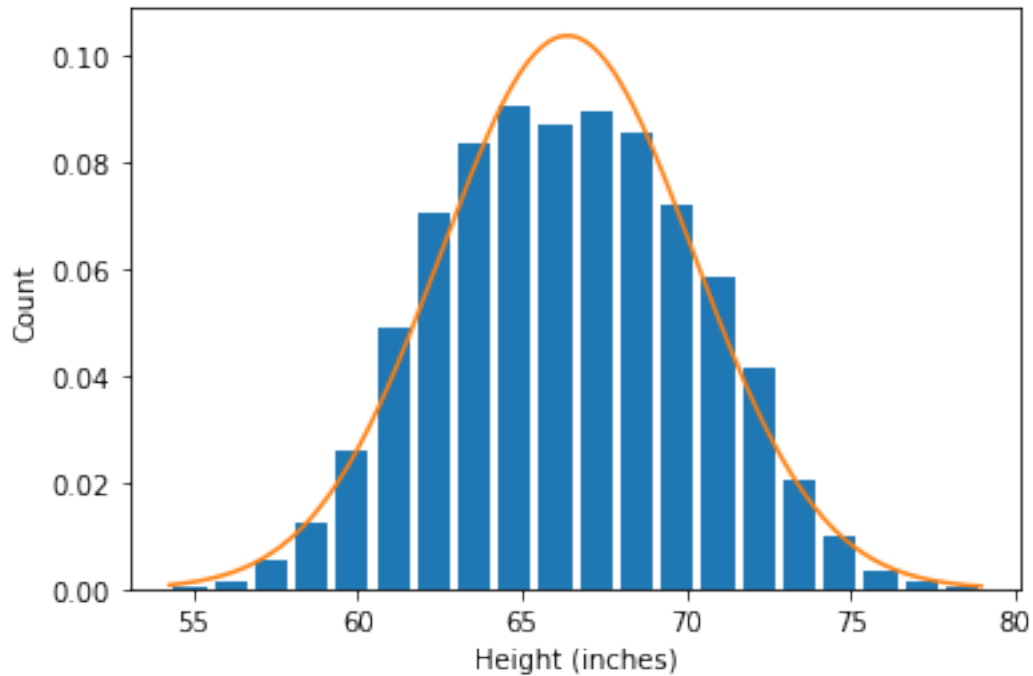
Refer to <https://www.mathsisfun.com/data/standard-normal-distribution.html> for more details

```
[44]: 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()))
```

```
[44]: [<matplotlib.lines.Line2D at 0x1b565155790>]
```



```
[45]: #Calculating Mean
df.Height.mean()
```

```
[45]: 66.3675597548656
```

```
[46]: #Calculating Standard Deviation
df.Height.std()
```

```
[46]: 3.847528120795573
```

```
[47]: #Calculating upper_limit
upper_limit = df.Height.mean() + 3 * df.Height.std()
upper_limit
```

```
[47]: 77.91014411725232
```

```
[48]: #Calculating lower_limit
lower_limit = df.Height.mean() - 3 * df.Height.std()
lower_limit
```

```
[48]: 54.824975392478876
```

```
[49]: # checking the outliers
df[(df.Height>upper_limit )| (df.Height <lower_limit)]
```

```
[49]:      Gender      Height
      994      Male  78.095867
      1317     Male  78.462053
      2014     Male  78.998742
      3285     Male  78.528210
      3757     Male  78.621374
      6624  Female  54.616858
      9285  Female  54.263133
```

```
[50]: df_no_outlier_std_dev = df[(df.Height<upper_limit ) & (df.Height >lower_limit)]
      df_no_outlier_std_dev.shape
```

```
[50]: (9993, 2)
```

```
[51]: df.shape[0] - df_no_outlier_std_dev.shape[0]
```

```
[51]: 7
```

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

$$1.4 \quad z = (x -) /$$

```
[52]: df['zscore'] = (df.Height - df.Height.mean())/df.Height.std()
```

```
[53]: df.head(5)
```

```
[53]:      Gender      Height      zscore
      0      Male  73.847017  1.943964
      1      Male  68.781904  0.627505
      2      Male  74.110105  2.012343
      3      Male  71.730978  1.393991
      4      Male  69.881796  0.913375
```

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

```
[54]:      Gender      Height      zscore
      994      Male  78.095867  3.048271
```



```

1317    Male    78.462053    3.143445
2014    Male    78.998742    3.282934
3285    Male    78.528210    3.160640
3757    Male    78.621374    3.184854
6624   Female    54.616858   -3.054091
9285   Female    54.263133   -3.146027

```

```

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

```

```

[55]:   Gender      Height      zscore
0    Male    73.847017    1.943964
1    Male    68.781904    0.627505
2    Male    74.110105    2.012343
3    Male    71.730978    1.393991
4    Male    69.881796    0.913375

```

```

[56]: #checkig the number of ouliers
df.shape[0] - df_no_outliers.shape[0]

```

```

[56]: 7

```

```

[57]: #Load the Dataset
df = pd.read_csv('imdb_1000.csv')

```

```

[58]: #preview of dataset
df.head()

```

```

[58]:   star_rating      title content_rating  genre  duration \
0          9.3  The Shawshank Redemption        R   Crime      142
1          9.2             The Godfather        R   Crime      175
2          9.1   The Godfather: Part II        R   Crime      200
3          9.0       The Dark Knight    PG-13  Action      152
4          8.9       Pulp Fiction        R   Crime      154

```

```

                                actors_list
0  [u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1  [u'Marlon Brando', u'Al Pacino', u'James Caan']
2  [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3  [u'Christian Bale', u'Heath Ledger', u'Aaron E...
4  [u'John Travolta', u'Uma Thurman', u'Samuel L...

```

```

[59]: df['genre']

```

```

[59]: 0      Crime
      1      Crime

```

```

2         Crime
3         Action
4         Crime
...
974        Comedy
975    Adventure
976        Action
977        Horror
978        Crime
Name: genre, Length: 979, dtype: object

```

```
[60]: from collections import Counter
```

```
[61]: Counter(df.genre)
```

```
[61]: Counter({'Crime': 124,
              'Action': 136,
              'Drama': 278,
              'Western': 9,
              'Adventure': 75,
              'Biography': 77,
              'Comedy': 156,
              'Animation': 62,
              'Mystery': 16,
              'Horror': 29,
              'Film-Noir': 3,
              'Sci-Fi': 5,
              'History': 1,
              'Thriller': 5,
              'Family': 2,
              'Fantasy': 1})
```

```
[62]: # to get the frequency
df.genre.value_counts()
```

```
[62]: Drama          278
      Comedy         156
      Action         136
      Crime          124
      Biography        77
      Adventure        75
      Animation        62
      Horror           29
      Mystery          16
      Western           9
      Sci-Fi           5
      Thriller          5
```

```
Film-Noir      3
Family         2
History        1
Fantasy        1
Name: genre, dtype: int64
```

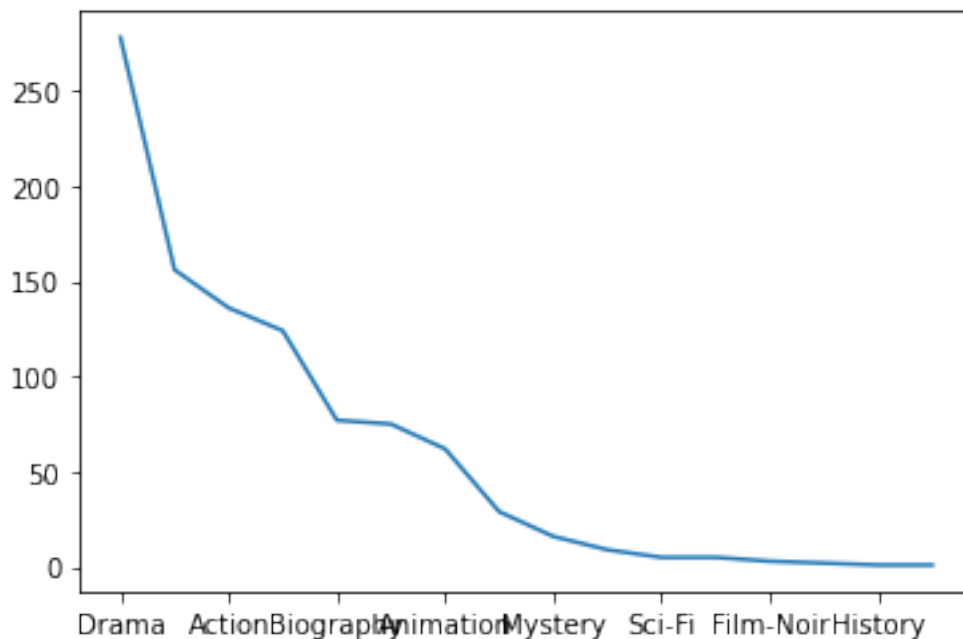
```
[63]: #checking the type
type(df.genre.value_counts())
```

```
[63]: pandas.core.series.Series
```

```
[64]: gc = df['genre'].value_counts()
```

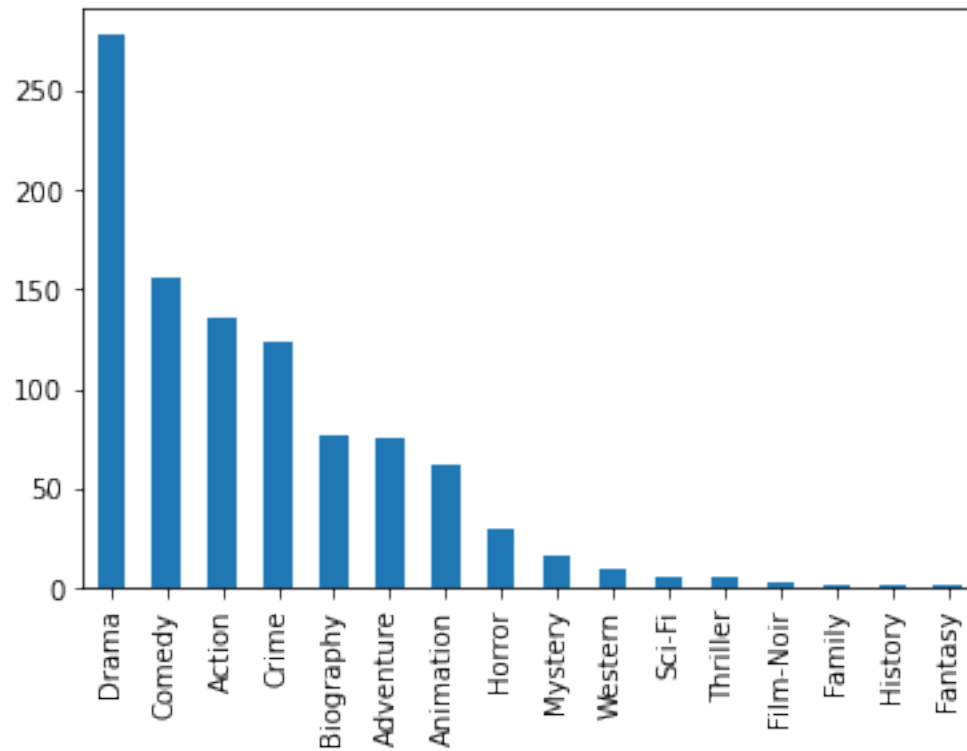
```
[65]: gc.plot()
```

```
[65]: <AxesSubplot:>
```



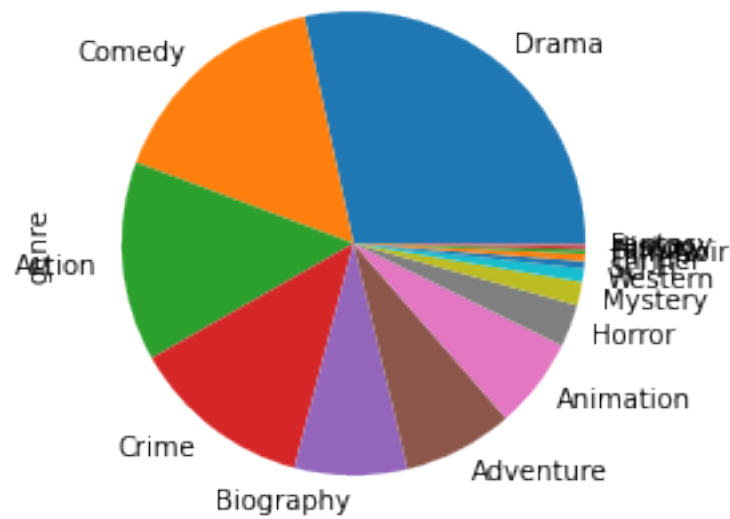
```
[66]: #Visualizing through a bar chart
gc.plot(kind='bar')
```

```
[66]: <AxesSubplot:>
```



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

```
[67]: <AxesSubplot:ylabel='genre'>
```



Refer to <https://medium.com/analytics-vidhya/intro-to-univariate-analysis-de75454b4719> for more details

```
[68]: titanic_df = pd.read_csv('titanic_df.csv')
```

```
[69]: 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
```

```
[70]: #checking top 5 records of the dataset
titanic_df.head()
```

```
[70]:
```

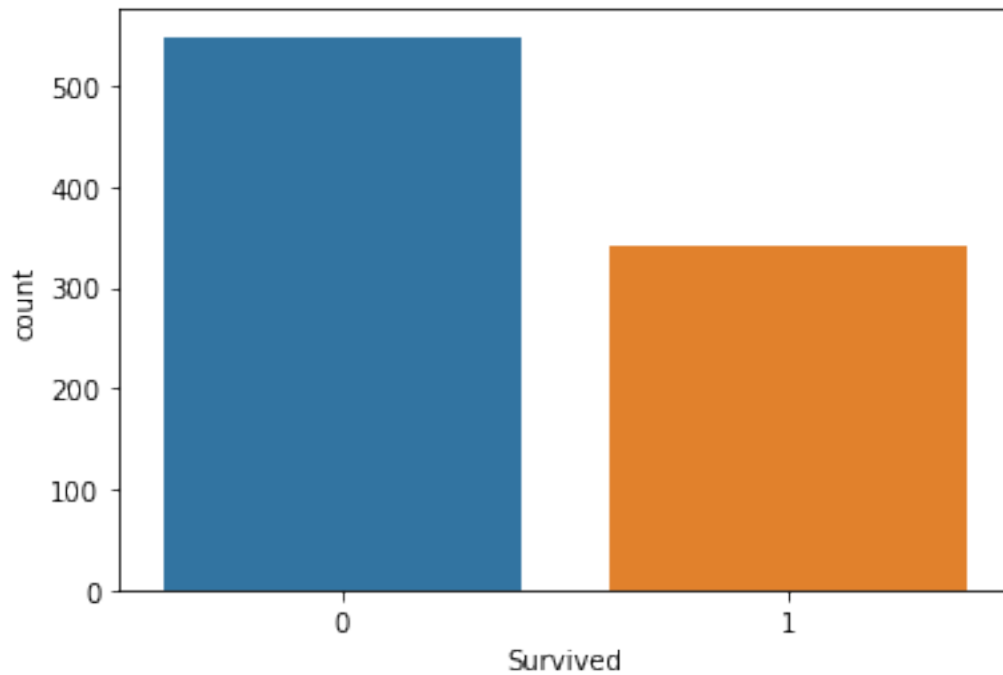
| | PassengerId | Survived | Pclass | \ |
|---|-------------|----------|--------|---|
| 0 | 1 | 0 | 3 | |
| 1 | 2 | 1 | 1 | |
| 2 | 3 | 1 | 3 | |
| 3 | 4 | 1 | 1 | |
| 4 | 5 | 0 | 3 | |

| | Name | Sex | Age | SibSp | \ |
|---|---|--------|------|-------|---|
| 0 | Braund, Mr. Owen Harris | male | 22.0 | 1 | |
| 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | |
| 2 | Heikkinen, Miss. Laina | female | 26.0 | 0 | |
| 3 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | |
| 4 | Allen, Mr. William Henry | male | 35.0 | 0 | |

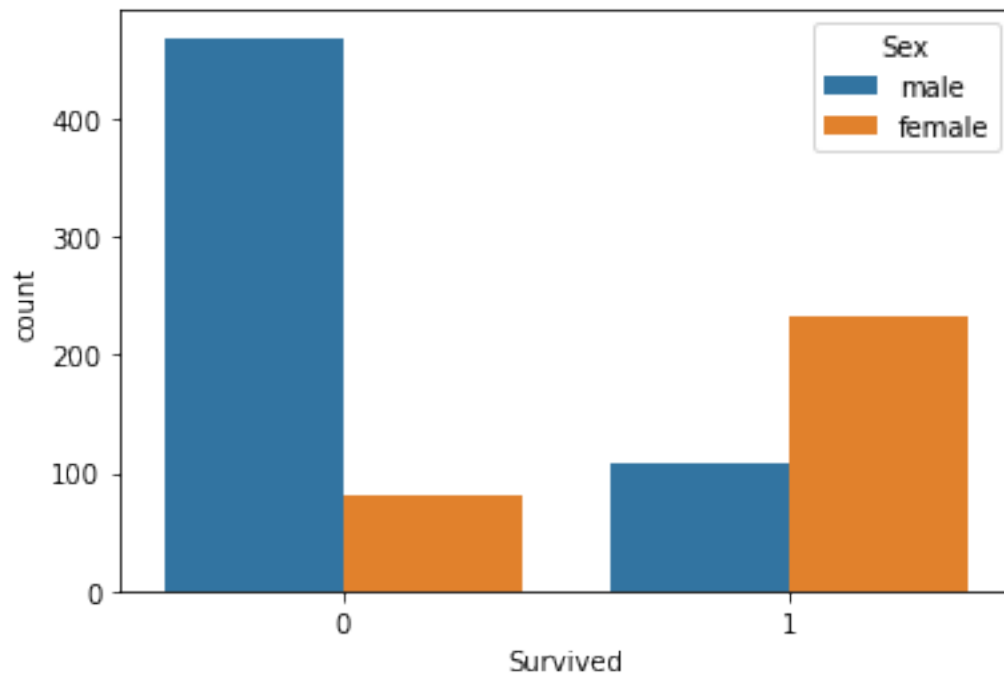
| | Parch | Ticket | Fare | Cabin | Embarked |
|--|-------|--------|------|-------|----------|
|--|-------|--------|------|-------|----------|

| | | | | | | |
|---|---|----------|---------|---------|------|---|
| 0 | 0 | A/5 | 21171 | 7.2500 | NaN | S |
| 1 | 0 | PC | 17599 | 71.2833 | C85 | C |
| 2 | 0 | STON/O2. | 3101282 | 7.9250 | NaN | S |
| 3 | 0 | | 113803 | 53.1000 | C123 | S |
| 4 | 0 | | 373450 | 8.0500 | NaN | S |

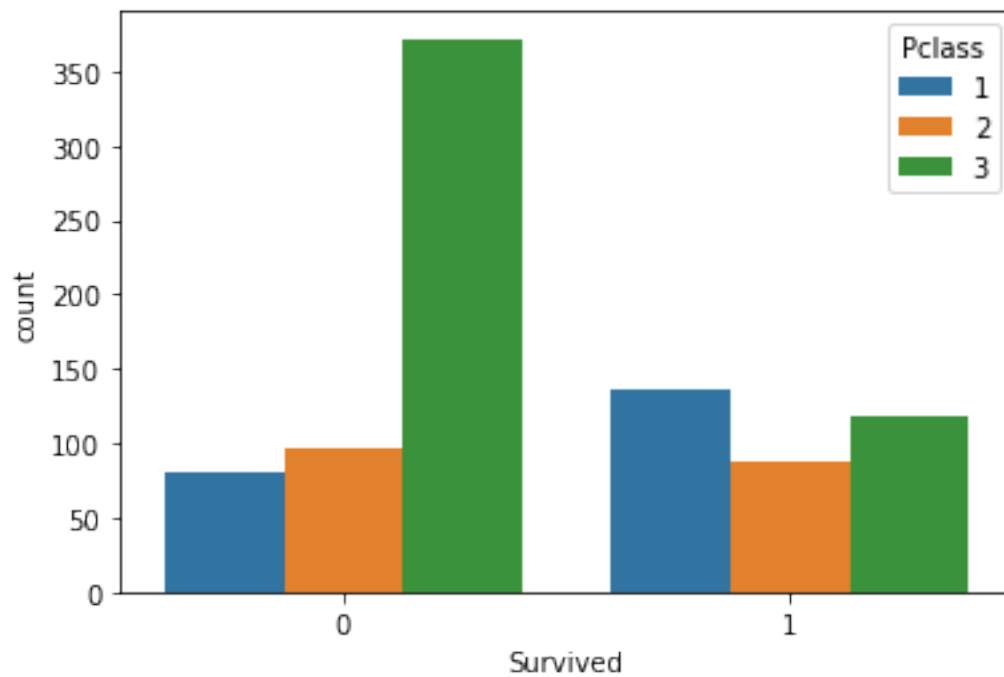
```
[71]: #Let's visualize the survivors count
sns.countplot(x=titanic_df.Survived)
plt.show()
```



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



```
[73]: # 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

```
[74]: tips_df = pd.read_csv('tips.csv')
```

```
[75]: tips_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 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
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

