1007_basic_analysis

January 29, 2023

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
df= pd.read_csv('winequality-red.csv')
df.head()
```

[2]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

[3]:
```
df.dtypes # type check
```

[3]:
```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
```

```
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

1 EDA

[4]:
```
df.describe()
```

[4]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

1.1 Filtering

[5]:
```
df['fixed acidity']  # Return as Series when singe []
```

[5]:
```
0        7.4
1        7.8
2        7.8
3       11.2
```

```
4        7.4
        …
1594     6.2
1595     5.9
1596     6.3
1597     5.9
1598     6.0
Name: fixed acidity, Length: 1599, dtype: float64
```

[6]:
```
df[['fixed acidity']] # two [[]], return as dataframe or list of list
```

[6]:
```
      fixed acidity
0               7.4
1               7.8
2               7.8
3              11.2
4               7.4
…               …
1594            6.2
1595            5.9
1596            6.3
1597            5.9
1598            6.0

[1599 rows x 1 columns]
```

[7]:
```
# Showing mutiple "COLUMN" as per necessary
df [['fixed acidity', 'density', 'quality']]   # misddle value shown as ...␣
↪later we will see how to see as full
```

[7]:
```
      fixed acidity  density  quality
0               7.4  0.99780        5
1               7.8  0.99680        5
2               7.8  0.99700        5
3              11.2  0.99800        6
4               7.4  0.99780        5
…               …        …       …
1594            6.2  0.99490        5
1595            5.9  0.99512        6
1596            6.3  0.99574        6
1597            5.9  0.99547        5
1598            6.0  0.99549        6

[1599 rows x 3 columns]
```

[8]:
```
# Test1: Check Spefic "ROW" as requrired or with any condition
```

```
df [df['fixed acidity'] > 9]  # Check when fixed acidity > 9; syntax df inside␣
 ↪df
```

[8]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
3              11.2              0.28         0.56             1.9      0.075
56             10.2              0.42         0.57             3.4      0.070
68              9.3              0.32         0.57             2.0      0.074
74              9.7              0.32         0.54             2.5      0.094
88              9.3              0.39         0.44             2.1      0.107
…               …                …            …               …          …
1470           10.0              0.69         0.11             1.4      0.084
1474            9.9              0.50         0.50            13.8      0.205
1476            9.9              0.50         0.50            13.8      0.205
1543           11.1              0.44         0.42             2.2      0.064
1548           11.2              0.40         0.50             2.0      0.099

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
3                    17.0                  60.0  0.99800  3.16       0.58
56                    4.0                  10.0  0.99710  3.04       0.63
68                   27.0                  65.0  0.99690  3.28       0.79
74                   28.0                  83.0  0.99840  3.28       0.82
88                   34.0                 125.0  0.99780  3.14       1.22
…                     …                    …        …     …          …
1470                  8.0                  24.0  0.99578  2.88       0.47
1474                 48.0                  82.0  1.00242  3.16       0.75
1476                 48.0                  82.0  1.00242  3.16       0.75
1543                 14.0                  19.0  0.99758  3.25       0.57
1548                 19.0                  50.0  0.99783  3.10       0.58

      alcohol  quality
3         9.8        6
56        9.6        5
68       10.7        5
74        9.6        5
88        9.5        5
…         …        …
1470      9.7        5
1474      8.8        5
1476      8.8        5
1543     10.4        6
1548     10.4        5

[441 rows x 12 columns]
```

[9]:
```
# Test 2
df [(df['fixed acidity'] > 9) & (df['citric acid'] > 0.5)]  # Multiple␣
 ↪condition over row, more condtion can be added inside parenthesis
```

[9]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
3              11.2              0.28         0.56             1.9      0.075
56             10.2              0.42         0.57             3.4      0.070
68              9.3              0.32         0.57             2.0      0.074
74              9.7              0.32         0.54             2.5      0.094
151             9.2              0.52         1.00             3.4      0.610
…             …                …            …              …         …
1221           10.9              0.32         0.52             1.8      0.132
1319            9.1              0.76         0.68             1.7      0.414
1414           10.0              0.32         0.59             2.2      0.077
1416           10.0              0.32         0.59             2.2      0.077
1454           11.7              0.45         0.63             2.2      0.073

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
3                    17.0                  60.0  0.99800  3.16       0.58
56                    4.0                  10.0  0.99710  3.04       0.63
68                   27.0                  65.0  0.99690  3.28       0.79
74                   28.0                  83.0  0.99840  3.28       0.82
151                  32.0                  69.0  0.99960  2.74       2.00
…                   …                   …      …     …    …
1221                 17.0                  44.0  0.99734  3.28       0.77
1319                 18.0                  64.0  0.99652  2.90       1.33
1414                  3.0                  15.0  0.99940  3.20       0.78
1416                  3.0                  15.0  0.99940  3.20       0.78
1454                  7.0                  23.0  0.99974  3.21       0.69

      alcohol  quality
3         9.8        6
56        9.6        5
68       10.7        5
74        9.6        5
151       9.4        4
…       …      …
1221     11.5        6
1319      9.1        6
1414      9.6        5
1416      9.6        5
1454     10.9        6

[142 rows x 12 columns]
```

[10]:
```python
df [(df['fixed acidity'] > 9) & (df['citric acid'] > 0.5) & (df['pH'] >=3)]   # Test 3
```

[10]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
3              11.2              0.28         0.56             1.9      0.075
56             10.2              0.42         0.57             3.4      0.070
```

```
68              9.3              0.32         0.57              2.0      0.074
74              9.7              0.32         0.54              2.5      0.094
197            11.5              0.30         0.60              2.0      0.067
…               …                …           …                …         …
1220           10.9              0.32         0.52              1.8      0.132
1221           10.9              0.32         0.52              1.8      0.132
1414           10.0              0.32         0.59              2.2      0.077
1416           10.0              0.32         0.59              2.2      0.077
1454           11.7              0.45         0.63              2.2      0.073

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
3                    17.0                  60.0  0.99800  3.16       0.58
56                    4.0                  10.0  0.99710  3.04       0.63
68                   27.0                  65.0  0.99690  3.28       0.79
74                   28.0                  83.0  0.99840  3.28       0.82
197                  12.0                  27.0  0.99810  3.11       0.97
…                     …                    …       …      …        …
1220                 17.0                  44.0  0.99734  3.28       0.77
1221                 17.0                  44.0  0.99734  3.28       0.77
1414                  3.0                  15.0  0.99940  3.20       0.78
1416                  3.0                  15.0  0.99940  3.20       0.78
1454                  7.0                  23.0  0.99974  3.21       0.69

      alcohol  quality
3         9.8        6
56        9.6        5
68       10.7        5
74        9.6        5
197      10.1        6
…        …       …
1220     11.5        6
1221     11.5        6
1414      9.6        5
1416      9.6        5
1454     10.9        6

[131 rows x 12 columns]
```

[11]:
```python
# In case of row wise selection, its returning all
df [(df['fixed acidity'] > 9) | (df['citric acid'] > 0.5)]  # Or Function
```

[11]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
3              11.2              0.28         0.56             1.9      0.075
16              8.5              0.28         0.56             1.8      0.092
19              7.9              0.32         0.51             1.8      0.341
47              8.7              0.29         0.52             1.6      0.113
56             10.2              0.42         0.57             3.4      0.070
```

```
…          …            …        …          …      …
1548          11.2           0.40       0.50             2.0      0.099
1566           6.7           0.16       0.64             2.1      0.059
1570           6.4           0.36       0.53             2.2      0.230
1574           5.6           0.31       0.78            13.9      0.074
1576           8.0           0.30       0.63             1.6      0.081

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
3                    17.0                  60.0  0.99800  3.16       0.58
16                   35.0                 103.0  0.99690  3.30       0.75
19                   17.0                  56.0  0.99690  3.04       1.08
47                   12.0                  37.0  0.99690  3.25       0.58
56                    4.0                  10.0  0.99710  3.04       0.63
…                     …                    …       …     …       …
1548                 19.0                  50.0  0.99783  3.10       0.58
1566                 24.0                  52.0  0.99494  3.34       0.71
1570                 19.0                  35.0  0.99340  3.37       0.93
1574                 23.0                  92.0  0.99677  3.39       0.48
1576                 16.0                  29.0  0.99588  3.30       0.78

      alcohol  quality
3         9.8        6
16       10.5        7
19        9.2        6
47        9.5        5
56        9.6        5
…       …       …
1548     10.4        5
1566     11.2        6
1570     12.4        6
1574     10.5        6
1576     10.8        6

[489 rows x 12 columns]
```

[12]:

```python
# Row Wise File - with only selective columns as per condtion
df.loc[df['fixed acidity'] == 9.2, ['fixed acidity', 'citric acid', 'pH']]  #
 ↪syntax: loc means, we are locationg, then condtion as ==9.2 & name of col as
 ↪want to show
# as per given condtion, the total retunr number / qnt not show, we can see it
 ↪as next code
```

[12]:

	fixed acidity	citric acid	pH
151	9.2	1.00	2.74
457	9.2	0.21	3.28
460	9.2	0.52	3.35
491	9.2	0.50	3.34

```
524             9.2          0.49  3.23
540             9.2          0.24  3.26
614             9.2          0.18  2.87
691             9.2          0.24  3.48
741             9.2          0.24  3.21
765             9.2          0.10  3.31
880             9.2          0.18  3.15
905             9.2          0.20  3.23
1093            9.2          0.36  3.33
1170            9.2          0.34  3.20
1225            9.2          0.23  3.15
1360            9.2          0.31  3.24
```

[13]:
```python
data = df.loc[df['fixed acidity'] == 9.2, ['fixed acidity', 'citric acid',␣
 ↪'pH']]  # just bring the previous code in a var and see the shape
data.shape
```

