Linee plot

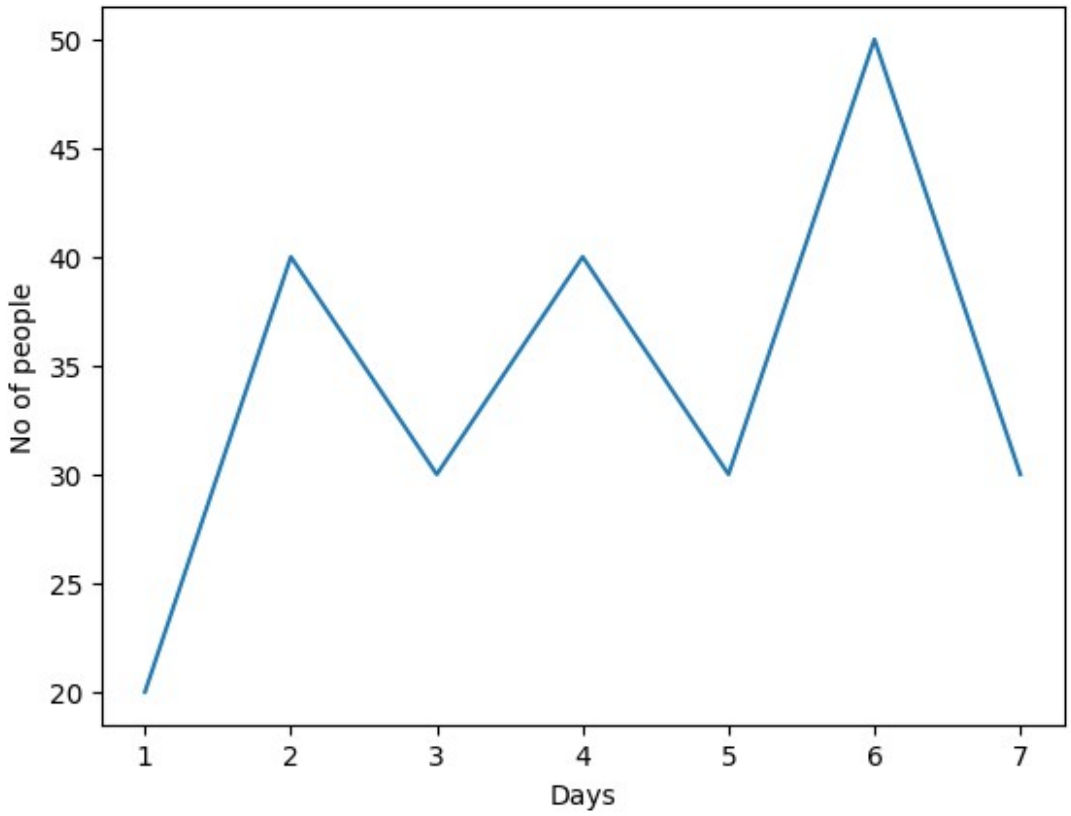
DataFrame is a 2-dimensional labeled data structure from the pandas library that is used to hold data in tabular form. the head() method returns the first few rows of the DataFrame. By default, it shows the first 5 rows, but you can specify a different number if needed.

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

x = [1,2,3,4,5,6,7]
y = [20,40,30,40,30,50,30]

df = pd.DataFrame({"Days":x, "No of people": y})
df.head(4)
```

```
   Days  No of people
0     1            20
1     2            40
2     3            30
3     4            40
```

seaborn itself doesn't have a show() function. You need to use matplotlib.pyplot.show() to display plots created with Seaborn.

```
sns.lineplot(x = "Days", y = "No of people", data = df)
plt.show()
```




```
z = sns.load_dataset("penguins").head(51)
z
```

```
   species     island  bill_length_mm  bill_depth_mm
flipper_length_mm  \
0   Adelie  Torgersen            39.1           18.7
181.0
1   Adelie  Torgersen            39.5           17.4
186.0
2   Adelie  Torgersen            40.3           18.0
195.0
3   Adelie  Torgersen             NaN            NaN
NaN
4   Adelie  Torgersen            36.7           19.3
193.0
5   Adelie  Torgersen            39.3           20.6
190.0
6   Adelie  Torgersen            38.9           17.8
181.0
7   Adelie  Torgersen            39.2           19.6
195.0
8   Adelie  Torgersen            34.1           18.1
193.0
9   Adelie  Torgersen            42.0           20.2
```

```
190.0
10  Adelie  Torgersen            37.8           17.1
186.0
11  Adelie  Torgersen            37.8           17.3
180.0
12  Adelie  Torgersen            41.1           17.6
182.0
13  Adelie  Torgersen            38.6           21.2
191.0
14  Adelie  Torgersen            34.6           21.1
198.0
15  Adelie  Torgersen            36.6           17.8
185.0
16  Adelie  Torgersen            38.7           19.0
195.0
17  Adelie  Torgersen            42.5           20.7
197.0
18  Adelie  Torgersen            34.4           18.4
184.0
19  Adelie  Torgersen            46.0           21.5
194.0
20  Adelie     Biscoe            37.8           18.3
174.0
21  Adelie     Biscoe            37.7           18.7
180.0
22  Adelie     Biscoe            35.9           19.2
189.0
23  Adelie     Biscoe            38.2           18.1
185.0
24  Adelie     Biscoe            38.8           17.2
180.0
25  Adelie     Biscoe            35.3           18.9
187.0
26  Adelie     Biscoe            40.6           18.6
183.0
27  Adelie     Biscoe            40.5           17.9
187.0
28  Adelie     Biscoe            37.9           18.6
172.0
29  Adelie     Biscoe            40.5           18.9
180.0
30  Adelie      Dream            39.5           16.7
178.0
31  Adelie      Dream            37.2           18.1
178.0
32  Adelie      Dream            39.5           17.8
188.0
33  Adelie      Dream            40.9           18.9
184.0
```

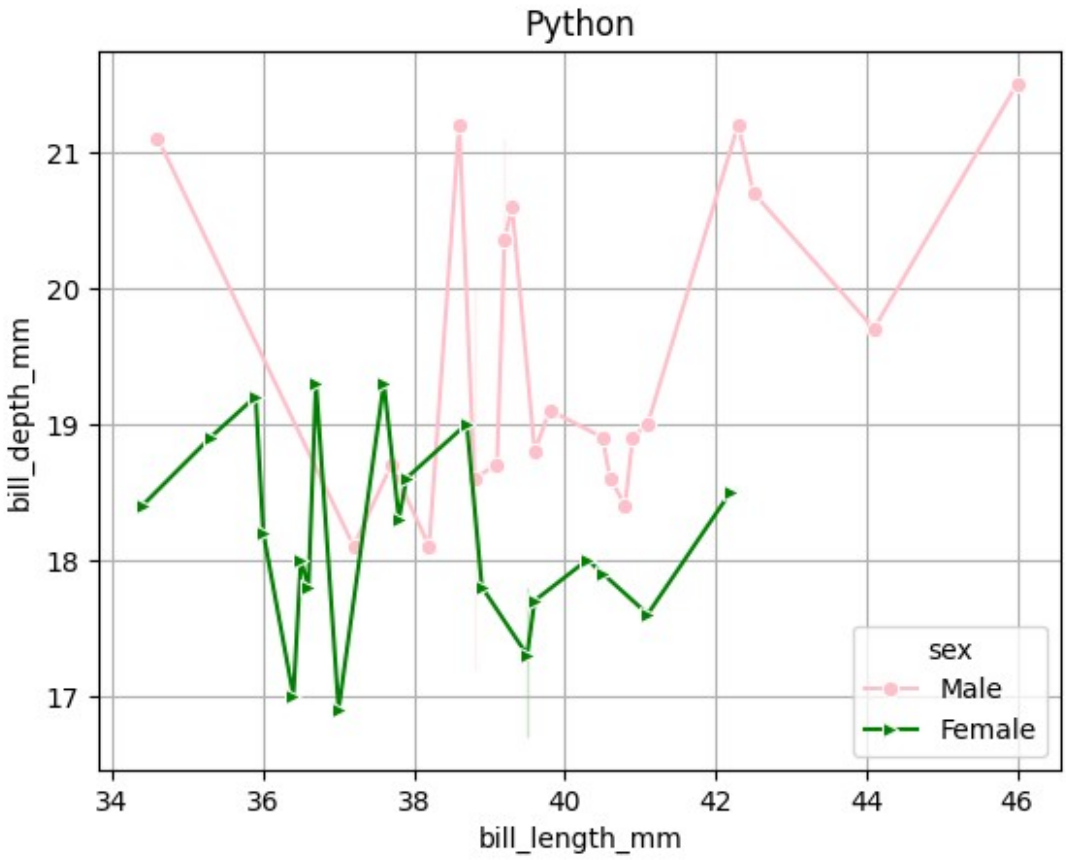
```
34  Adelie      Dream             36.4            17.0
195.0
35  Adelie      Dream             39.2            21.1
196.0
36  Adelie      Dream             38.8            20.0
190.0
37  Adelie      Dream             42.2            18.5
180.0
38  Adelie      Dream             37.6            19.3
181.0
39  Adelie      Dream             39.8            19.1
184.0
40  Adelie      Dream             36.5            18.0
182.0
41  Adelie      Dream             40.8            18.4
195.0
42  Adelie      Dream             36.0            18.5
186.0
43  Adelie      Dream             44.1            19.7
196.0
44  Adelie      Dream             37.0            16.9
185.0
45  Adelie      Dream             39.6            18.8
190.0
46  Adelie      Dream             41.1            19.0
182.0
47  Adelie      Dream             37.5            18.9
179.0
48  Adelie      Dream             36.0            17.9
190.0
49  Adelie      Dream             42.3            21.2
191.0
50  Adelie     Biscoe             39.6            17.7
186.0

    body_mass_g     sex
0        3750.0    Male
1        3800.0  Female
2        3250.0  Female
3           NaN     NaN
4        3450.0  Female
5        3650.0    Male
6        3625.0  Female
7        4675.0    Male
8        3475.0     NaN
9        4250.0     NaN
10       3300.0     NaN
11       3700.0     NaN
12       3200.0  Female
```

```
13        3800.0    Male
14        4400.0    Male
15        3700.0  Female
16        3450.0  Female
17        4500.0    Male
18        3325.0  Female
19        4200.0    Male
20        3400.0  Female
21        3600.0    Male
22        3800.0  Female
23        3950.0    Male
24        3800.0    Male
25        3800.0  Female
26        3550.0    Male
27        3200.0  Female
28        3150.0  Female
29        3950.0    Male
30        3250.0  Female
31        3900.0    Male
32        3300.0  Female
33        3900.0    Male
34        3325.0  Female
35        4150.0    Male
36        3950.0    Male
37        3550.0  Female
38        3300.0  Female
39        4650.0    Male
40        3150.0  Female
41        3900.0    Male
42        3100.0  Female
43        4400.0    Male
44        3000.0  Female
45        4600.0    Male
46        3425.0    Male
47        2975.0     NaN
48        3450.0  Female
49        4150.0    Male
50        3500.0  Female
```

The blue color comes from Seaborn adding a default color because there's likely another category in the sex variable that you didn't account for. To fix this, explicitly define the colors like this: ci=None)

```
sns.lineplot(x = "bill_length_mm", y = "bill_depth_mm", data = z,
hue="sex", style="sex",
             palette=["pink", "green"], markers=["o",">"],
dashes=False, legend=True)
plt.grid()
```

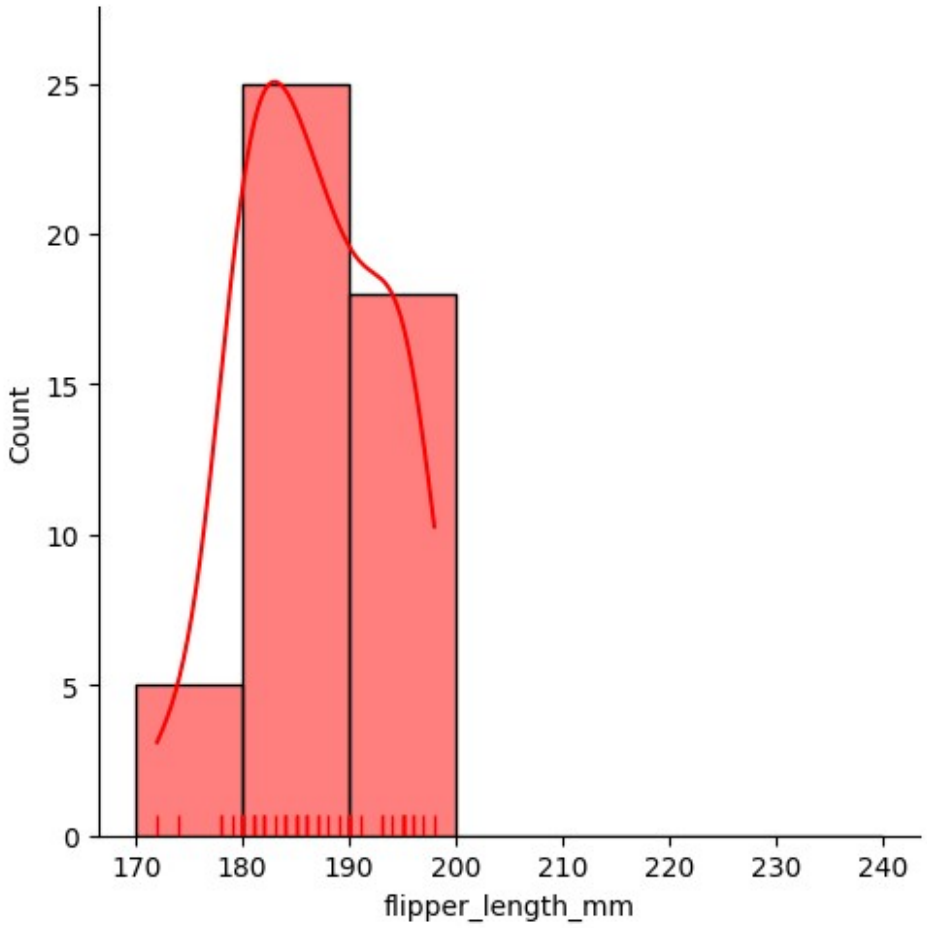
```
plt.title("Python")
plt.show()
```



histogram

You see <seaborn.axisgrid.FacetGrid at 0x...> because the FacetGrid object is being printed directly. To display the plots, use plt.show() after creating and configuring the FacetGrid. ploting address

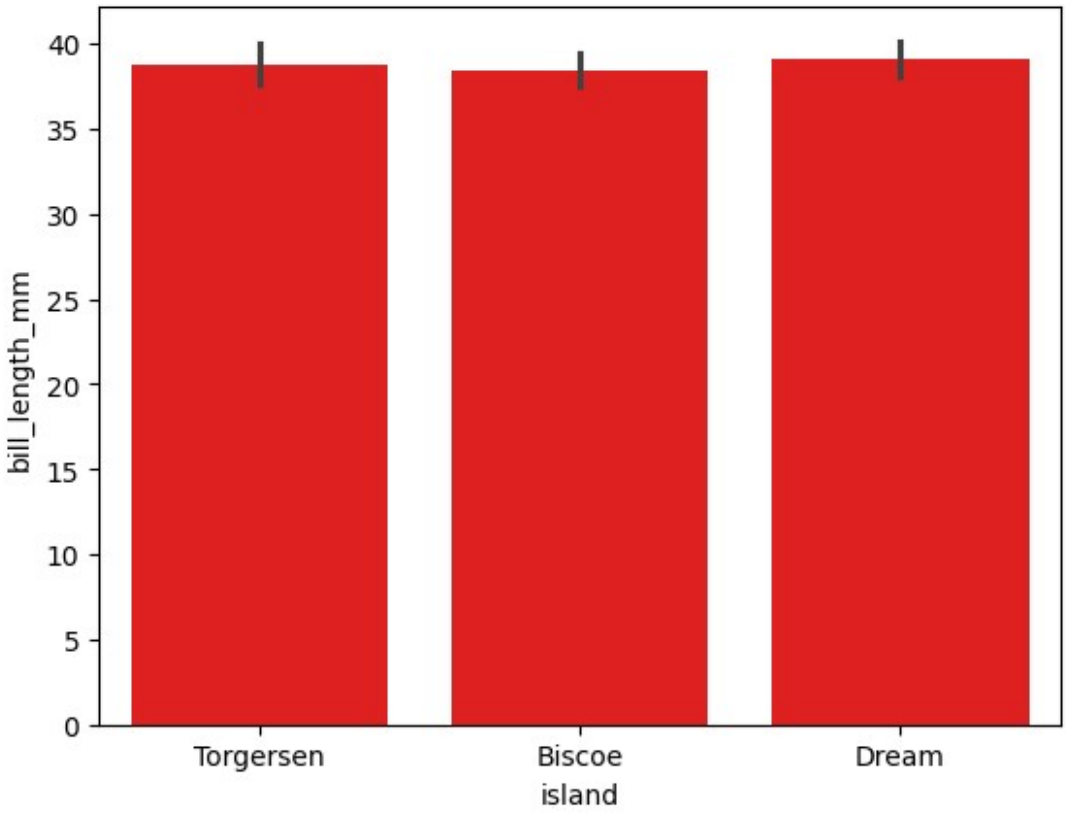
```
sns.displot(z["flipper_length_mm"],bins=[170,180,190,200,210,220,230,240],kde=True, rug=True, color="r")
plt.show()
```



BAR PLOT

Color: Refers to a single color applied to a plot element Palette: Refers to a set of multiple colors used across different elements (e.g., bars or categories).