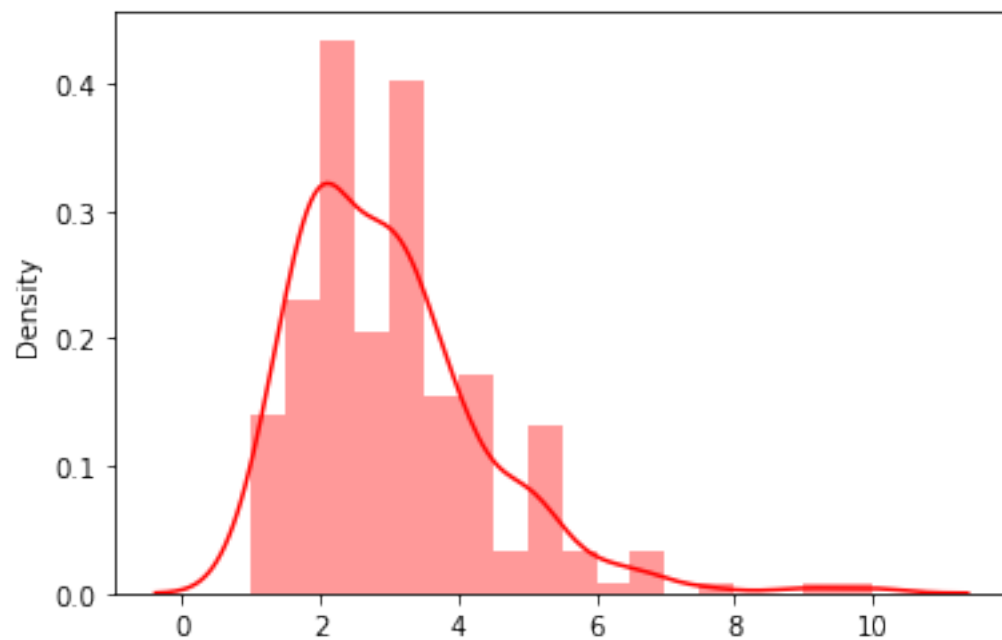
```
[76]: #checking top 5 records of the dataset
tips_df.head()
```

```
[76]:
```

| | total_bill | tip | sex | smoker | day | time | size |
|---|------------|------|--------|--------|-----|--------|------|
| 0 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 1 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 2 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 3 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 4 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

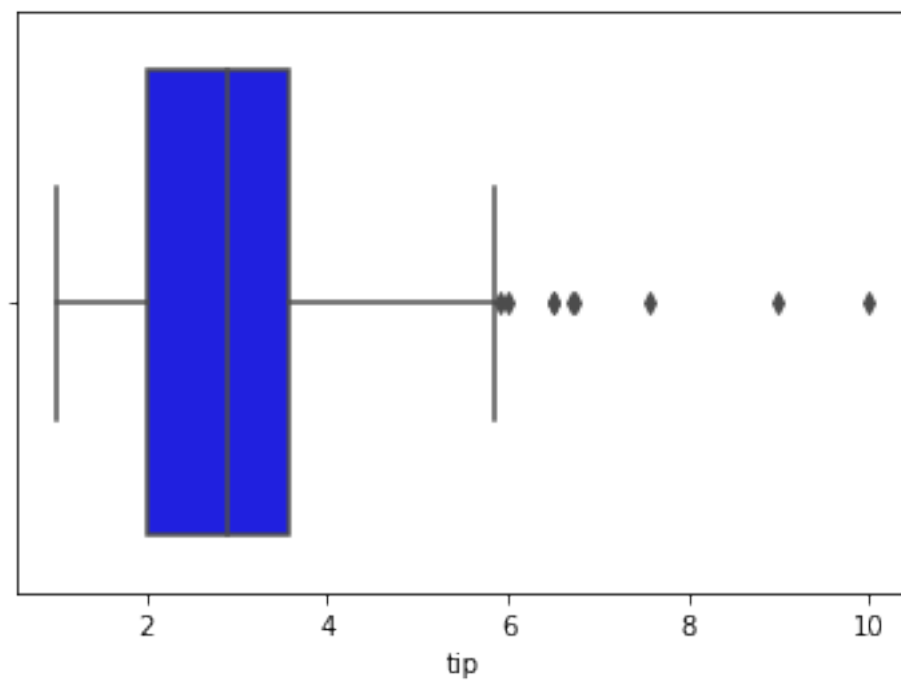
```
[77]: #distribution of tips
sns.distplot(x=tips_df.tip, hist=True, kde=True, color='r')
```

```
[77]: <AxesSubplot:ylabel='Density'>
```

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

```
[78]: <AxesSubplot:xlabel='tip'>
```



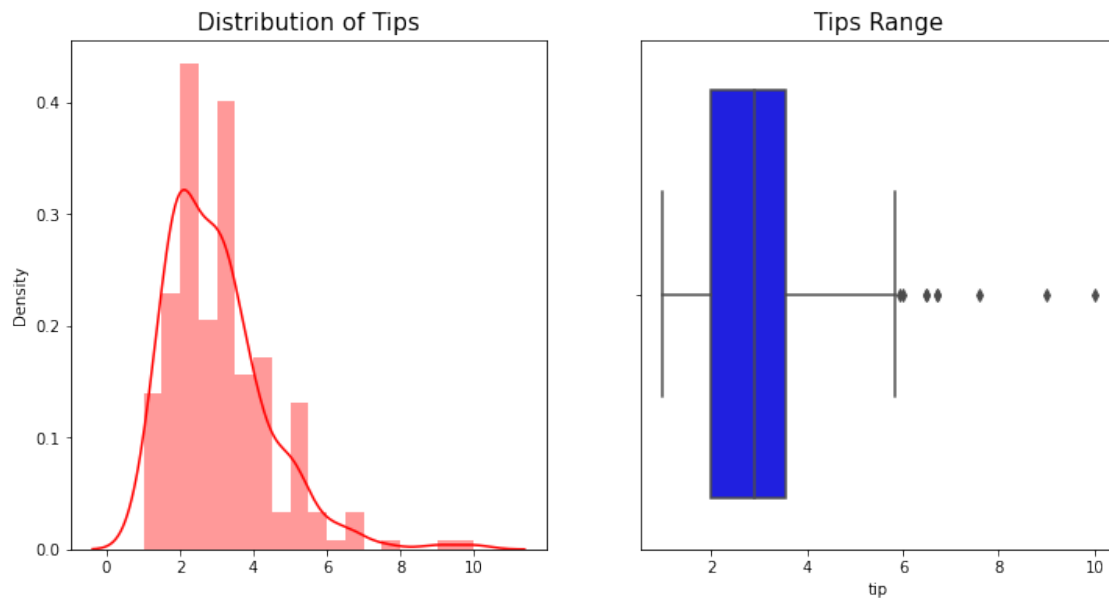
```
[79]: fig, axes = plt.subplots(1,2, figsize =(12,6))

#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)
```

```
[79]: Text(0.5, 1.0, 'Tips Range')
```

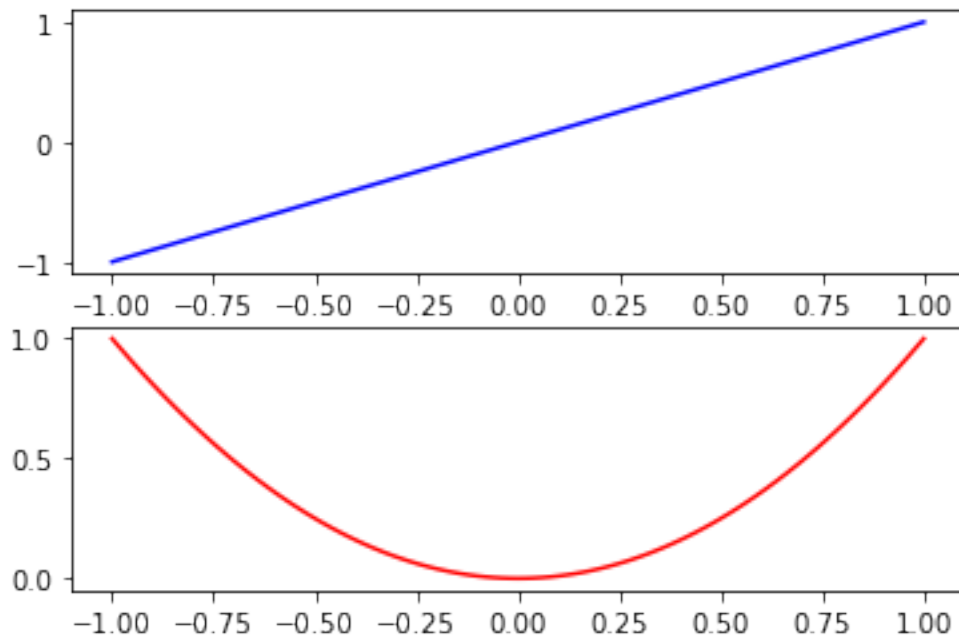


1.5 Subplots

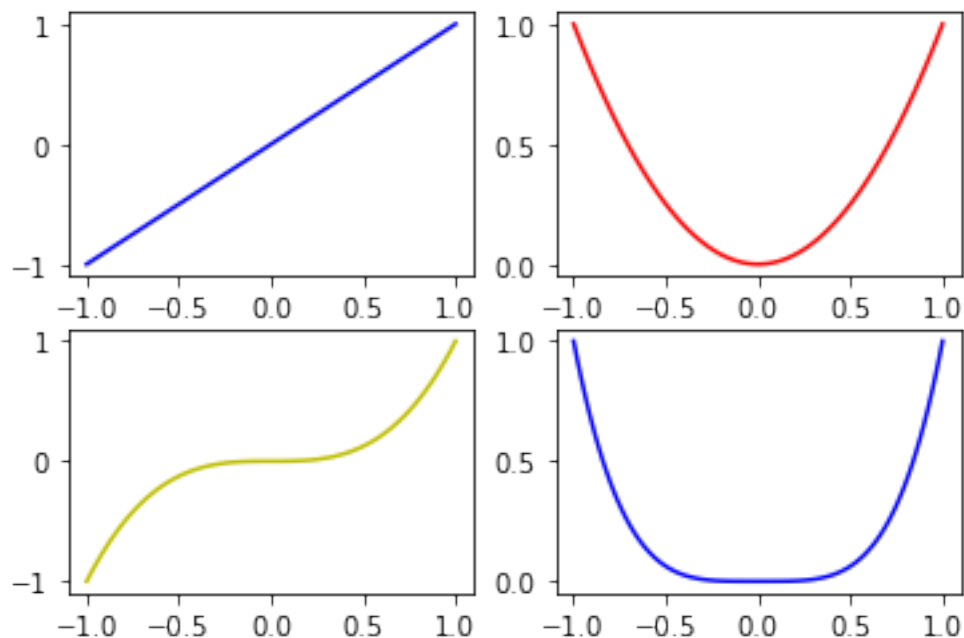
```
[80]: x = np.linspace(-1,1,101)
y1 = x
y2 = x**2
y3 = x**3
y4 = x**4
```

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

[81]: [



```
[82]: 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()
```



```
[83]: #Load the dataset
diamonds = pd.read_csv('diamonds.csv')
```

```
[84]: diamonds.head()
```

```
[84]:
```

| | Unnamed: 0 | carat | cut | color | clarity | depth | table | price | x | y | \ |
|---|------------|-------|---------|-------|---------|-------|-------|-------|------|------|---|
| 0 | 1 | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | |
| 1 | 2 | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | |
| 2 | 3 | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | |
| 3 | 4 | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | |
| 4 | 5 | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | |

| | z |
|---|------|
| 0 | 2.43 |
| 1 | 2.31 |
| 2 | 2.31 |
| 3 | 2.63 |
| 4 | 2.75 |

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

```
[85]: (53940, 11)
```

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

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

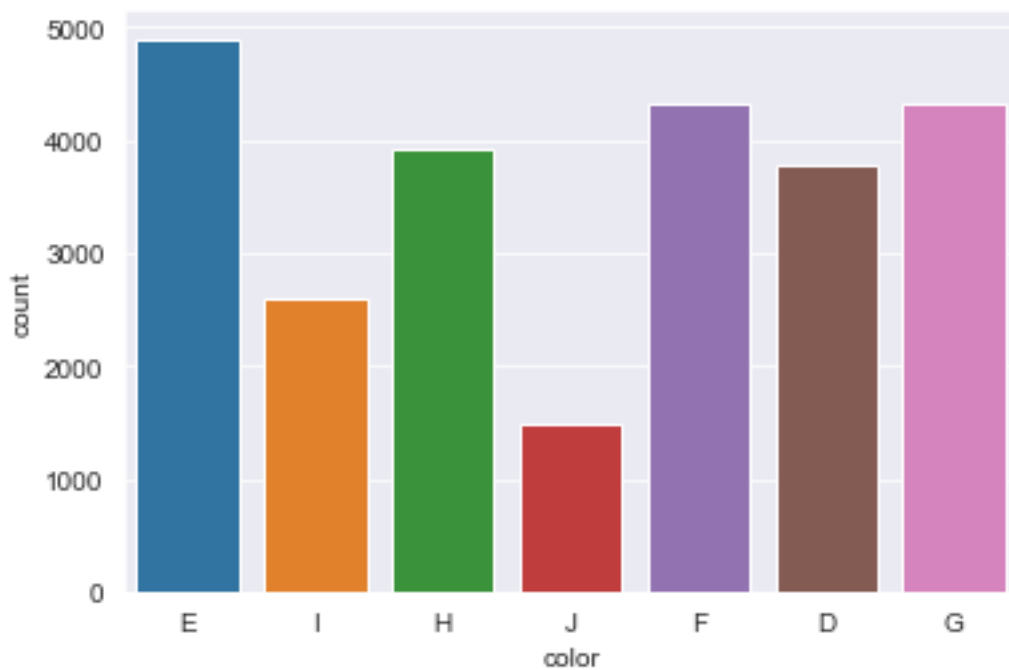
```
[87]: (25323, 11)
```

Countplots

```
[88]: sns.set_style('darkgrid')
```

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

```
[89]: <AxesSubplot:xlabel='color', ylabel='count'>
```

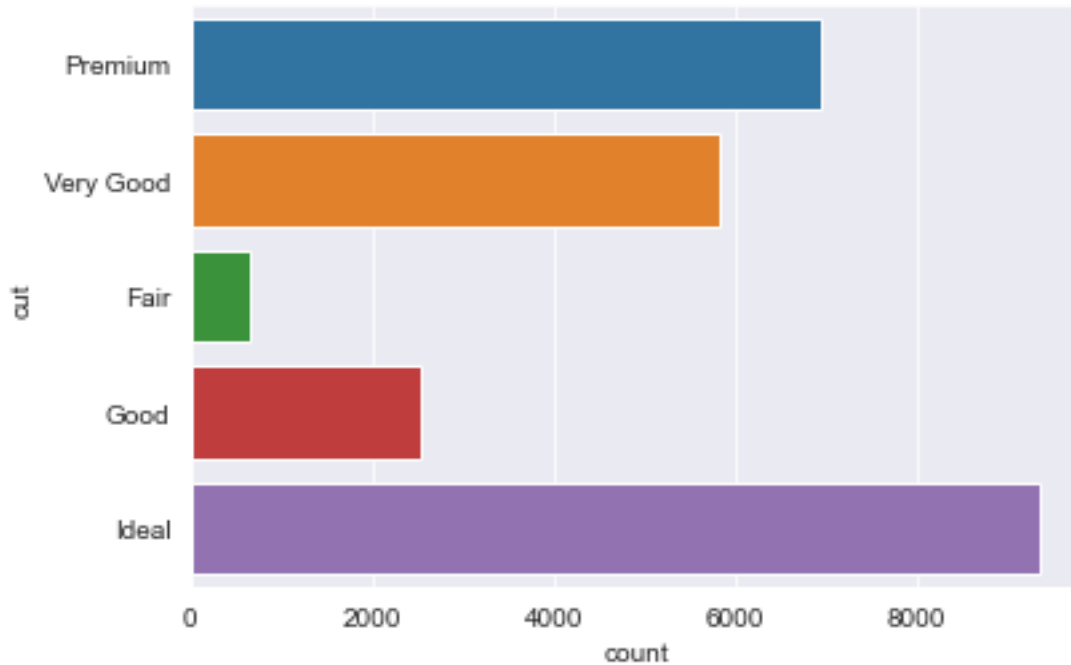


```
[90]: #checking the value counts of color  
diamond.color.value_counts()
```