[13]:
```
(16, 3)
```

[14]:
```python
df.tail(7)
```

[14]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
1592            6.3             0.510         0.13             2.3      0.076
1593            6.8             0.620         0.08             1.9      0.068
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
1592                 29.0                  40.0  0.99574  3.42       0.75
1593                 28.0                  38.0  0.99651  3.42       0.82
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol  quality
1592     11.0        6
1593      9.5        6
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6
```

[15]:
```
df[5:11] # want to see from index 5 to 10
```

[15]:
```
    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
5             7.4              0.66         0.00             1.8      0.075
6             7.9              0.60         0.06             1.6      0.069
7             7.3              0.65         0.00             1.2      0.065
8             7.8              0.58         0.02             2.0      0.073
9             7.5              0.50         0.36             6.1      0.071
10            6.7              0.58         0.08             1.8      0.097

    free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
5                  13.0                  40.0   0.9978  3.51       0.56
6                  15.0                  59.0   0.9964  3.30       0.46
7                  15.0                  21.0   0.9946  3.39       0.47
8                   9.0                  18.0   0.9968  3.36       0.57
9                  17.0                 102.0   0.9978  3.35       0.80
10                 15.0                  65.0   0.9959  3.28       0.54

    alcohol  quality
5       9.4        5
6       9.4        5
7      10.0        7
8       9.5        7
9      10.5        5
10      9.2        5
```

View Full Rows & Column, middle ... will not be shown

[16]:
```
# Required codes for see full, but its not recommended actually or no needed
# pd.set_option('display.max_rows',None)
# pd.set_option('display.max_columns', None)
# df
```

1.2 Processed File Export

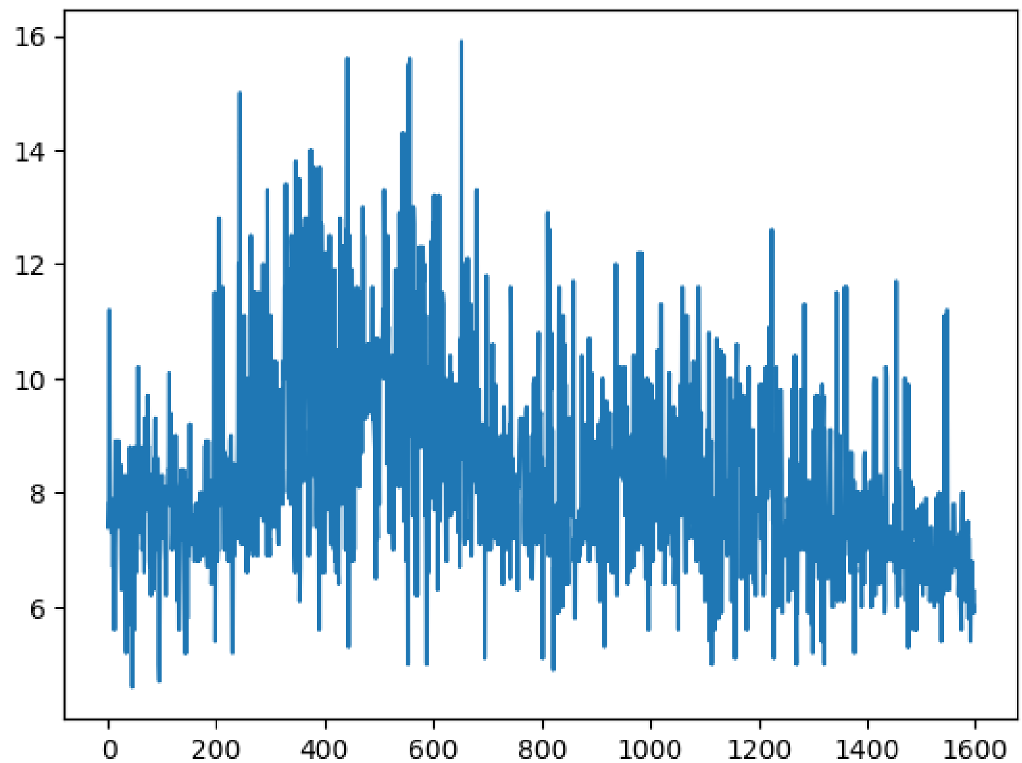
[17]:
```
# x = df [(df['fixed acidity'] > 9) | (df['citric acid'] > 0.5)]   # processed␣
↪file stored into a variable file as x
# x.to_csv('1007_processedfile.csv')
```

2 Visualization

Very basic or Ordinary Visualization

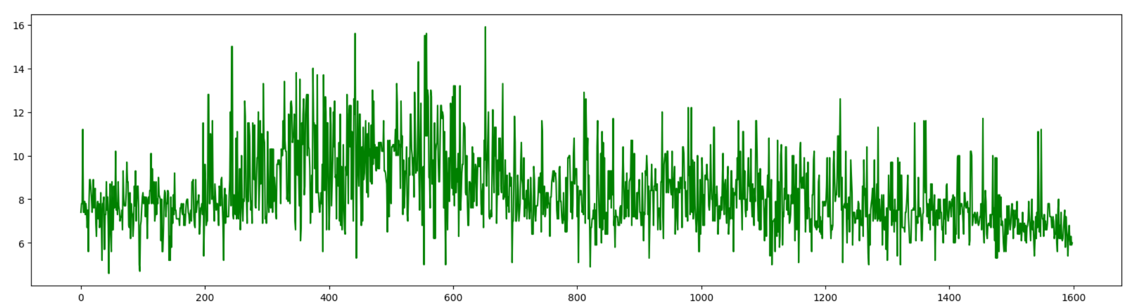
[18]:
```
# Plot with a single columns
df['fixed acidity'].plot.line()
```

[18]:
```
<AxesSubplot:>
```