```
sns.barplot(x="island", y="bill_length_mm", data=z, color="r")
plt.show()
```

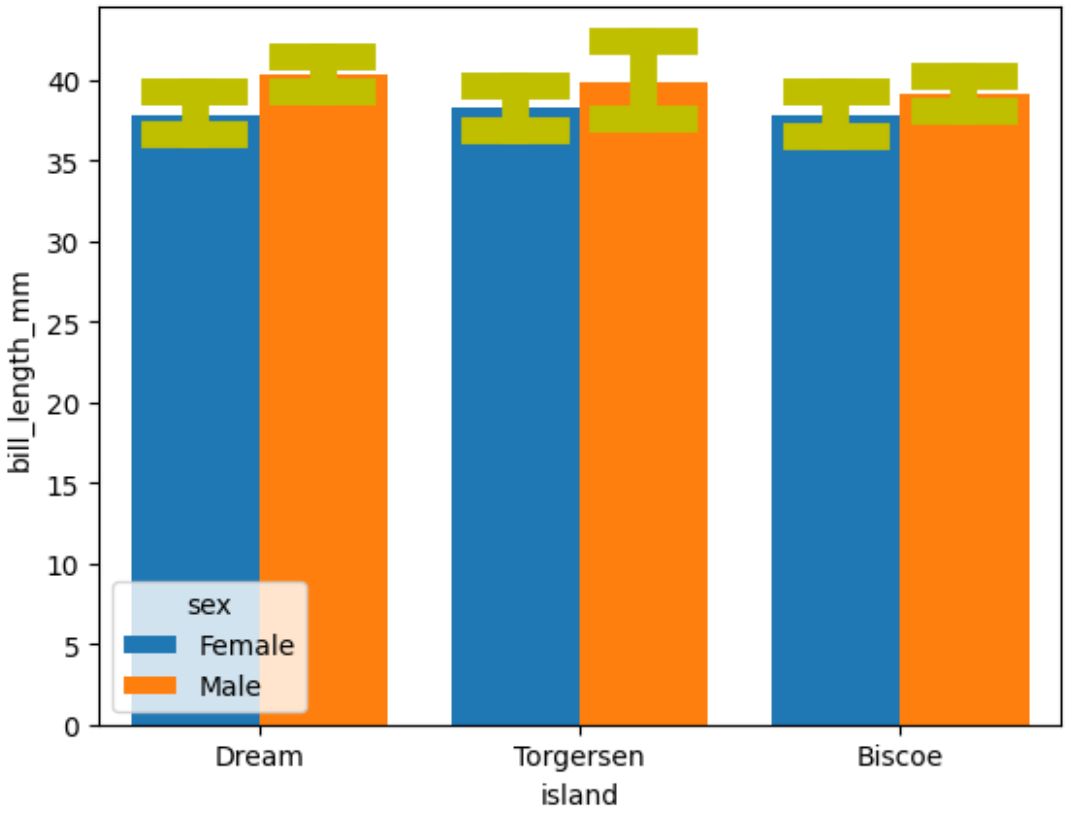


saturation set darkness or light

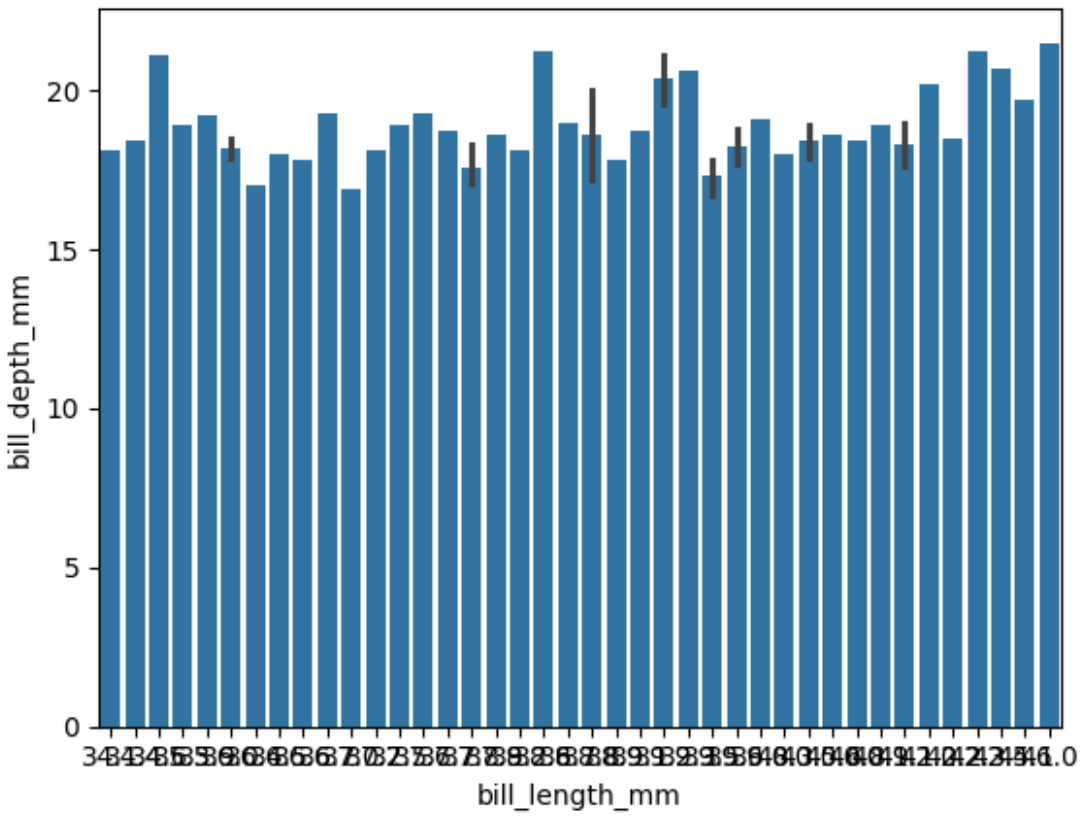
```
order_1 = ["Dream", "Torgersen", "Biscoe"]
sns.barplot(x="island", y="bill_length_mm", data=z, hue="sex",
order=order_1,
            hue_order=["Female","Male"], orient="v", saturation=100,
errcolor="y",
            errwidth=10, capsize=0.5)
plt.show()
```

```
C:\Users\JIya\AppData\Local\Temp\ipykernel_14800\3275064449.py:2:
FutureWarning:

The `errcolor` parameter is deprecated. And will be removed in
v0.15.0. Pass `err_kws={'color': 'y'}` instead.

  sns.barplot(x="island", y="bill_length_mm", data=z, hue="sex",
order=order_1,
C:\Users\JIya\AppData\Local\Temp\ipykernel_14800\3275064449.py:2:
FutureWarning:

The `errwidth` parameter is deprecated. And will be removed in
v0.15.0. Pass `err_kws={'linewidth': 10}` instead.
```

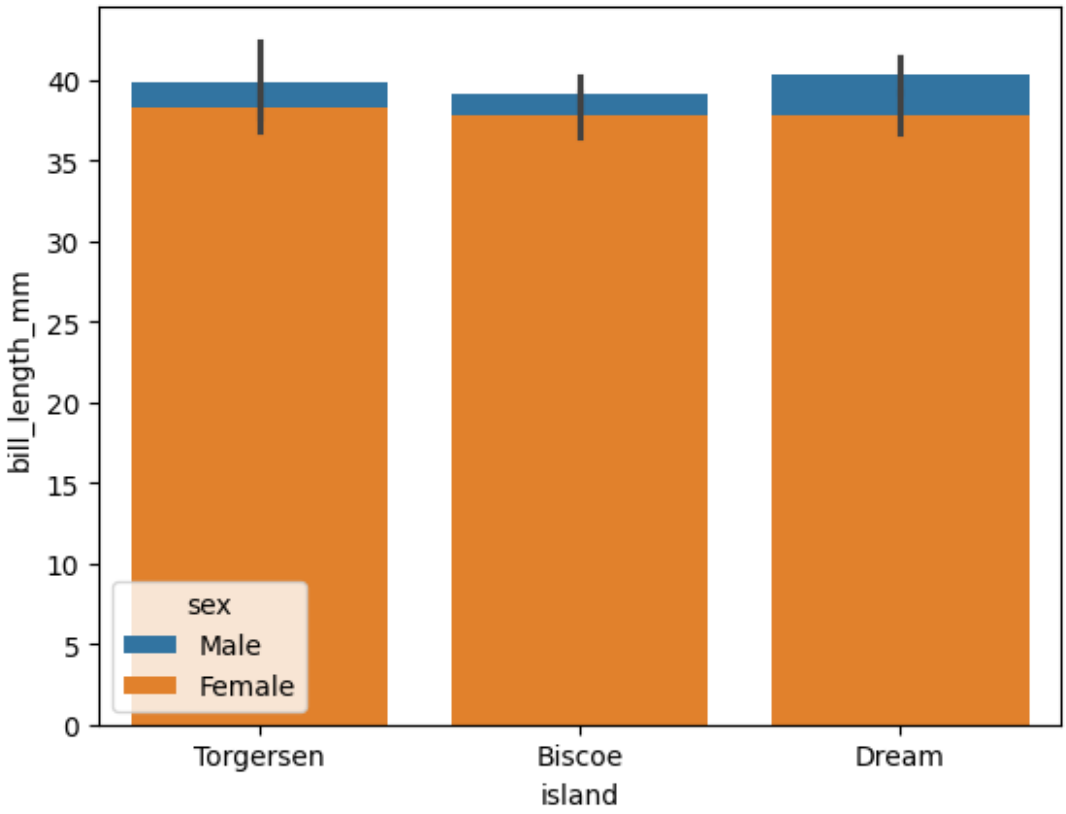
```
  sns.barplot(x="island", y="bill_length_mm", data=z, hue="sex",
order=order_1,
```



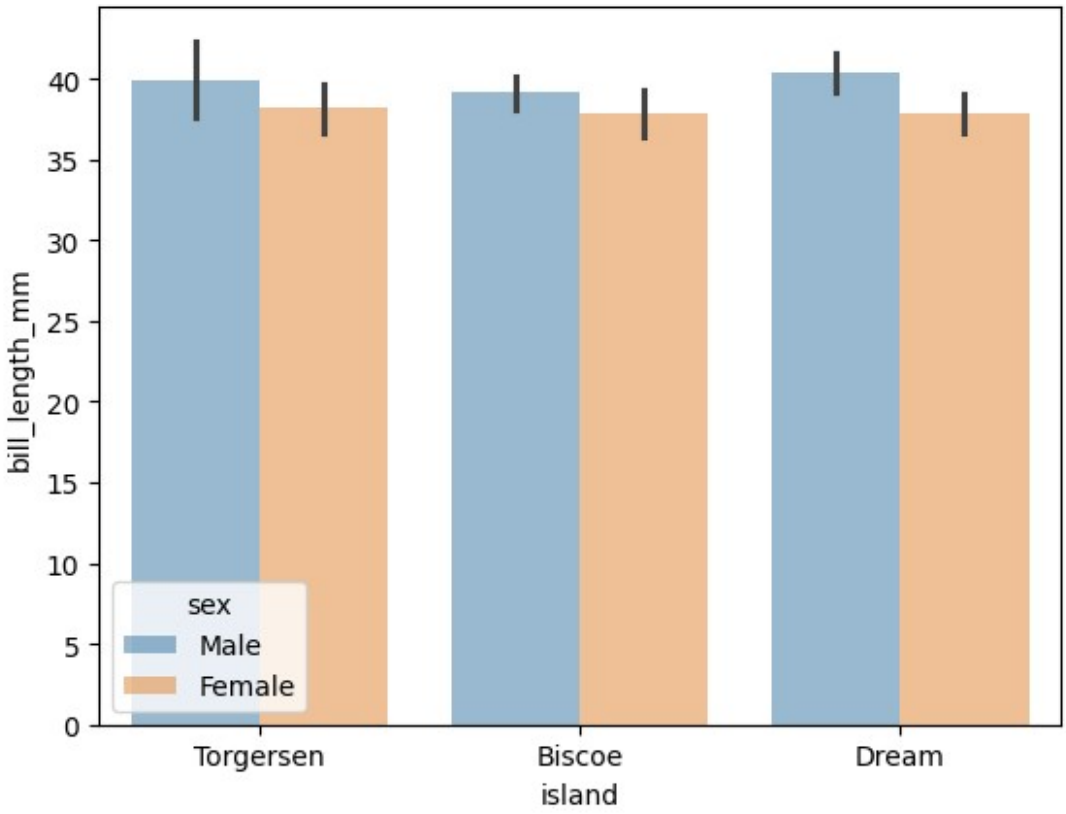

```
order_1 = ["Dream", "Torgersen", "Biscoe"]
sns.barplot(x="bill_length_mm", y="bill_depth_mm", data=z, orient="v")
plt.show()
```



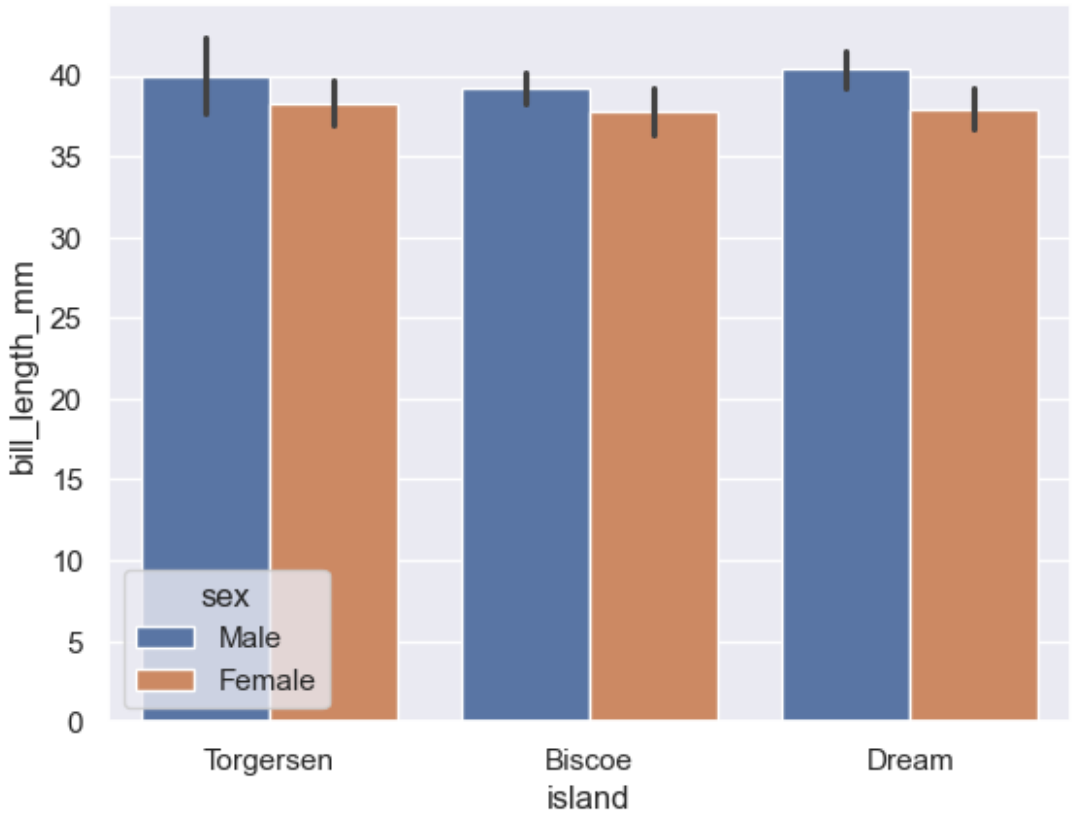

```
order_1 = ["Dream", "Torgersen", "Biscoe"]
sns.barplot(x="island", y="bill_length_mm", data=z, hue="sex",
dodge=False)
plt.show()
```




```
order_1 = ["Dream", "Torgersen", "Biscoe"]
sns.barplot(x="island", y="bill_length_mm", data=z, hue="sex",
alpha=0.5)
plt.show()
```




```
sns.set(style="darkgrid")
order_1 = ["Dream", "Torgersen", "Biscoe"]
sns.barplot(x="island", y="bill_length_mm", data=z, hue="sex")
plt.show()
```



Scatter plot