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

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

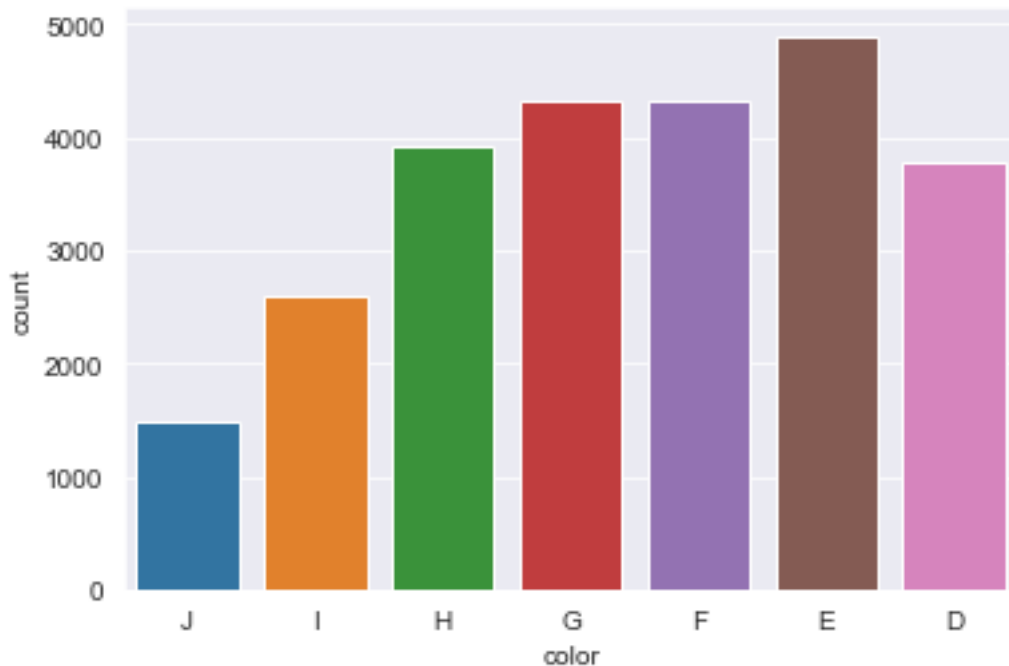
```
[91]: <AxesSubplot:xlabel='count', ylabel='cut'>
```



```
[92]: #checking the datatypes
diamond.dtypes
```

```
[92]: 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
```

```
[93]: #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

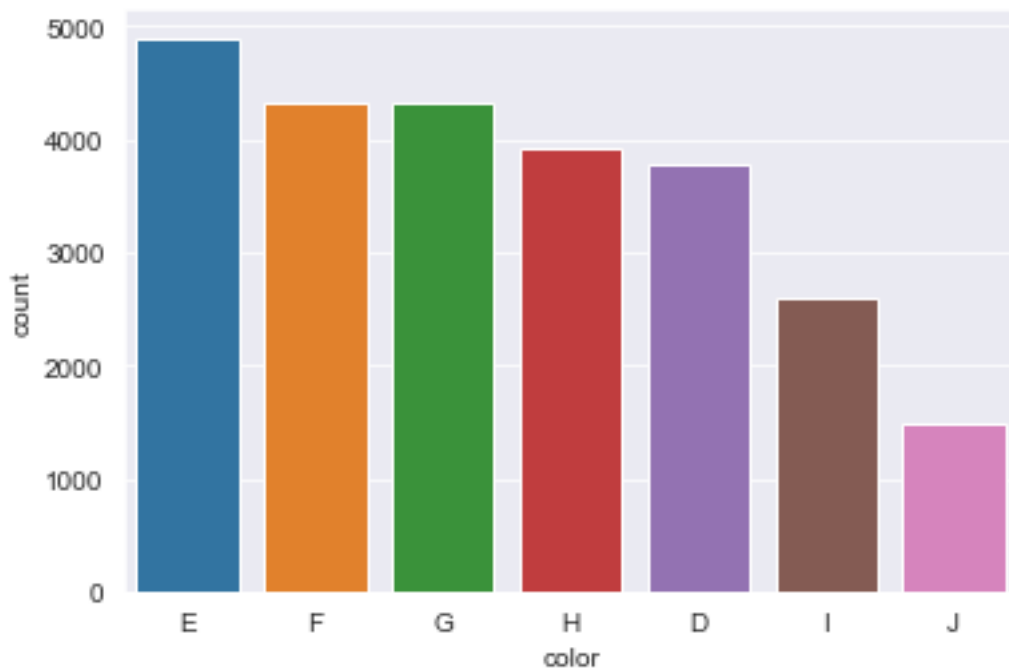
```
[94]: #checking the value counts
diamond.color.value_counts()
```

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

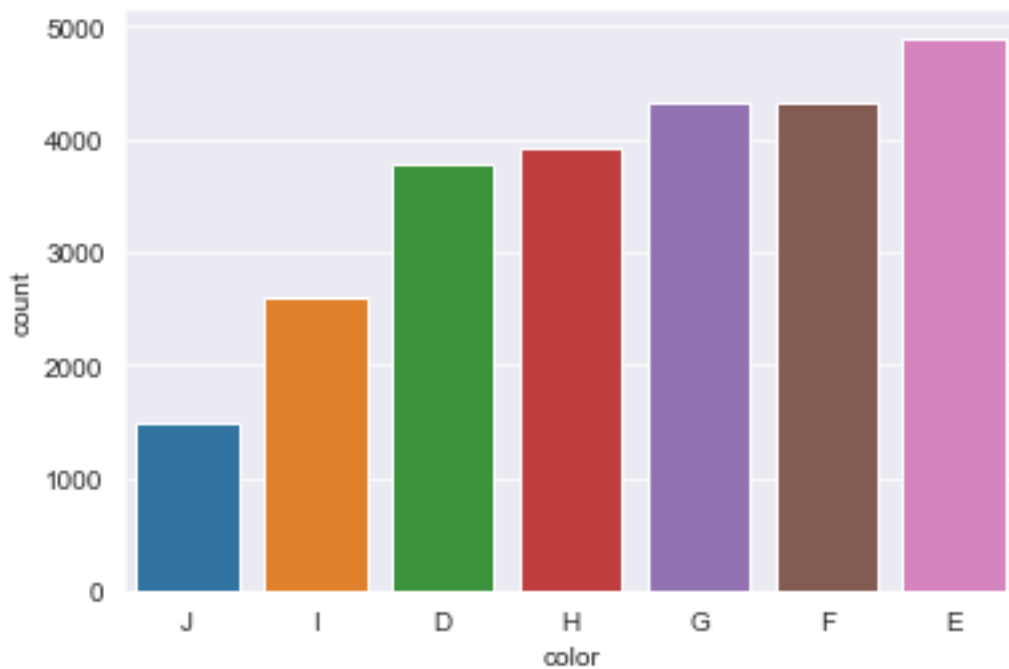
```
[95]: #checking the Index
diamond.color.value_counts().index
```

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

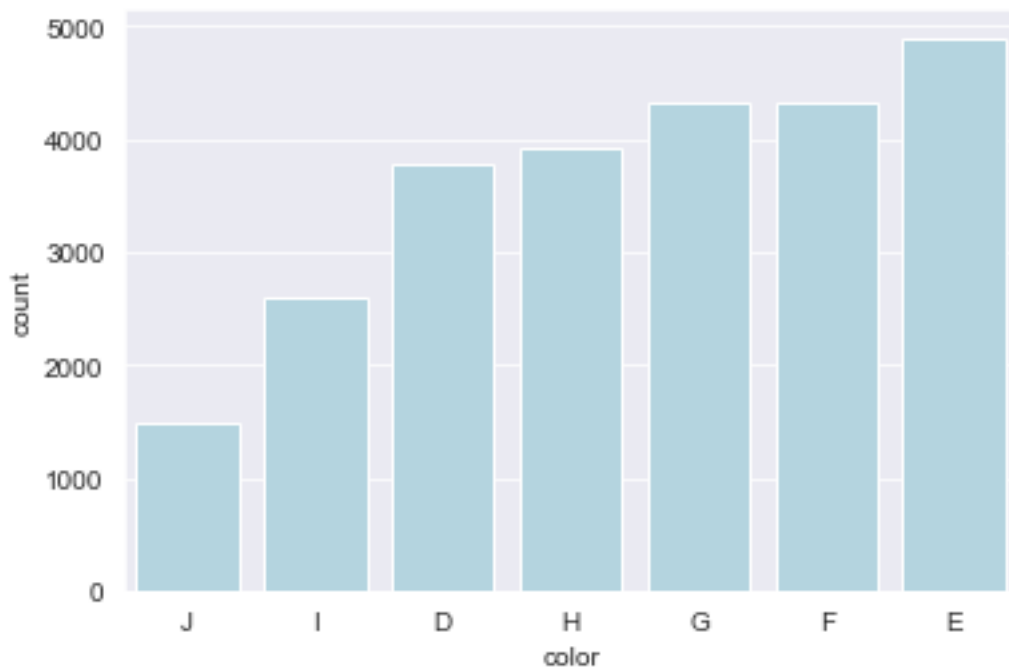
```
[96]: #plotting as per index (plotting in descending order)
sns.countplot(x='color', data=diamond, order=diamond.color.value_counts().
    ↳ index);
```



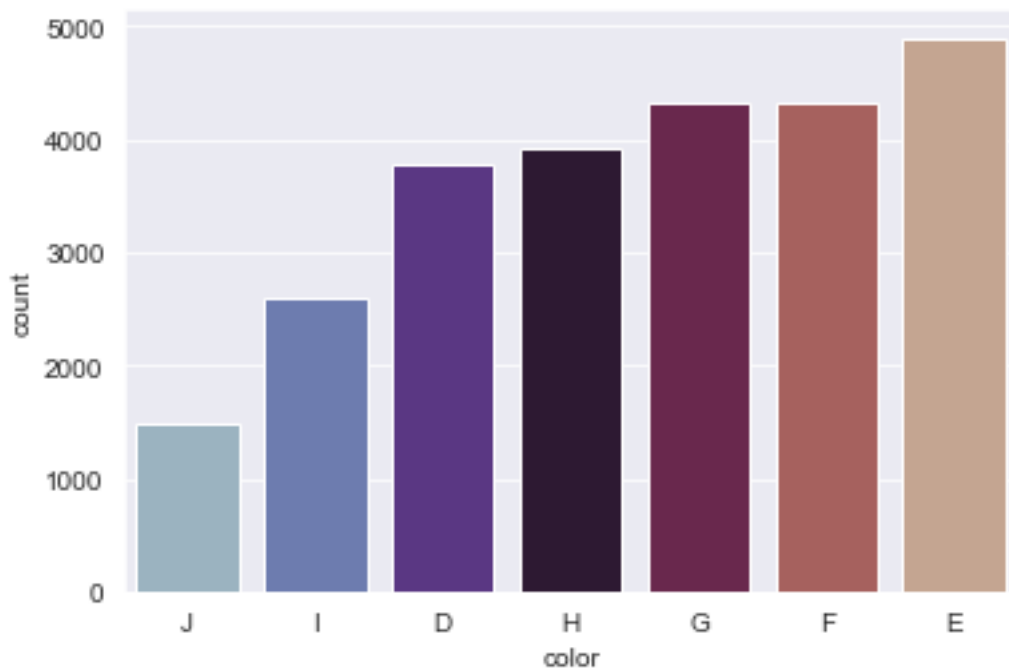
```
[97]: #plotting in ascending order
sns.countplot(x='color', data=diamond, order=diamond.color.value_counts().
↪index[::-1]);
```




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

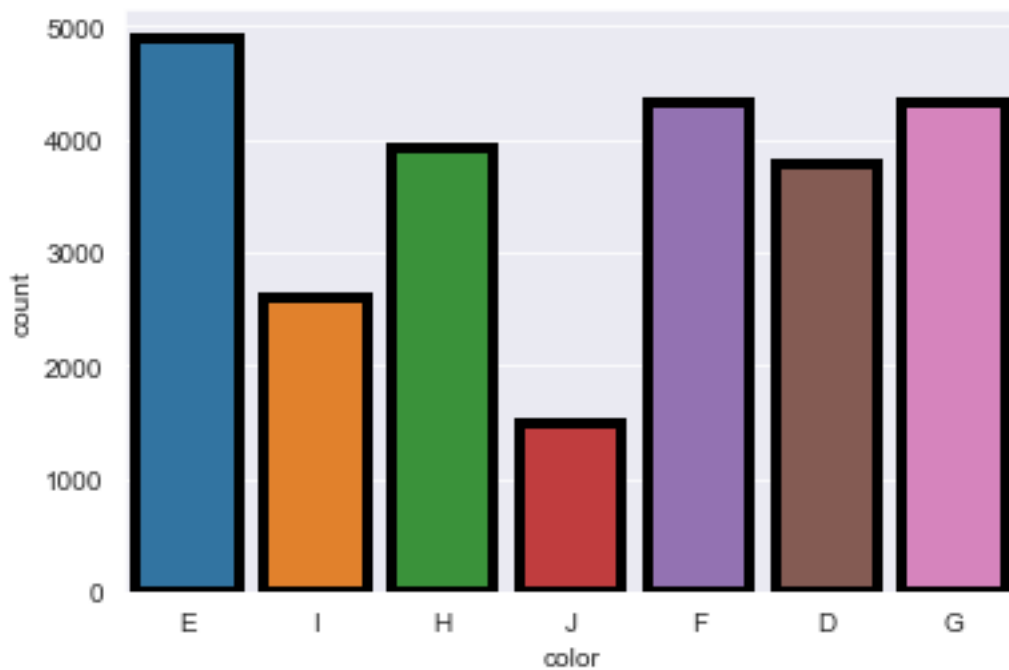


```
[99]: # setting the color to Black
sns.countplot(x='color', data=diamond, order=diamond.color.value_counts().
↳index[::-1],palette='twilight');
```



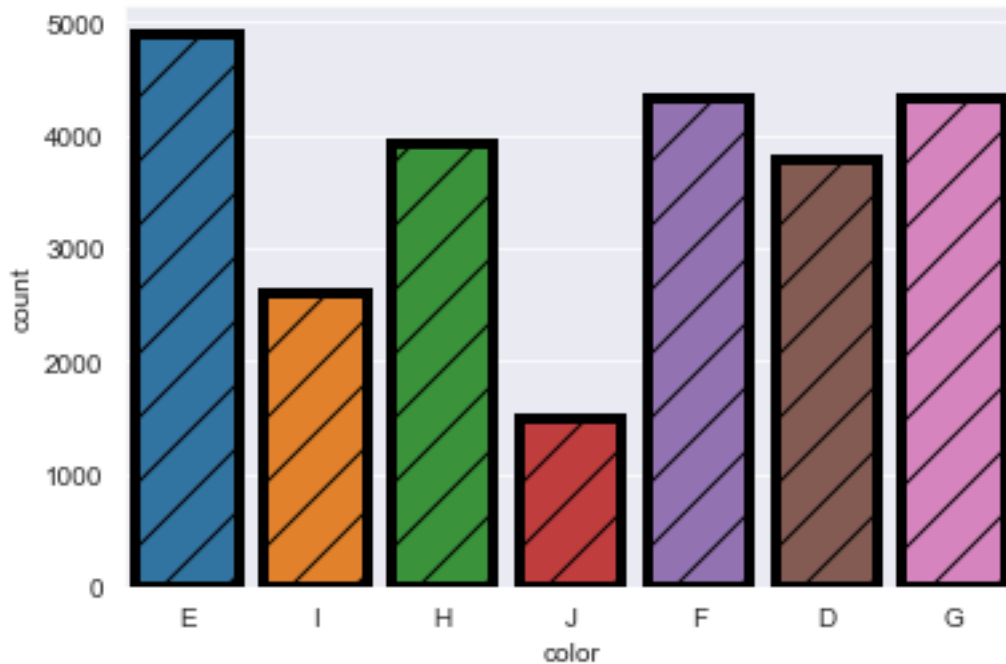
```
[100]: #line width and edge color
sns.countplot(x='color',data=diamond, lw=4, ec='black')
```

```
[100]: <AxesSubplot:xlabel='color', ylabel='count'>
```



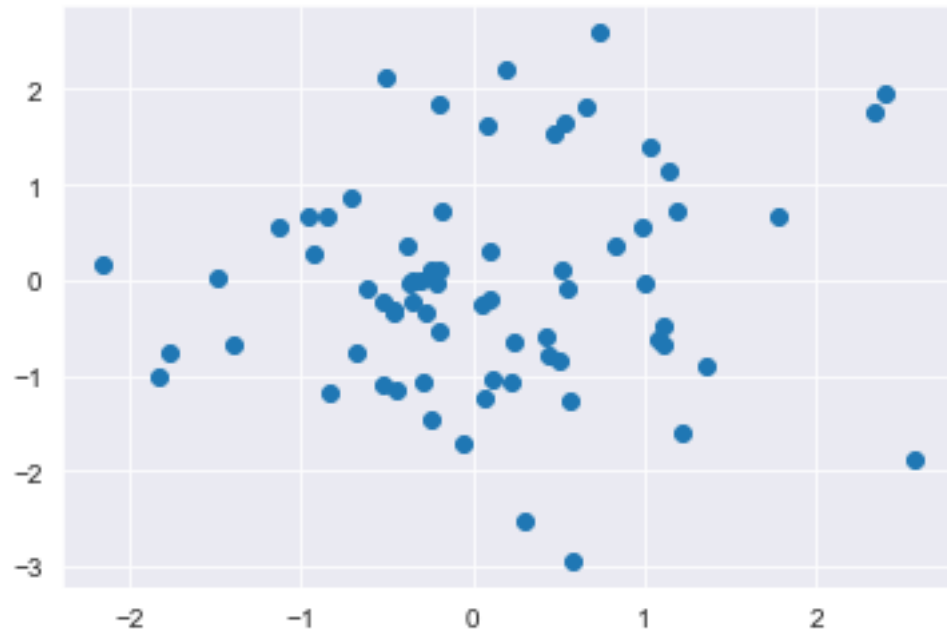
```
[101]: #hatching
sns.countplot(x='color',data=diamond, lw=4, ec='black',hatch='/')
```

```
[101]: <AxesSubplot:xlabel='color', ylabel='count'>
```

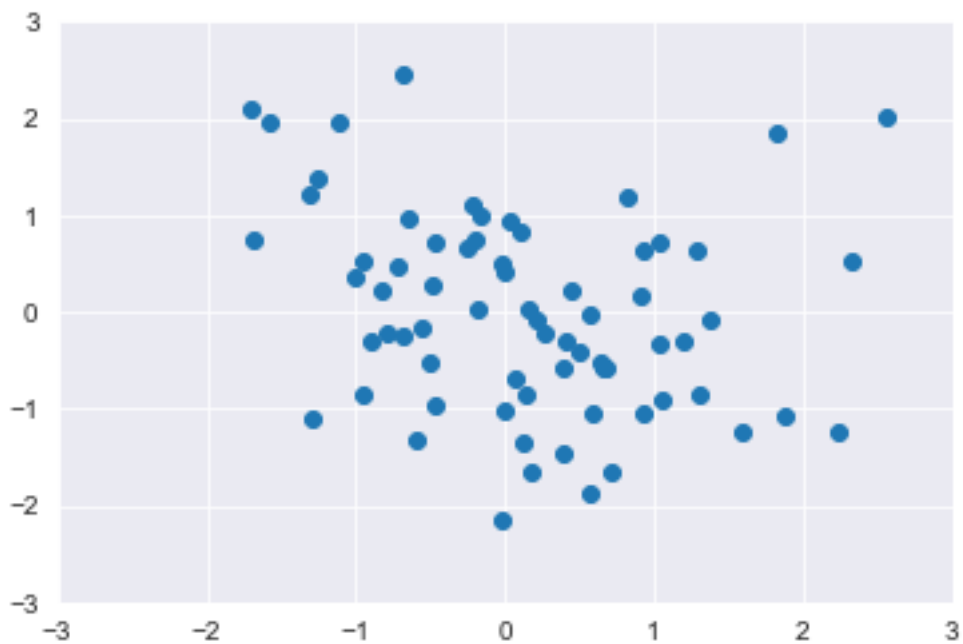


1.7 Scatter plot

```
[102]: 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()
```

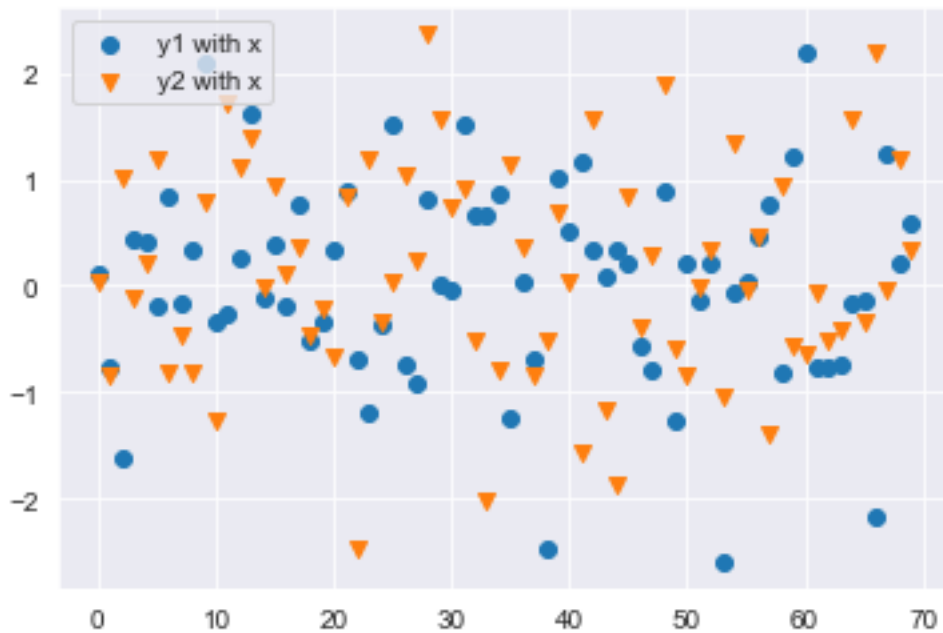


```
[103]: ## adjust axis, if required
y1 = np.random.randn(70)
y2 = np.random.randn(70)
plt.scatter(y1,y2)
plt.axis([-3,3,-3,3]) #plt.axis([Xmin,Xmax,Ymin,Ymax])
plt.show()
```



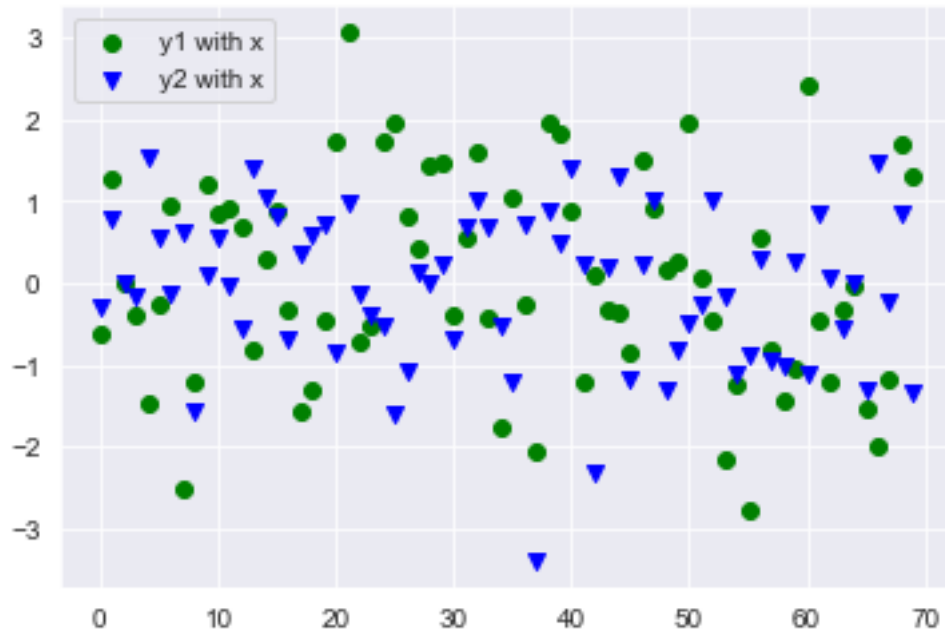
```
[104]: # using the scatter plot with common axis
x = np.arange(70)
y1 = np.random.randn(70)
y2 = np.random.randn(70)
plt.scatter(x,y1, marker='o', label = 'y1 with x')
plt.scatter(x,y2, marker='v', label = 'y2 with x')
plt.legend(loc='upper left')
```

[104]: <matplotlib.legend.Legend at 0x1b565e3e370>



```
[105]: #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')
```

[105]: <matplotlib.legend.Legend at 0x1b565eb22e0>



1.8 Line Plot

```
[106]: # extract data from various Internet sources into a pandas DataFrame
import pandas_datareader as pdr
```

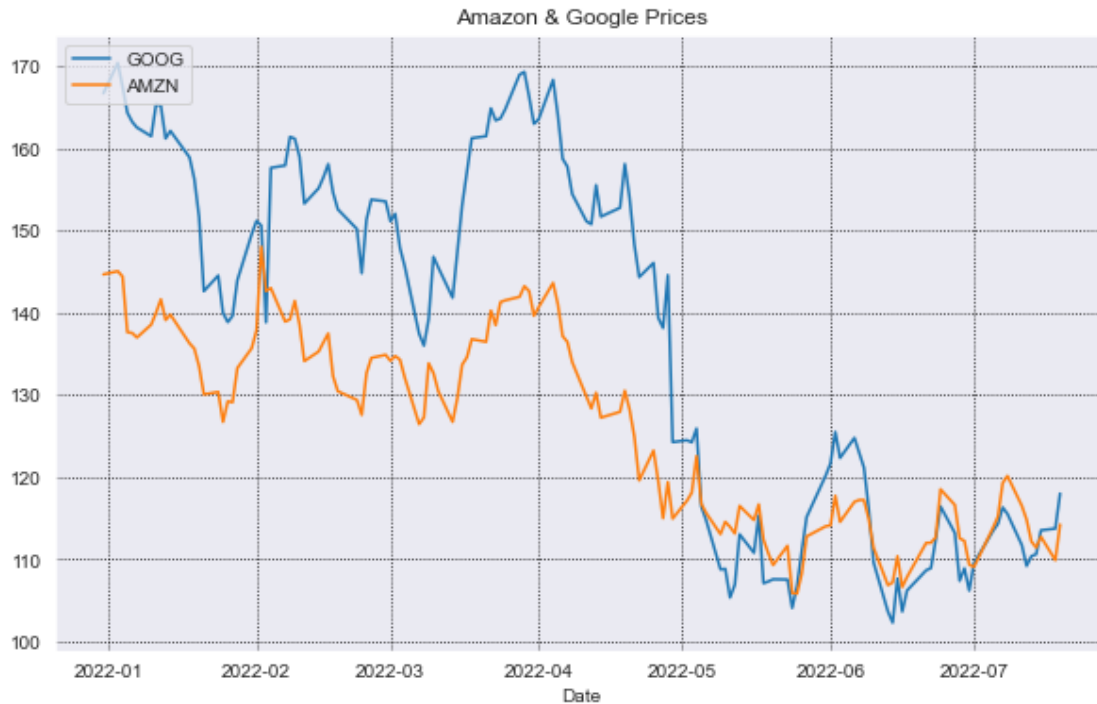
```
[107]: #extracting Data
stocks = ['GOOG', 'AMZN']
data = pdr.get_data_yahoo(stocks, start = '2022-01-01')['Close']
data.head()
```