```
[19]: df['fixed acidity'].plot.line(figsize =(20,5), color='green')
```

[19]: <AxesSubplot:>



3 Better Visualization

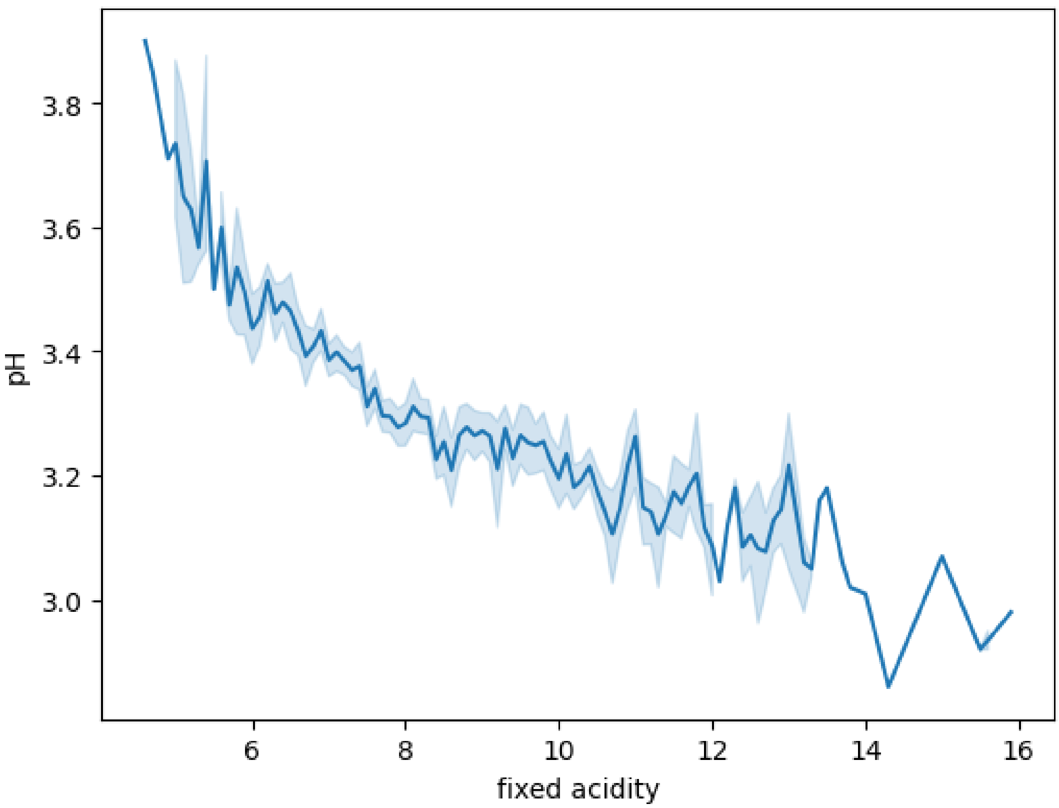
3.1 Seaborn

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

- There is Negative Relationship between Acidity & Ph, lets visualize the relationship between these two
- Sns lineplot must require independent & dependent variable, here x is indendendent & y is dependent over x

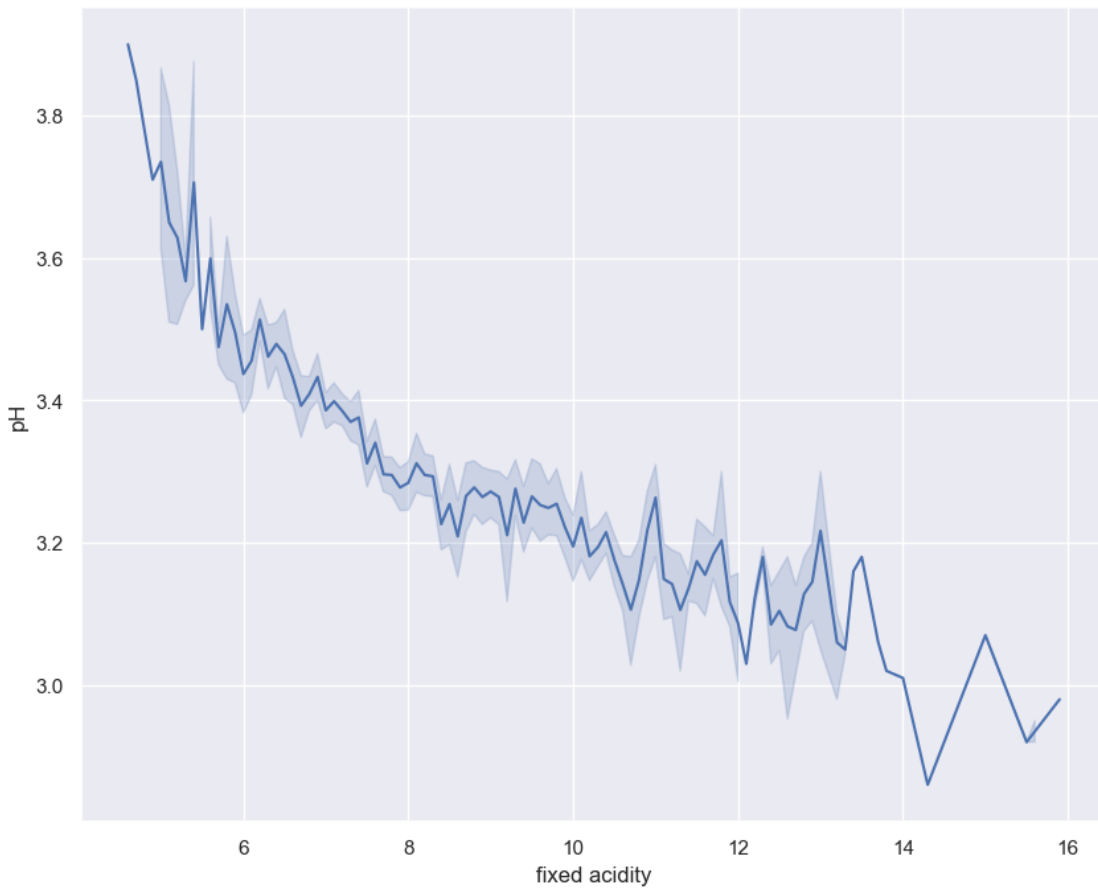
[21]:
```
sns.lineplot(data=df, x='fixed acidity', y='pH')
```

[21]: <AxesSubplot:xlabel='fixed acidity', ylabel='pH'>



[32]:
```
# Figure Size Customization
sns.set(rc={'figure.figsize':(10, 8)})
sns.lineplot(data=df, x='fixed acidity', y='pH')
# The shadow represent the deviation / standard deviation / error rate or how
↪much deviation is there
```

[32]: <AxesSubplot:xlabel='fixed acidity', ylabel='pH'>

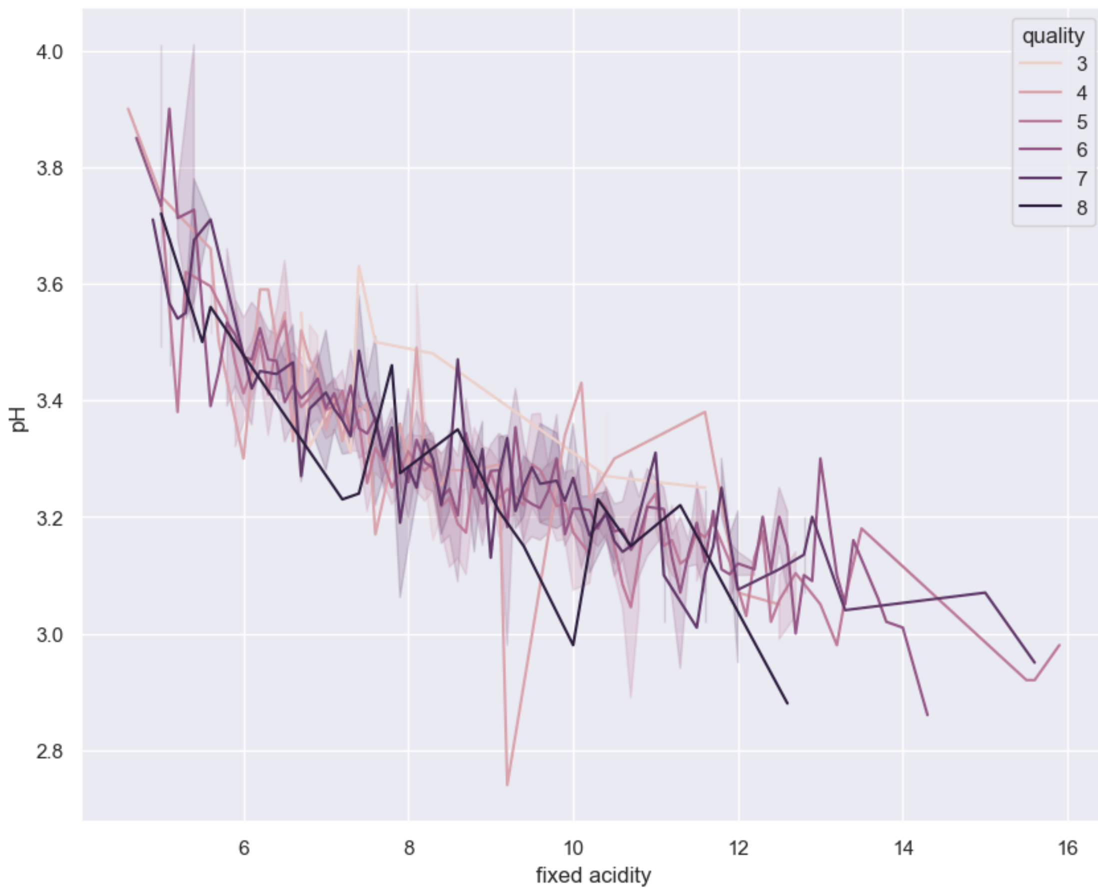


3.1.1 Categorical Data Visualization

- Male / Famale type
- In this dataset, quality is categorical data
- Event
- By the category you want to represent data, that must be passed in “hue”
- Checking Facebook traffic for day & night

[34]:
```
sns.lineplot(data=df, x='fixed acidity', y='pH', hue='quality')  # We have 6␣
↪type of category
```

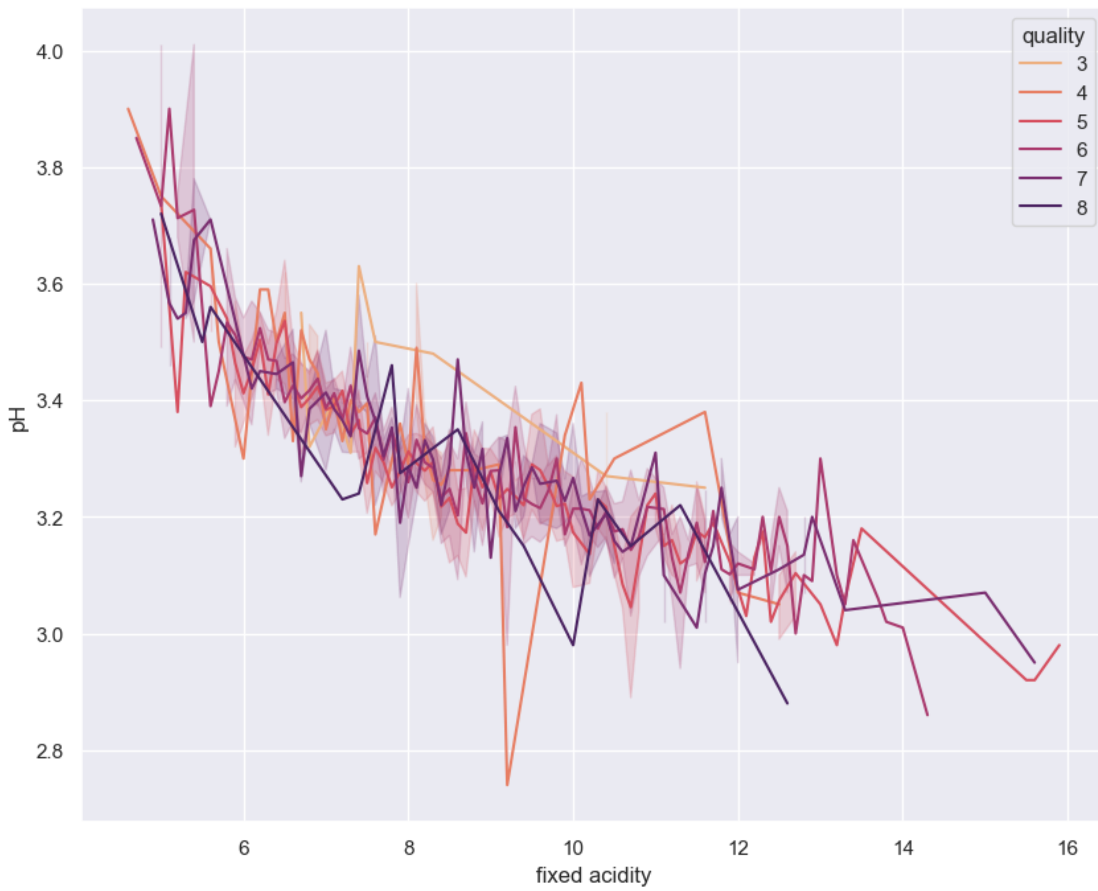
[34]:
```
<AxesSubplot:xlabel='fixed acidity', ylabel='pH'>
```



3.2 Color Customization

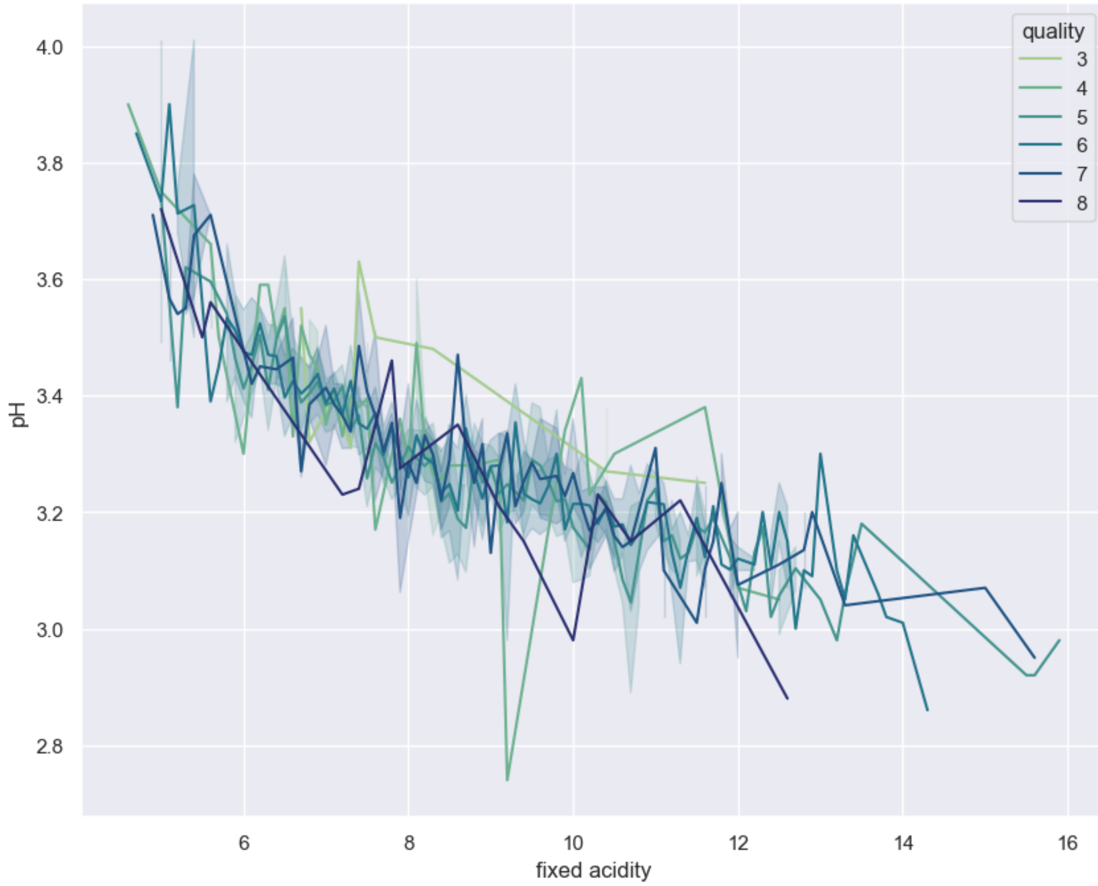
[36]:
```
p = sns.color_palette("flare", as_cmap=True)  # customized color palette stored␣
↪in p & then pass to following code for color customization
sns.lineplot(data=df, x='fixed acidity', y='pH', hue='quality', palette=p)
```

[36]: `<AxesSubplot:xlabel='fixed acidity', ylabel='pH'>`



[38]:
```
p = sns.color_palette("crest", as_cmap=True)  # Another color palette