```
j = sns.load_dataset("penguins").head(20)
j
```

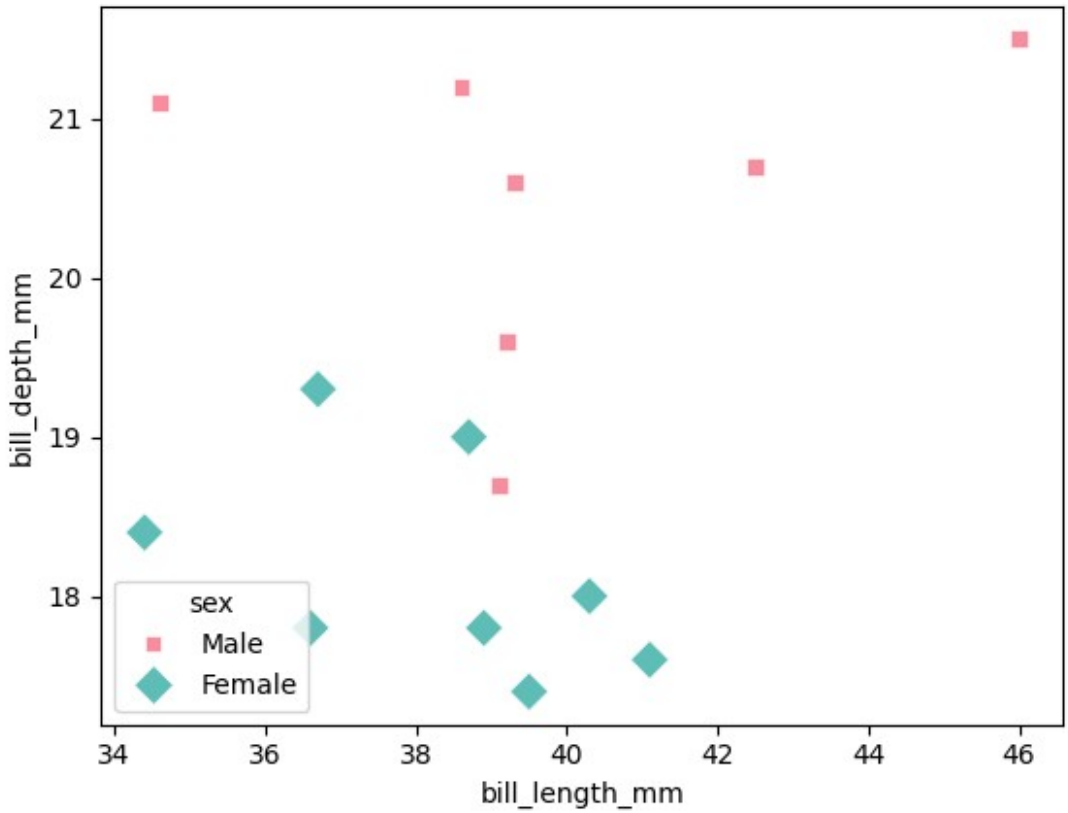
```
   species     island  bill_length_mm  bill_depth_mm
flipper_length_mm  \
0   Adelie  Torgersen            39.1           18.7
181.0
1   Adelie  Torgersen            39.5           17.4
186.0
2   Adelie  Torgersen            40.3           18.0
195.0
3   Adelie  Torgersen             NaN            NaN
NaN
4   Adelie  Torgersen            36.7           19.3
193.0
5   Adelie  Torgersen            39.3           20.6
190.0
6   Adelie  Torgersen            38.9           17.8
181.0
7   Adelie  Torgersen            39.2           19.6
195.0
8   Adelie  Torgersen            34.1           18.1
```

```
193.0
9   Adelie  Torgersen            42.0           20.2
190.0
10  Adelie  Torgersen            37.8           17.1
186.0
11  Adelie  Torgersen            37.8           17.3
180.0
12  Adelie  Torgersen            41.1           17.6
182.0
13  Adelie  Torgersen            38.6           21.2
191.0
14  Adelie  Torgersen            34.6           21.1
198.0
15  Adelie  Torgersen            36.6           17.8
185.0
16  Adelie  Torgersen            38.7           19.0
195.0
17  Adelie  Torgersen            42.5           20.7
197.0
18  Adelie  Torgersen            34.4           18.4
184.0
19  Adelie  Torgersen            46.0           21.5
194.0
```

```
    body_mass_g     sex
0        3750.0    Male
1        3800.0  Female
2        3250.0  Female
3           NaN     NaN
4        3450.0  Female
5        3650.0    Male
6        3625.0  Female
7        4675.0    Male
8        3475.0     NaN
9        4250.0     NaN
10       3300.0     NaN
11       3700.0     NaN
12       3200.0  Female
13       3800.0    Male
14       4400.0    Male
15       3700.0  Female
16       3450.0  Female
17       4500.0    Male
18       3325.0  Female
19       4200.0    Male
```

```
m={"Male":"s", "Female":"D"}
sns.scatterplot(x="bill_length_mm", y="bill_depth_mm", data=j,
hue="sex",
                style="sex", size="sex", sizes=(100, 40),
```

```
palette="husl",
                alpha=0.8, markers=m)
plt.show()
```



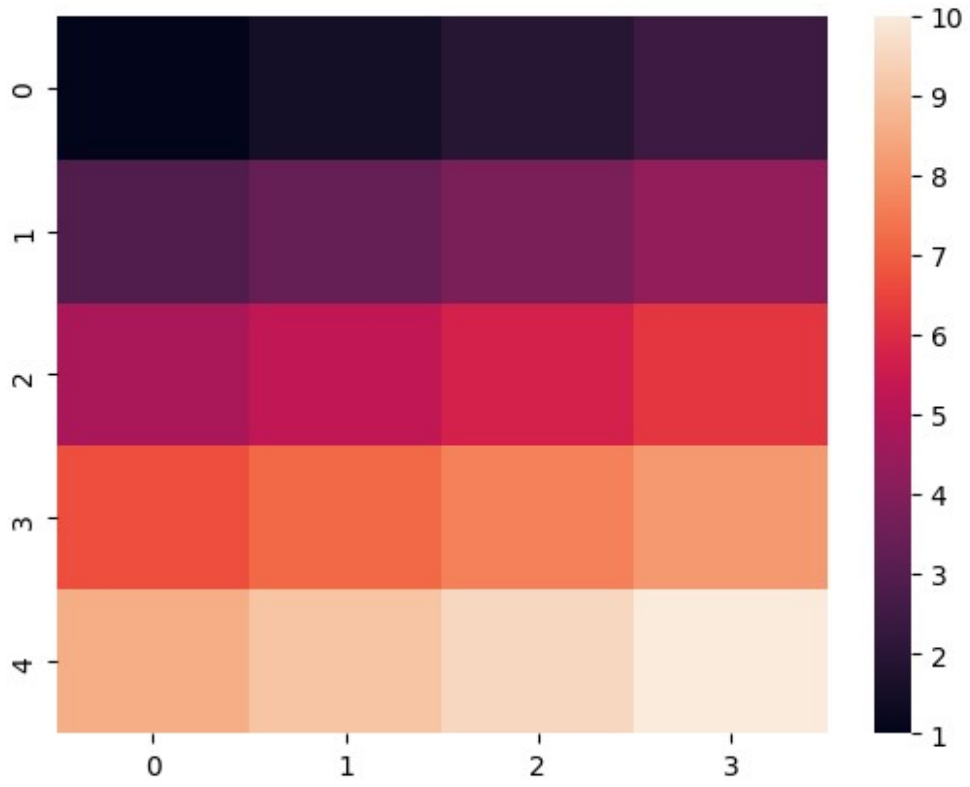
Heatmap

```
import numpy as np

var = np.linspace(1,10,20).reshape(5,4)
var
```

```
array([[ 1.        ,  1.47368421,  1.94736842,  2.42105263],
       [ 2.89473684,  3.36842105,  3.84210526,  4.31578947],
       [ 4.78947368,  5.26315789,  5.73684211,  6.21052632],
       [ 6.68421053,  7.15789474,  7.63157895,  8.10526316],
       [ 8.57894737,  9.05263158,  9.52631579, 10.        ]])
```

```
sns.heatmap(var)
var
```

```
array([[ 1.        ,  1.47368421,  1.94736842,  2.42105263],
       [ 2.89473684,  3.36842105,  3.84210526,  4.31578947],
       [ 4.78947368,  5.26315789,  5.73684211,  6.21052632],
       [ 6.68421053,  7.15789474,  7.63157895,  8.10526316],
       [ 8.57894737,  9.05263158,  9.52631579, 10.        ]])
```

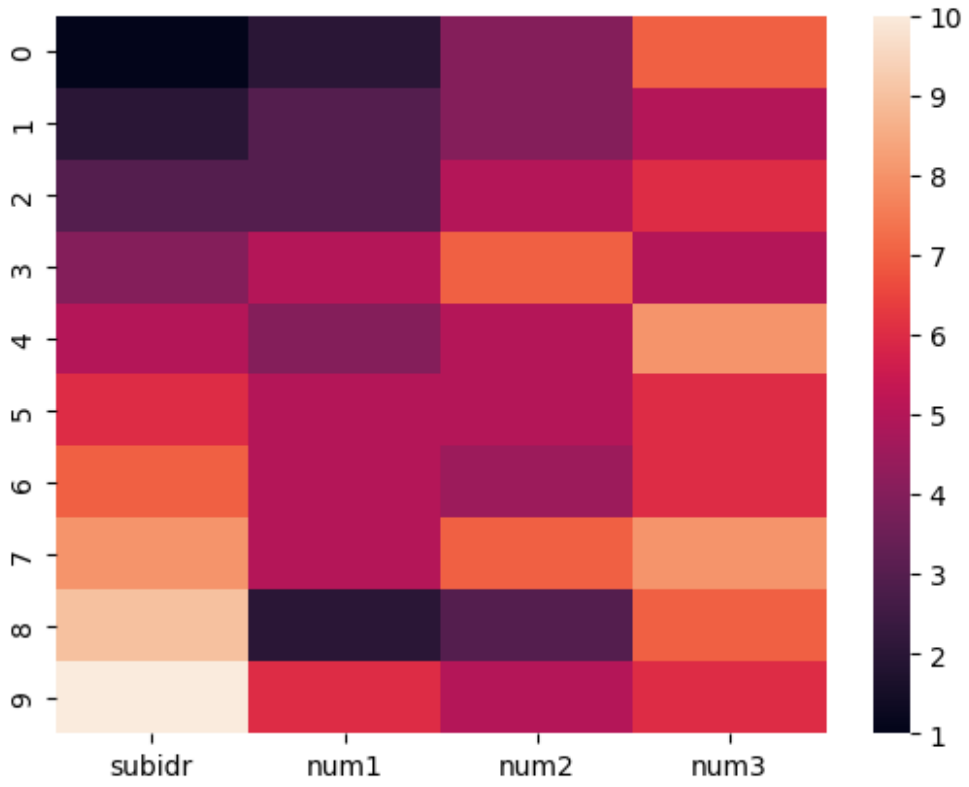



```
data = sns.load_dataset("anagrams").head(10)
x = data.drop(columns=["attnr"],axis=1)
x
```

	subidr	num1	num2	num3
0	1	2	4.0	7
1	2	3	4.0	5
2	3	3	5.0	6
3	4	5	7.0	5
4	5	4	5.0	8
5	6	5	5.0	6
6	7	5	4.5	6
7	8	5	7.0	8
8	9	2	3.0	7
9	10	6	5.0	6

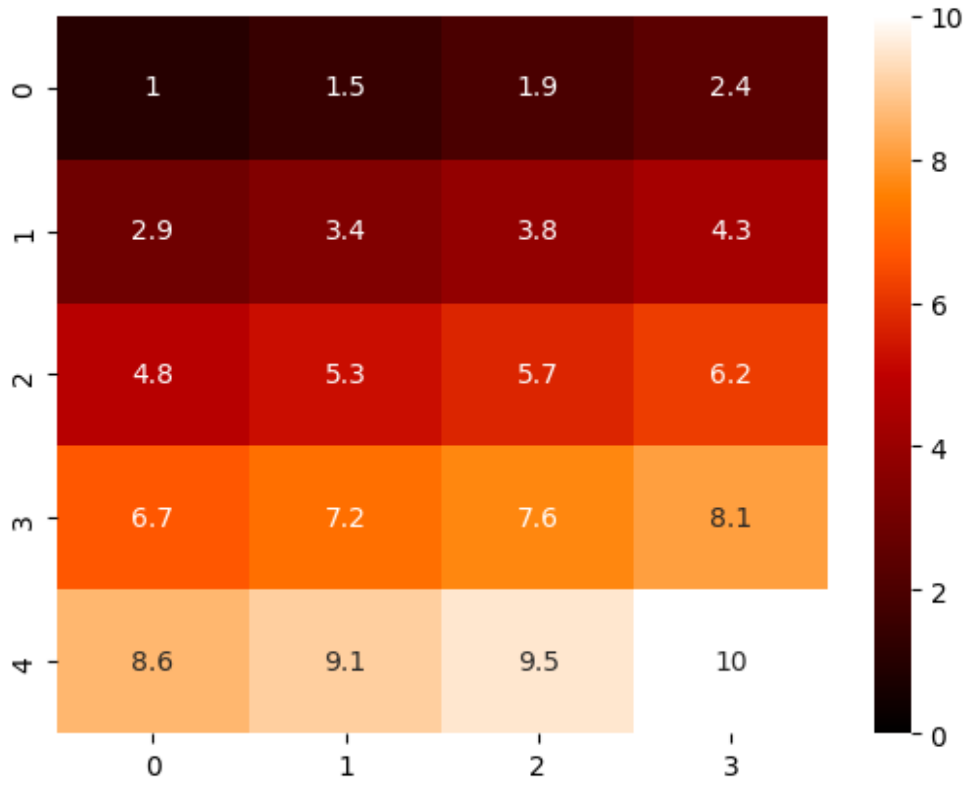
```
sns.heatmap(x)
```

```
<Axes: >
```




```
sns.heatmap(var, vmin=0, vmax=10, cmap='gist_heat', annot=True)
var
```

```
array([[ 1.        ,  1.47368421,  1.94736842,  2.42105263],
       [ 2.89473684,  3.36842105,  3.84210526,  4.31578947],
       [ 4.78947368,  5.26315789,  5.73684211,  6.21052632],
       [ 6.68421053,  7.15789474,  7.63157895,  8.10526316],
       [ 8.57894737,  9.05263158,  9.52631579, 10.        ]])
```




```
var1 = np.linspace(1,10,10).reshape(2,5)
var1
```

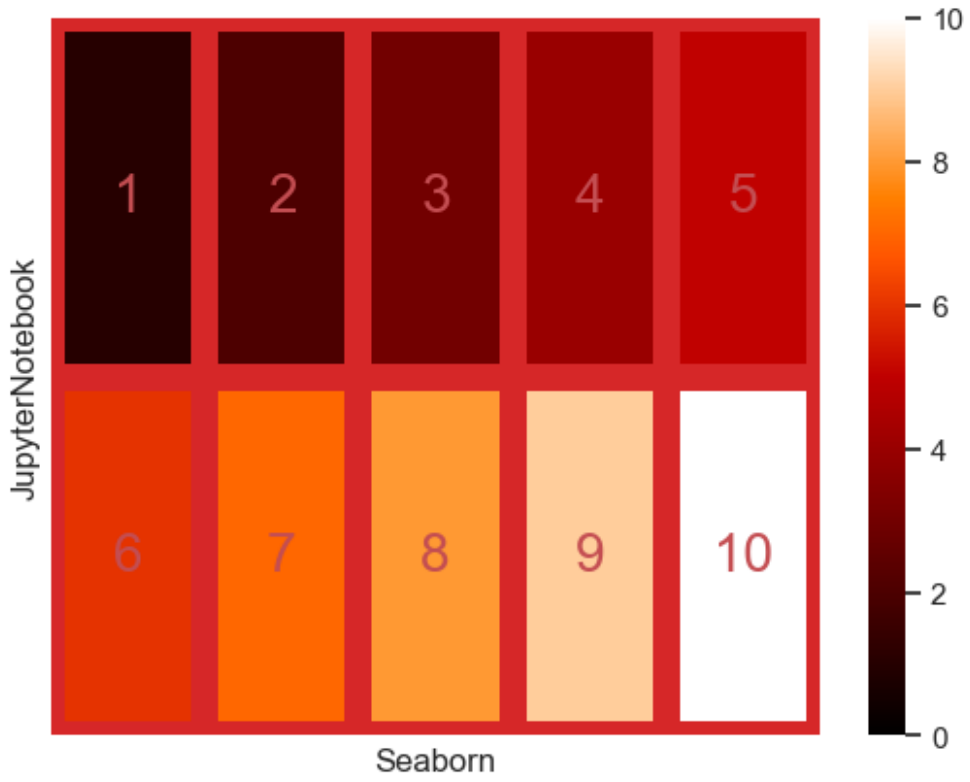
```
array([[ 1.,  2.,  3.,  4.,  5.],
       [ 6.,  7.,  8.,  9., 10.]])
```

In Seaborn, annot is a parameter used in the heatmap function to add numerical annotations to each cell in the heatmap.

annot_kws is a parameter in Seaborn's heatmap function that allows you to customize the appearance of the annotations

cbar=True to display the color bar.