```
[107]: Symbols      GOOG      AMZN
Date
2021-12-31  144.679504  166.716995
2022-01-03  145.074493  170.404495
2022-01-04  144.416504  167.522003
2022-01-05  137.653503  164.356995
2022-01-06  137.550995  163.253998
```

```
[108]: from matplotlib import rcParams
rcParams['figure.figsize'] = 10,6
plt.plot(data.AMZN)
plt.plot(data.GOOG)
plt.grid(True, color='k', linestyle=':')
plt.title("Amazon & Google Prices")
plt.xlabel("Date")
```

```
plt.legend(['GOOG', 'AMZN'], loc = 2)
```

[108]: <matplotlib.legend.Legend at 0x1b5666b9d60>



1.8.1 Refer to below link for more details

- https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html
- https://matplotlib.org/3.1.1/api/markers_api.html formore details

```
[109]: #loading the Dataset
tips = pd.read_csv('tips.csv')
```

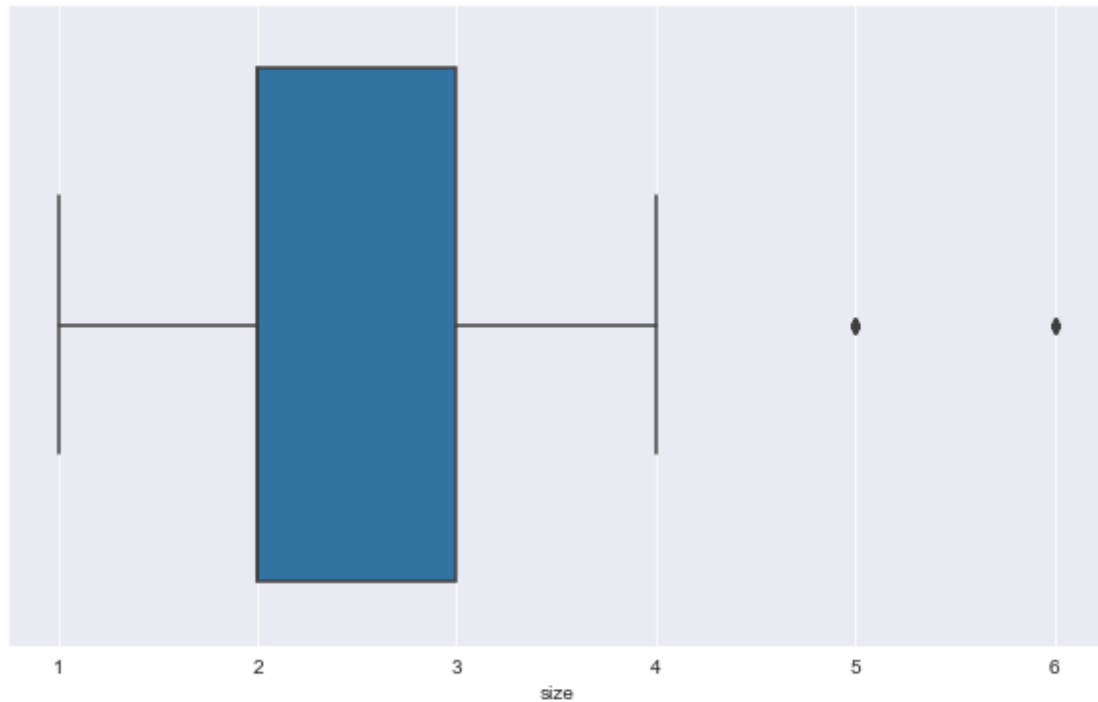
```
[110]: #checking the head of data
tips.head()
```

```
[110]:   total_bill  tip  sex smoker  day  time  size
0      16.99  1.01 Female    No  Sun  Dinner    2
1      10.34  1.66  Male    No  Sun  Dinner    3
2      21.01  3.50  Male    No  Sun  Dinner    3
3      23.68  3.31  Male    No  Sun  Dinner    2
4      24.59  3.61 Female    No  Sun  Dinner    4
```

1.9 Boxplot

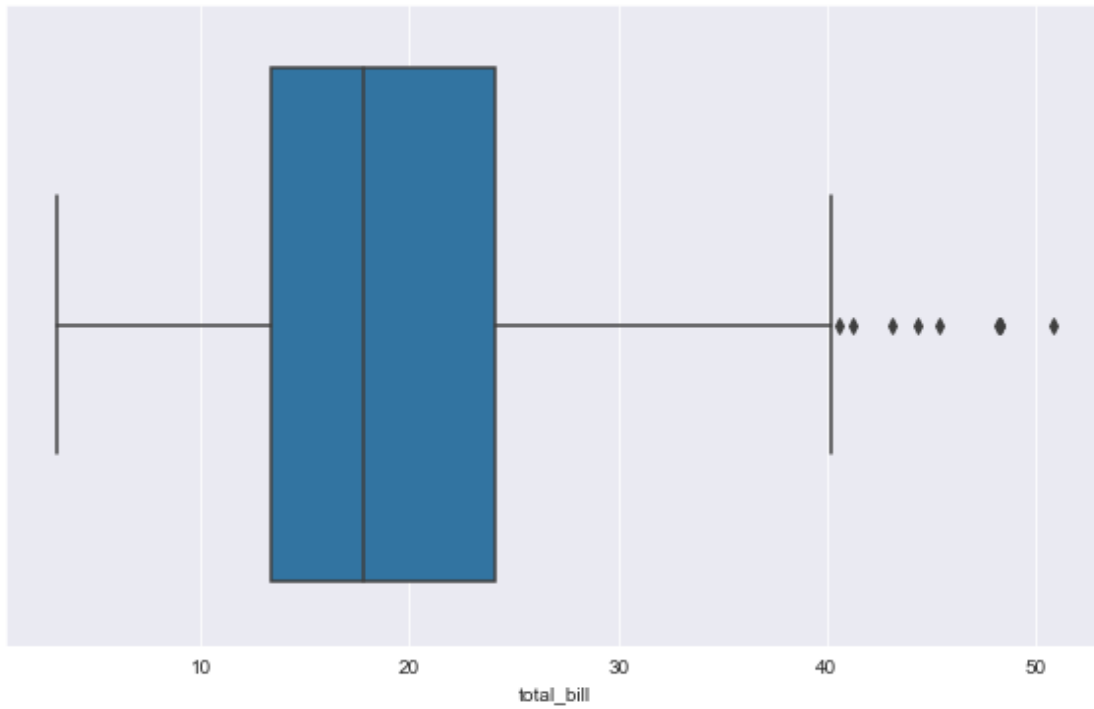
```
[111]: sns.boxplot(tips['size'])
```

```
[111]: <AxesSubplot:xlabel='size'>
```



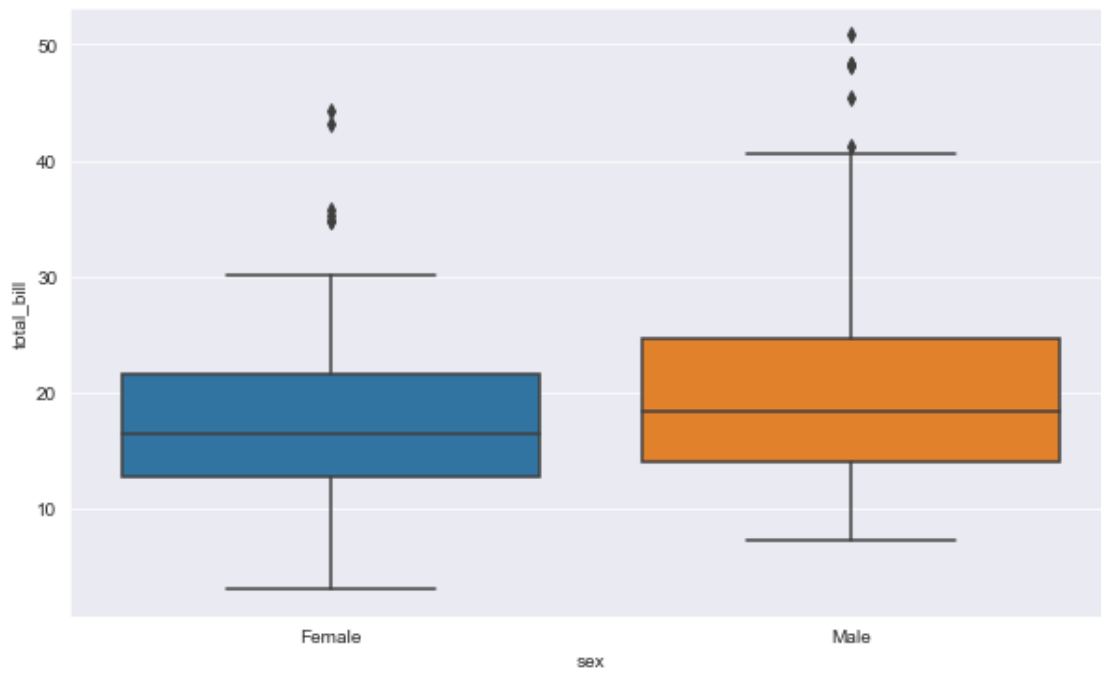
```
[112]: sns.boxplot(tips['total_bill'])
```

```
[112]: <AxesSubplot:xlabel='total_bill'>
```

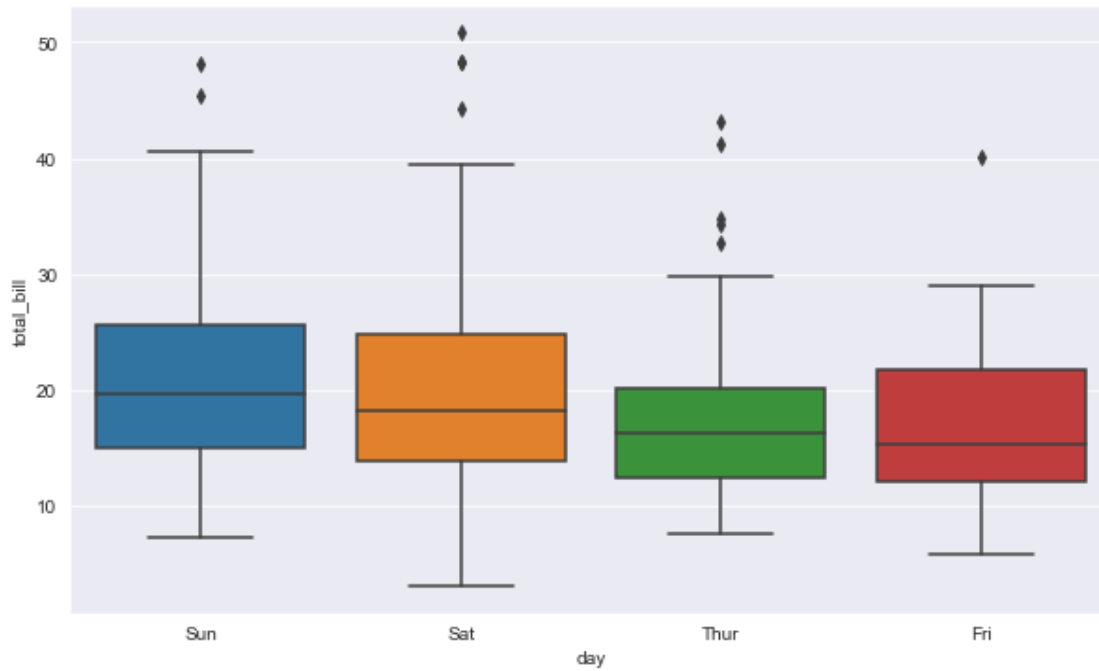
```
[113]: sns.boxplot(x='sex', y='total_bill', data=tips)
```

```
[113]: <AxesSubplot:xlabel='sex', ylabel='total_bill'>
```



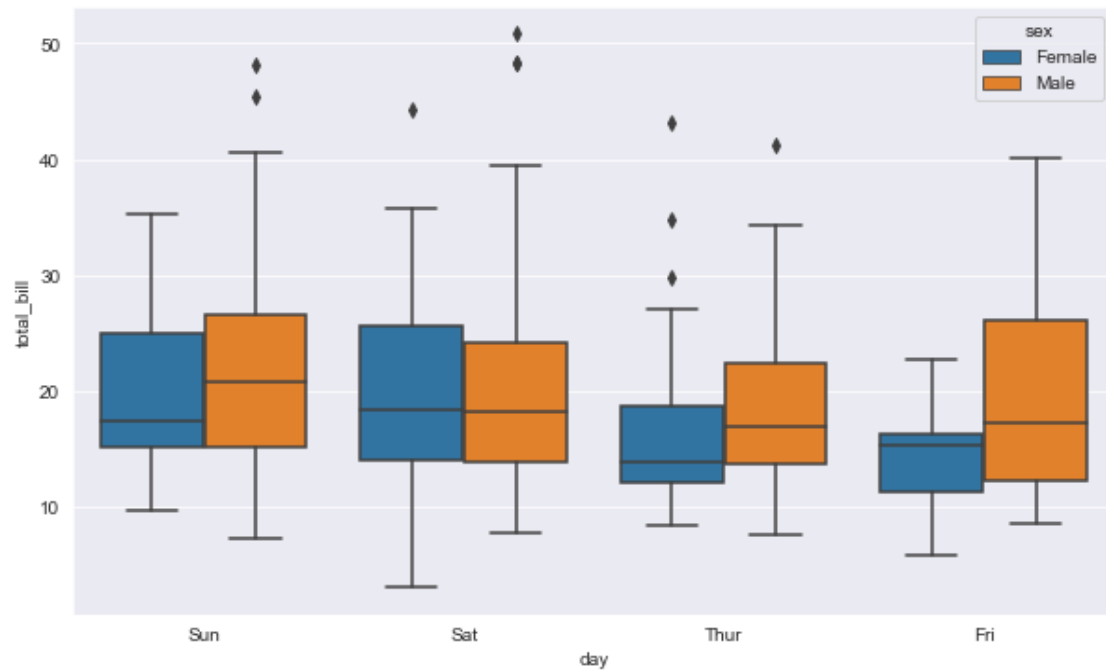
```
[114]: sns.boxplot(x='day', y='total_bill', data=tips)
```

```
[114]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



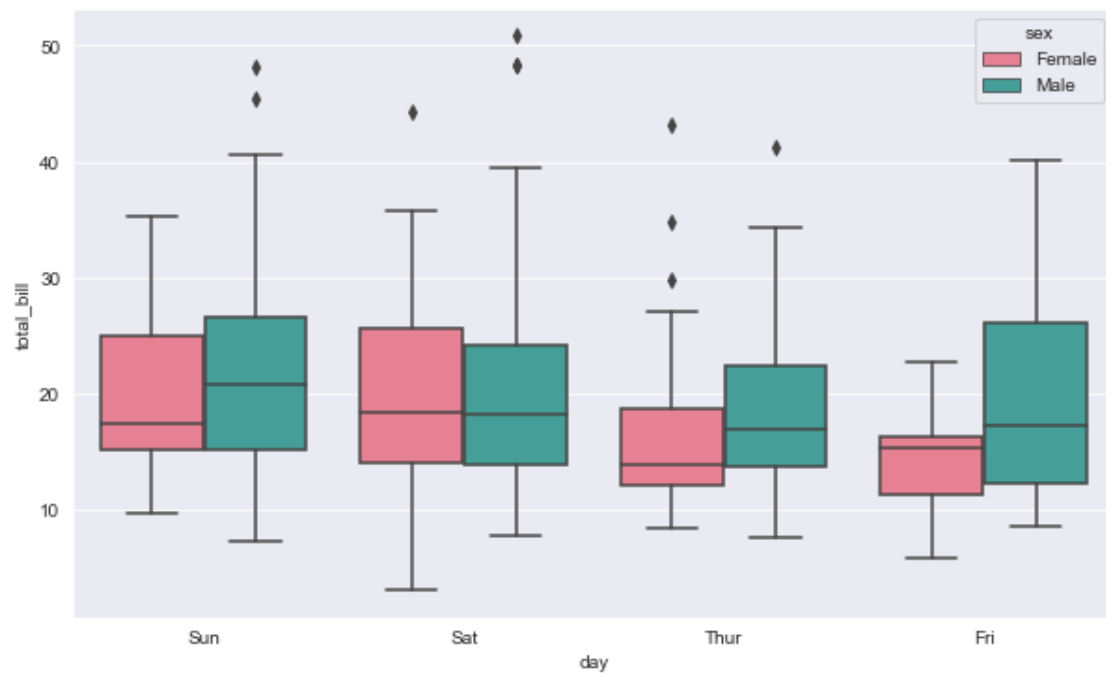
```
[115]: sns.boxplot(x='day', y='total_bill', data=tips, hue='sex')
```

```
[115]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



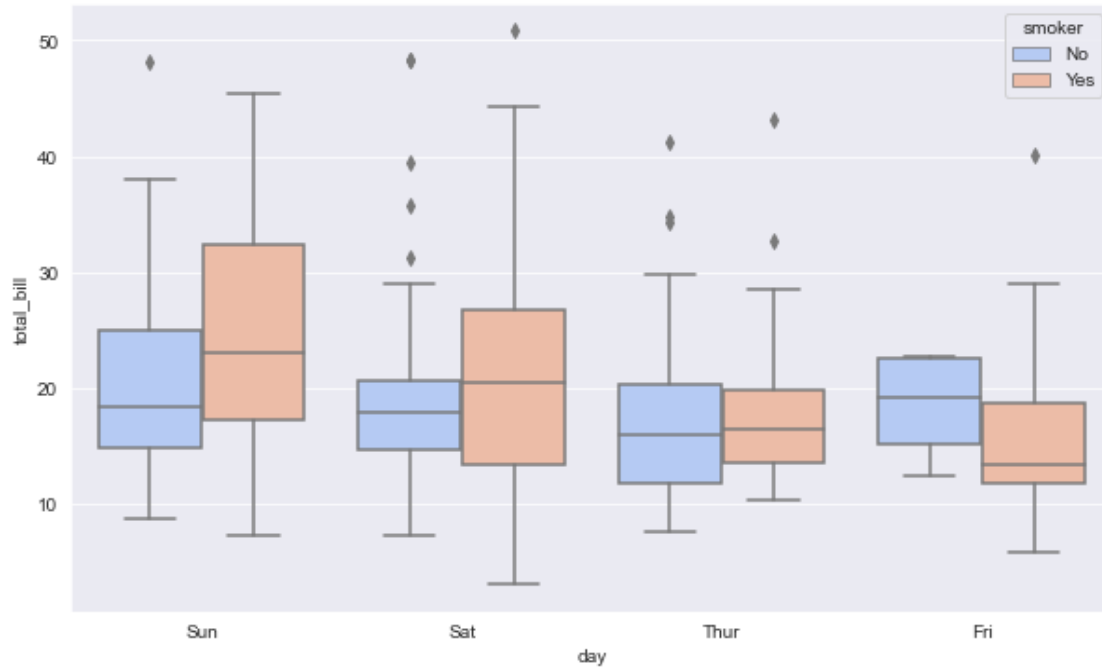
```
[116]: sns.boxplot(x='day', y='total_bill', data=tips, hue='sex', palette='husl')
```

```
[116]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



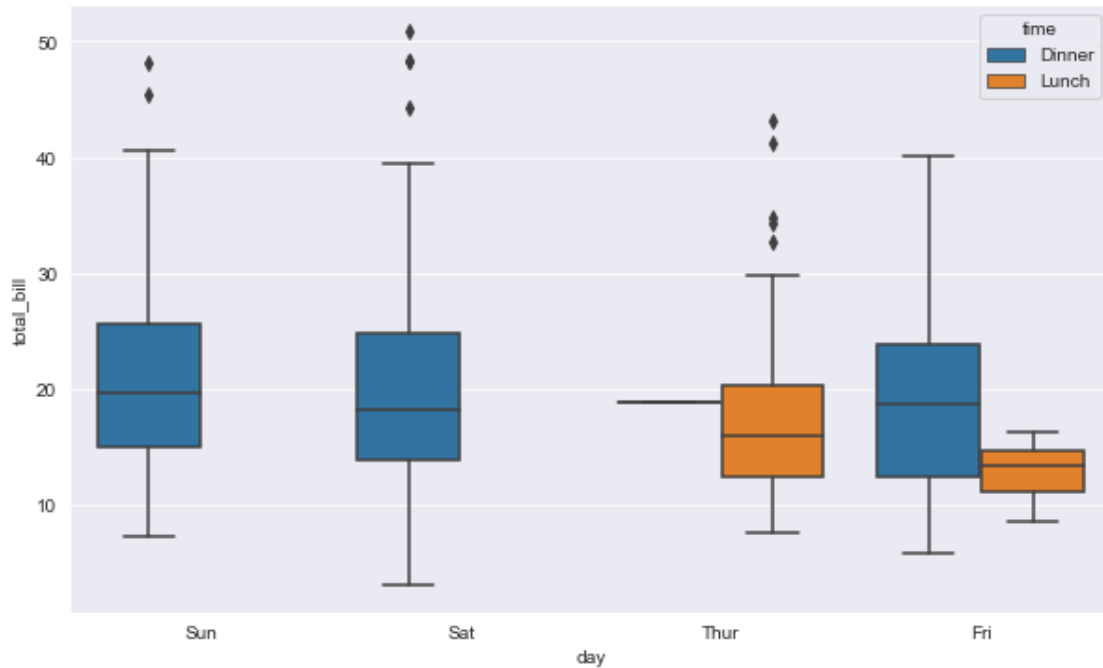
```
[117]: sns.boxplot(x='day', y='total_bill', data=tips, hue='smoker', palette_
↳='coolwarm')
```

```
[117]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



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