sns.lineplot(data=df, x='fixed acidity', y='pH', hue='quality', palette=p)
```

[38]: <AxesSubplot:xlabel='fixed acidity', ylabel='pH'>



[39]:
```
p = sns.color_palette("Spectral", as_cmap=True)  # Another color palette
sns.lineplot(data=df, x='fixed acidity', y='pH', hue='quality', palette=p)
```

[39]: <AxesSubplot:xlabel='fixed acidity', ylabel='pH'>

3.3 Creationg New Dataset for Visualization

[52]:
```python
df2 = [[ 'Alice', 30, 'Male', 55 ], ['Bobe', 17,'Female', 25 ], ['Jeba', 11,␣
↪'Female', 12], ['Tom', 45,'Male', 72], ['Rita', 21, 'Female', 35],␣
↪['Jackline', 51, 'Female', 65], ['Peter', 55, 'Male', 72]]
df2 = pd.DataFrame(df2, columns=['Name', 'Age', 'Gender','Weight in KG'])  #␣
↪Column declaration
df2
```

[52]:

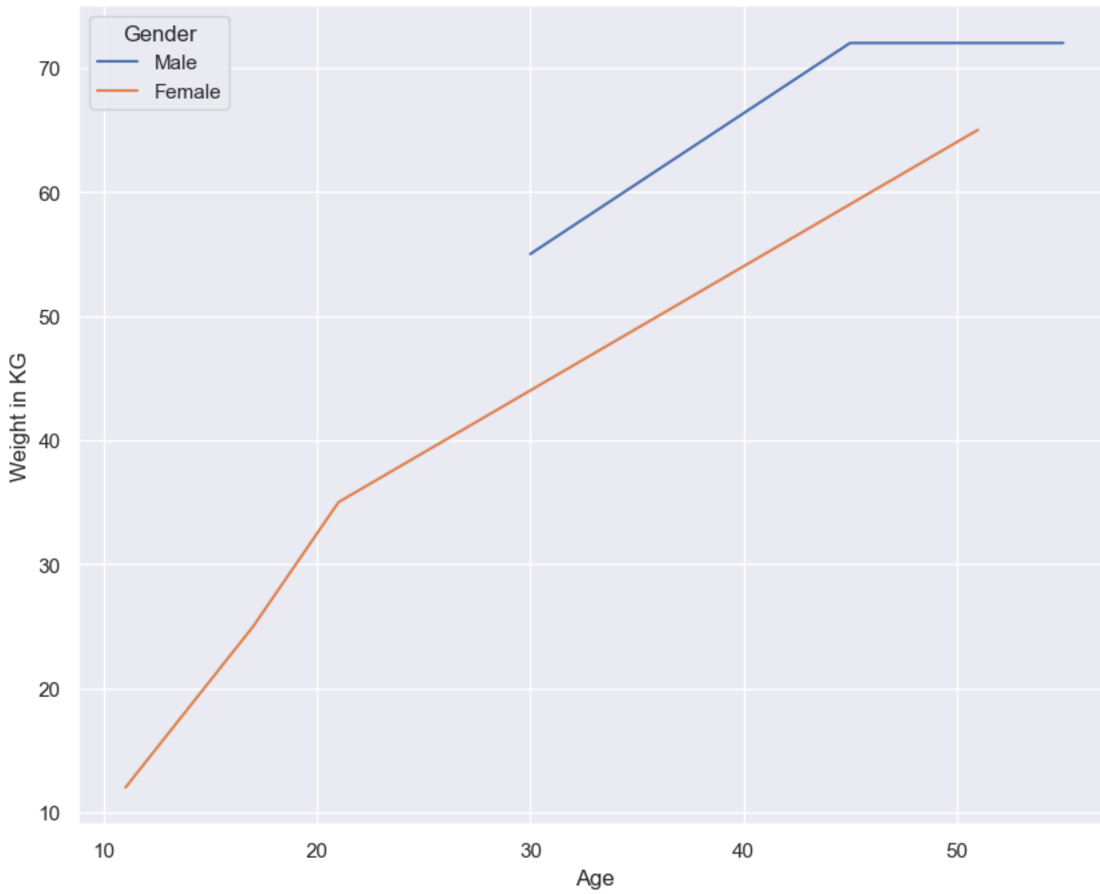
	Name	Age	Gender	Weight in KG
0	Alice	30	Male	55
1	Bobe	17	Female	25
2	Jeba	11	Female	12
3	Tom	45	Male	72
4	Rita	21	Female	35
5	Jackline	51	Female	65
6	Peter	55	Male	72

- We will see a visualization for Age & Weight in respect of Gender

- More Application: facebook traffic day & night

[55]:
```
sns.lineplot(data=df2, x='Age', y='Weight in KG', hue='Gender')  # hue param is␣
↪usefull when divided by category
```

[55]: `<AxesSubplot:xlabel='Age', ylabel='Weight in KG'>`



3.4 Creating Test Data

[60]:
```
tips = [
    [16.99, 1.01, 'Female', 'No', 'Sun', 'Dinner', 2],
    [10.34, 1.66, 'Male', 'No', 'Sun', 'Dinner', 3],
    [21.01, 3.50, 'Male', 'No', 'Sun', 'Dinner', 3],
    [23.65, 3.50, 'Male', 'No', 'Mon', 'Dinner', 4],
    [24.59, 3.61, 'Female', 'No', 'Sun', 'Dinner', 3],
]
tips = pd.DataFrame(data=tips, columns=['Total Bill', 'Tips','Gender',␣
↪'Smoker', 'Day', 'Time', 'Size'])
tips
```

[60]:

	Total Bill	Tips	Gender	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	23.65	3.50	Male	No	Mon	Dinner	4
4	24.59	3.61	Female	No	Sun	Dinner	3

[64]:

```
# Alternative Way for Dataset creation with seperate col
tips2 = [
    [16.99, 1.01, 'Female', 'No', 'Sun', 'Dinner', 2],
    [10.34, 1.66, 'Male', 'No', 'Sun', 'Dinner', 3],
    [21.01, 3.50, 'Male', 'No', 'Sun', 'Dinner', 3],
    [23.65, 3.50, 'Male', 'No', 'Mon', 'Dinner', 4],
    [24.59, 3.61, 'Female', 'No', 'Sun', 'Dinner', 3],
]
c = ['Total Bill', 'Tips','Gender', 'Smoker', 'Day', 'Time', 'Size']    # Just␣
↪keep a column var as c & keep the col name here
tips2 = pd.DataFrame(data=tips, columns= c )  # Pass the c here
tips2
```

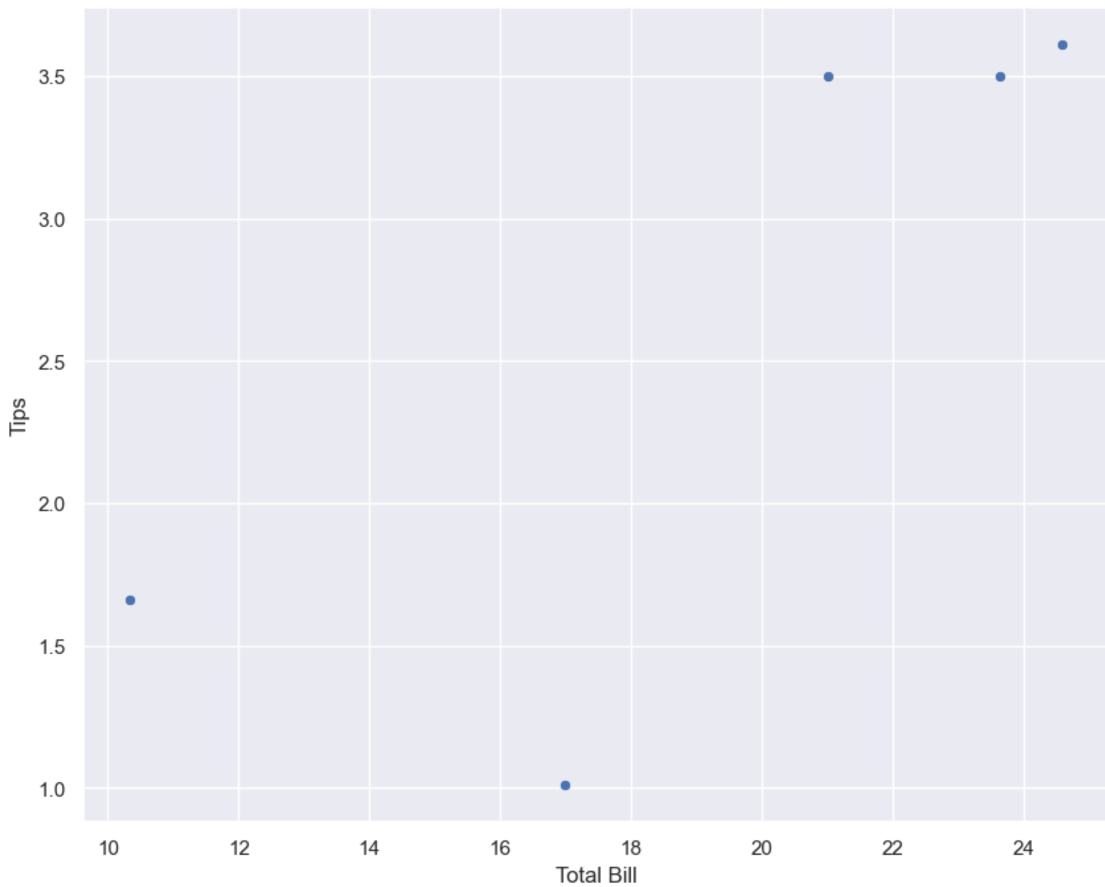
[64]:

	Total Bill	Tips	Gender	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	23.65	3.50	Male	No	Mon	Dinner	4
4	24.59	3.61	Female	No	Sun	Dinner	3

[65]:

```
sns.scatterplot(data=tips, x='Total Bill', y='Tips')
```

[65]: <AxesSubplot:xlabel='Total Bill', ylabel='Tips'>



[71]:
```
tips = pd.read_csv('tips.csv')
tips.head()
```

[71]:

	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

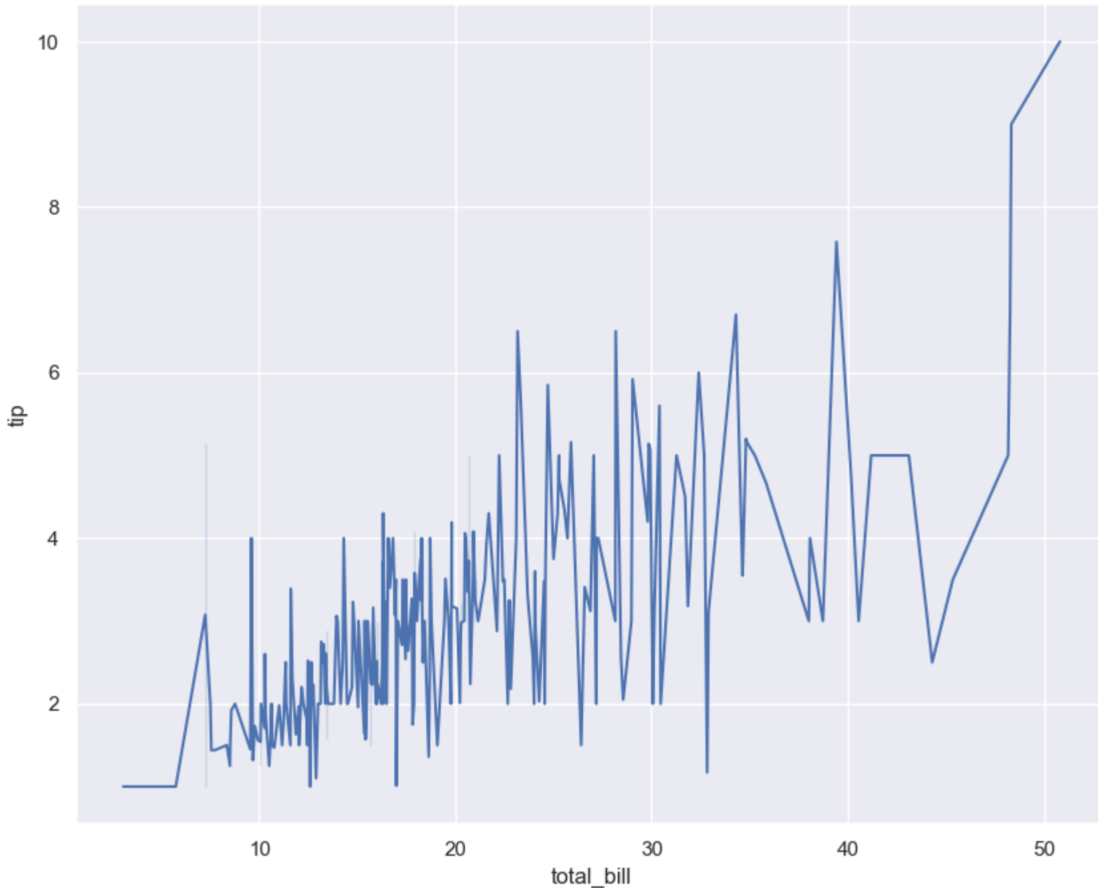
[74]:
```
tips.shape
```

[74]: (244, 7)

3.4.1 Line Plot

[75]:
```
sns.lineplot(data=tips, x='total_bill', y='tip')
```

[75]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



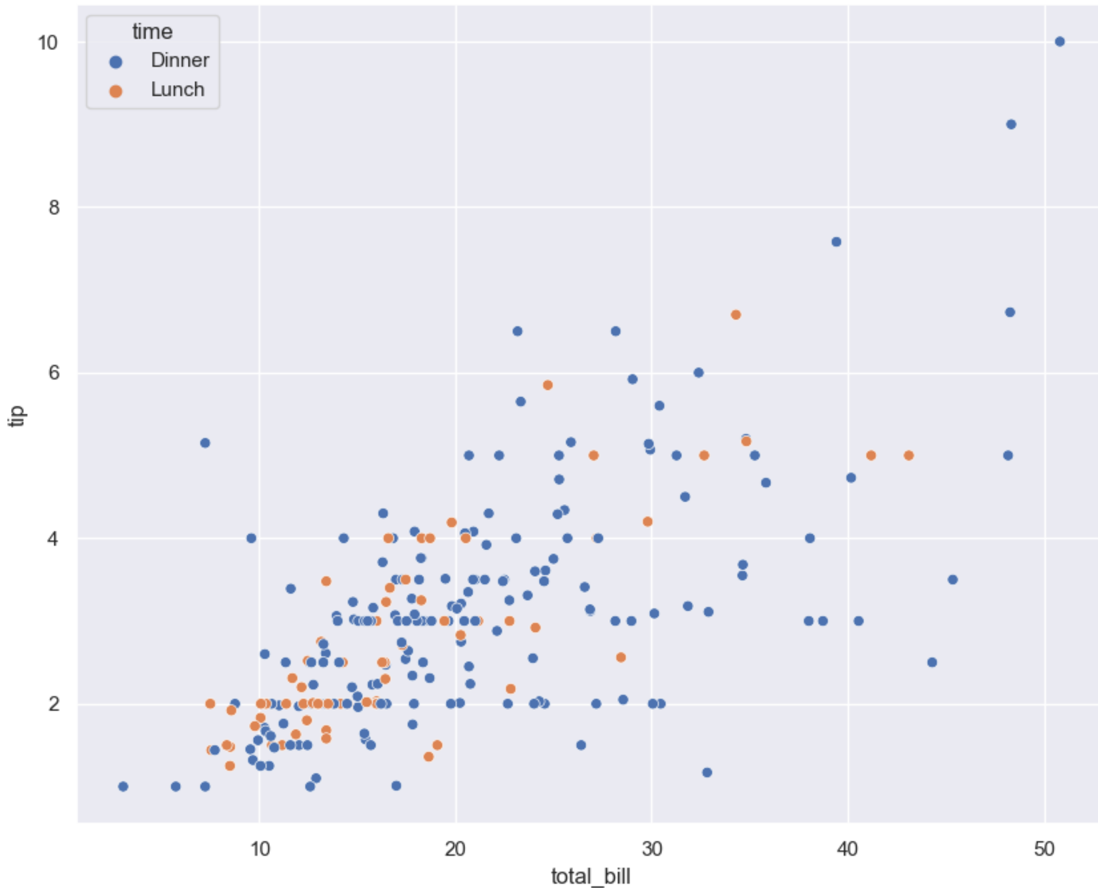
3.4.2 Scatter plot

[73]:
```
sns.scatterplot(data=tips, x='total_bill', y='tip')
```

[73]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>

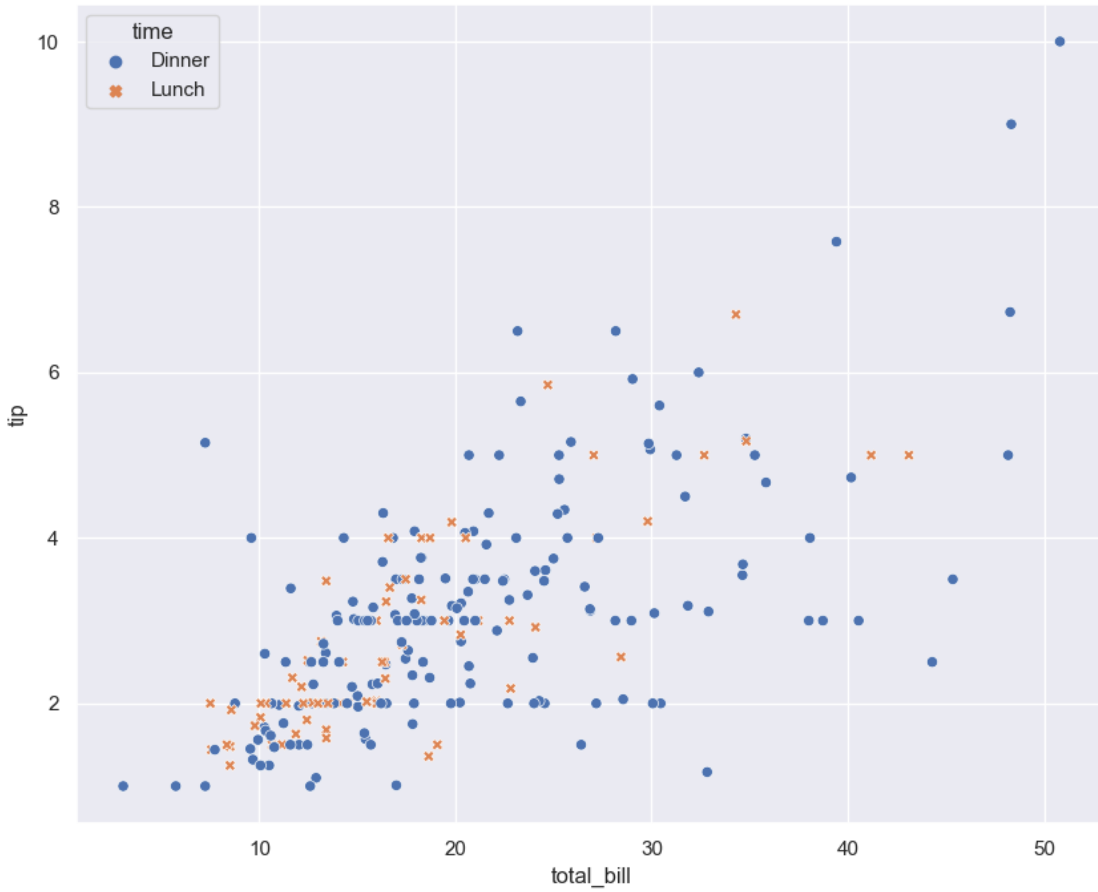
[76]:
```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='time')
```

[76]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



[77]:
```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='time', style='time')
```

[77]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>

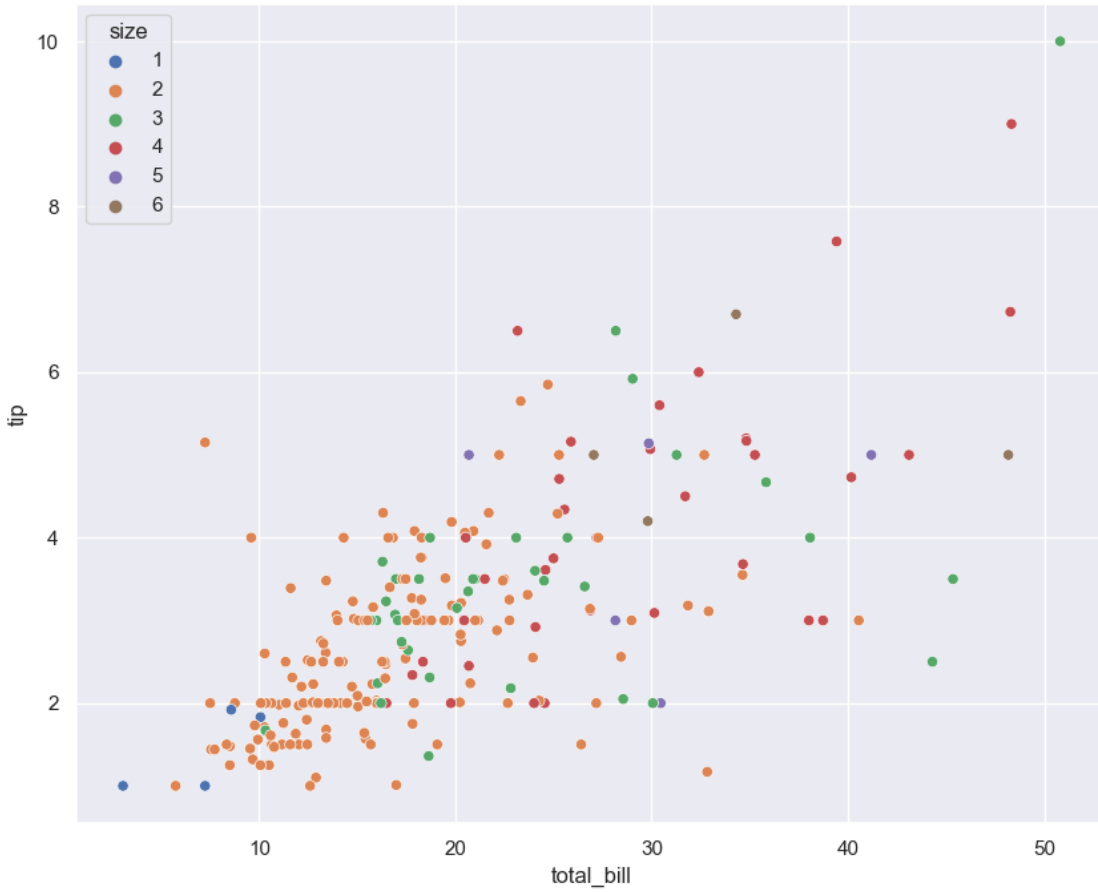


[78]:
```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='size')
```

[78]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>

[79]:
```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='size', palette="deep")
```

[79]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>

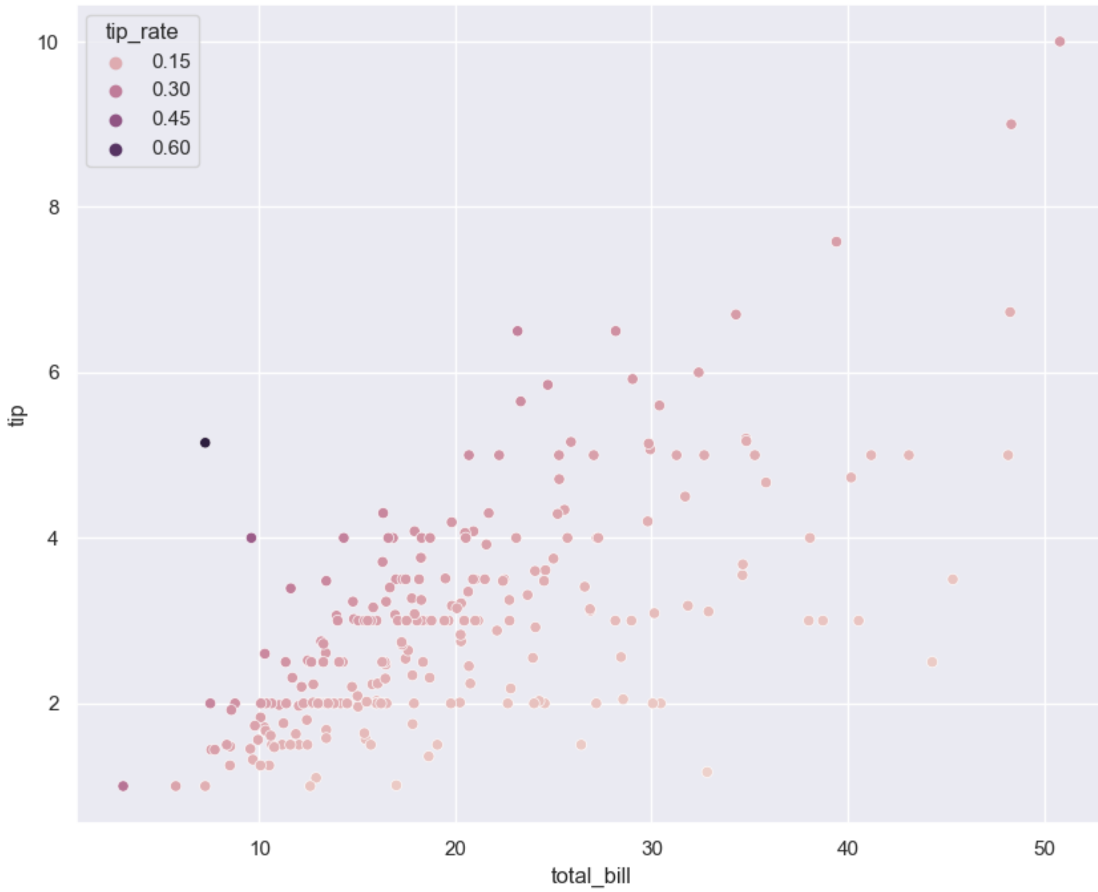


3.4.3 Tip rate