```
y={"fontsize":20, "color":"r"}
v = sns.heatmap(var1, vmin=0, vmax=10, cmap='gist_heat', annot=True,
annot_kws=y,
            linewidth=10, linecolor="#d62728", cbar=True,
xticklabels=False, yticklabels=False)

v.set(xlabel="Seaborn", ylabel="JupyterNotebook")
sns.set(font_scale=0.4)
plt.show()
```

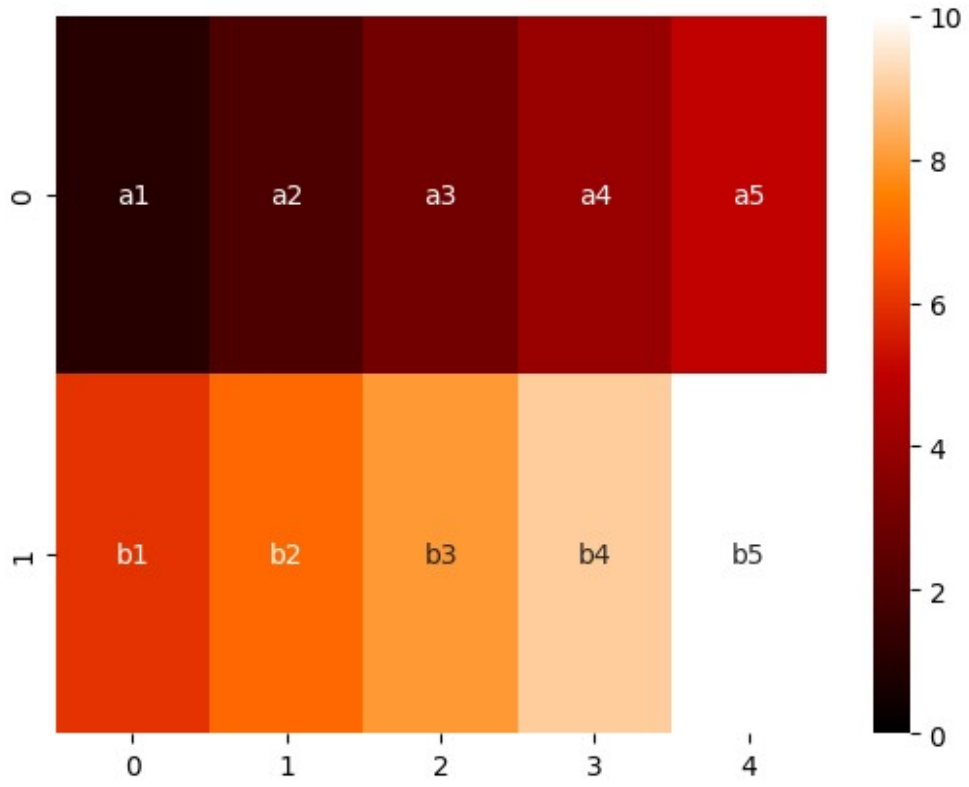



```
arr = np.array([["a1", "a2", "a3", "a4", "a5"],
                ["b1", "b2", "b3", "b4", "b5"]])
arr
```

```
array([['a1', 'a2', 'a3', 'a4', 'a5'],
       ['b1', 'b2', 'b3', 'b4', 'b5']], dtype='<U2')
```

```
sns.heatmap(var1, vmin=0, vmax=10, cmap='gist_heat', annot=arr,
            fmt="s")
```

```
<Axes: >
```

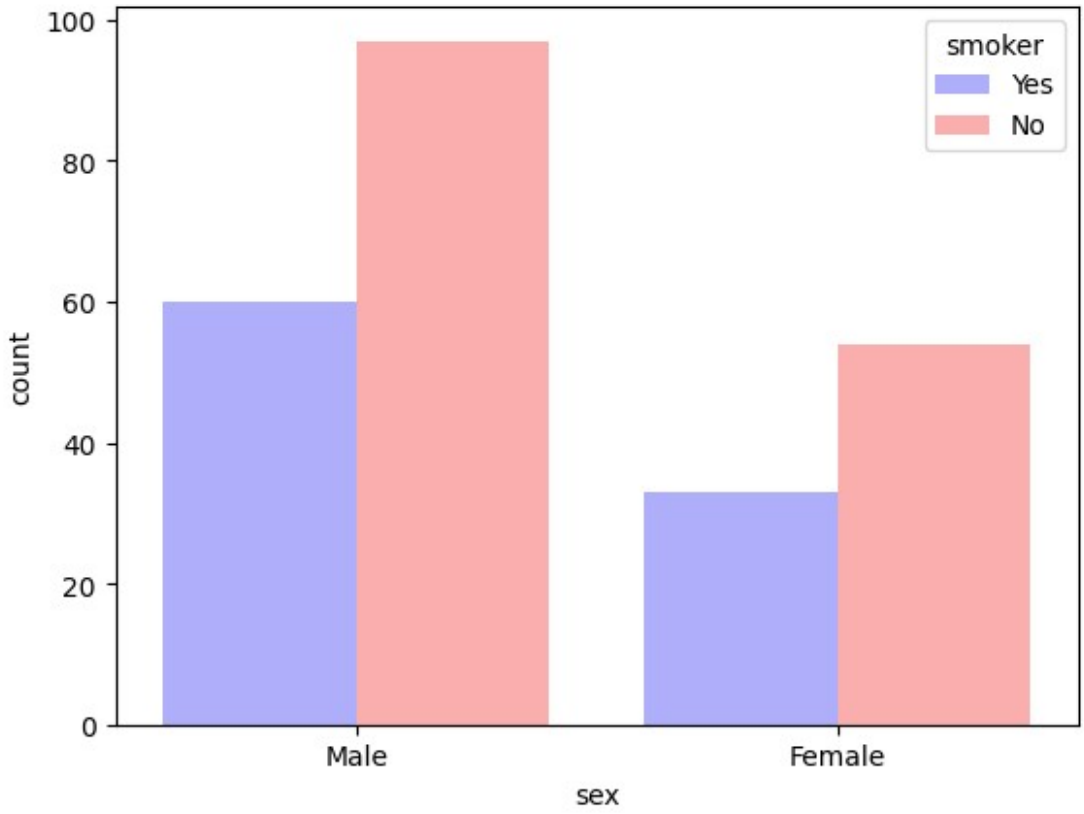


Count Plot

```
var1 = sns.load_dataset("tips")
var1
```

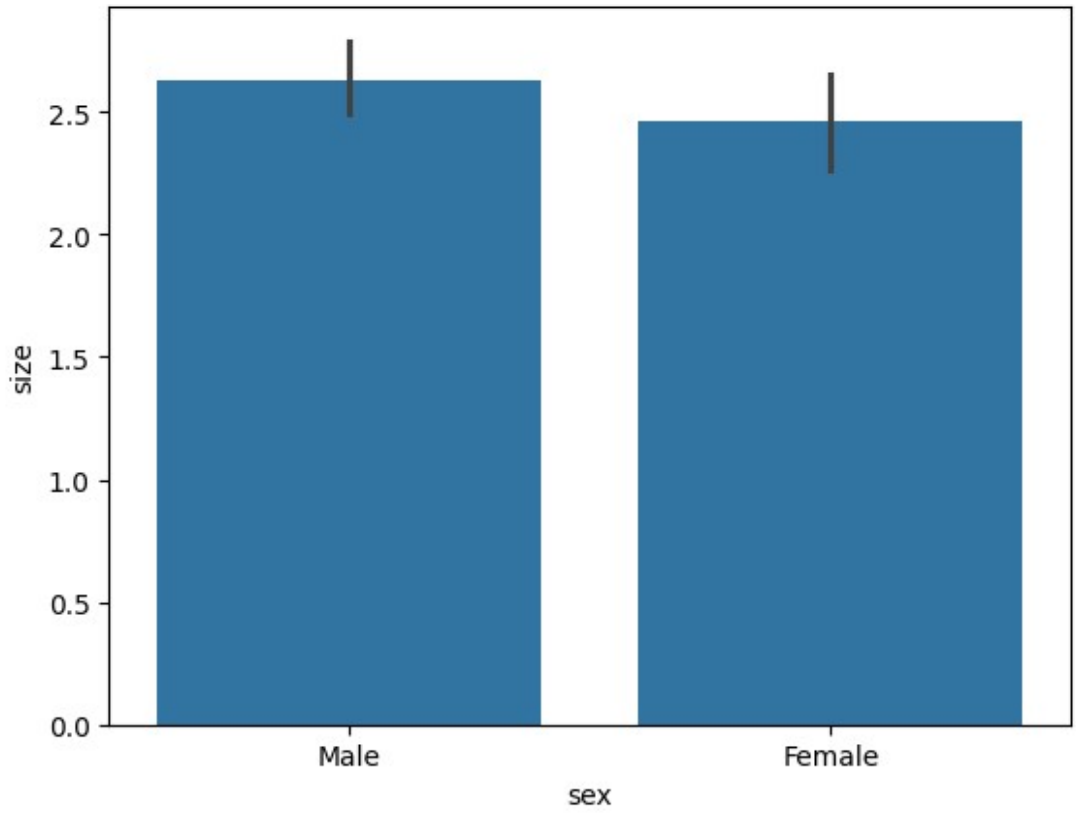
```
     total_bill   tip     sex smoker   day    time  size
0          16.99  1.01  Female     No   Sun  Dinner     2
1          10.34  1.66    Male     No   Sun  Dinner     3
2          21.01  3.50    Male     No   Sun  Dinner     3
3          23.68  3.31    Male     No   Sun  Dinner     2
4          24.59  3.61  Female     No   Sun  Dinner     4
..           ...   ...     ...    ...   ...     ...   ...
239        29.03  5.92    Male     No   Sat  Dinner     3
240        27.18  2.00  Female    Yes   Sat  Dinner     2
241        22.67  2.00    Male    Yes   Sat  Dinner     2
242        17.82  1.75    Male     No   Sat  Dinner     2
243        18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```

```
sns.countplot(x="sex", data=var1, hue="smoker",palette="bwr",
saturation=0.9)
plt.show()
```



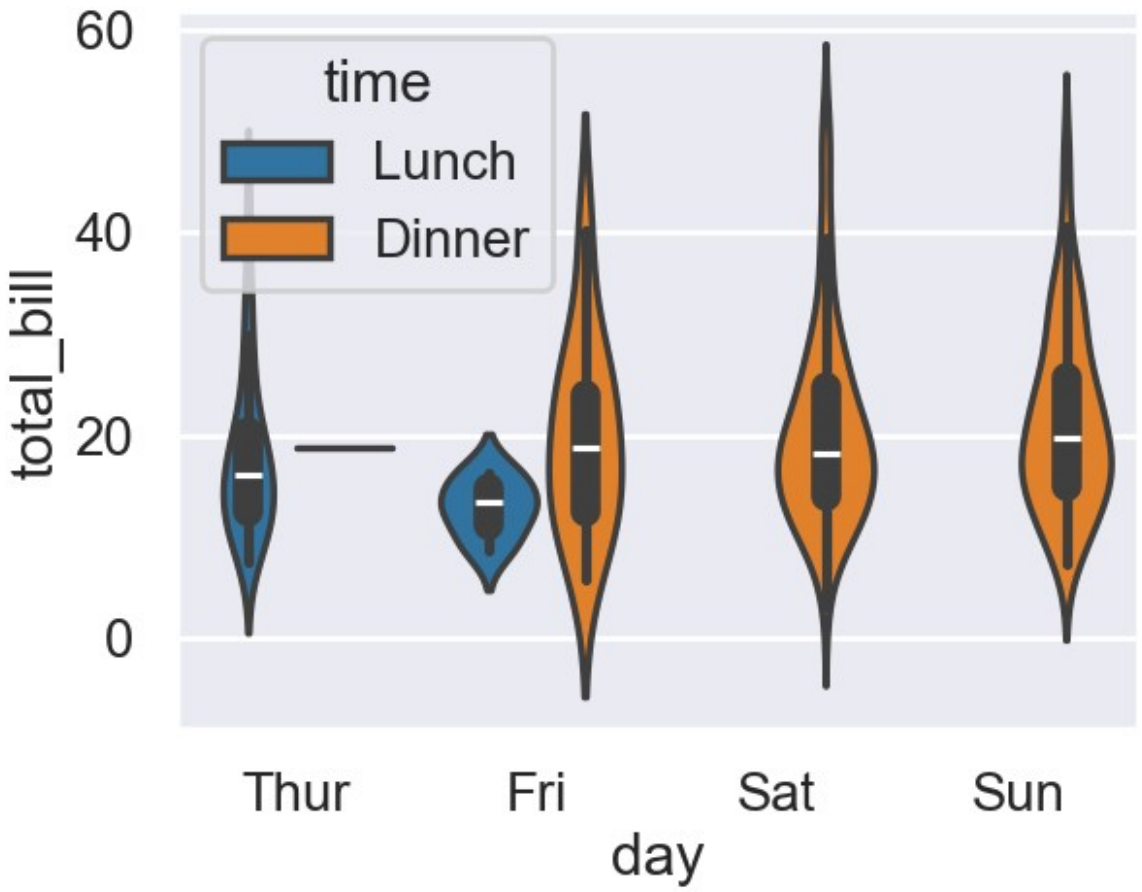
Bar plot (sns.barplot()): You apply size (or any numeric variable) because it shows the average value of that numeric variable for each category (e.g., average size for males and females).

```
sns.barplot(x="sex", y ="size", data=var1)
plt.show()
```



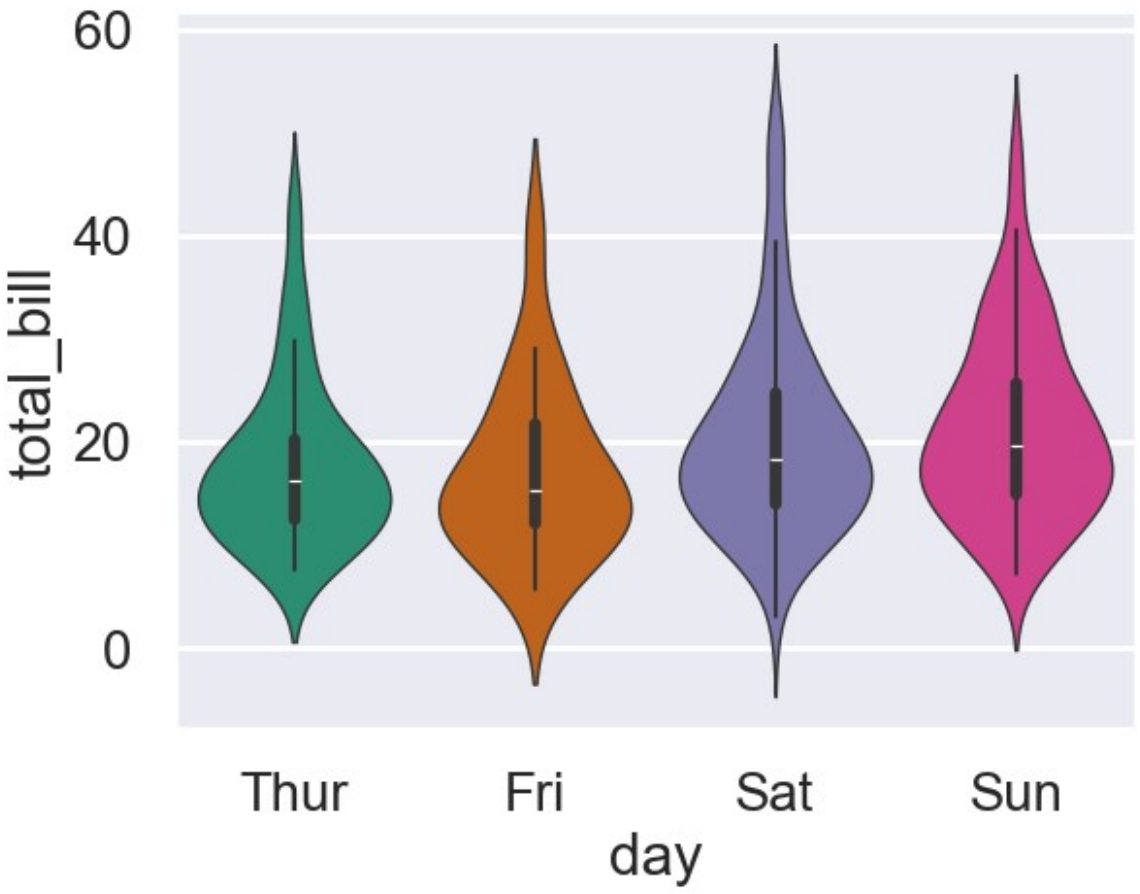
Violin plot

```
sns.violinplot(x="day", y="total_bill", data=var1, hue="time")
plt.show()
```

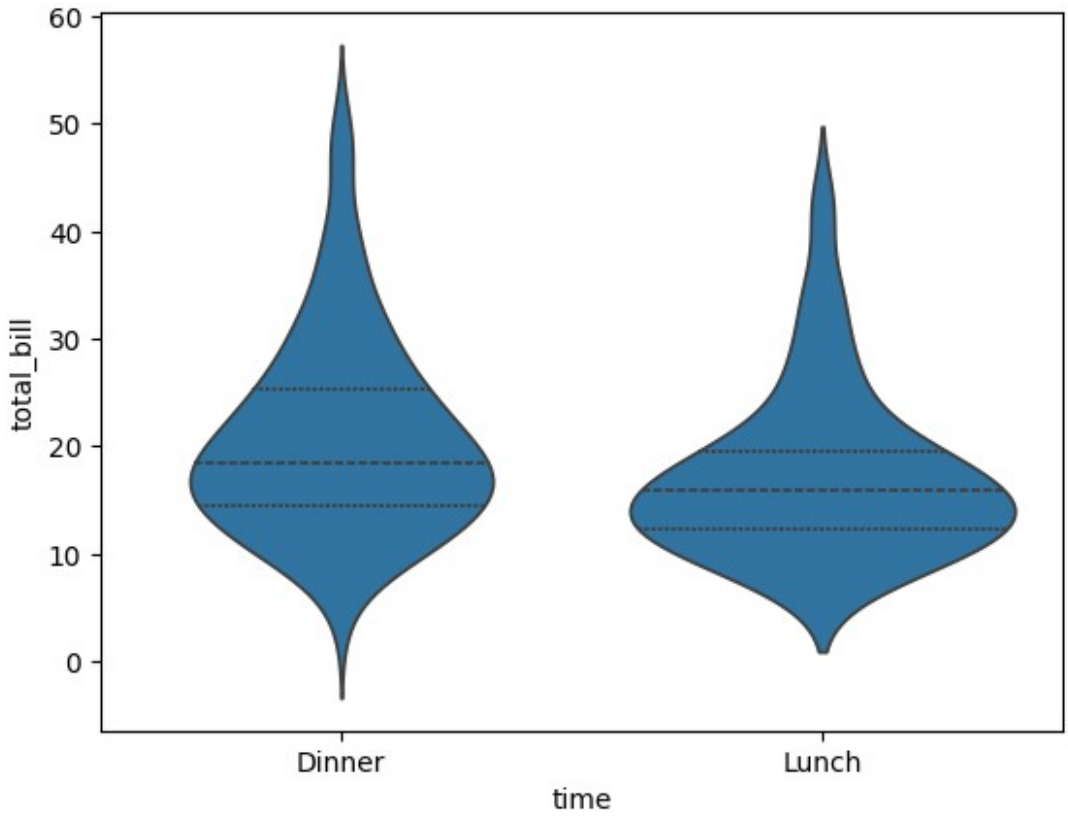



```
sns.violinplot(x="day", y="total_bill", data=var1, linewidth=1,
palette='Dark2')
plt.show()
```

```
C:\Users\JIya\AppData\Local\Temp\ipykernel_9404\204958709.py:1:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.violinplot(x="day", y="total_bill", data=var1, linewidth=1,
palette='Dark2')
```