```
[118]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



1.10 Joint Distribution

```
[119]: iris = pd.read_csv('Iris.csv')
```

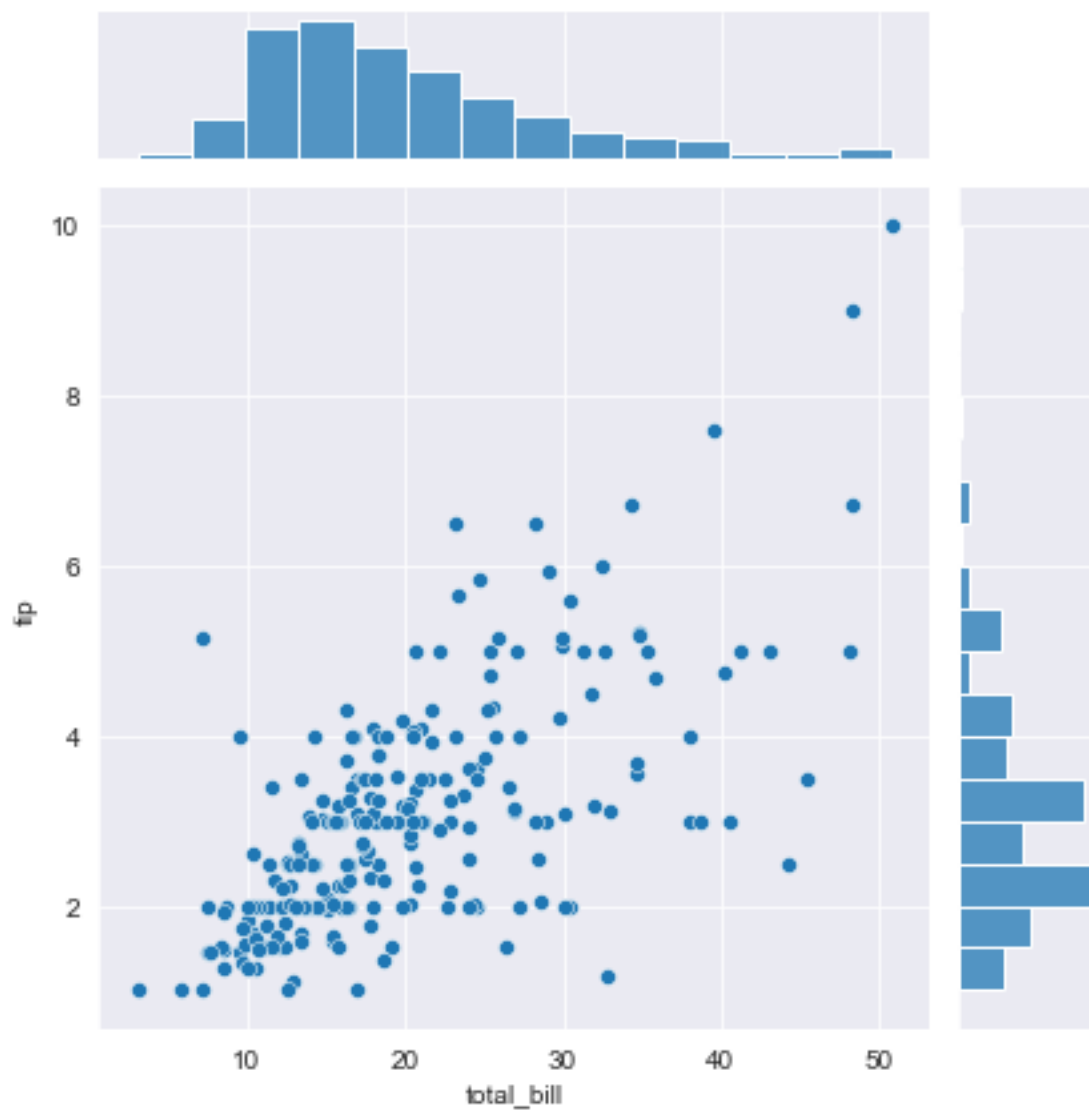
```
[120]: iris.head()
```

```
[120]:
```

| | 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 |

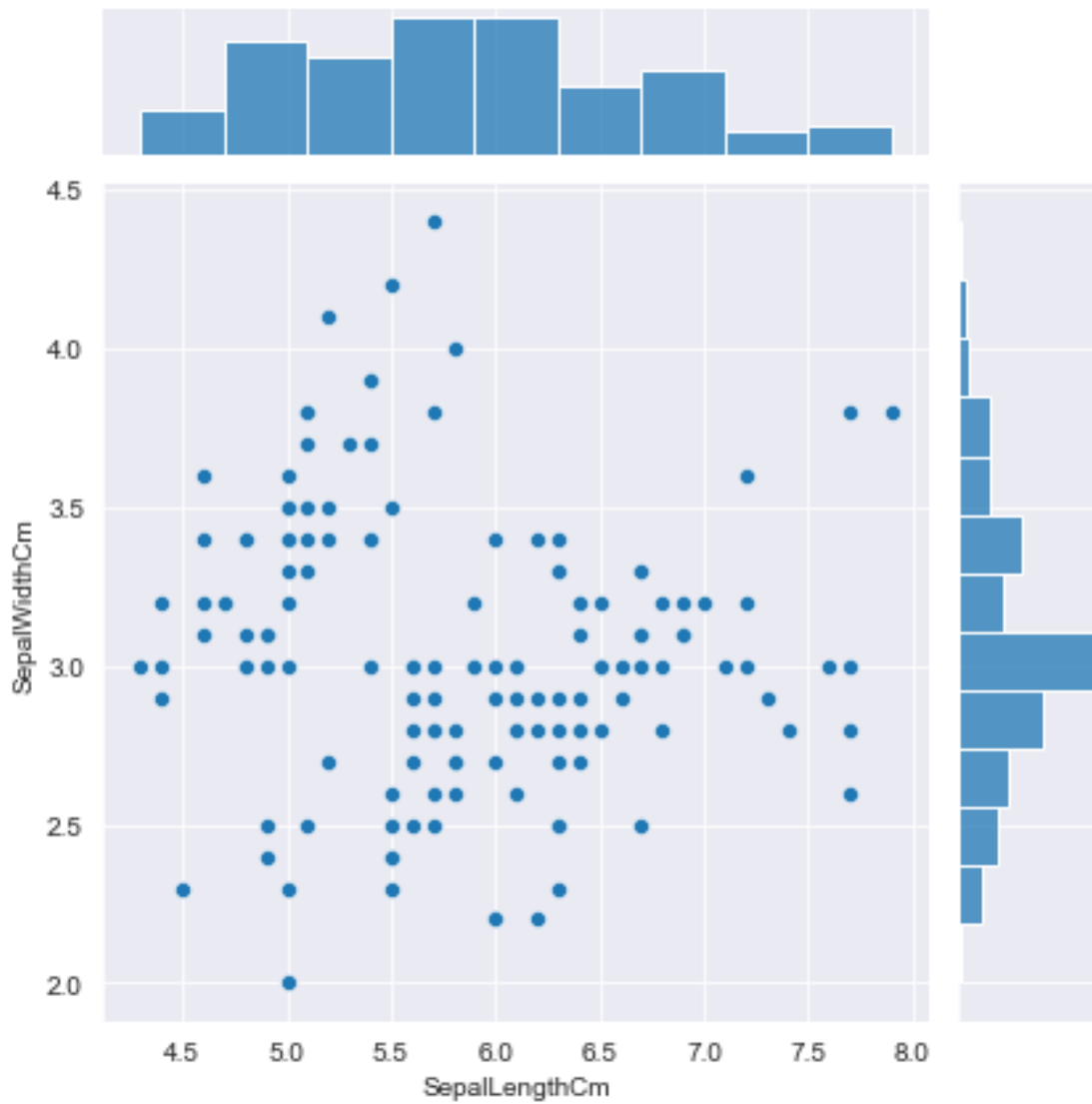
```
[121]: sns.jointplot(x='total_bill', y='tip', data=tips)
```

```
[121]: <seaborn.axisgrid.JointGrid at 0x1b566796fa0>
```



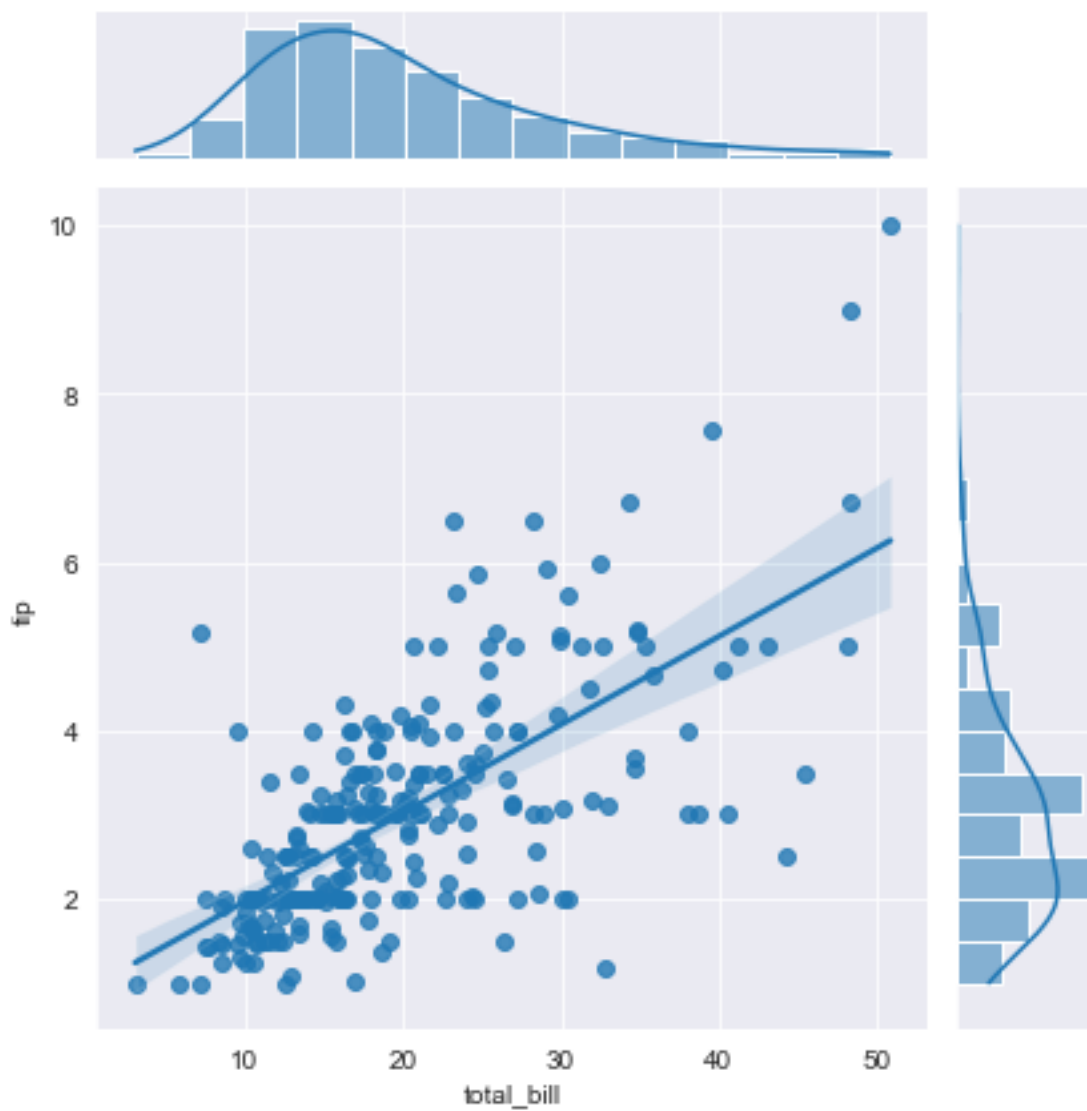
```
[122]: sns.jointplot(x='SepalLengthCm', y='SepalWidthCm', data=iris)
```

```
[122]: <seaborn.axisgrid.JointGrid at 0x1b566654400>
```



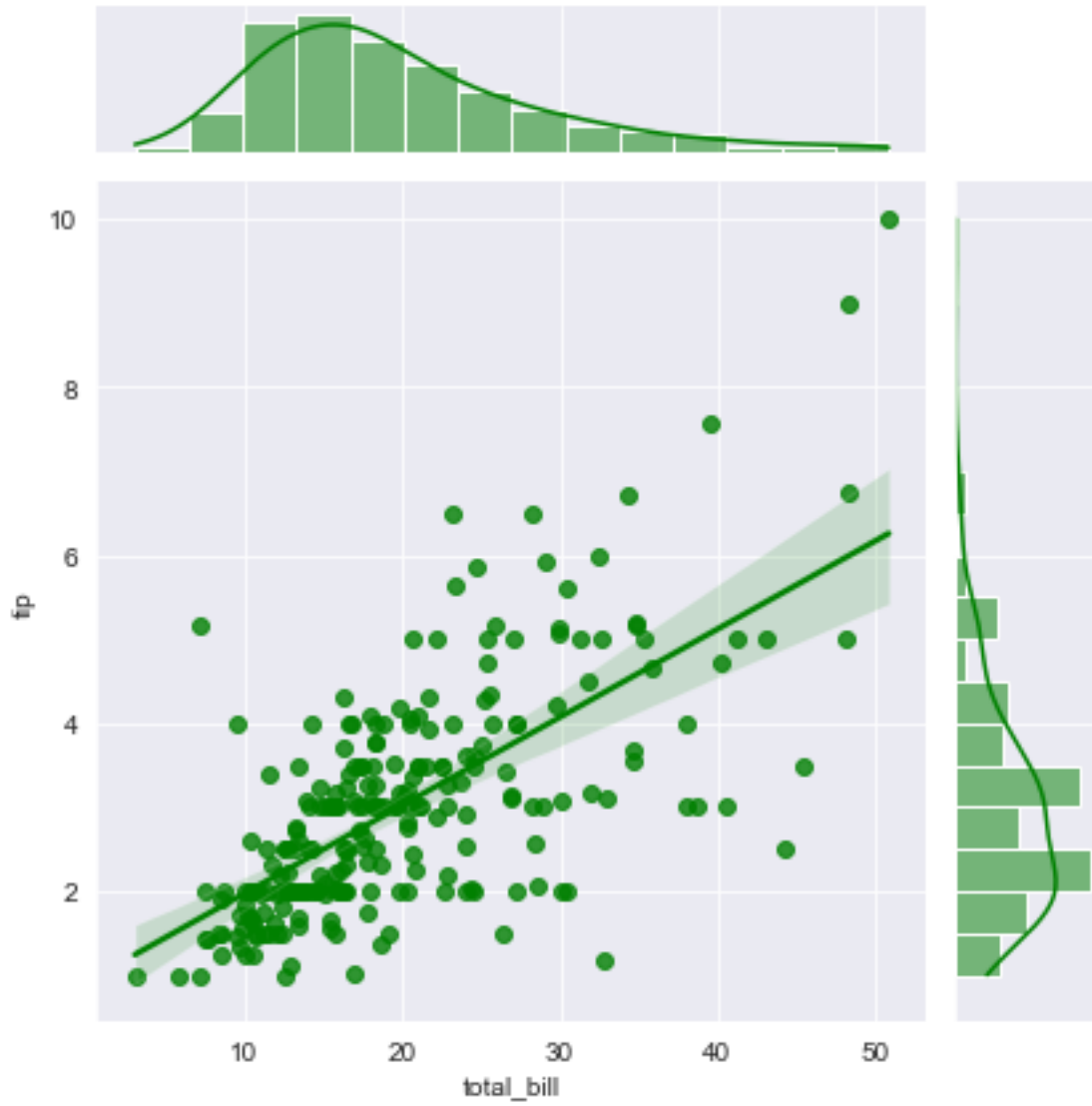
```
[123]: # adding regression lines
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg')
```

```
[123]: <seaborn.axisgrid.JointGrid at 0x1b566a11be0>
```



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

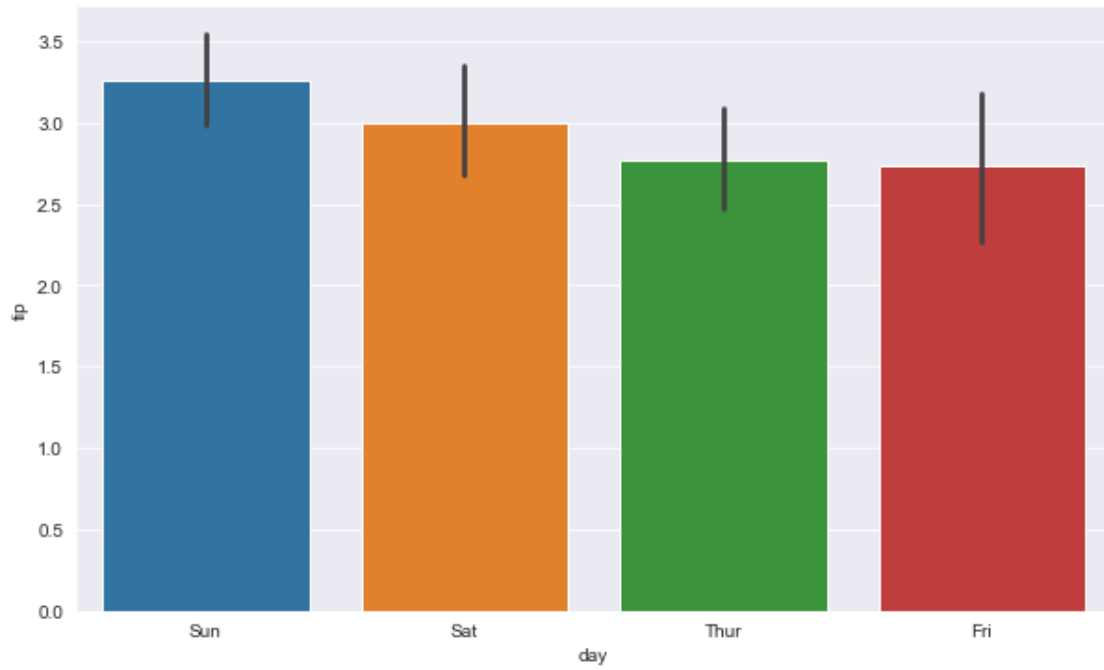
```
[124]: <seaborn.axisgrid.JointGrid at 0x1b567d33d90>
```

1.11 Bar Plots

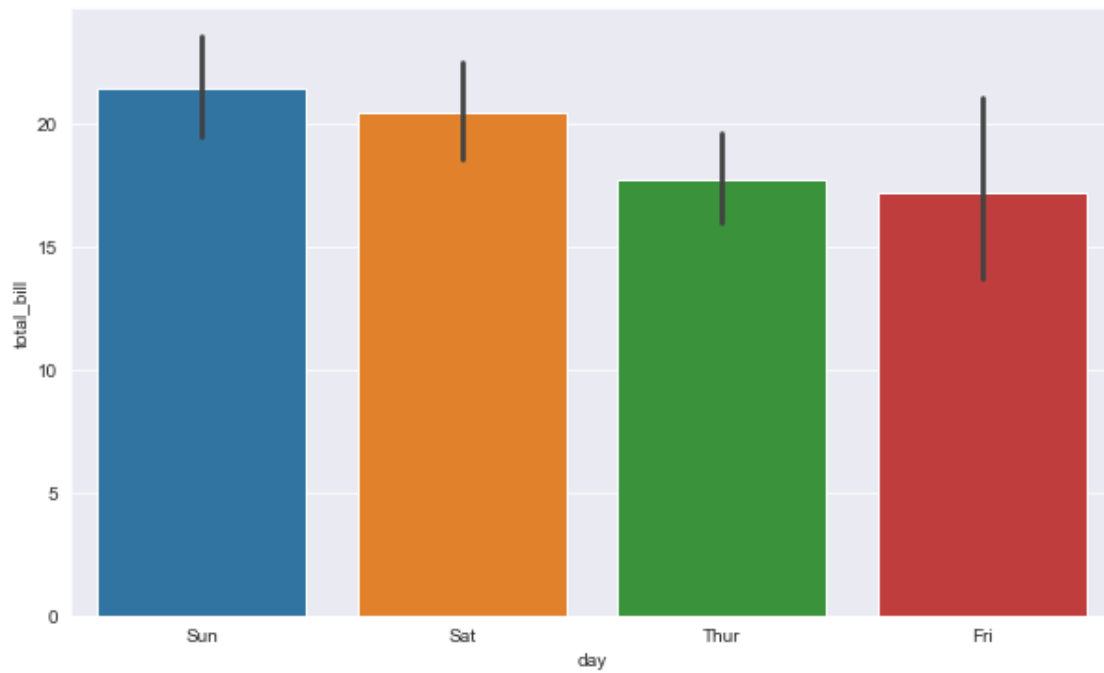
```
[125]: sns.barplot(x='day', y='tip', data=tips)
```

```
[125]: <AxesSubplot:xlabel='day', ylabel='tip'>
```