- If there are a large number of unique numeric values, the legend will show a representative, evenly-spaced set:

[84]:
```
tip_rate = tips.eval("tip / total_bill").rename("tip_rate")
sns.scatterplot(data=tips, x="total_bill", y="tip", hue=tip_rate)
```

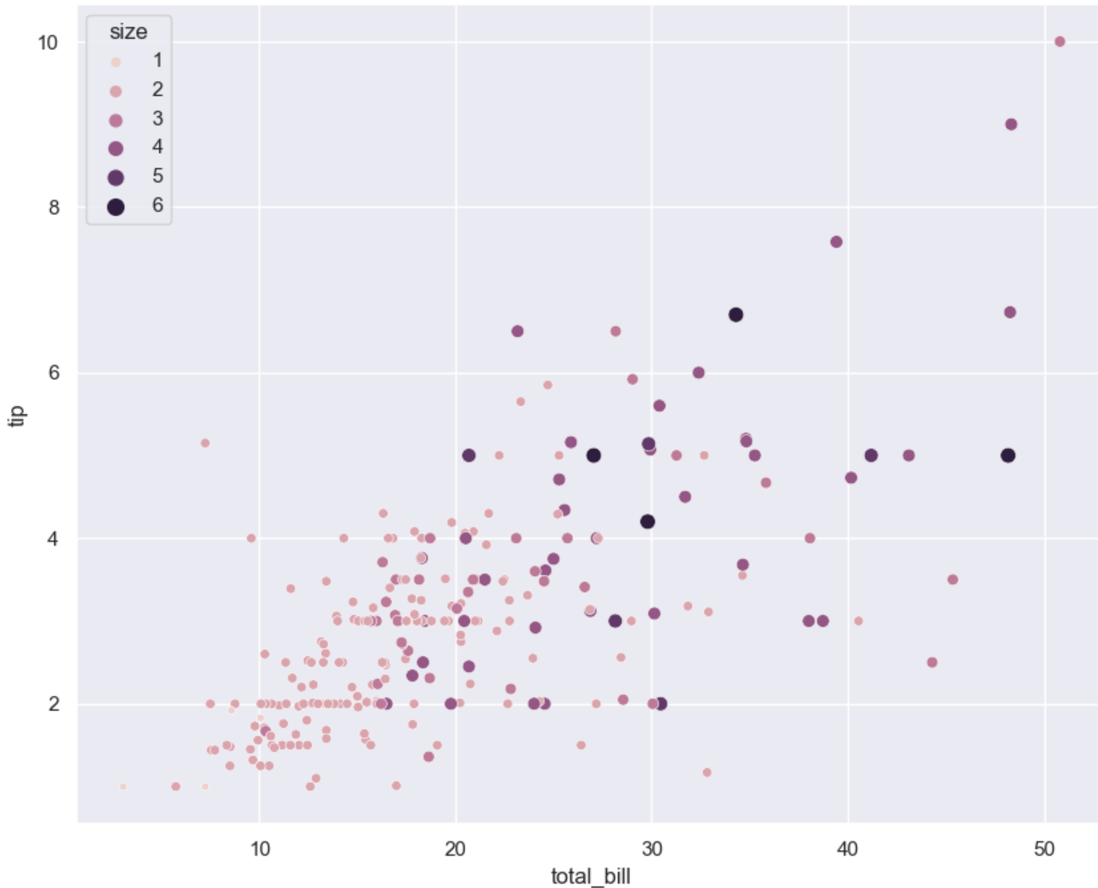
[84]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



- A numeric variable can also be assigned to size to apply a semantic mapping to the areas of the points:

[85]:
```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='size', size="size")
```

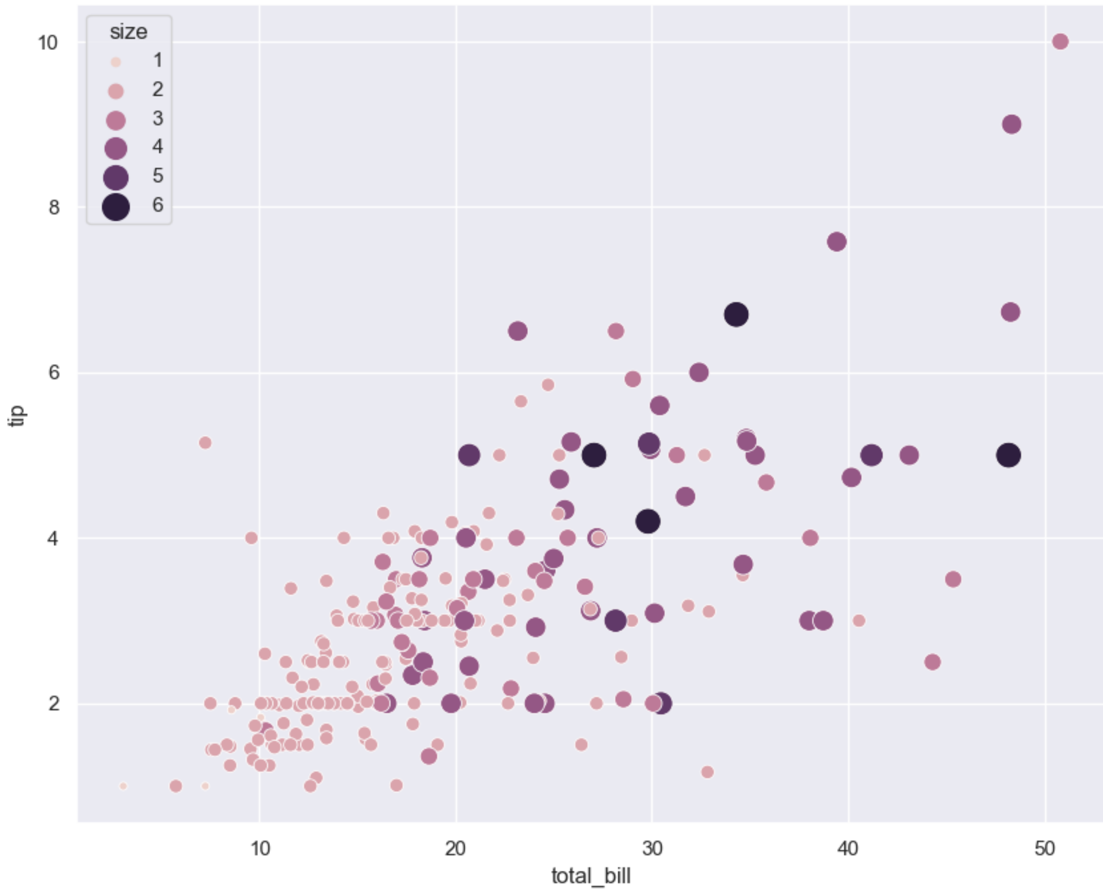
[85]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



- Control the range of marker areas with sizes, and set lengend="full" to force every unique value to appear in the legend:

[86]:
```
sns.scatterplot(
    data=tips, x="total_bill", y="tip", hue="size", size="size",
    sizes=(20, 200), legend="full"
)
```

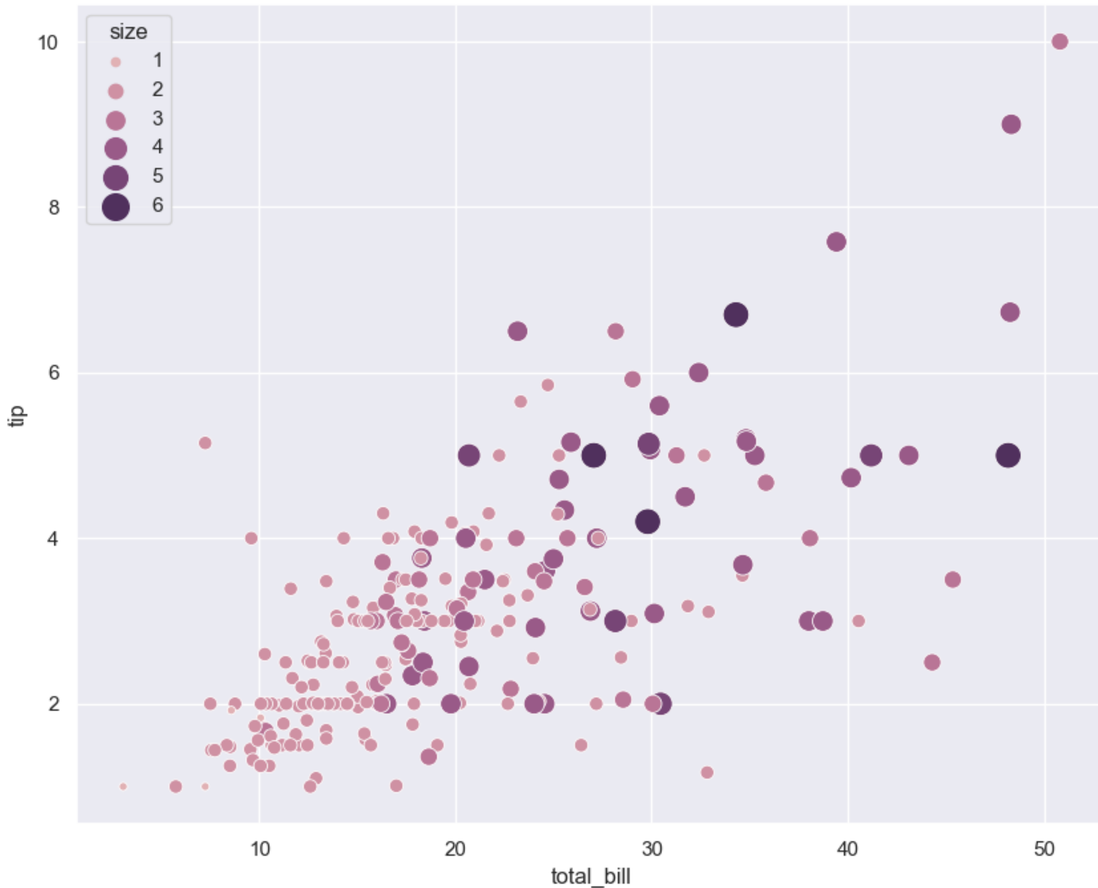
[86]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



- Pass a tuple of values or a matplotlib.colors.Normalize object to hue_norm to control the quantitative hue mapping:

[87]:
```
sns.scatterplot(
    data=tips, x="total_bill", y="tip", hue="size", size="size",
    sizes=(20, 200), hue_norm=(0, 7), legend="full"
)
```

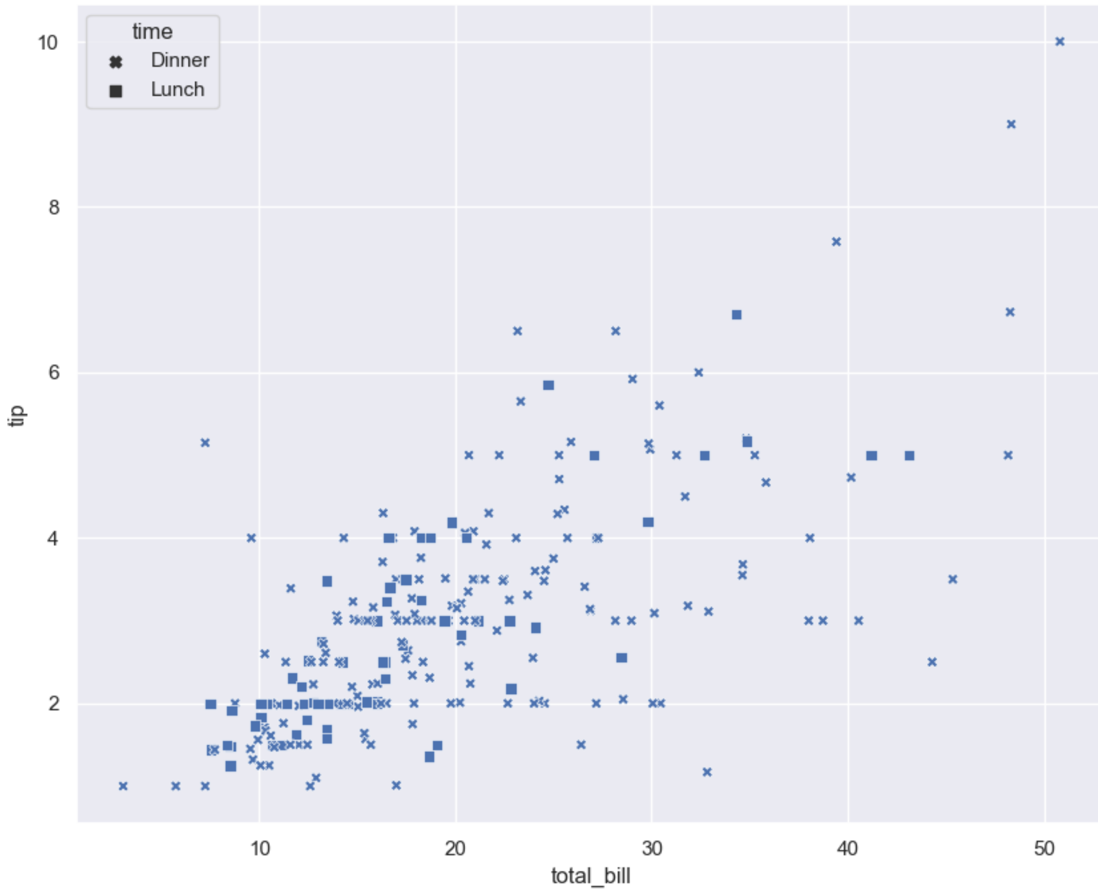
[87]:
```
<AxesSubplot:xlabel='total_bill', ylabel='tip'>
```