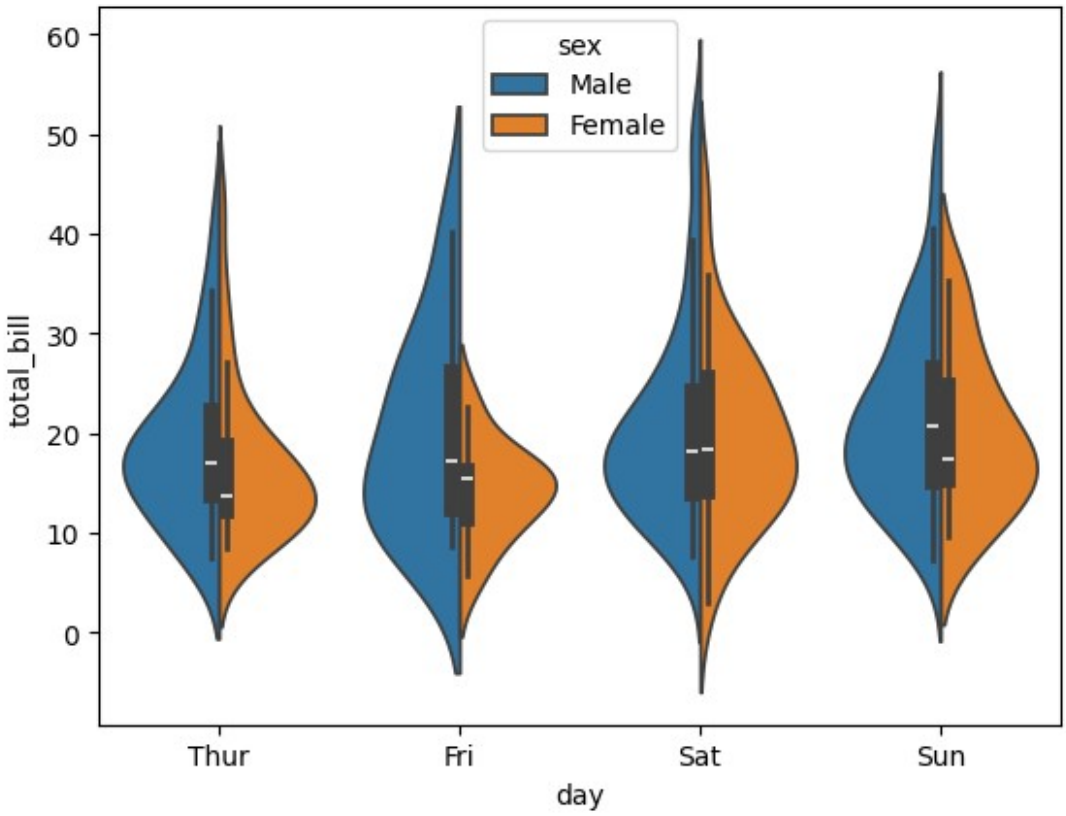



```
sns.violinplot(x="time", y="total_bill", data=var1, order=["Dinner",
"Lunch"],
                inner="quart") # ("quartile", "point", "stick", None)
plt.show()
```

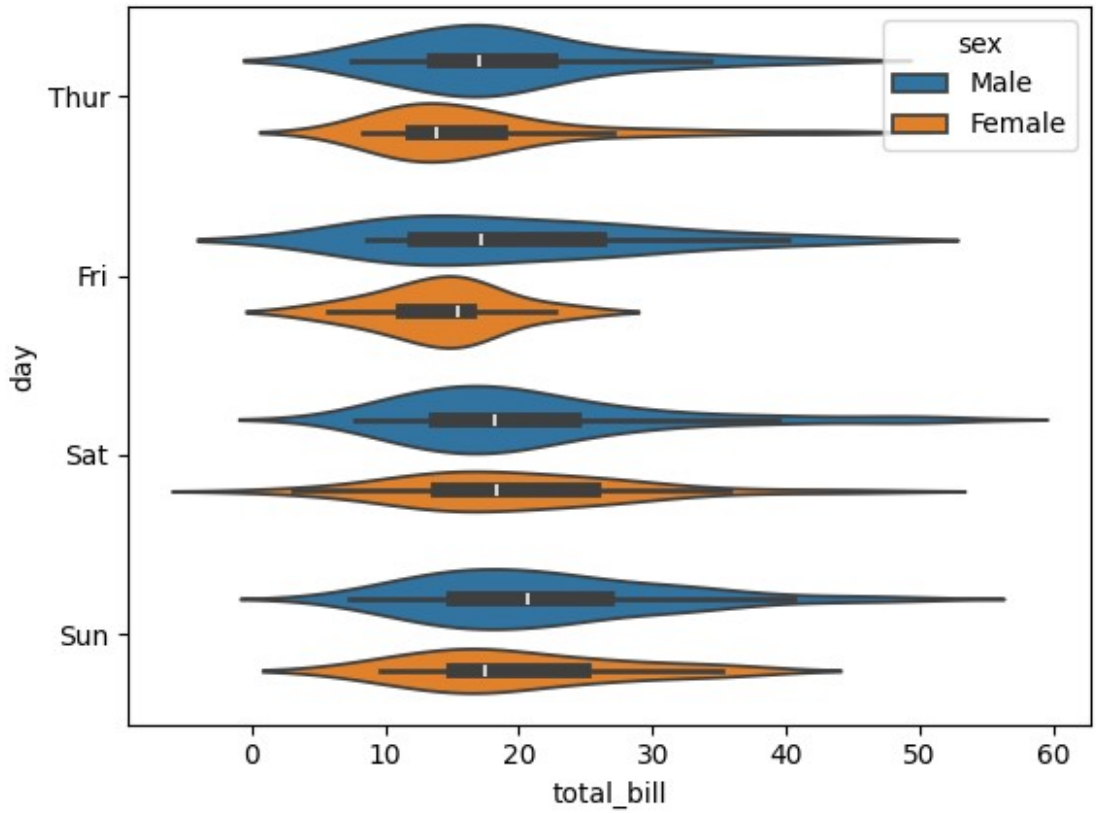



```
sns.violinplot(x="day", y="total_bill", data=var1, hue="sex",
split=True, scale="width")
plt.show()
```

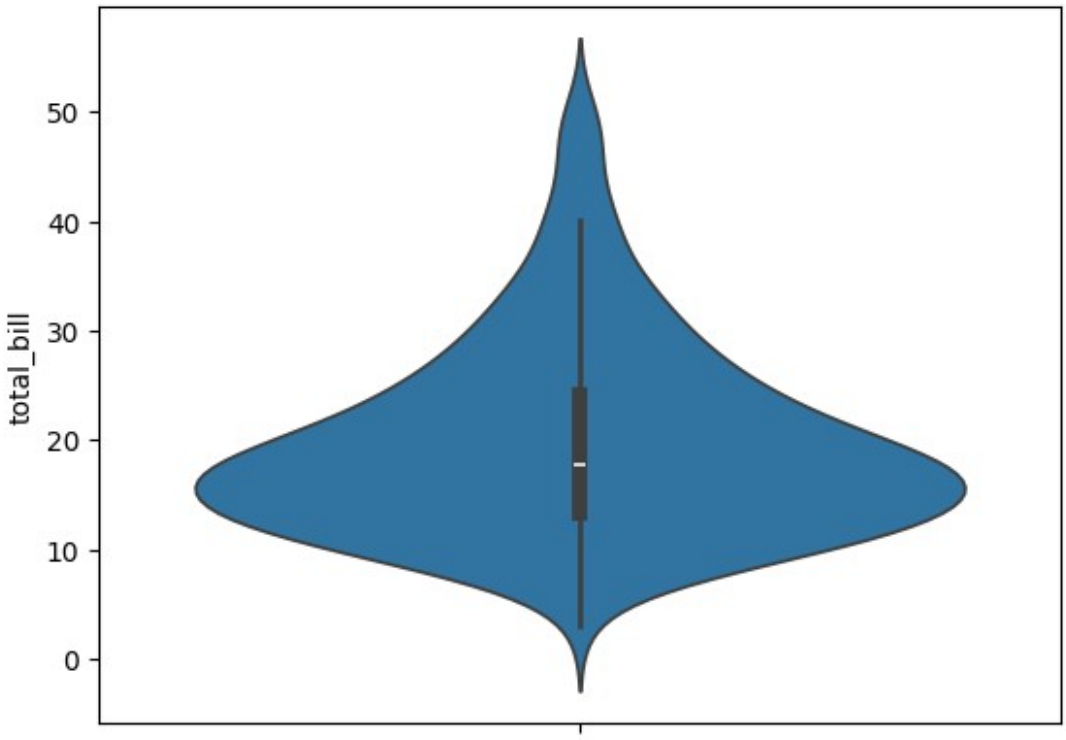
```
C:\Users\JIya\AppData\Local\Temp\ipykernel_13164\3343507532.py:1:
FutureWarning:

The `scale` parameter has been renamed and will be removed in v0.15.0.
Pass `density_norm='width'` for the same effect.
  sns.violinplot(x="day", y="total_bill", data=var1, hue="sex",
split=True, scale="width")
```




```
sns.violinplot(x="total_bill", y="day", data=var1, hue="sex")
plt.show()
```




```
sns.violinplot(y=var1["total_bill"])
plt.show()
```



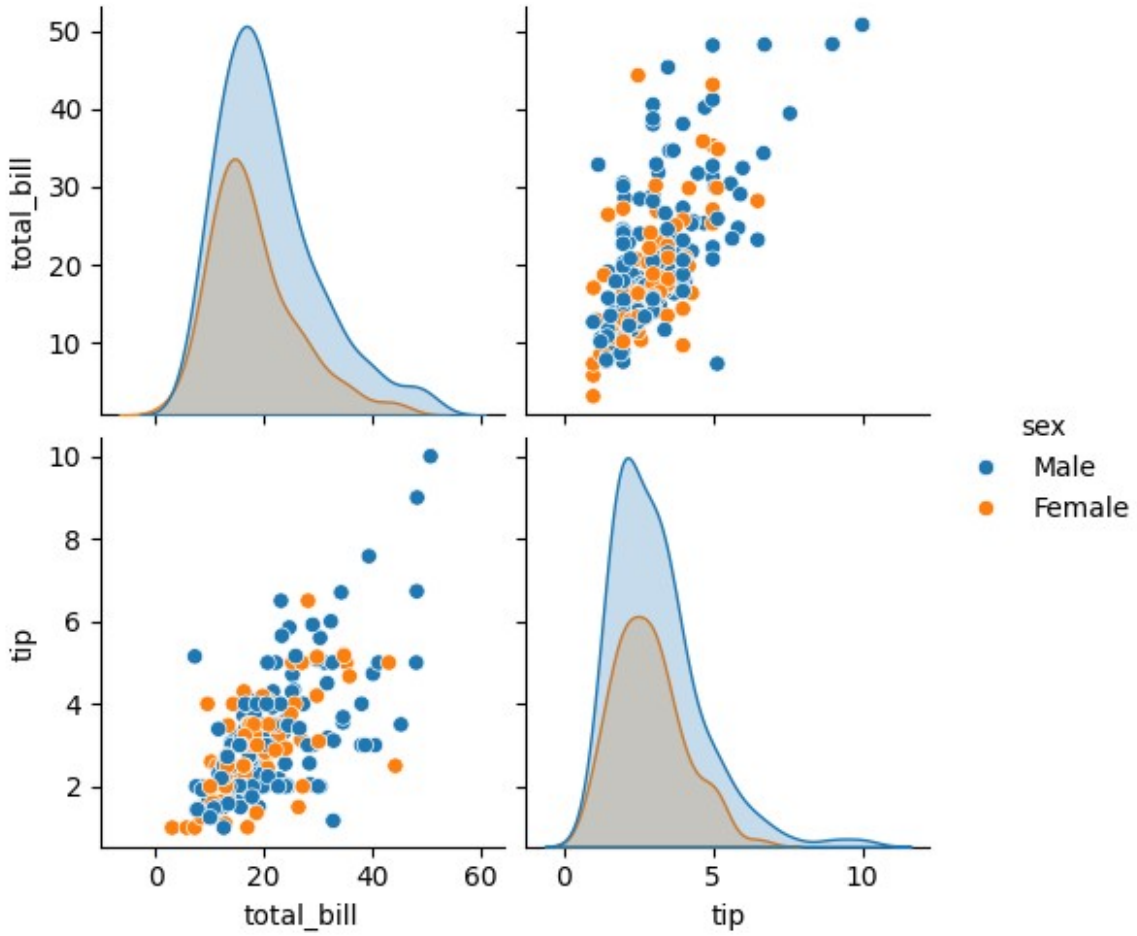
pair plot

```
var2 = sns.load_dataset("tips")
var2
```

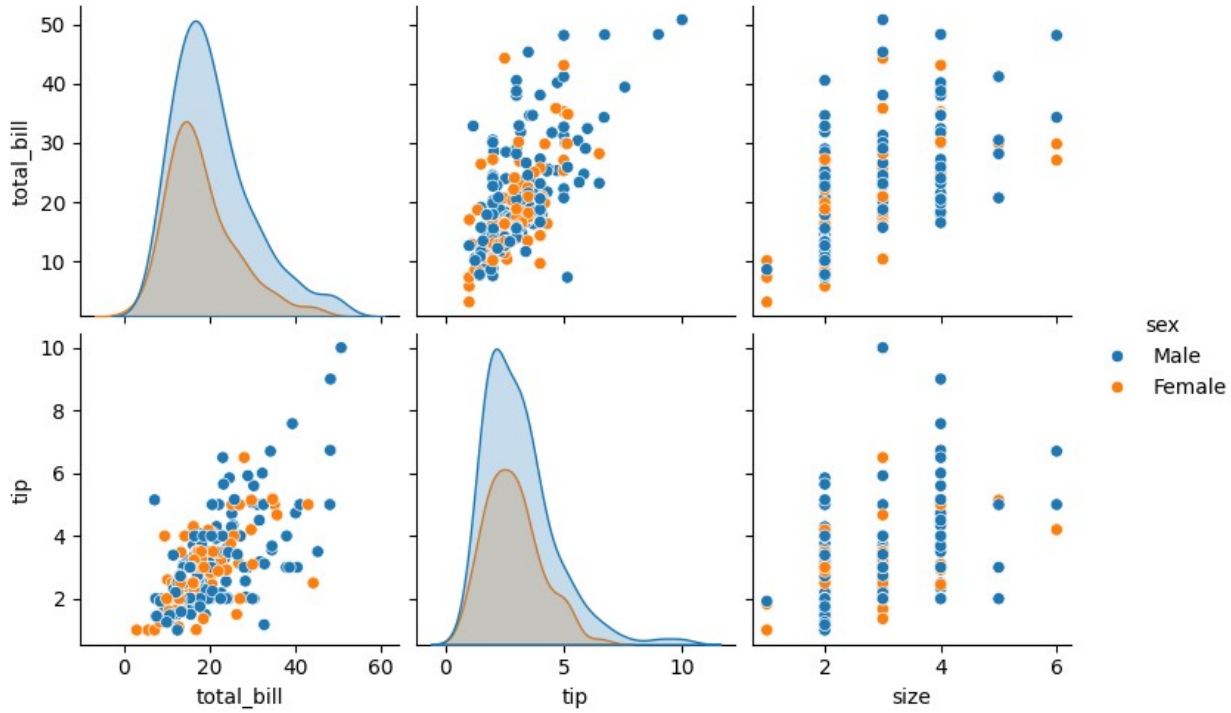
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
..	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

[244 rows x 7 columns]

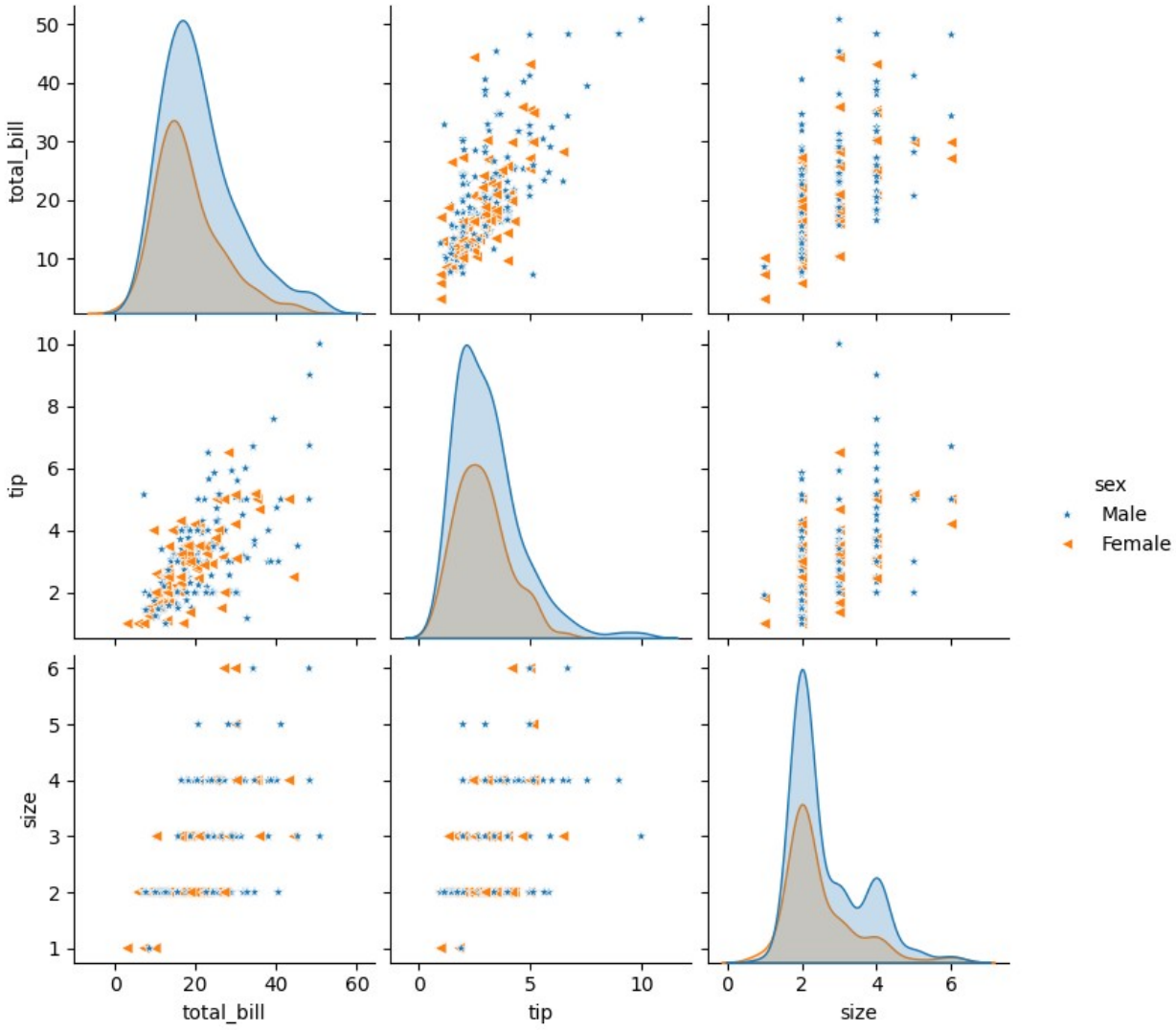
```
sns.pairplot(var2, vars=["total_bill", "tip"],hue="sex")
plt.show()
```




```
sns.pairplot(var2,hue="sex", y_vars=["total_bill", "tip"])
plt.show()
```




```
sns.pairplot(var2,hue="sex", markers=["*","<"])
plt.show()
```



Using kind="reg" adds regression lines to the scatter plots, helping visualize the linear relationship between variables. It makes trends or correlations easier to spot.