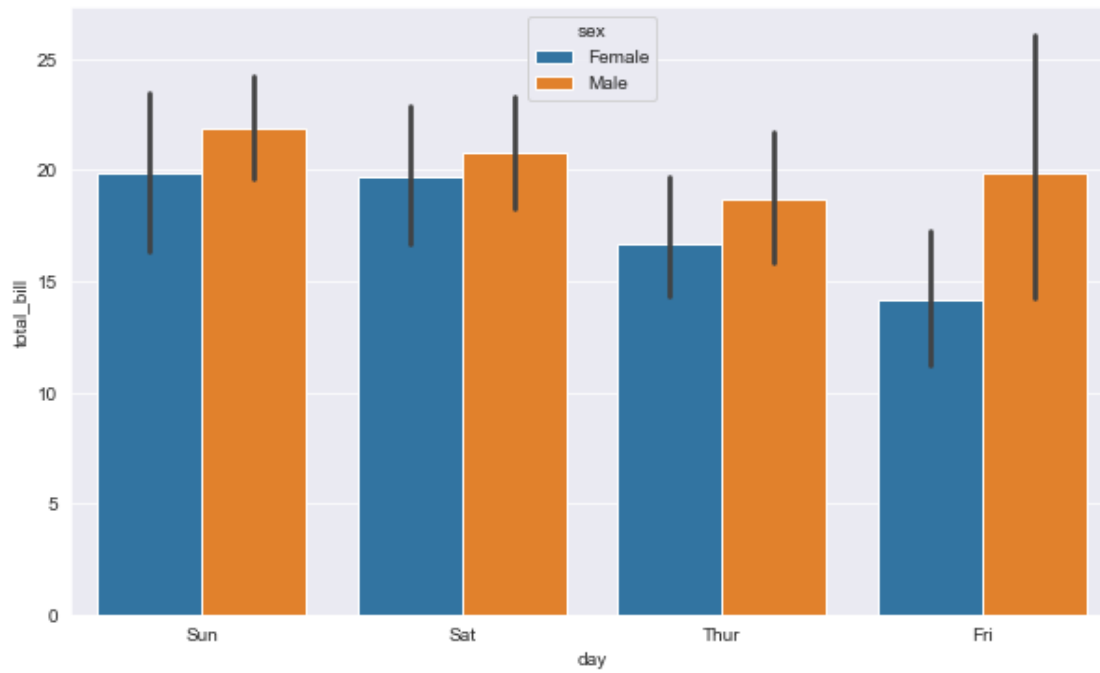
```
[126]: sns.barplot(x='day', y='total_bill',data=tips)
```

```
[126]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



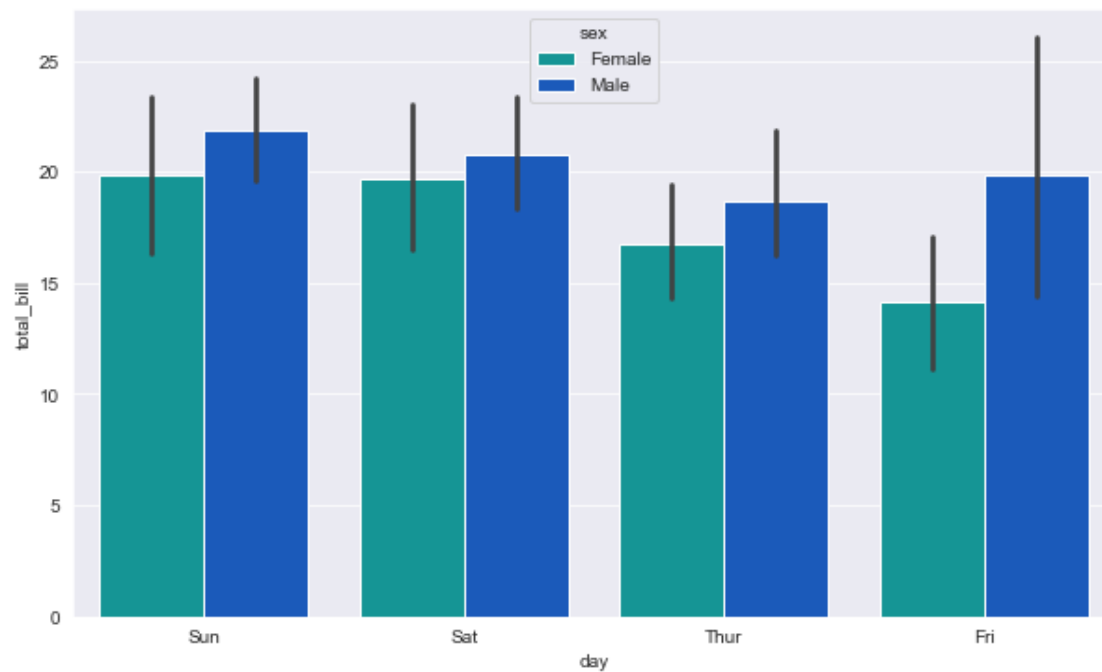
```
[127]: sns.barplot(x='day', y='total_bill',data=tips,hue='sex')
```

```
[127]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



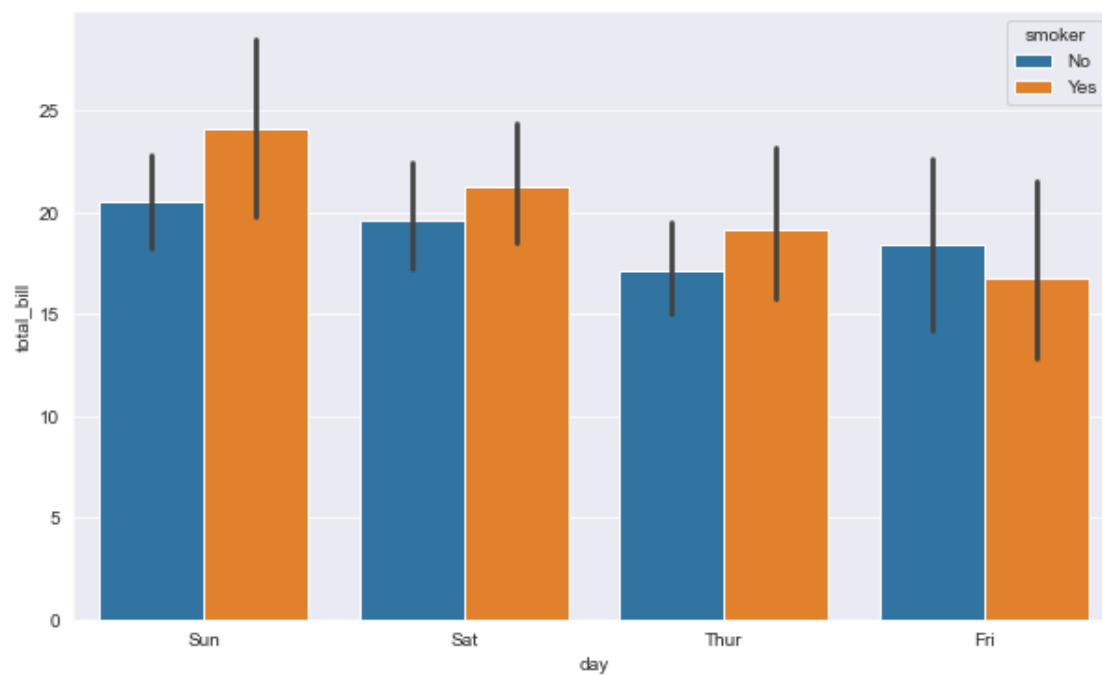
```
[128]: sns.barplot(x='day', y='total_bill',data=tips,hue='sex', palette='winter_r')
```

```
[128]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



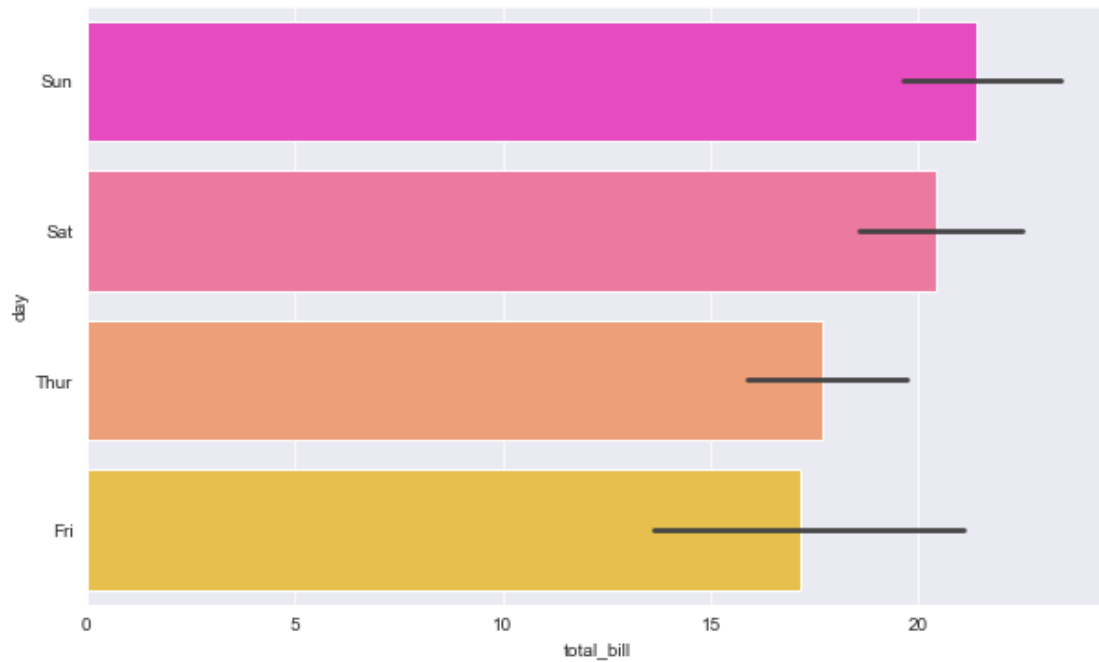
```
[129]: sns.barplot(x='day', y='total_bill',data=tips,hue='smoker')
```

```
[129]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



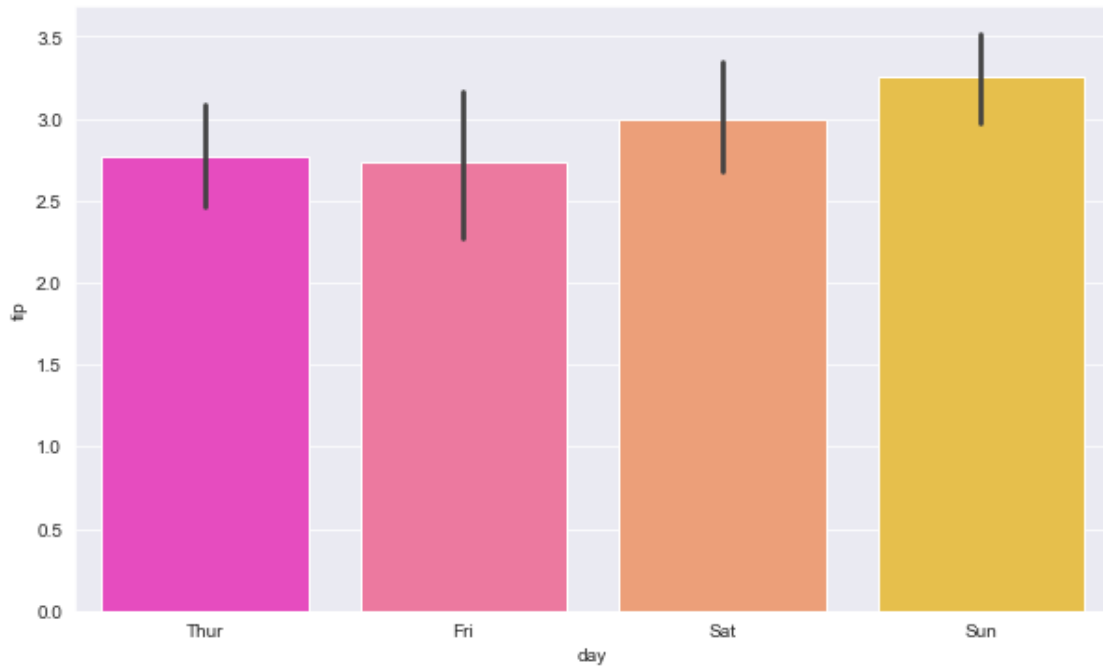
```
[130]: sns.barplot(x='total_bill', y='day' , data = tips, palette = 'spring')
```

```
[130]: <AxesSubplot:xlabel='total_bill', ylabel='day'>
```



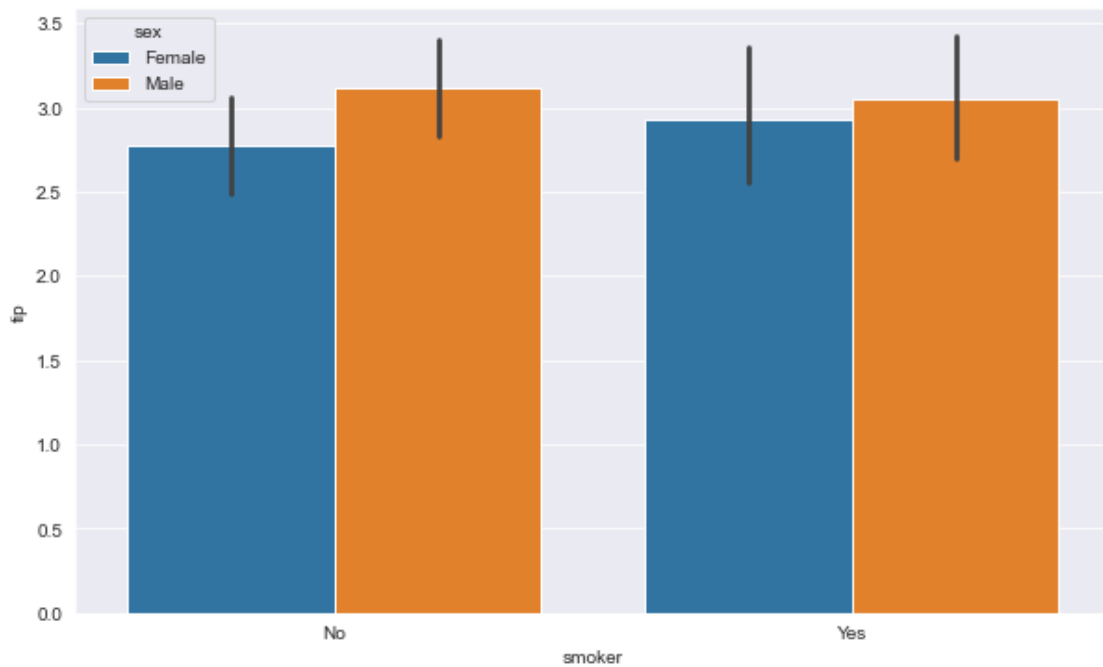
```
[131]: sns.barplot(x='day', y='tip' , data = tips, palette='spring', order=['Thur', 'Fri', 'Sat', 'Sun'])
```

```
[131]: <AxesSubplot:xlabel='day', ylabel='tip'>
```



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

```
[132]: <AxesSubplot:xlabel='smoker', ylabel='tip'>
```



```
[133]: #Load the dataset
df = pd.read_csv('mtcars.csv')
```

```
[134]: #Viewing first 5 observations
df.head()
```

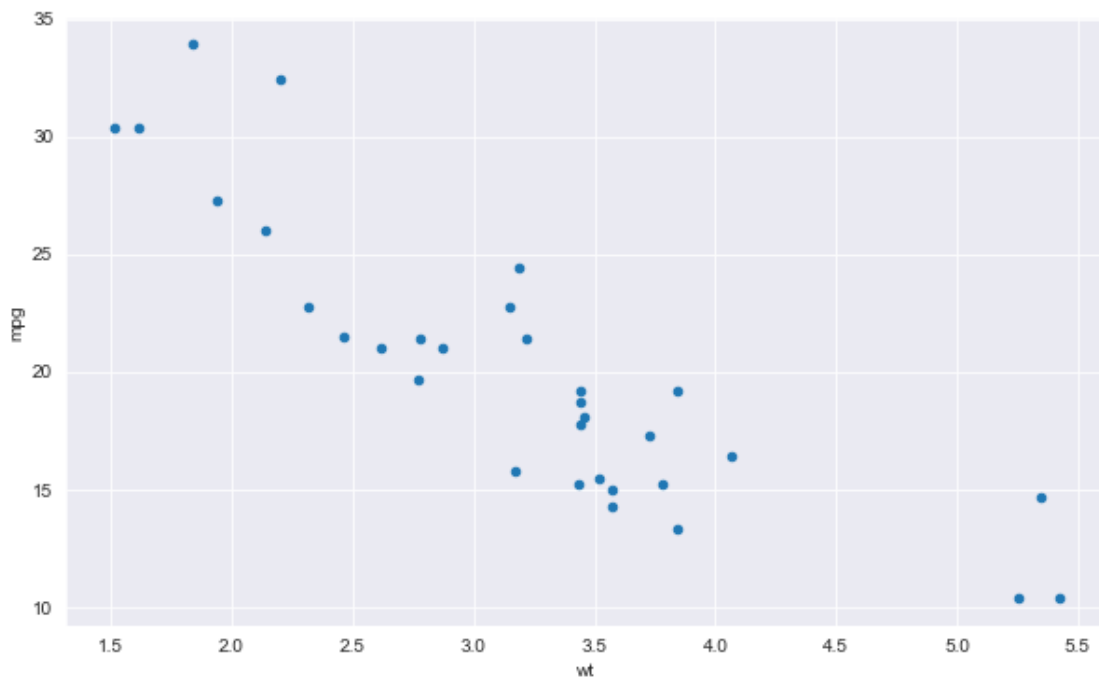
```
[134]:
```

| | model | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | \ |
|---|-------------------|------|-----|-------|-----|------|-------|-------|----|----|------|---|
| 0 | Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | |
| 1 | Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | |
| 2 | Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | |
| 3 | Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | |
| 4 | Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | |


```
carb
0    4
1    4
2    1
3    1
4    2
```

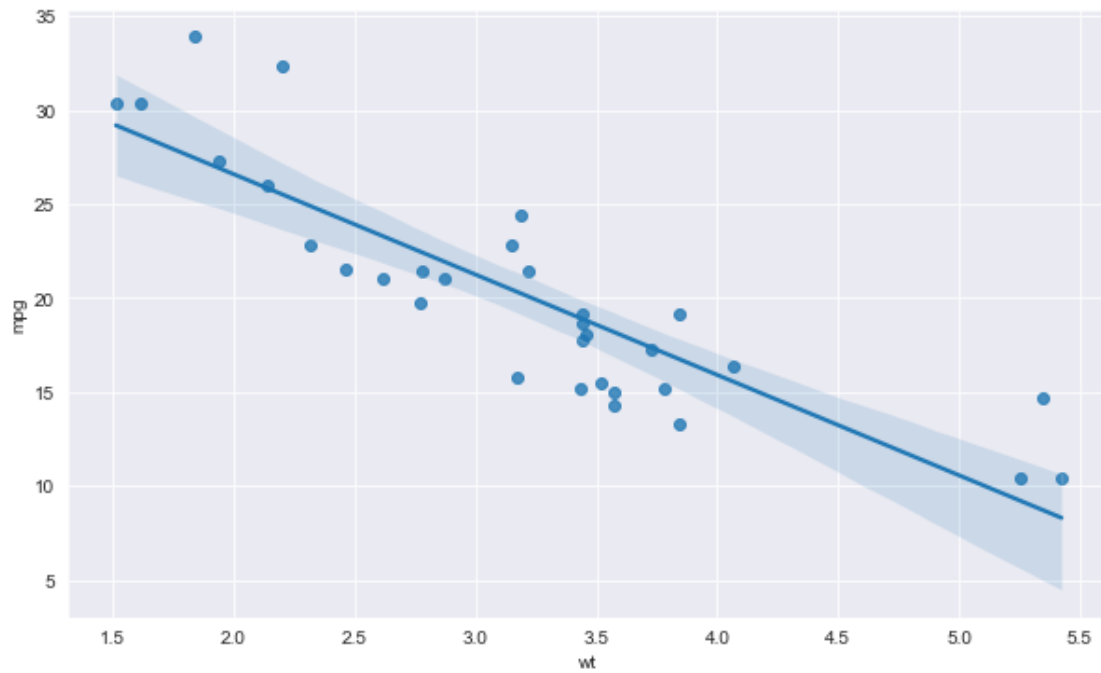
```
[135]: df.plot.scatter(x='wt', y='mpg')
```

```
[135]: <AxesSubplot:xlabel='wt', ylabel='mpg'>
```



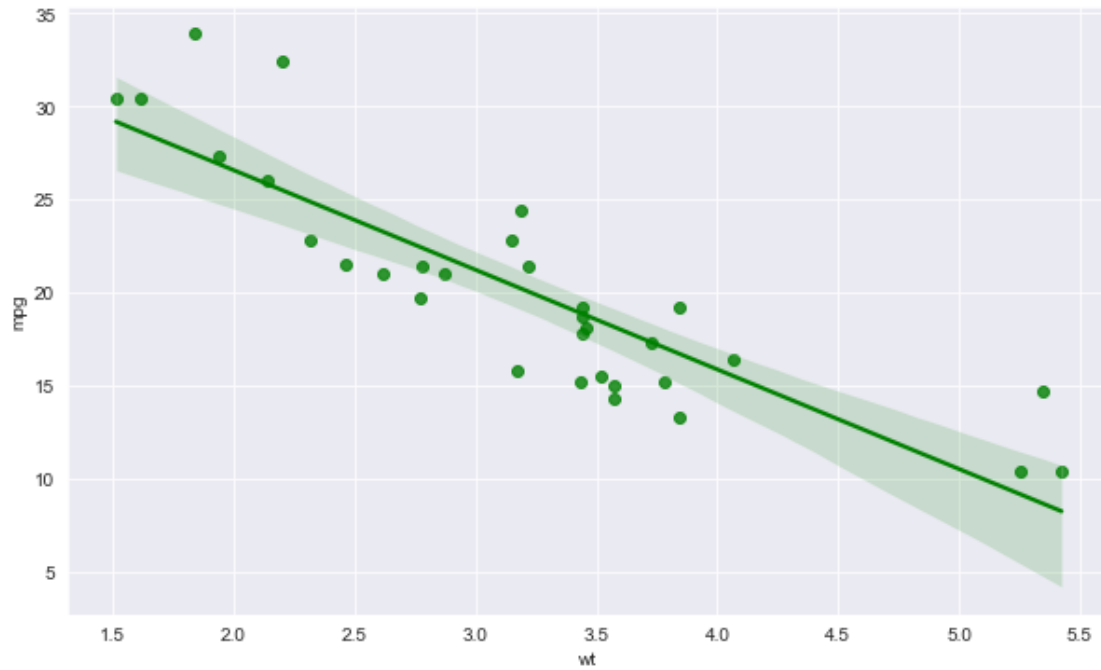
```
[136]: sns.regplot(x='wt', y='mpg', data=df)
```

```
[136]: <AxesSubplot:xlabel='wt', ylabel='mpg'>
```