- Control the specific markers used to map the style variable by passing a Python list or dictionary of marker codes:

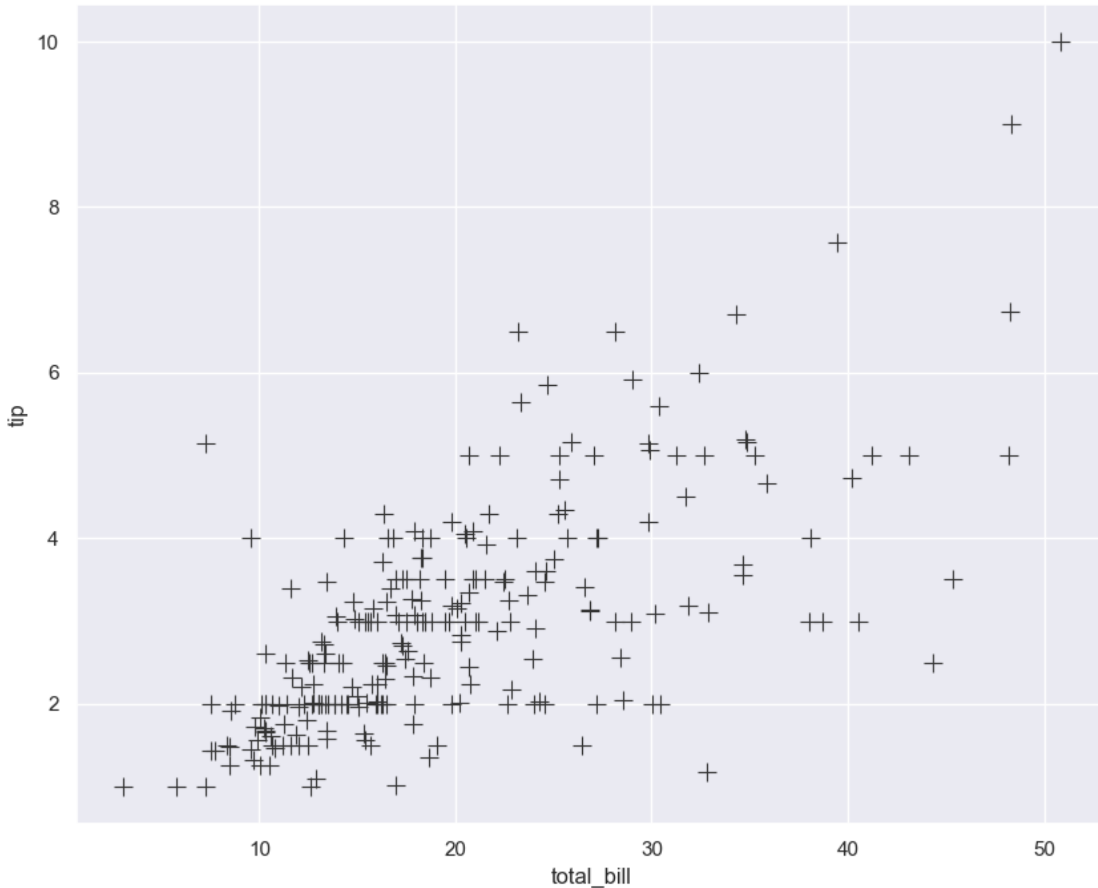
[88]:
```
markers = {"Lunch": "s", "Dinner": "X"}
sns.scatterplot(data=tips, x="total_bill", y="tip", style="time",␣
 ↪markers=markers)
```

[88]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



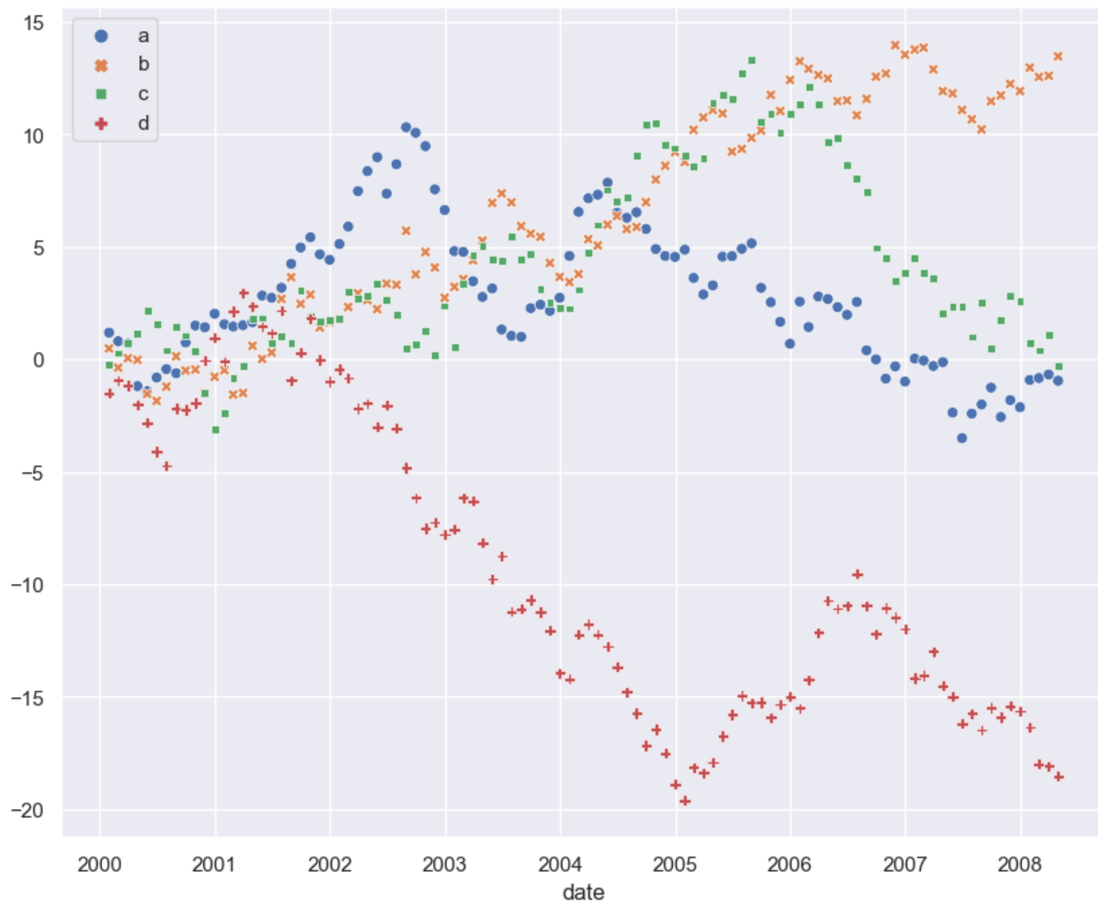
[89]:
```
sns.scatterplot(data=tips, x="total_bill", y="tip", s=100, color=".2",
    marker="+")
```

[89]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>



[90]:
```
index = pd.date_range("1 1 2000", periods=100, freq="m", name="date")
data = np.random.randn(100, 4).cumsum(axis=0)
wide_df = pd.DataFrame(data, index, ["a", "b", "c", "d"])
sns.scatterplot(data=wide_df)
```

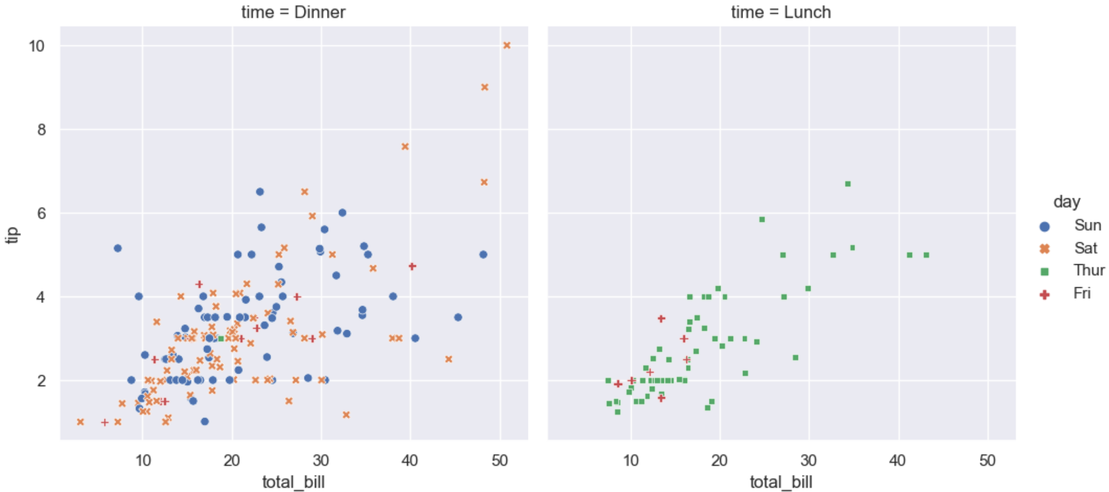
[90]: <AxesSubplot:xlabel='date'>



- Use relplot() to combine scatterplot() and FacetGrid. This allows grouping within additional categorical variables, and plotting them across multiple subplots.
- Using relplot() is safer than using FacetGrid directly, as it ensures synchronization of the semantic mappings across facets.

[91]:
```
sns.relplot(
    data=tips, x="total_bill", y="tip",
    col="time", hue="day", style="day",
    kind="scatter"
)
```

[91]: `<seaborn.axisgrid.FacetGrid at 0x21614e09e40>`



[93]:
```
# https://pandas.pydata.org/docs/user_guide/10min.html    # ***
```