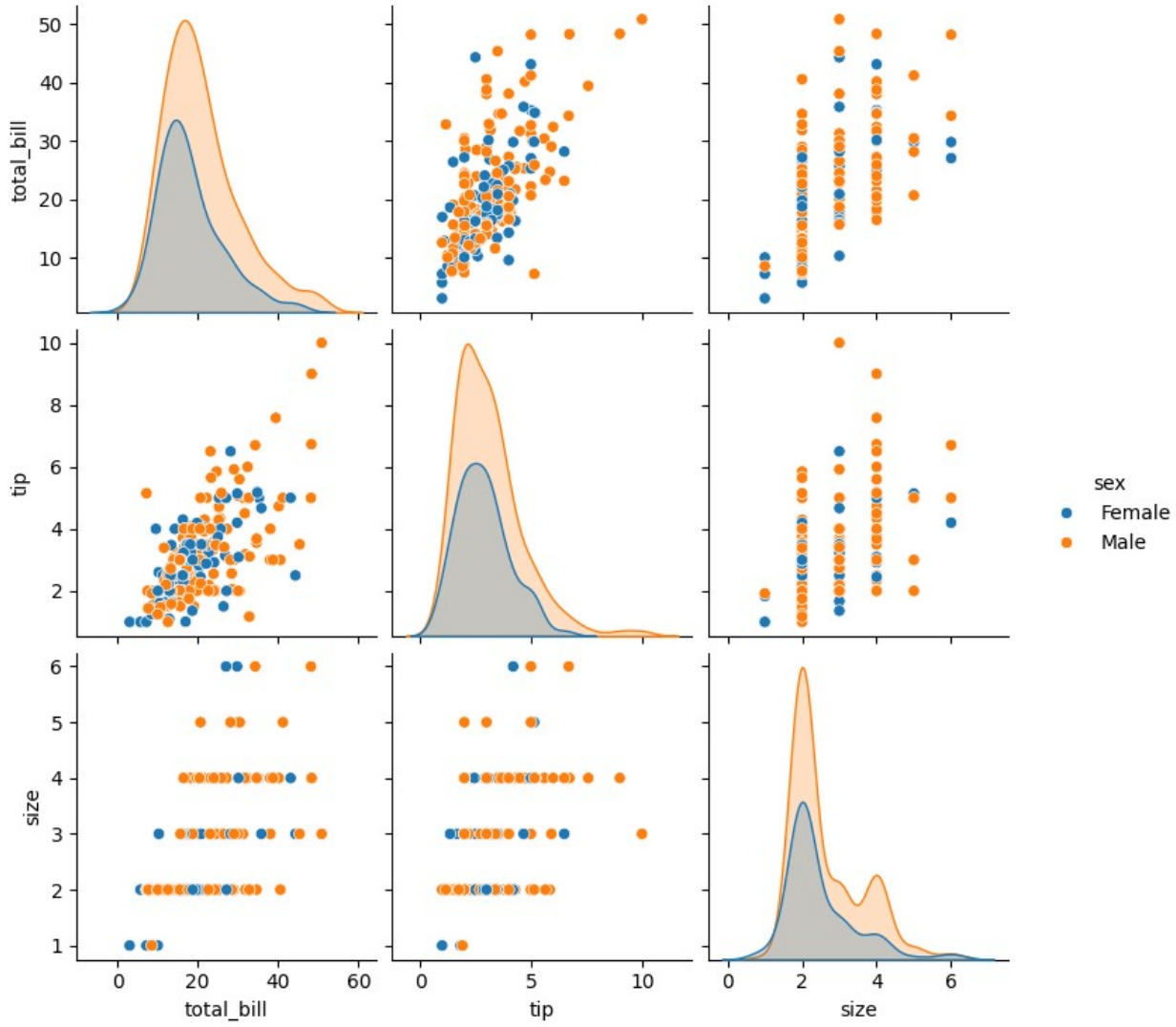
kind="kde" = it creates a smooth curve that shows how data points are distributed.

kind='hist' creates a histogram.

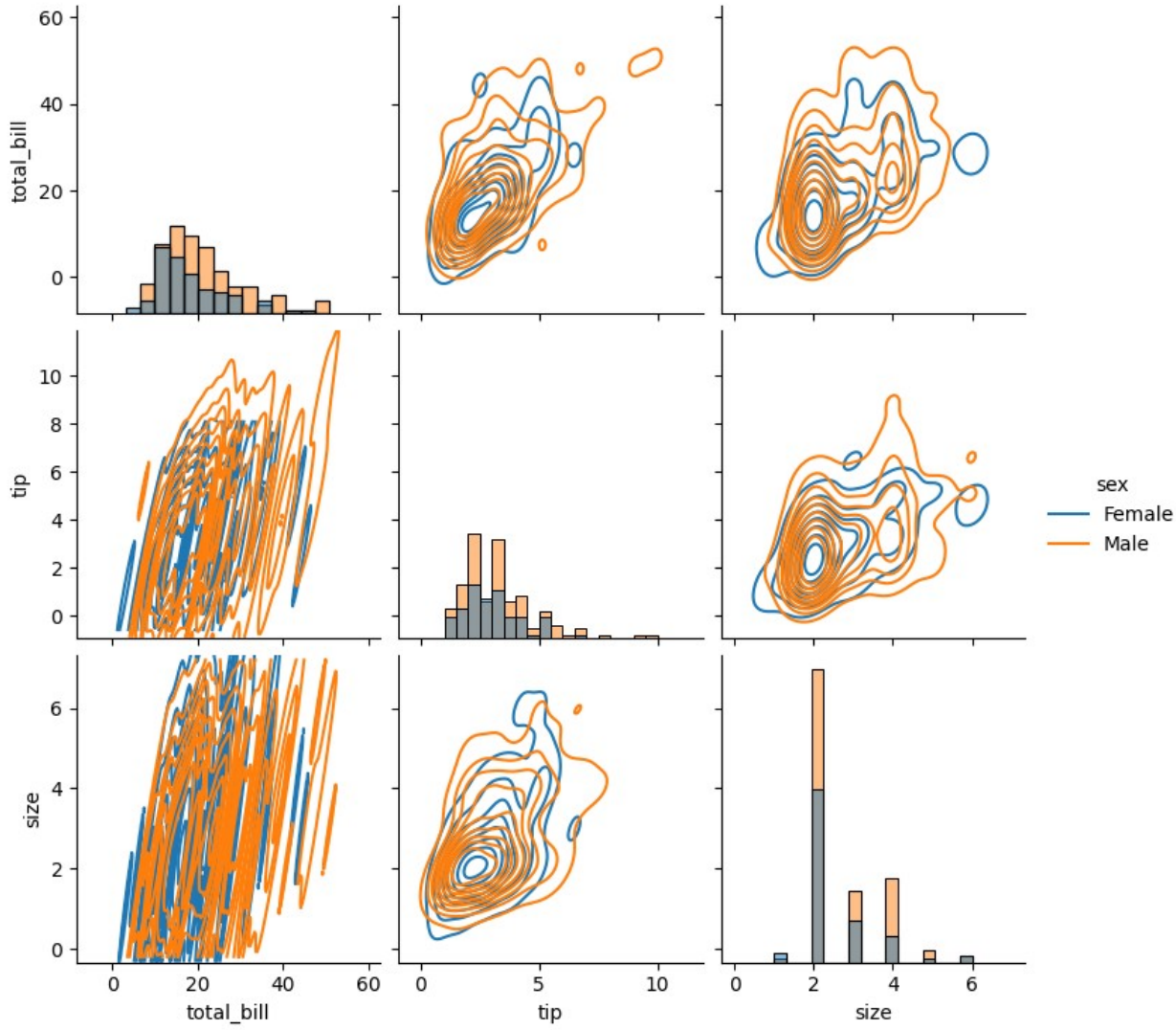
kind='scatter' creates a scatter plot

Diagonal = Distribution of single variables (KDE/histograms). Off-diagonal = Pairwise relationships (scatter plots) for each pair of variables.

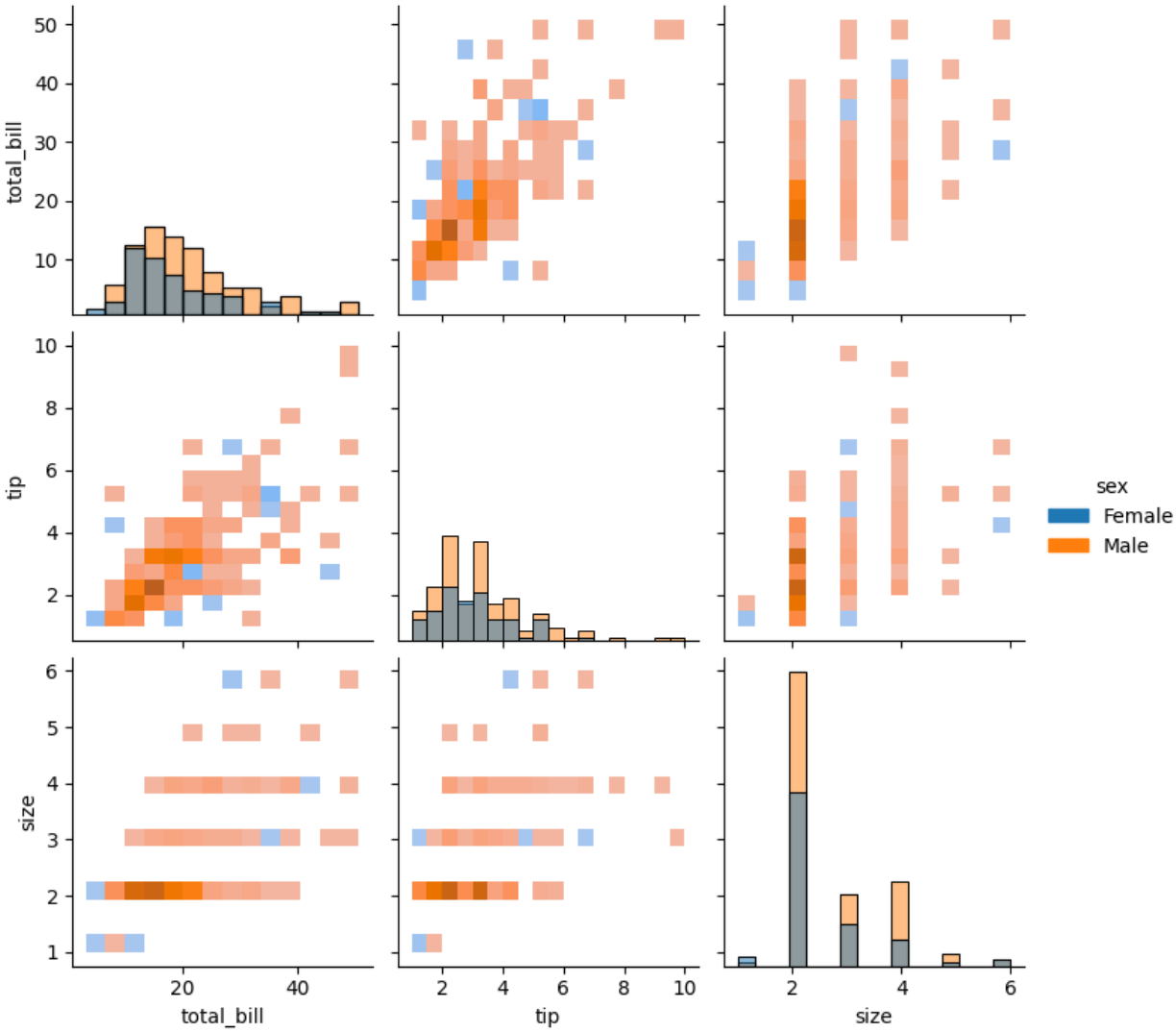
```
sns.pairplot(var2,hue="sex", hue_order=["Female", "Male"],
kind='scatter', ) # 'scatter', 'kde', 'hist', 'reg'
plt.show()
```



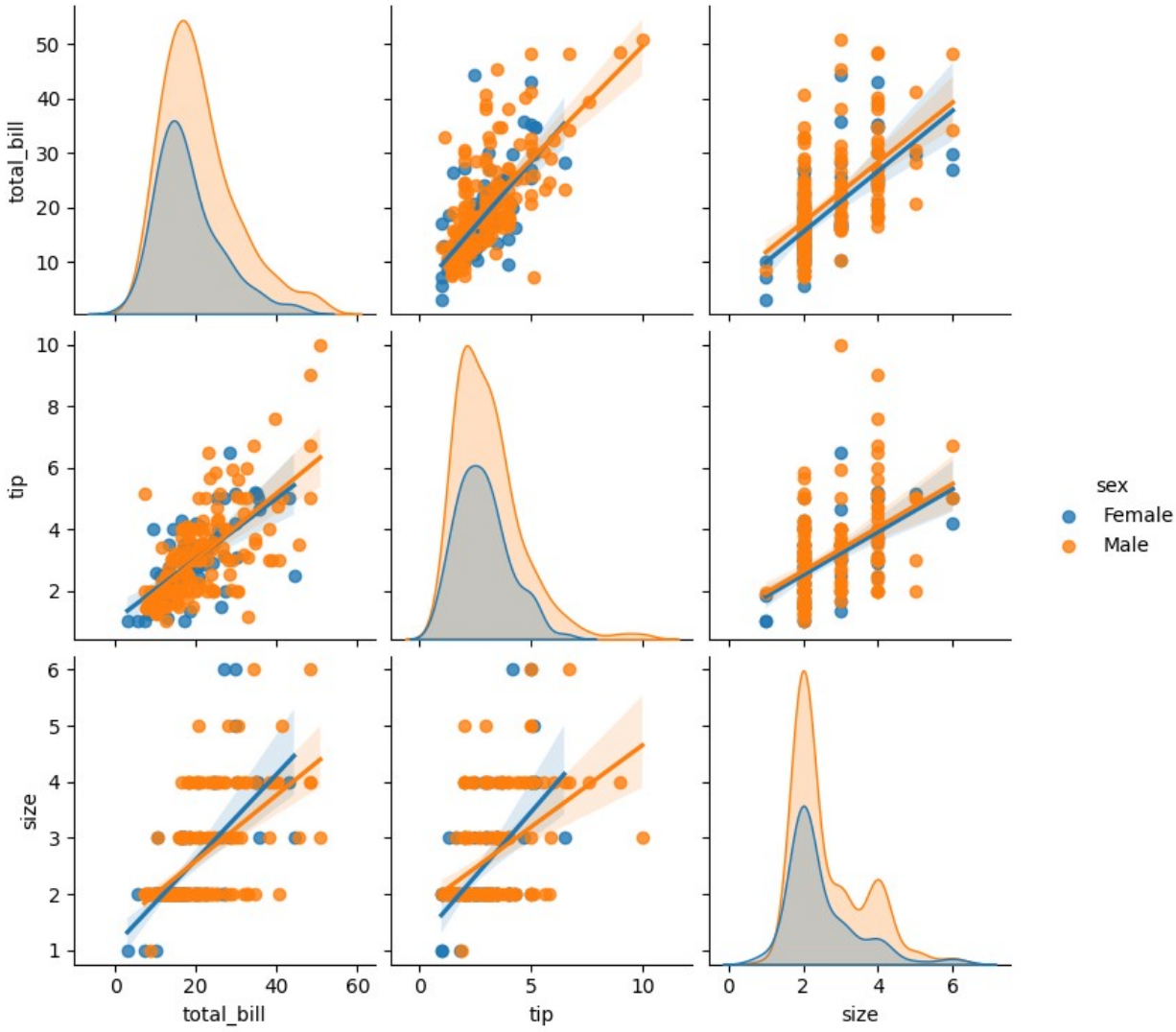

```
sns.pairplot(var2,hue="sex", hue_order=["Female", "Male"], kind='kde',
diag_kind="hist") # 'scatter', 'kde', 'hist', 'reg'
plt.show()
```




```
sns.pairplot(var2,hue="sex", hue_order=["Female", "Male"],
kind='hist') # 'scatter', 'kde', 'hist', 'reg'
plt.show()
```