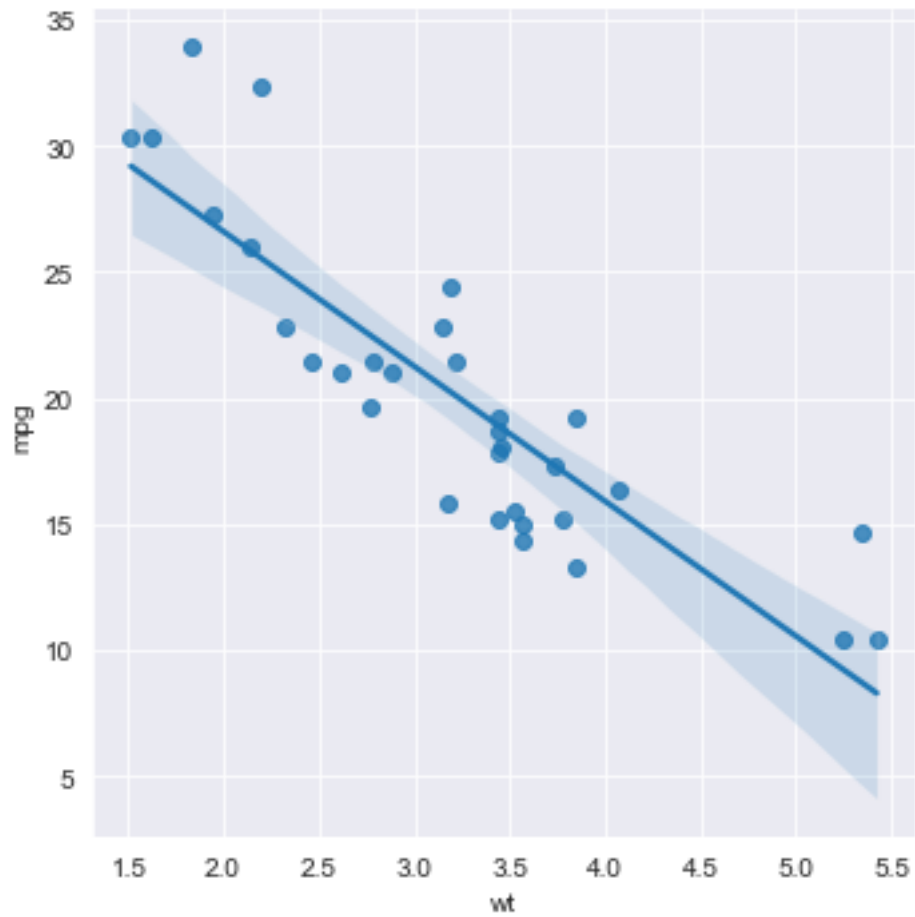
```
[137]: #changing color
sns.regplot(x='wt', y='mpg', data=df, color='g')
```

```
[137]: <AxesSubplot:xlabel='wt', ylabel='mpg'>
```

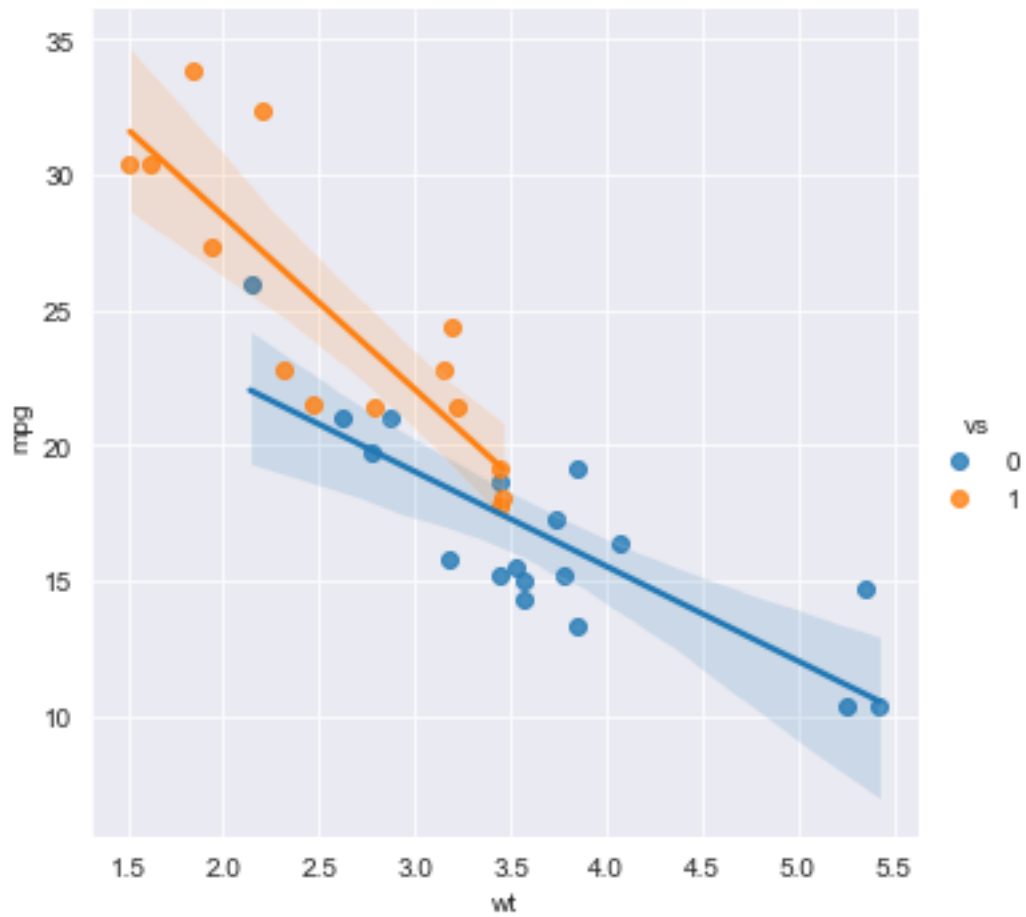
```
[138]: sns.lmplot(x='wt', y='mpg',data=df)
```

```
[138]: <seaborn.axisgrid.FacetGrid at 0x1b5683b1d60>
```



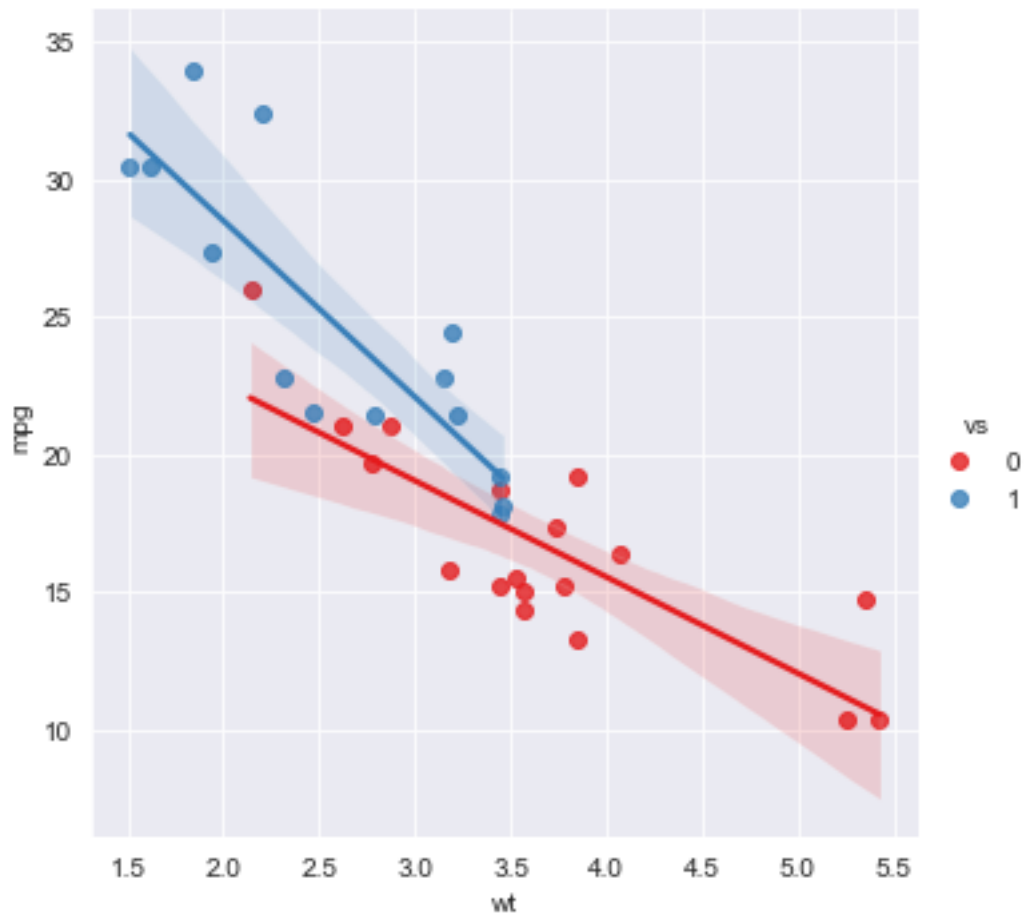
```
[139]: sns.lmplot(x='wt', y='mpg', hue='vs', data=df)
```

```
[139]: <seaborn.axisgrid.FacetGrid at 0x1b56840a9a0>
```



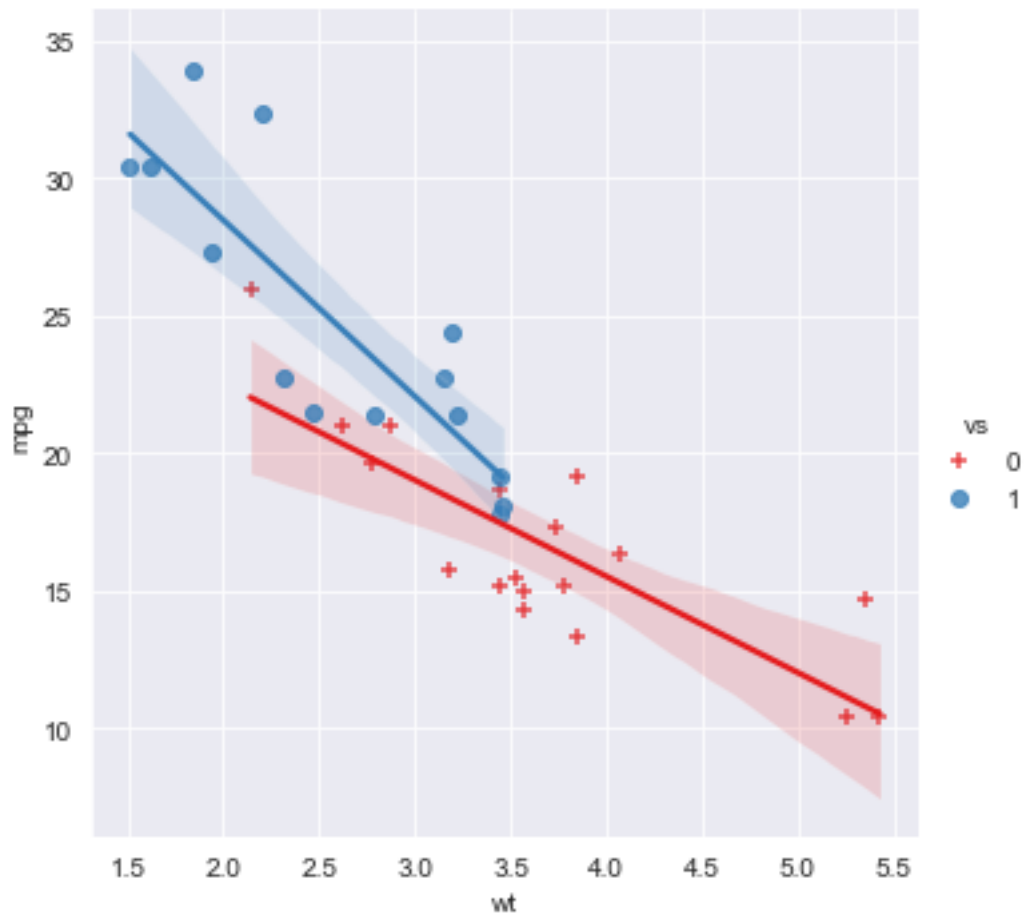
```
[140]: sns.lmplot(x='wt', y='mpg', hue='vs', palette='Set1', data=df)
```

```
[140]: <seaborn.axisgrid.FacetGrid at 0x1b568412fa0>
```



```
[141]: sns.lmplot(x='wt', y='mpg', hue='vs', markers=['+', 'o'], palette='Set1', data=df)
```

```
[141]: <seaborn.axisgrid.FacetGrid at 0x1b5684f3e80>
```



```
[142]: # Load the dataset
iris = pd.read_csv('Iris.csv')
```

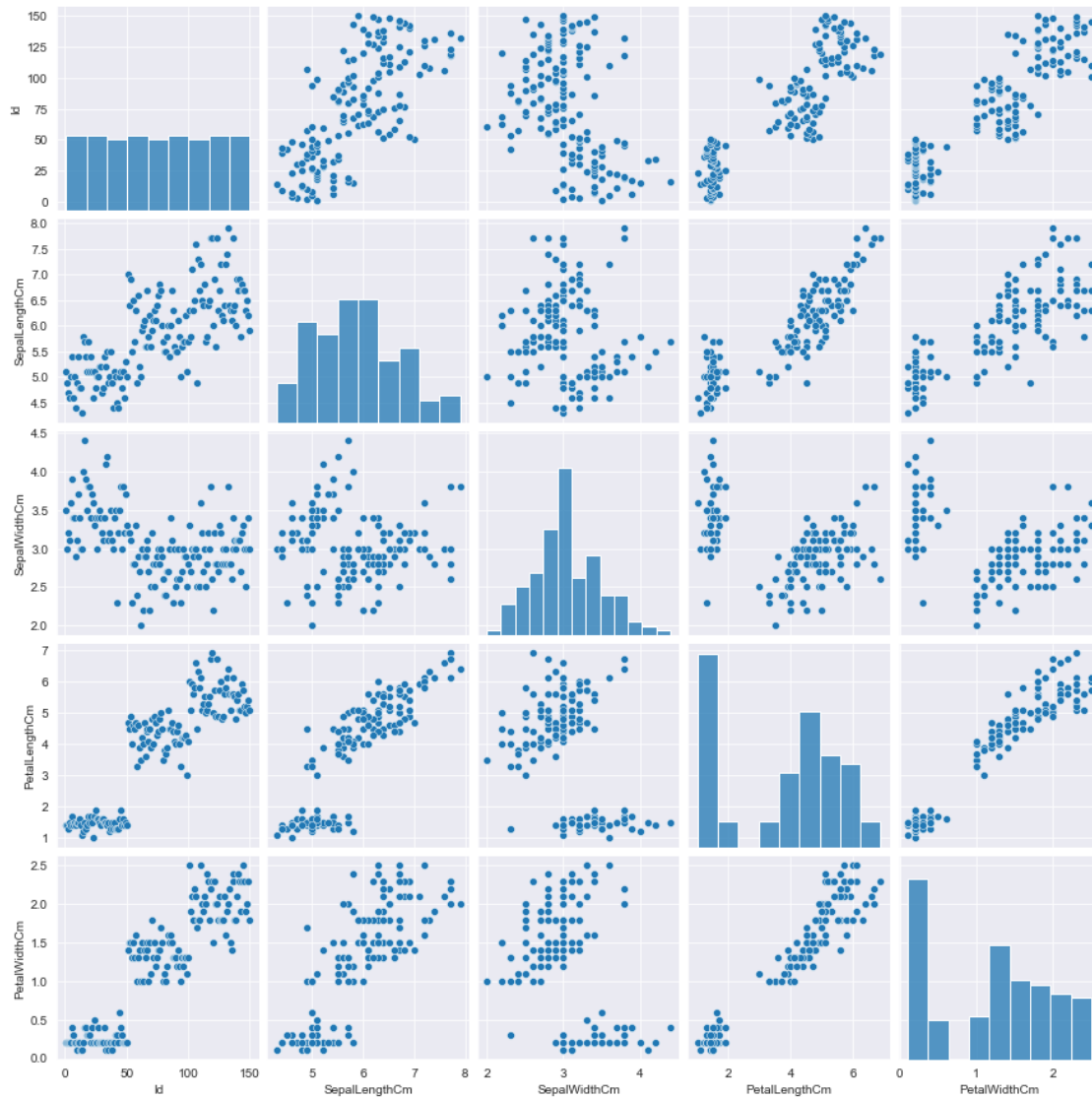
```
[143]: # viewing top 5 rows
iris.head()
```

```
[143]:
```

| | 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 |

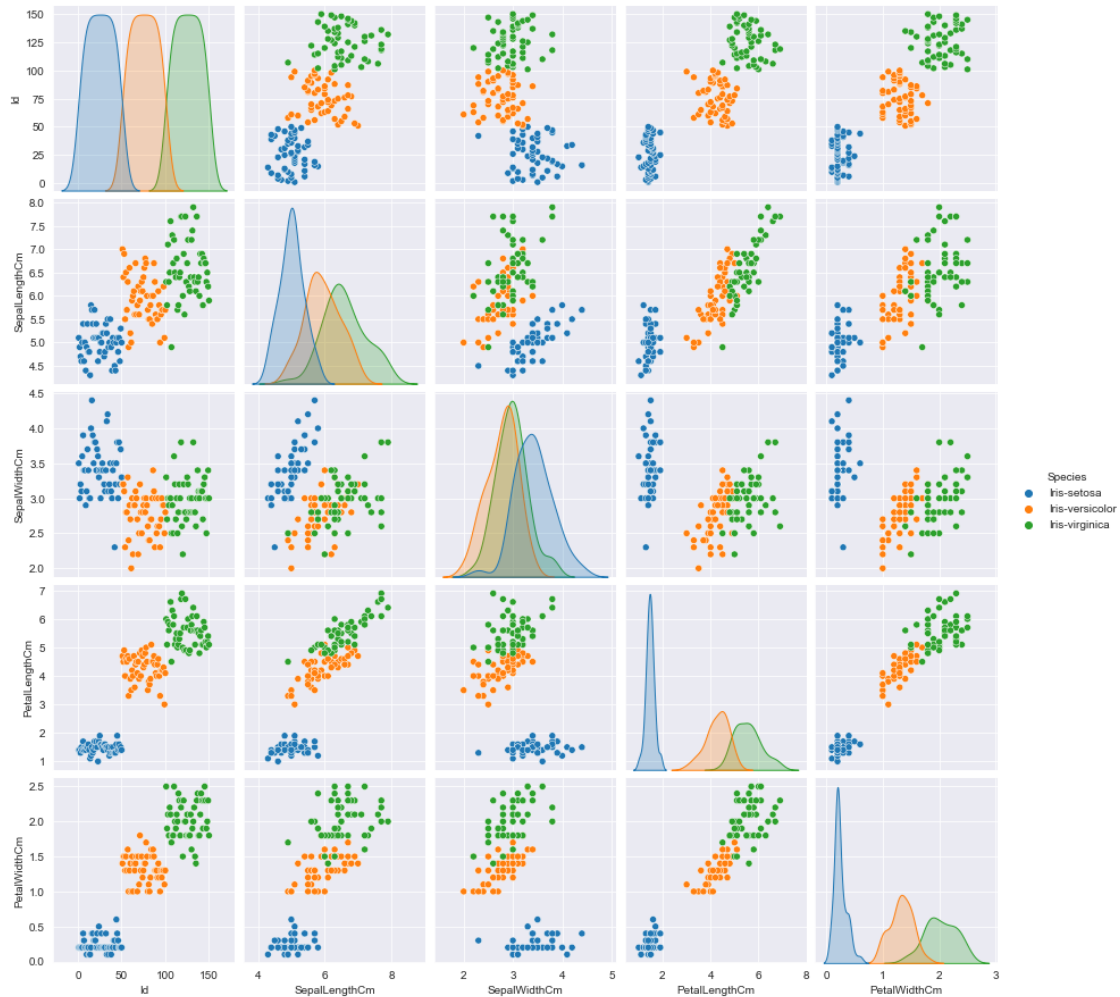
```
[144]: sns.pairplot(iris)
```

```
[144]: <seaborn.axisgrid.PairGrid at 0x1b568565c10>
```



```
[145]: sns.pairplot(iris,hue='Species')
```

```
[145]: <seaborn.axisgrid.PairGrid at 0x1b56860efd0>
```



```
[146]: ## Parallel coordinates
from pandas.plotting import parallel_coordinates
## Loading Dataset
df = pd.read_csv('NHANES.csv')
```

```
[147]: #Top 5 rows
df.head()
```

```
[147]:
```

| | SEQN | ALQ101 | ALQ110 | ALQ130 | SMQ020 | RIAGENDR | RIDAGEYR | RIDRETH1 | \ |
|---|-------|--------|--------|--------|--------|----------|----------|----------|---|
| 0 | 83732 | 1.0 | NaN | 1.0 | 1 | 1 | 62 | 3 | |
| 1 | 83733 | 1.0 | NaN | 6.0 | 1 | 1 | 53 | 3 | |
| 2 | 83734 | 1.0 | NaN | NaN | 1 | 1 | 78 | 3 | |
| 3 | 83735 | 2.0 | 1.0 | 1.0 | 2 | 2 | 56 | 3 | |
| 4 | 83736 | 2.0 | 1.0 | 1.0 | 2 | 2 | 42 | 4 | |

| | DMDCITZN | DMDEDUC2 | ... | BPXSY2 | BPXDI2 | BMXWT | BMXHT | BMXBMI | BMXLEG | \ |
|--|----------|----------|-----|--------|--------|-------|-------|--------|--------|---|
|--|----------|----------|-----|--------|--------|-------|-------|--------|--------|---|

| | | | | | | | | | |
|---|-----|-----|-----|-------|------|-------|-------|------|------|
| 0 | 1.0 | 5.0 | ... | 124.0 | 64.0 | 94.8 | 184.5 | 27.8 | 43.3 |
| 1 | 2.0 | 3.0 | ... | 140.0 | 88.0 | 90.4 | 171.4 | 30.8 | 38.0 |
| 2 | 1.0 | 3.0 | ... | 132.0 | 44.0 | 83.4 | 170.1 | 28.8 | 35.6 |
| 3 | 1.0 | 5.0 | ... | 134.0 | 68.0 | 109.8 | 160.9 | 42.4 | 38.5 |
| 4 | 1.0 | 4.0 | ... | 114.0 | 54.0 | 55.2 | 164.9 | 20.3 | 37.4 |

| | BMXARML | BMXARMC | BMXWAIST | HIQ210 |
|---|---------|---------|----------|--------|
| 0 | 43.6 | 35.9 | 101.1 | 2.0 |
| 1 | 40.0 | 33.2 | 107.9 | NaN |
| 2 | 37.0 | 31.0 | 116.5 | 2.0 |
| 3 | 37.7 | 38.3 | 110.1 | 2.0 |
| 4 | 36.0 | 27.2 | 80.4 | 2.0 |