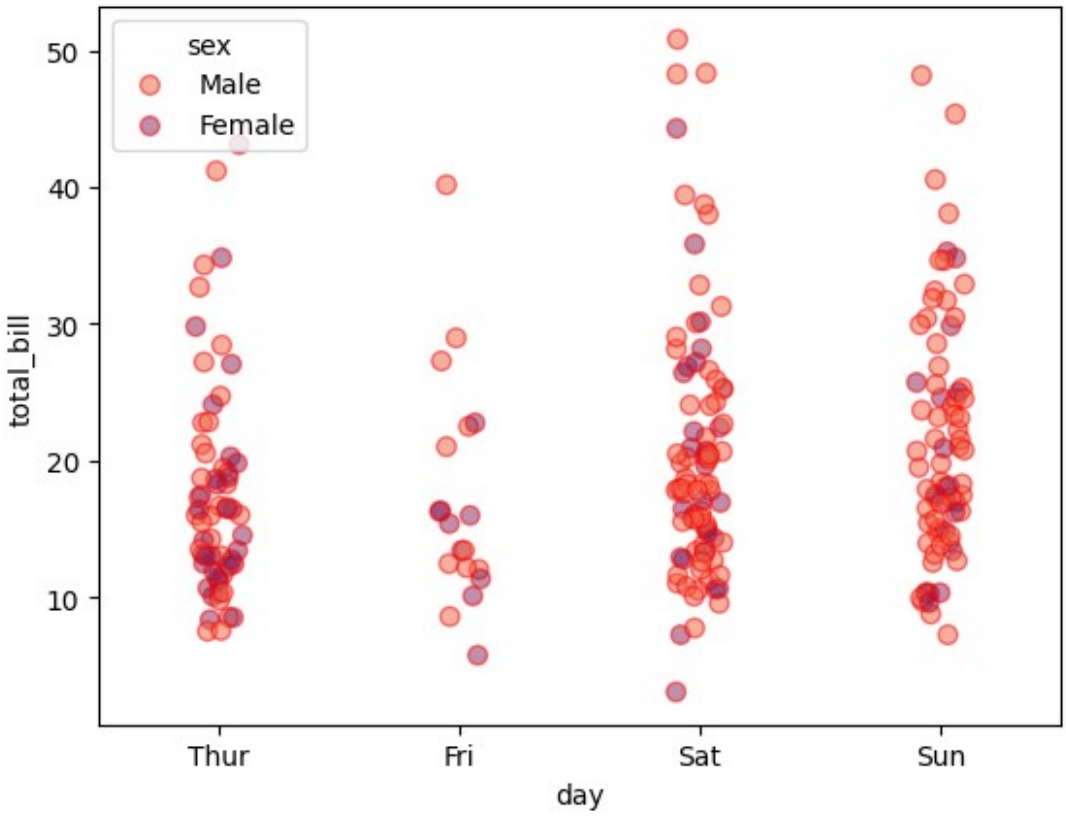



```
sns.pairplot(var2,hue="sex", hue_order=["Female", "Male"], kind='reg')
# 'scatter', 'kde', 'hist', 'reg'
plt.show()
```

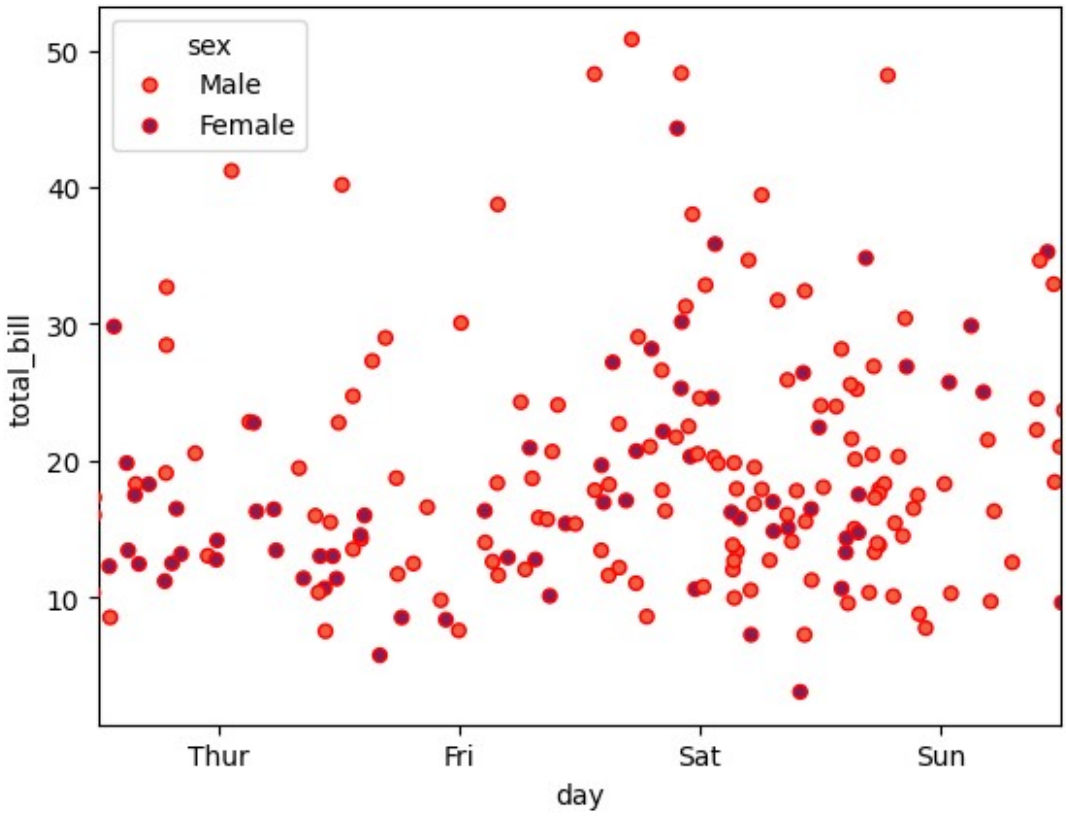


stripplot

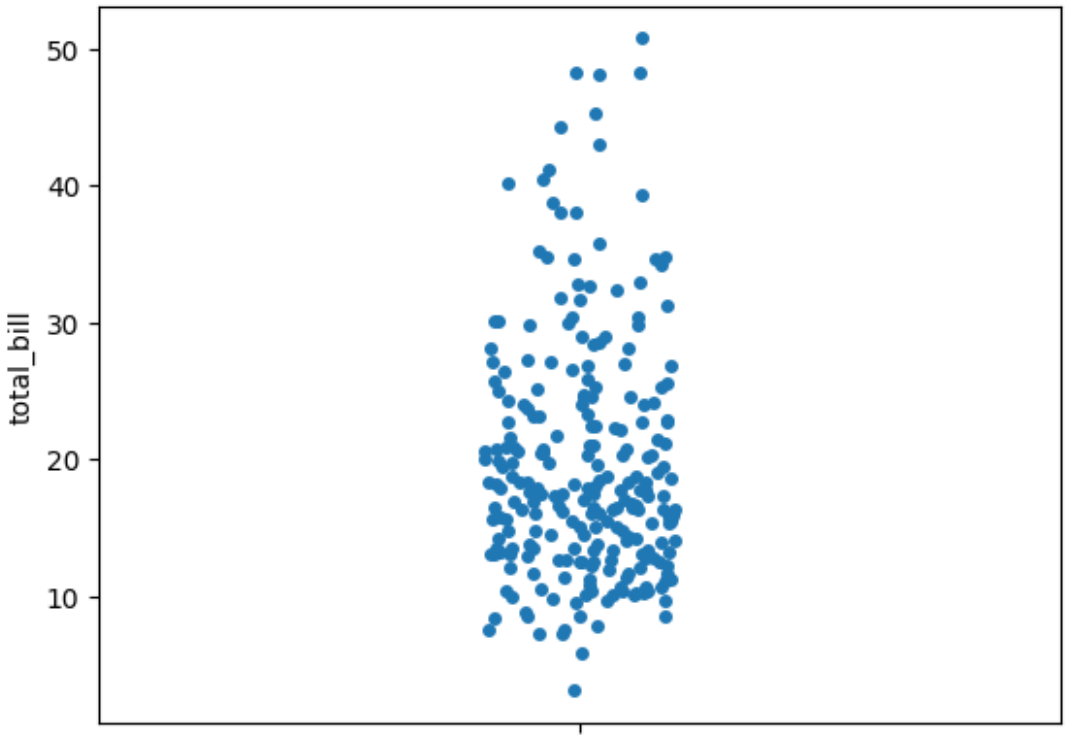
```
sns.stripplot(x="day", y="total_bill", data=var2, hue="sex",
palette='rocket_r', linewidth=1, edgecolor="r",
              size=7,  alpha=0.5)
plt.show()
```




```
sns.stripplot(x="day", y="total_bill", data=var2, hue="sex",
palette='rocket_r', linewidth=1,
              edgecolor="r", jitter=1)
plt.show()
```




```
sns.stripplot(y=var2["total_bill"])
plt.show()
```

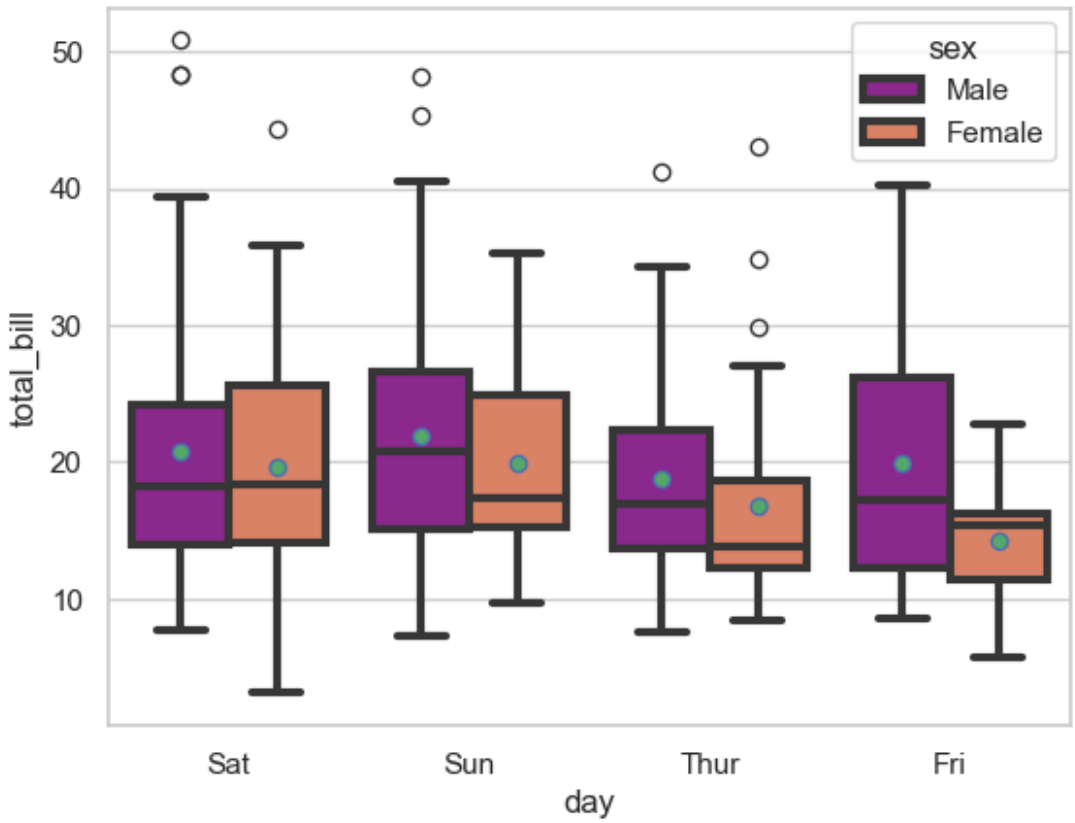


BoxPlot

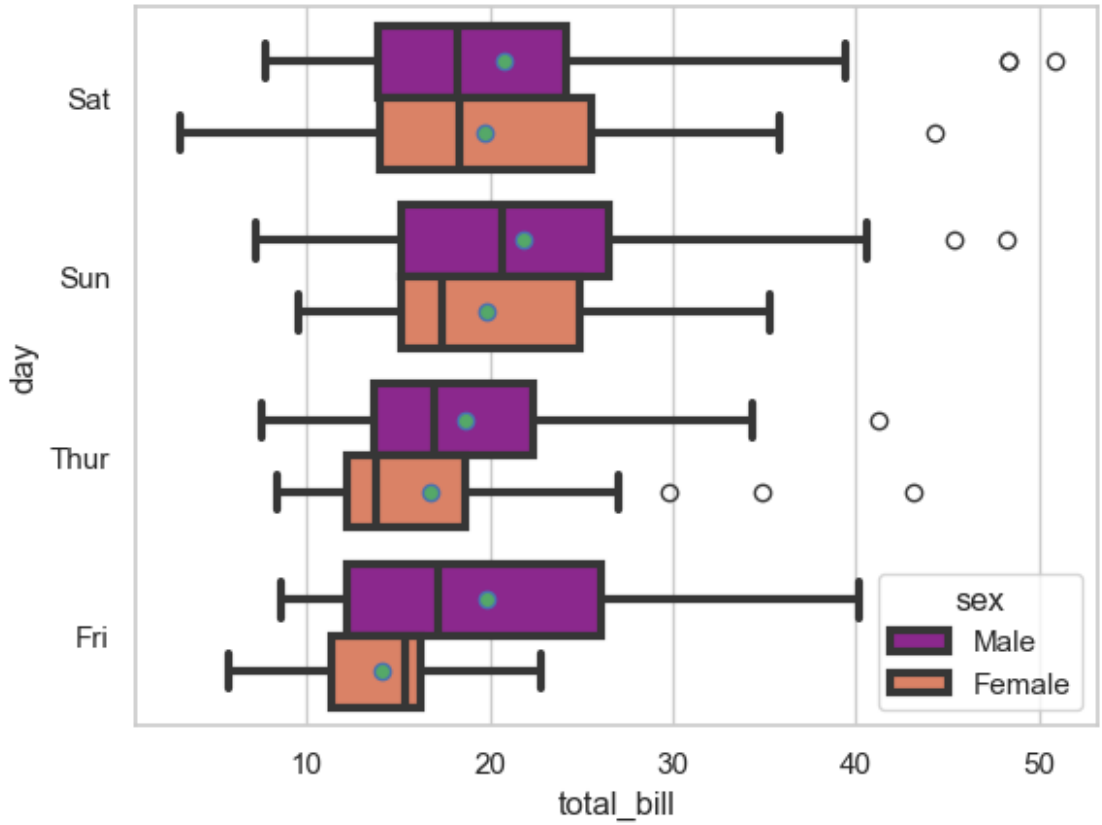
Triangle (Mean): Shows the average value Line (Median): Represents the middle value, dividing the data into two equal halves

To orient (horizontal x=String / vertical y=int) has to be If no, then you can do it by giving the details of X to Y and Y to X

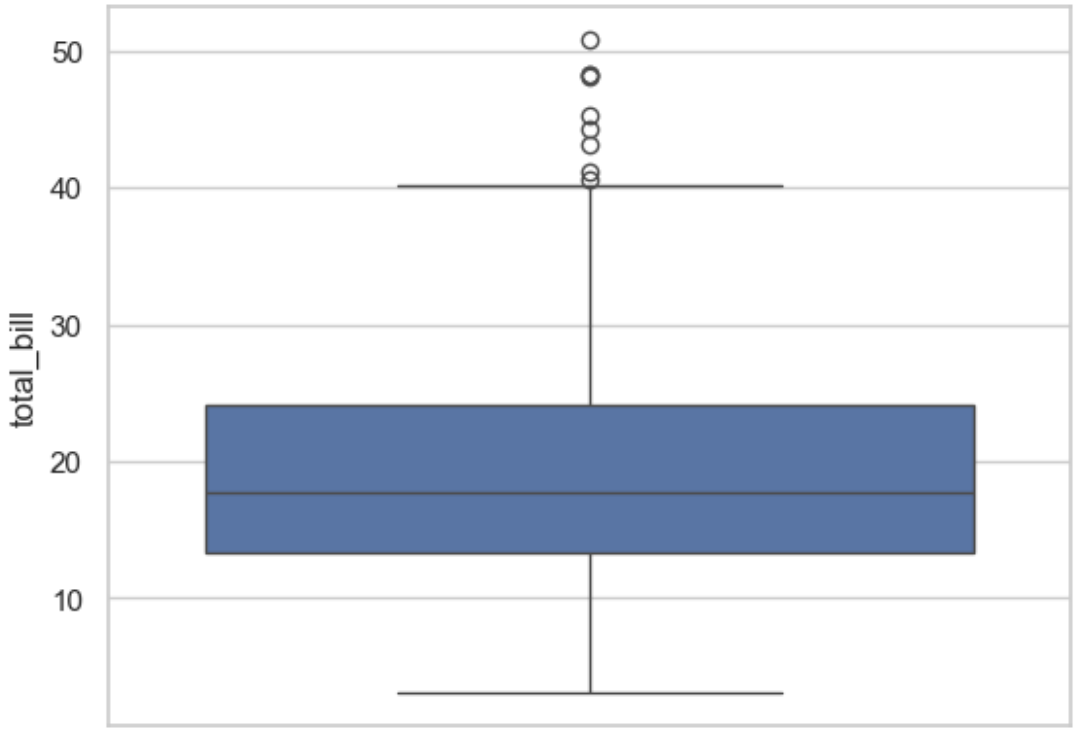
```
sns.set(style="whitegrid")
sns.boxplot(x="day", y="total_bill", data=var2, hue="sex", color="g",
order=["Sat","Sun","Thur","Fri"]
            ,showmeans=True, meanprops={"marker": "o",
"markeredgecolor":"b"}, linewidth=3, palette="plasma")
plt.show()
```




```
To orient (horizontal x=String /  vertical y=int) has to be
If no, then you can do it by giving the details of X to Y and Y to X

sns.set(style="whitegrid")
sns.boxplot(x="total_bill", y="day", data=var2, hue="sex", color="g",
order=["Sat","Sun","Thur","Fri"]
           ,showmeans=True, meanprops={"marker": "o",
"markeredgecolor":"b"}, linewidth=3, palette="plasma")
plt.show()
```




```
sns.set(style="whitegrid")
sns.boxplot(y=var2["total_bill"])
plt.show()
```

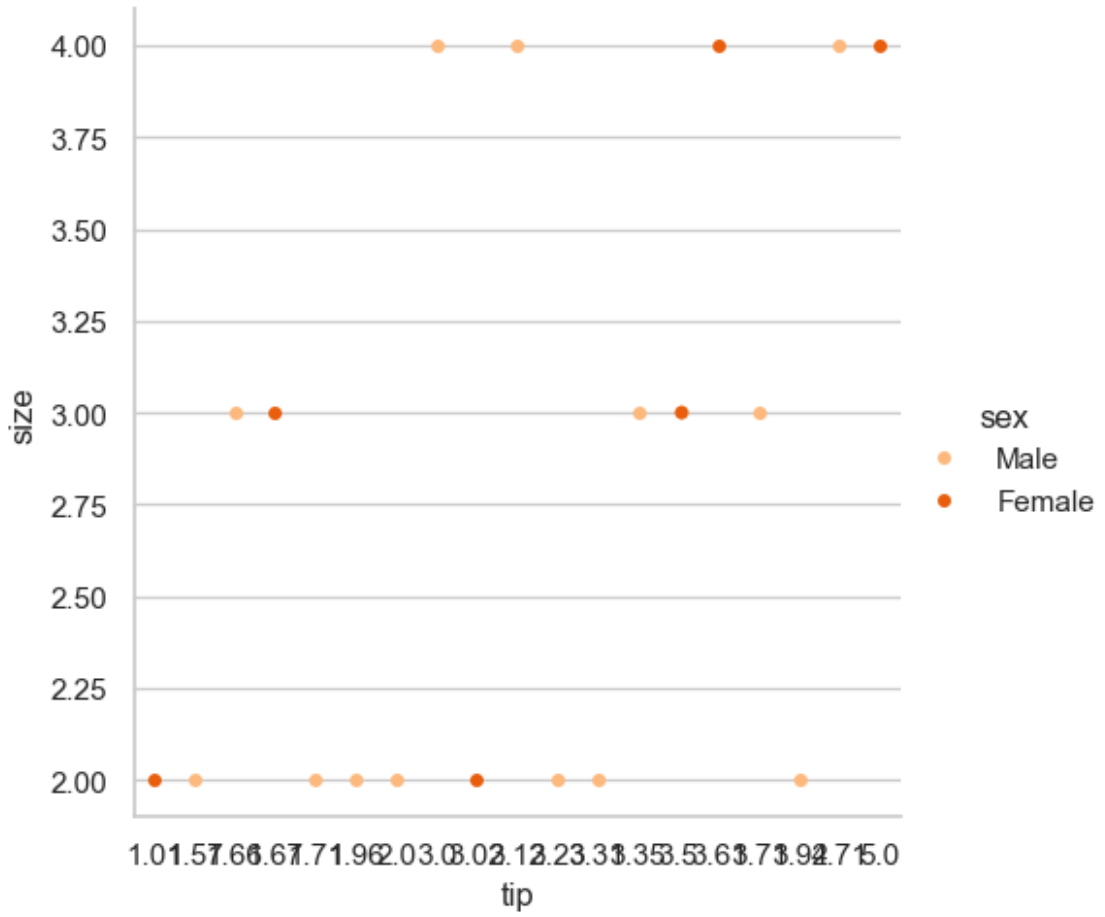


cat plot

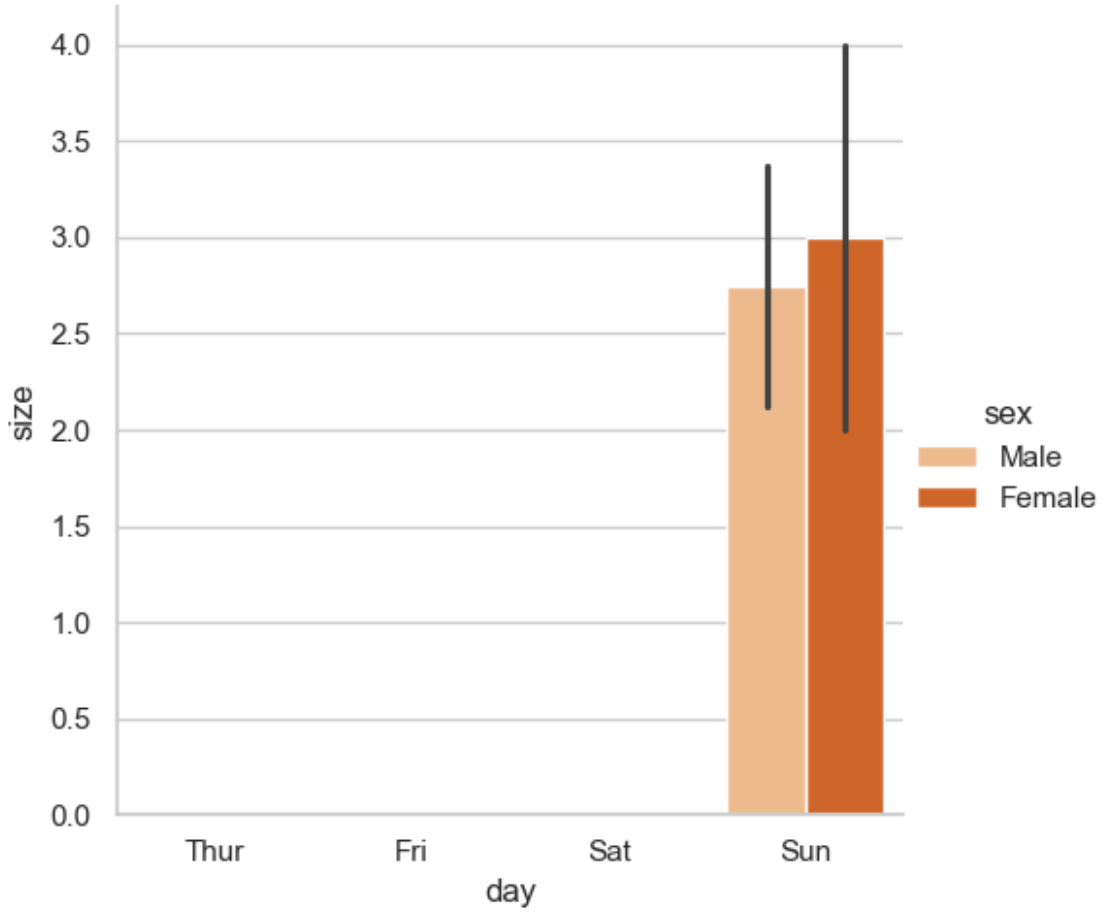
```
var3 = sns.load_dataset("tips").head(10)
var3
```

	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
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

```
sns.catplot(x="tip", y="size", data=var3, hue="sex",
palette='Oranges', height=5)
plt.show()
```




```
sns.catplot(x="day", y="size", data=var3, hue="sex",
palette='Oranges', height=5, kind="bar")
plt.show()
```



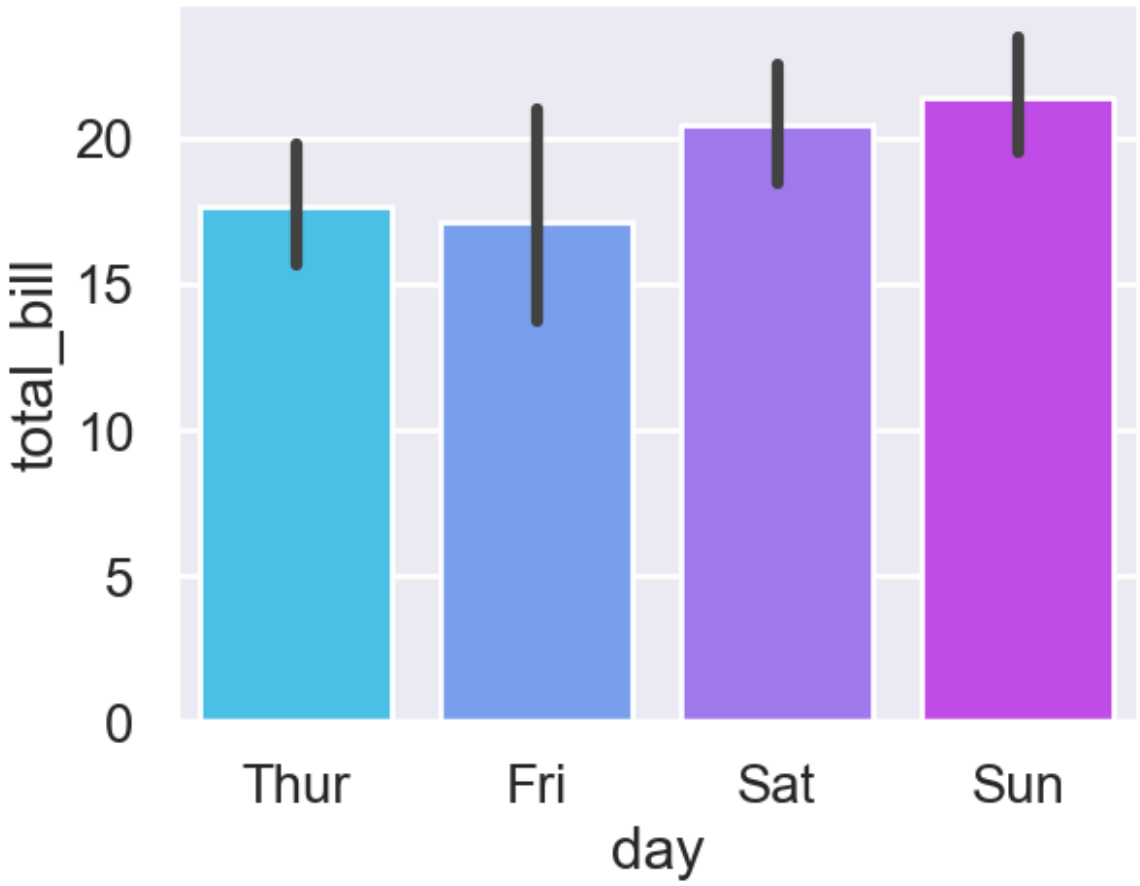
styling plot

-figure styles -removing axes spines -scale and context

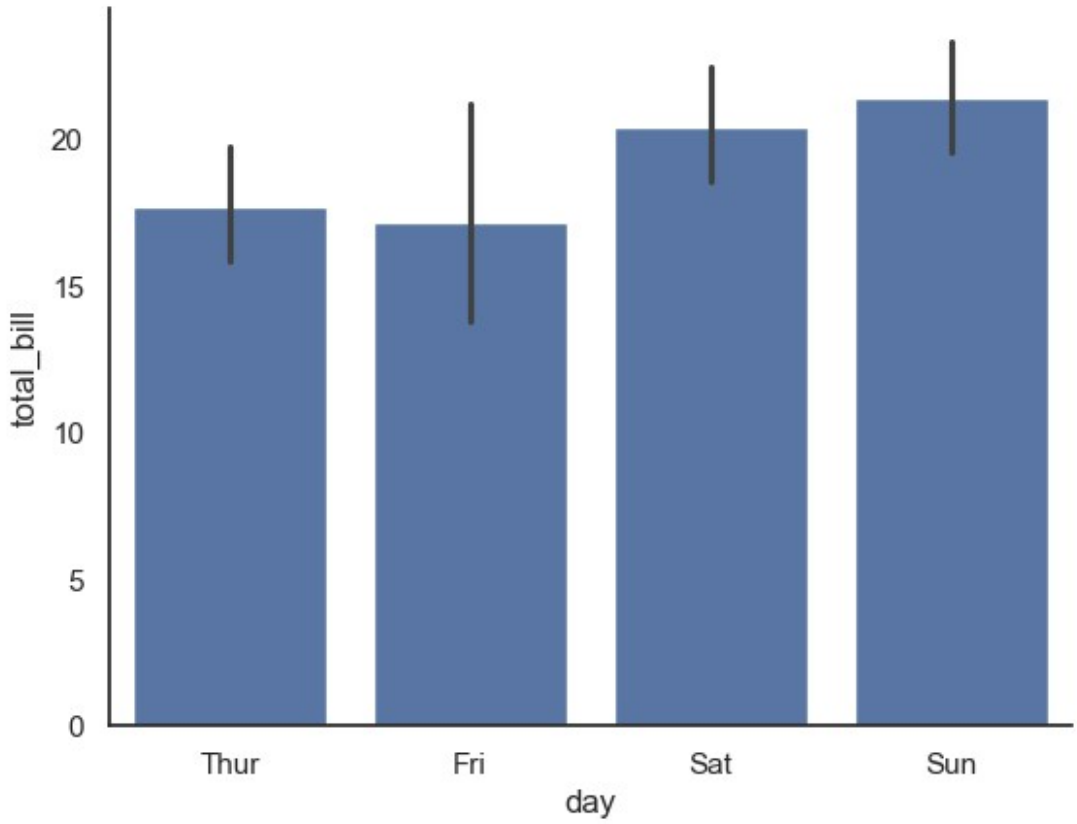
style = notebook, paper, poster

```
sns.set_style("darkgrid")
sns.set_context("poster", font_scale=0.9)
sns.barplot(x="day", y="total_bill", data=var2, palette="cool")
plt.show()
```

```
C:\Users\JIya\AppData\Local\Temp\ipykernel_9404\3182457328.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x="day", y="total_bill", data=var2, palette="cool")
```

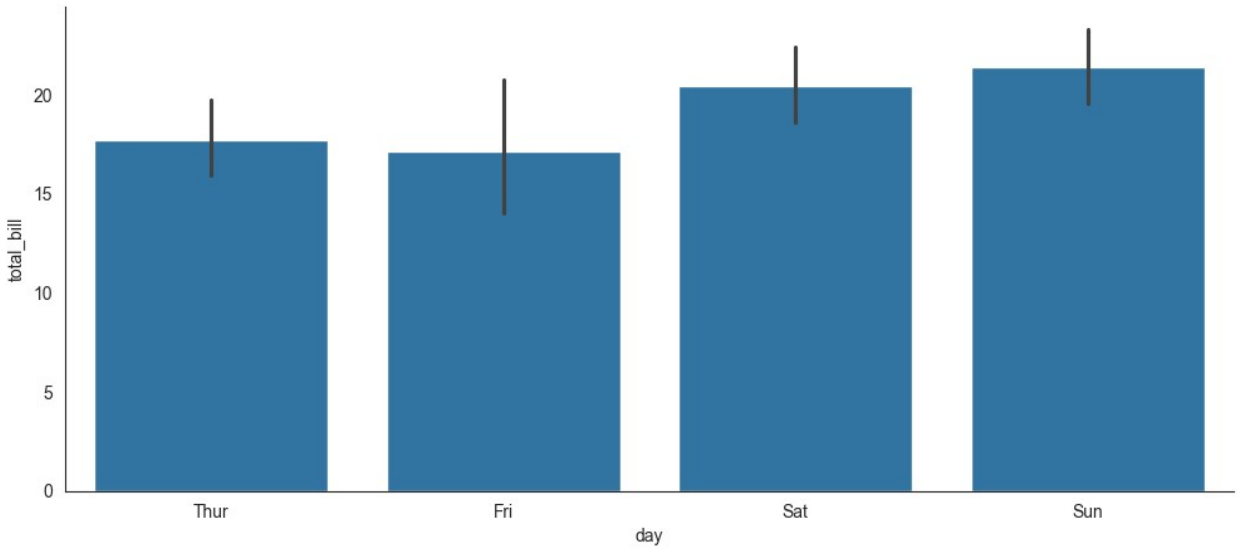



```
sns.set_style("white")
sns.barplot(x="day", y="total_bill", data=var2)
sns.despine()
plt.show()
```



despine() removing axes spines

```
sns.set_style("white")
plt.figure(figsize=(12,5))
sns.barplot(x="day", y="total_bill", data=var2)
sns.despine()
plt.show()
```



facet grid

```
fg = sns.FacetGrid(var2, col="day", hue="sex", height=8.5, aspect=1,
palette='summer')
fg.map(plt.bar, "total_bill", "tip").add_legend()
plt.show()
```