[5 rows x 28 columns]

```
[148]: #columns
df.columns
```

```
[148]: Index(['SEQN', 'ALQ101', 'ALQ110', 'ALQ130', 'SMQ020', 'RIAGENDR', 'RIDAGEYR',
        'RIDRETH1', 'DMDCITZN', 'DMDDEDUC2', 'DMDMARTL', 'DMDHHSIZ', 'WTINT2YR',
        'SDMVPSU', 'SDMVSTRA', 'INDFMPIR', 'BPXSY1', 'BPXDI1', 'BPXSY2',
        'BPXDI2', 'BMXWT', 'BMXHT', 'BMXBMI', 'BMXLEG', 'BMXARML', 'BMXARMC',
        'BMXWAIST', 'HIQ210'],
        dtype='object')
```

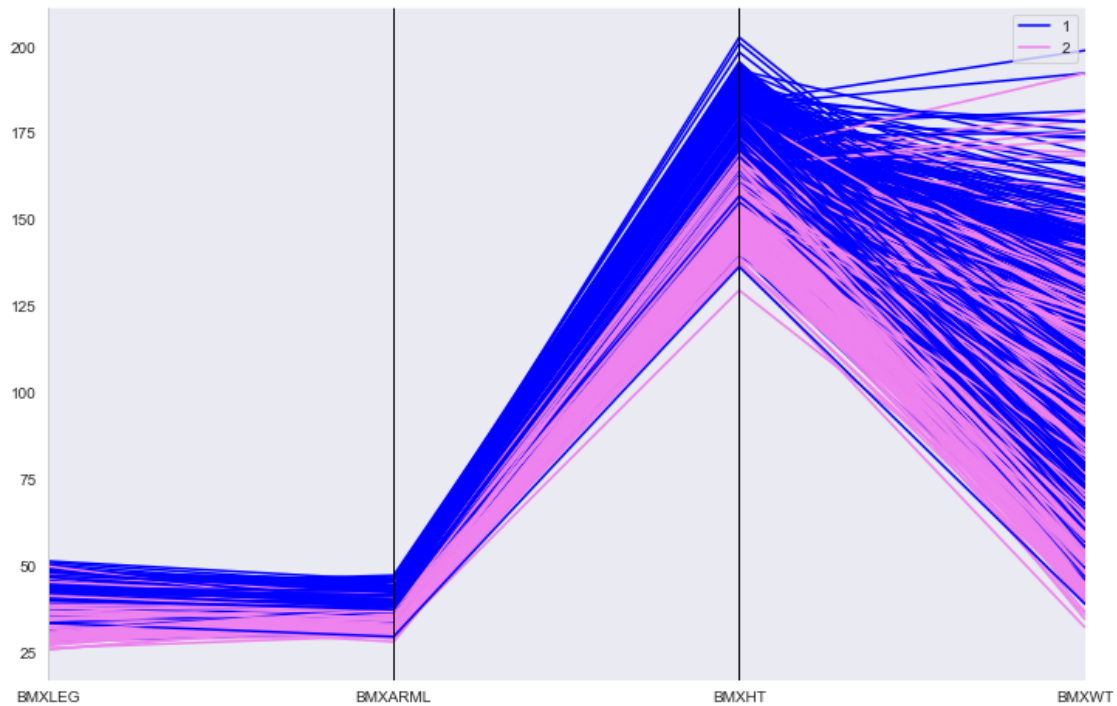
```
[149]: d = df[['BMXLEG', 'BMXARML', 'BMXHT', 'BMXWT', 'RIAGENDR']]
```

```
[150]: d.head()
```

| | BMXLEG | BMXARML | BMXHT | BMXWT | RIAGENDR |
|---|--------|---------|-------|-------|----------|
| 0 | 43.3 | 43.6 | 184.5 | 94.8 | 1 |
| 1 | 38.0 | 40.0 | 171.4 | 90.4 | 1 |
| 2 | 35.6 | 37.0 | 170.1 | 83.4 | 1 |
| 3 | 38.5 | 37.7 | 160.9 | 109.8 | 2 |
| 4 | 37.4 | 36.0 | 164.9 | 55.2 | 2 |

```
[151]: #gender based plotting
plt.figure(figsize=(12,8))
parallel_coordinates(d, 'RIAGENDR', color=['blue', 'violet'])
```

```
[151]: <AxesSubplot:>
```

```
[152]: #Loading Dataset
df = pd.read_csv('avocado.csv')
```

```
[153]: df.columns
```

```
[153]: Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046', '4225',
         '4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type',
         'year', 'region'],
        dtype='object')
```

```
[154]: df.head()
```

```
[154]:
```

| | Unnamed: 0 | Date | AveragePrice | Total Volume | 4046 | 4225 | \ |
|---|------------|------------|--------------|--------------|---------|-----------|---|
| 0 | 0 | 2015-12-27 | 1.33 | 64236.62 | 1036.74 | 54454.85 | |
| 1 | 1 | 2015-12-20 | 1.35 | 54876.98 | 674.28 | 44638.81 | |
| 2 | 2 | 2015-12-13 | 0.93 | 118220.22 | 794.70 | 109149.67 | |
| 3 | 3 | 2015-12-06 | 1.08 | 78992.15 | 1132.00 | 71976.41 | |
| 4 | 4 | 2015-11-29 | 1.28 | 51039.60 | 941.48 | 43838.39 | |

| | 4770 | Total Bags | Small Bags | Large Bags | XLarge Bags | type | \ |
|---|--------|------------|------------|------------|-------------|--------------|---|
| 0 | 48.16 | 8696.87 | 8603.62 | 93.25 | 0.0 | conventional | |
| 1 | 58.33 | 9505.56 | 9408.07 | 97.49 | 0.0 | conventional | |
| 2 | 130.50 | 8145.35 | 8042.21 | 103.14 | 0.0 | conventional | |
| 3 | 72.58 | 5811.16 | 5677.40 | 133.76 | 0.0 | conventional | |

```
4    75.78    6183.95    5986.26    197.69    0.0    conventional
```

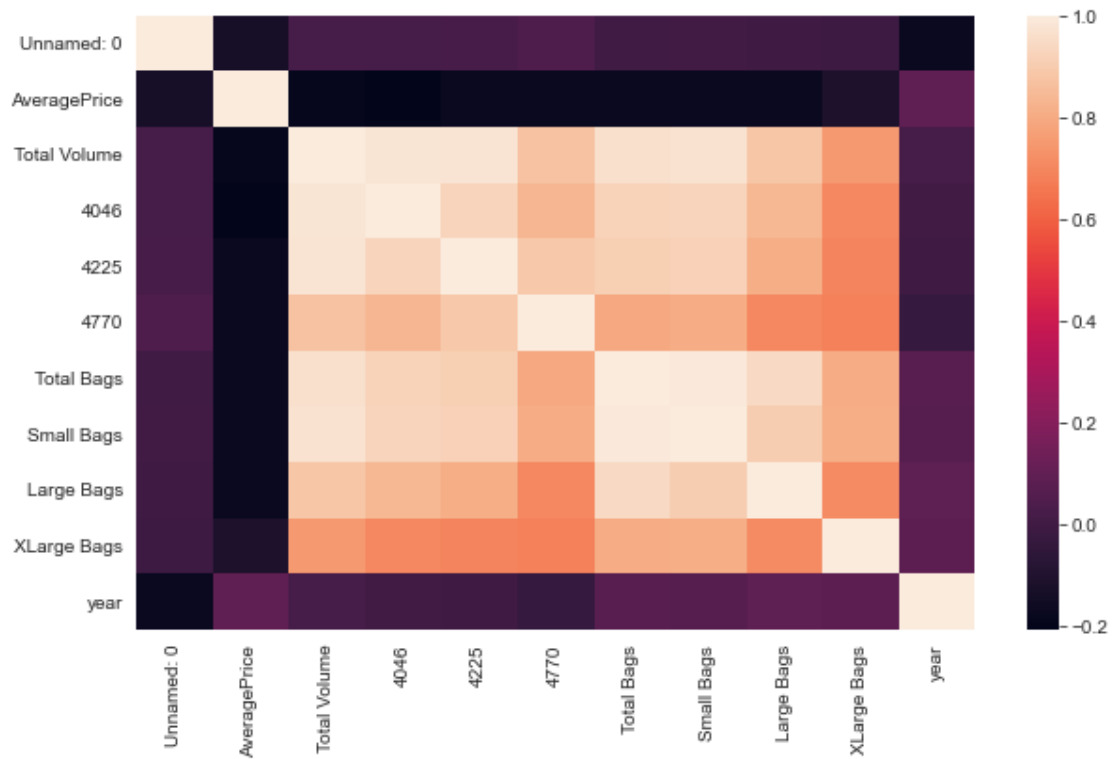
```
      year region
0  2015  Albany
1  2015  Albany
2  2015  Albany
3  2015  Albany
4  2015  Albany
```

```
[155]: df.dtypes
```

```
[155]: Unnamed: 0      int64
Date              object
AveragePrice      float64
Total Volume      float64
4046              float64
4225              float64
4770              float64
Total Bags        float64
Small Bags        float64
Large Bags        float64
XLarge Bags       float64
type              object
year              int64
region            object
dtype: object
```

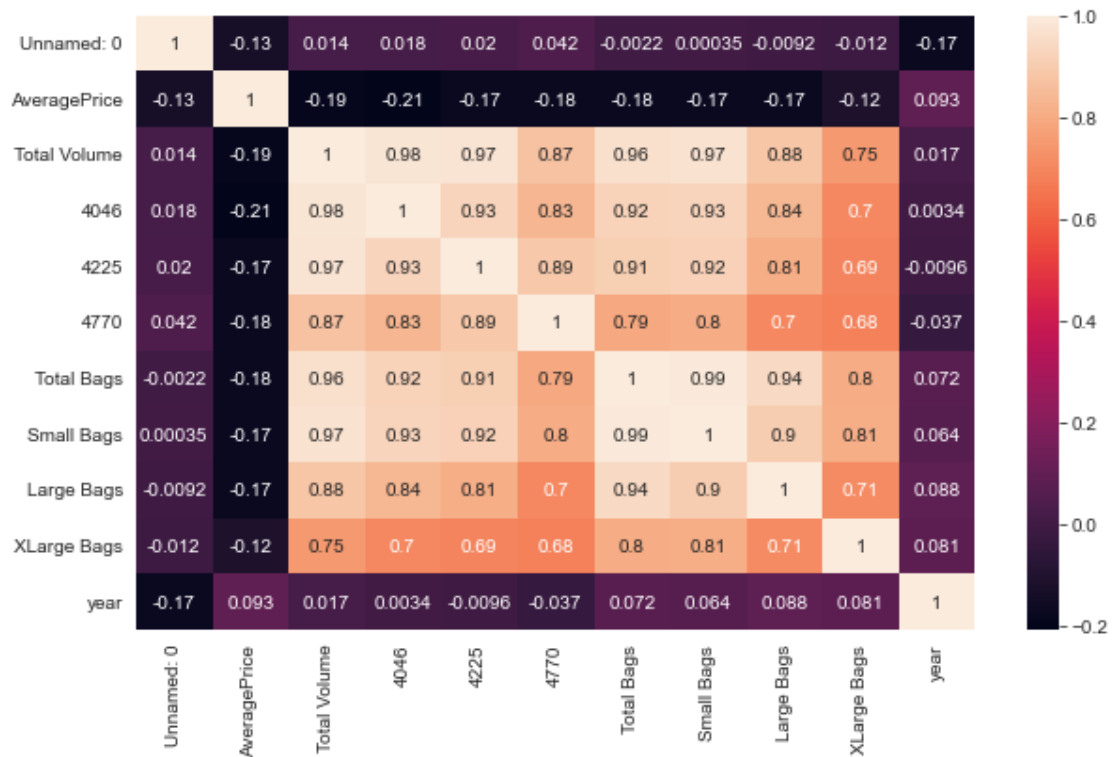
```
[156]: sns.heatmap(df.corr())
```

```
[156]: <AxesSubplot:>
```



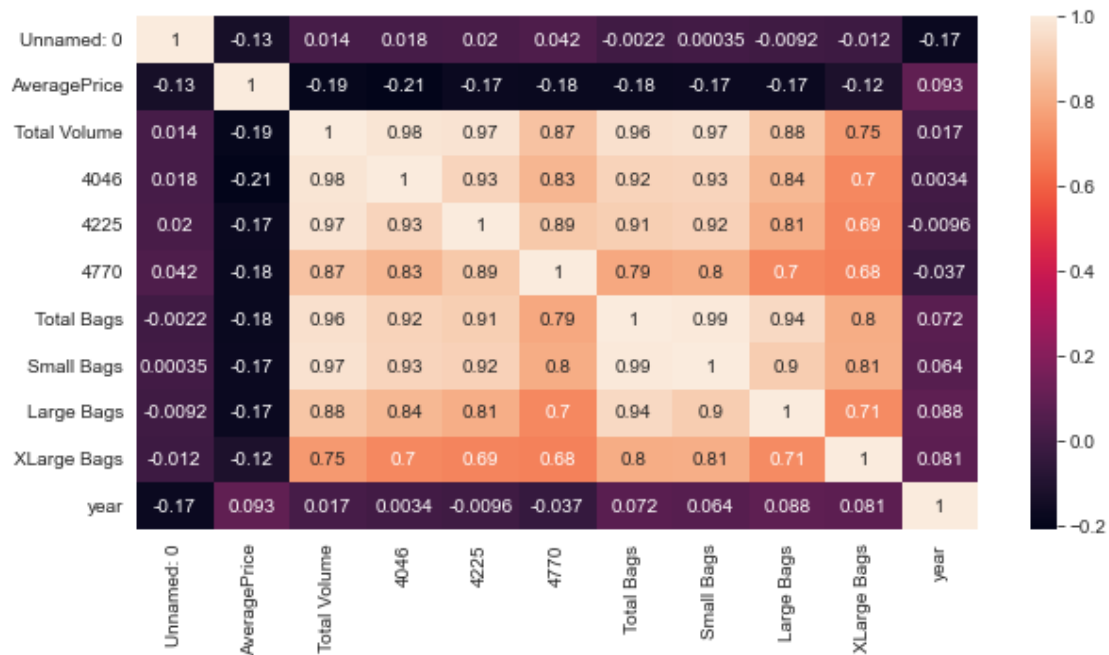
```
[157]: sns.heatmap(df.corr(), annot = True)
```

```
[157]: <AxesSubplot:>
```



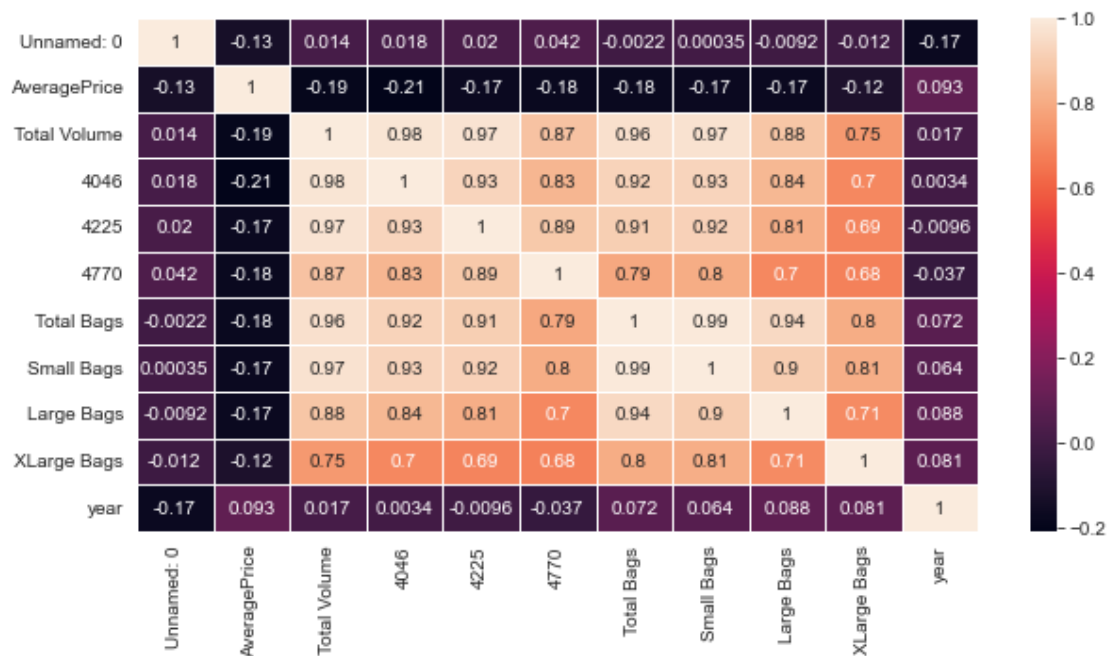
```
[158]: plt.figure(figsize=(10,5))
sns.heatmap(df.corr(), annot = True)
```

```
[158]: <AxesSubplot:>
```



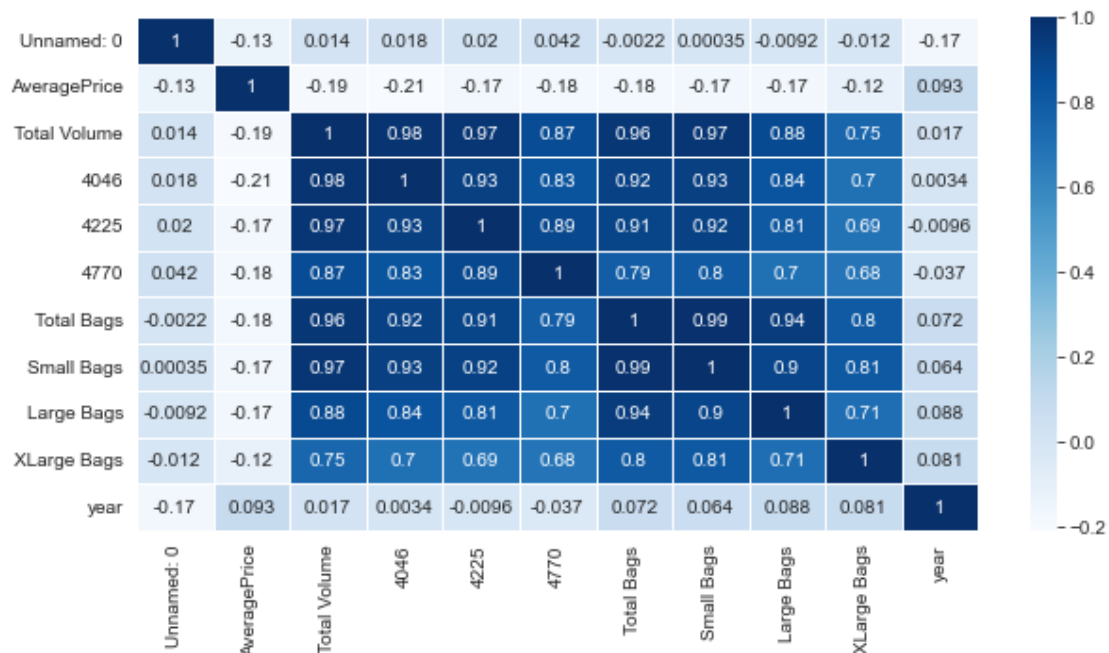
```
[159]: plt.figure(figsize=(10,5))
sns.heatmap(df.corr(), annot = True,linewidth = 0.5)
```

[159]: <AxesSubplot:>



```
[160]: plt.figure(figsize=(10,5))
sns.heatmap(df.corr(), annot = True,linewidth = 0.5, cmap='Blues')
```

[160]: <AxesSubplot:>



1.12 Refer to below Link for the datasets:

<https://drive.google.com/drive/folders/1QZxDigr3kJCTjtDBRvbIujSsiR9E3Nak?usp=sharing>

1.13 Refer to below link for “How to choose the right chart”

<https://towardsdatascience.com/data-visualization-how-to-choose-the-right-chart-part-1-d4c550085ea7>

[]: