seaborn

November 5, 2024

1 Seaborn Visualization

Seaborn

[3]:
```
# seaborn is a library mostly used for statistical plotting in python.It is␣
 ↪built top of Matplotlib and provides beautiful default  styles and color to␣
 ↪maake statistical plots more attractive.
```

[4]:
```
#Seaborn is an amazing visualization library for statistical graphs plotting in␣
 ↪python. It provides beautiful default styles and colour
# to make statistical plts more attractive.It is on top matplotlib library and␣
 ↪is also closely integrated with the data structures from pandas.
```

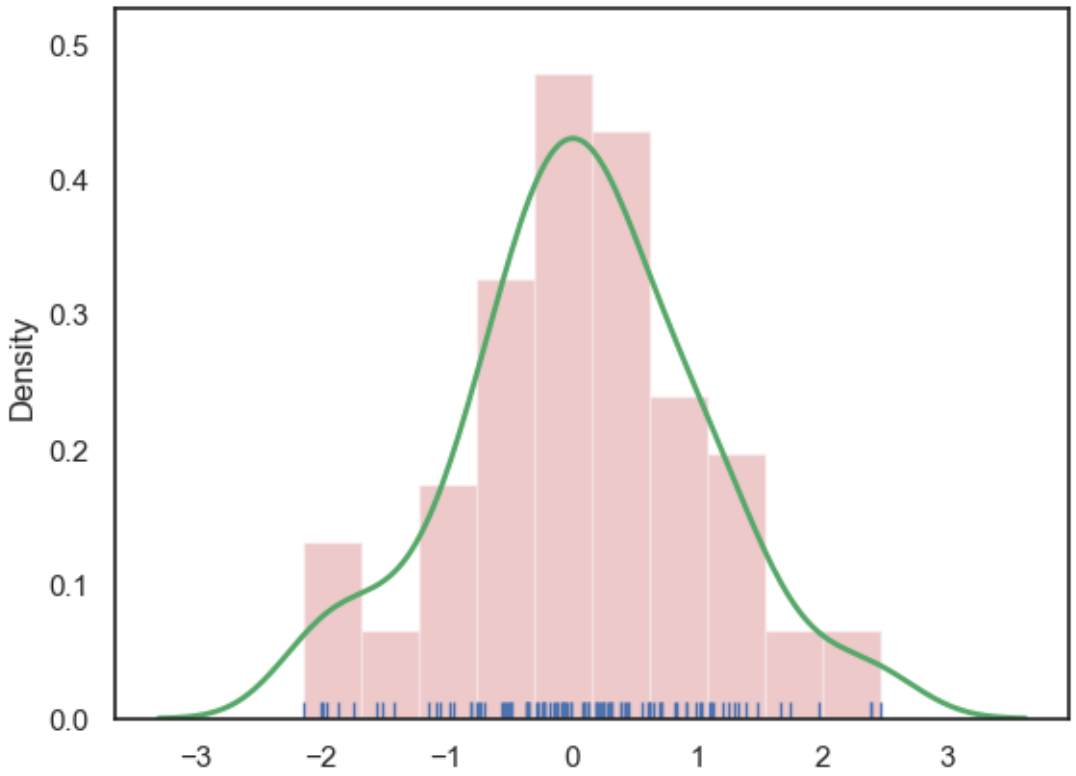
[5]:
```
# Relational plots : This plot is used to understand the realtion between two␣
 ↪variables.
# Categorical plots : This plot deals with categorical varibles and how they␣
 ↪can be visualized.
# Distribution plots : This plot is used for examining variable and how they␣
 ↪can be visualized.
# Regression plots : The regression plots in seaborn are primarily intended to␣
 ↪add a visual guide that helps to emphasize patterns ina dataset during␣
 ↪exploratory data anlyses.
# Matrix plots: A matrx plot is array of scatterplots.
# Multi-plot grids : It is a useful approach to draw multiple instances of the␣
 ↪same plot on different subsets of the dataset.
```

some basic plots using seaborn

[7]:
```
import numpy as np
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

sns.set(style="white")

# genrate a random univariate dataset
rs= np.random.RandomState(10)
```

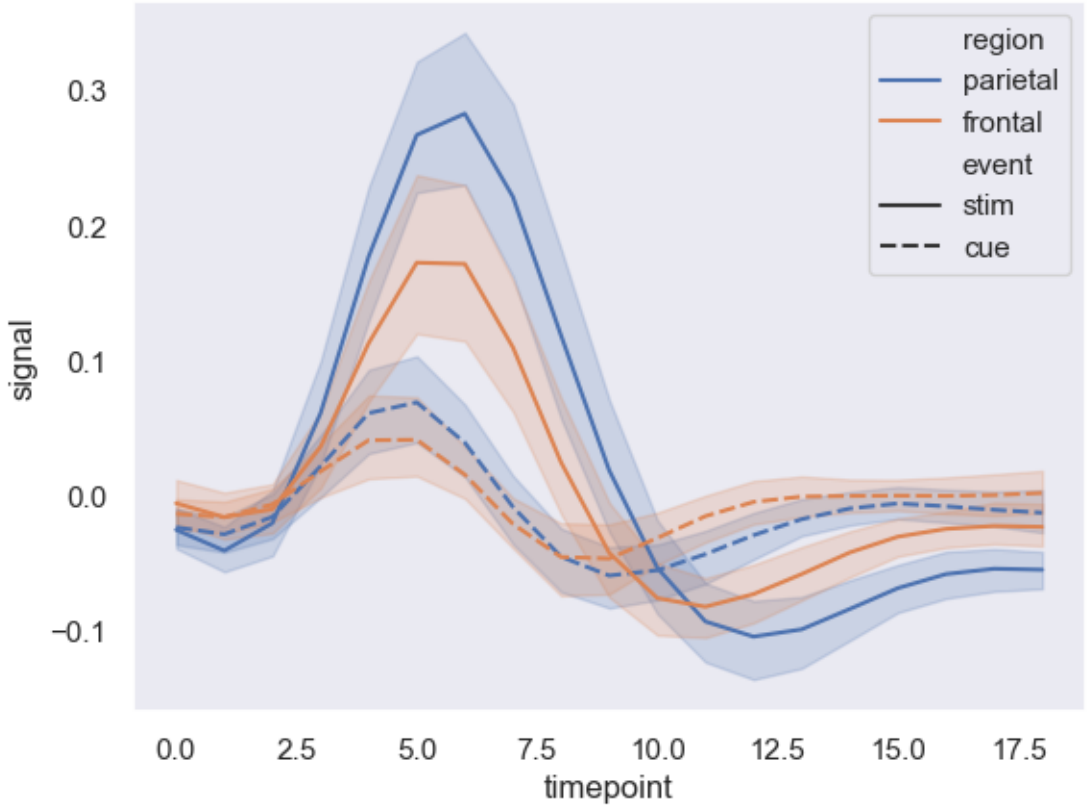
```
d=rs.normal(size=100)

colors=["r", "g","b"]
sns.distplot(d, kde=True, hist=True, bins=10, rug=True, hist_kws={"alpha": 0.3,
                                  "color": colors[0]},
             kde_kws={"color": colors[1], "lw": 2},
             rug_kws={"color": colors[2]})
```

[7]: <Axes: ylabel='Density'>



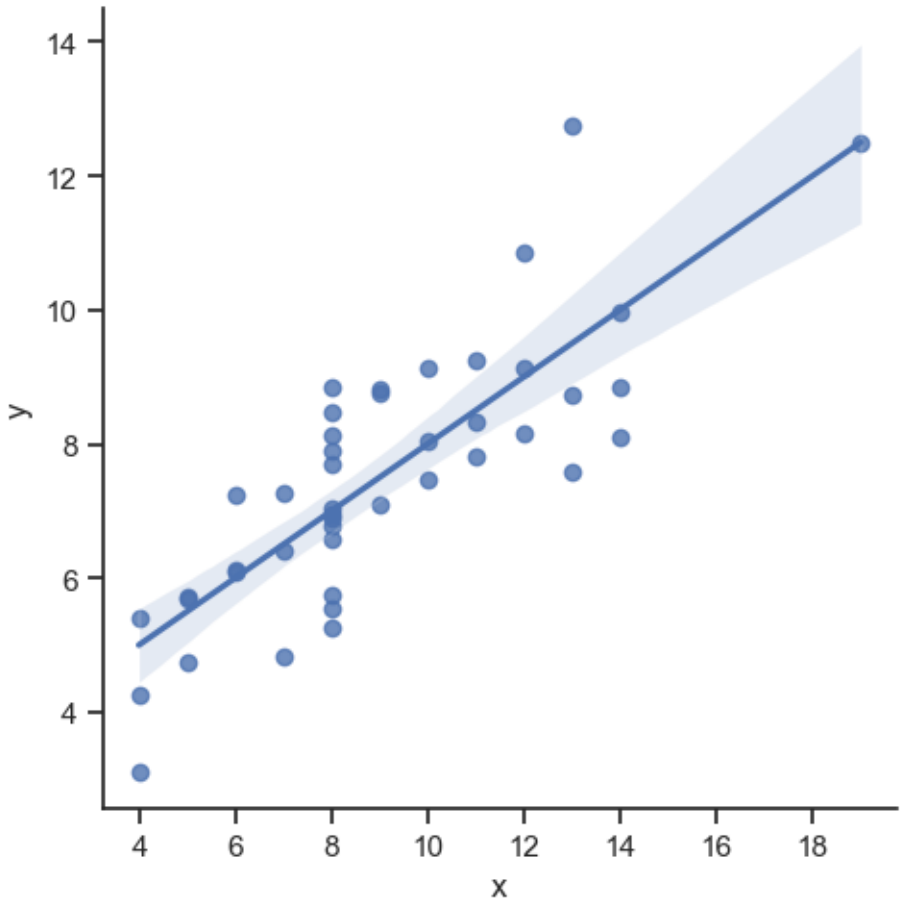
[8]:
```
sns.set(style='dark')
fmri= sns.load_dataset('fmri')

sns.lineplot(x="timepoint", y="signal",hue="region",style="event",data=fmri)
```

[8]: <Axes: xlabel='timepoint', ylabel='signal'>



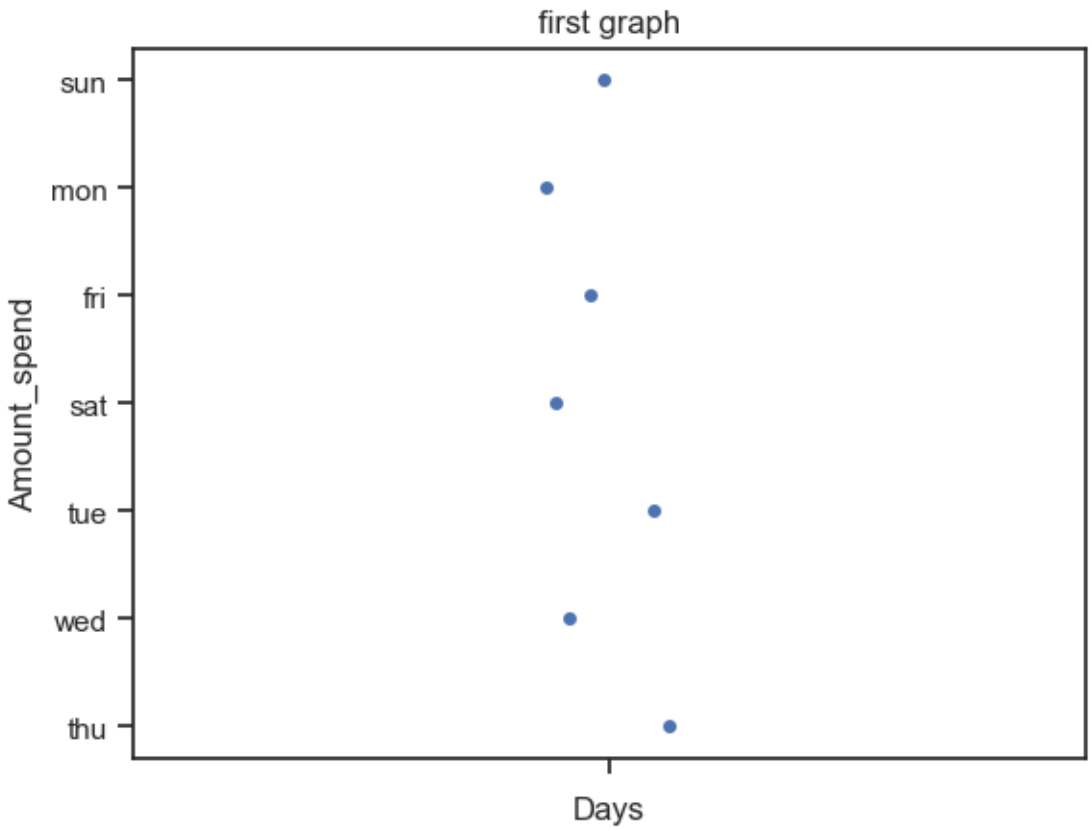
[9]:
```
sns.set(style="ticks")
df=sns.load_dataset("anscombe")
sns.lmplot(x="x", y="y", data=df)
```

[9]: <seaborn.axisgrid.FacetGrid at 0x1d3b4d96510>

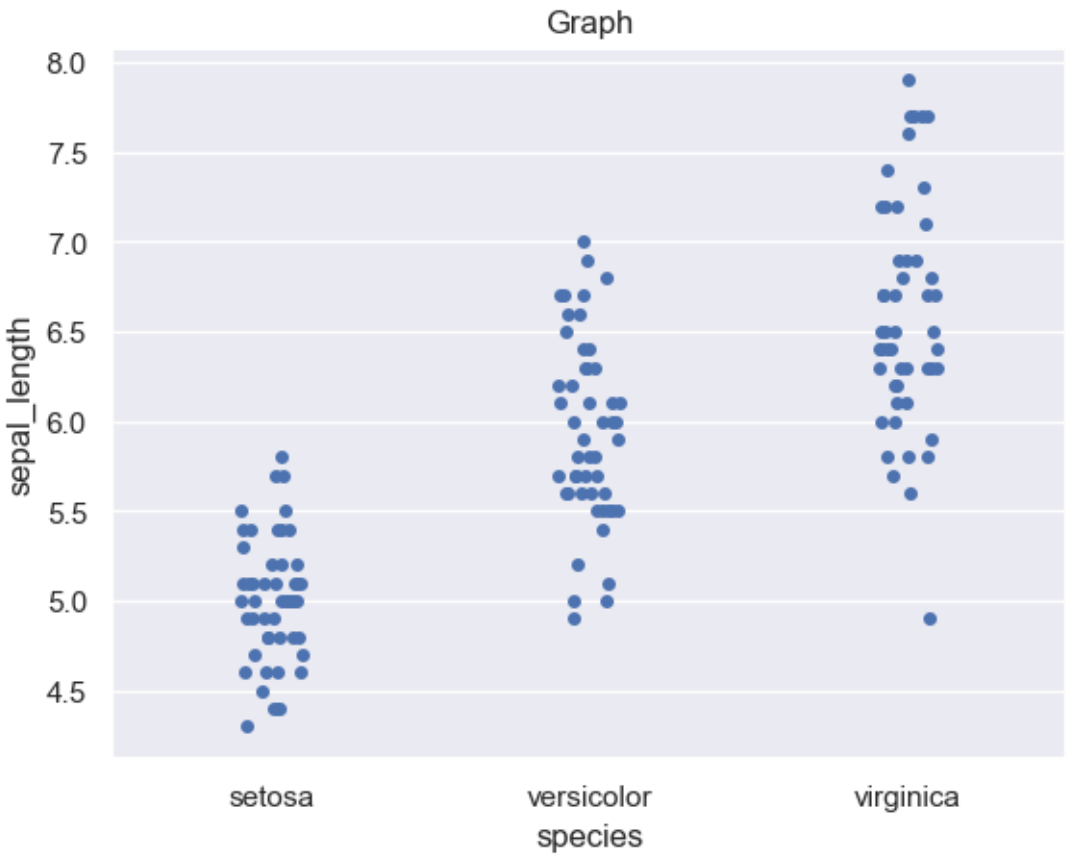


[10]:
```
# Plottimg graph using seaborn/ pyhton
```

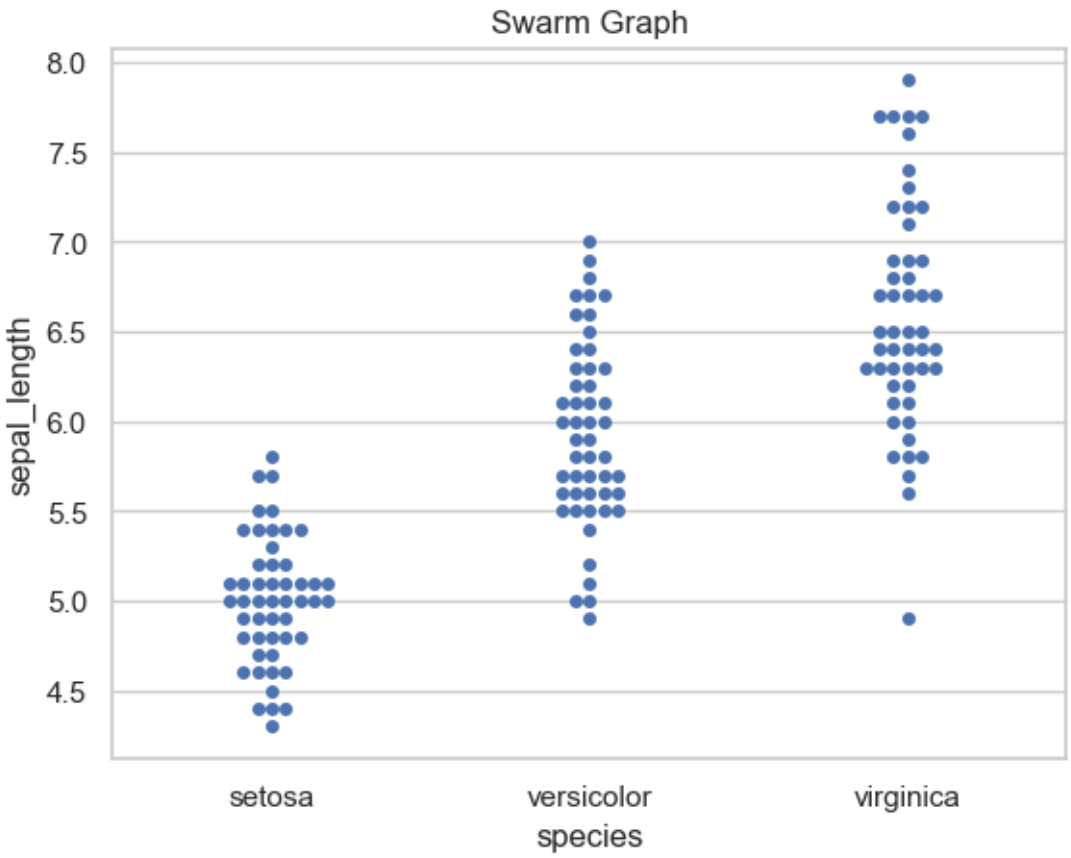
[11]:
```
import matplotlib.pyplot as plt
import seaborn as sns

x=['sun', 'mon', 'fri', 'sat', 'tue', 'wed', 'thu']
y =[5, 6.7, 4, 6, 2, 4.9, 1.8]
ax=sns.stripplot(x)
ax.set(xlabel='Days', ylabel='Amount_spend')
plt.title('first graph')
plt.show()
```



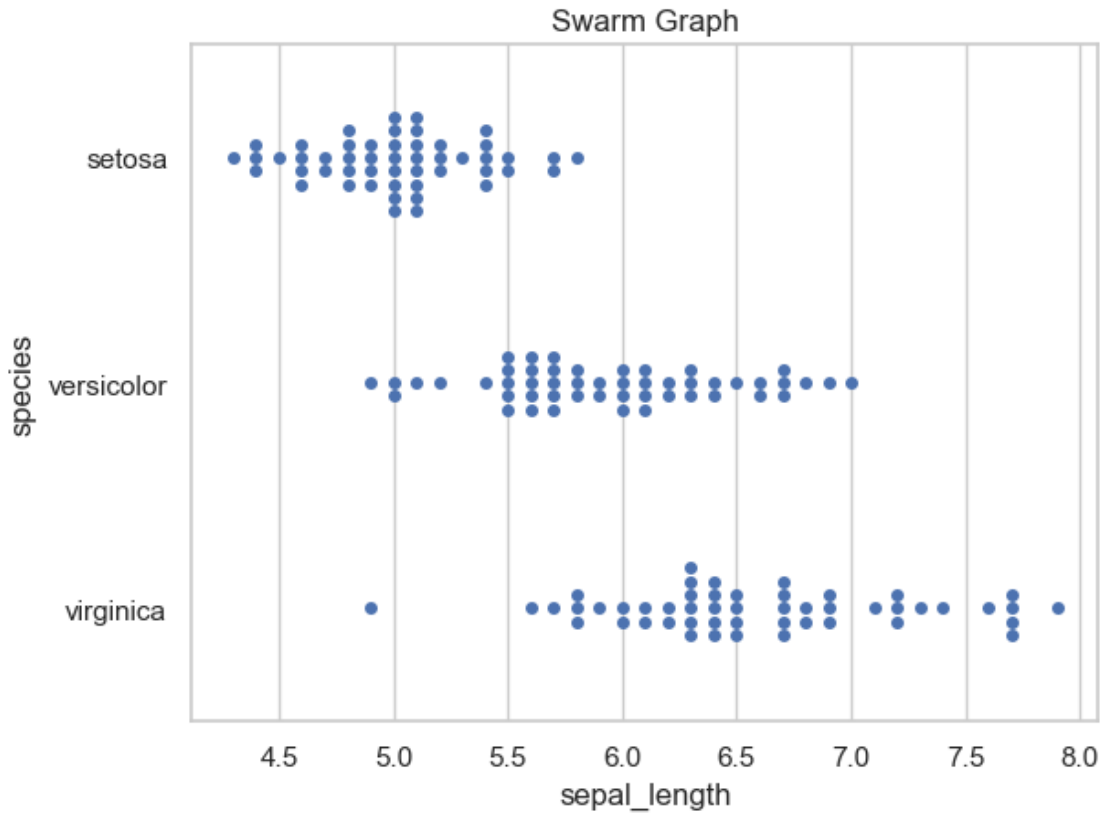
[12]:
```
# iris dataset already present in seaborn module for use
sns.set(style='darkgrid')
iris=sns.load_dataset('iris')
ax=sns.stripplot(x='species', y='sepal_length', data=iris)
plt.title('Graph')
plt.show()
```



[13]:
```
# SWARM PLOT
sns.set(style='whitegrid')
iris=sns.load_dataset("iris")
axis=sns.swarmplot(x='species', y='sepal_length', data=iris)
plt.title('Swarm Graph')
plt.show()
```

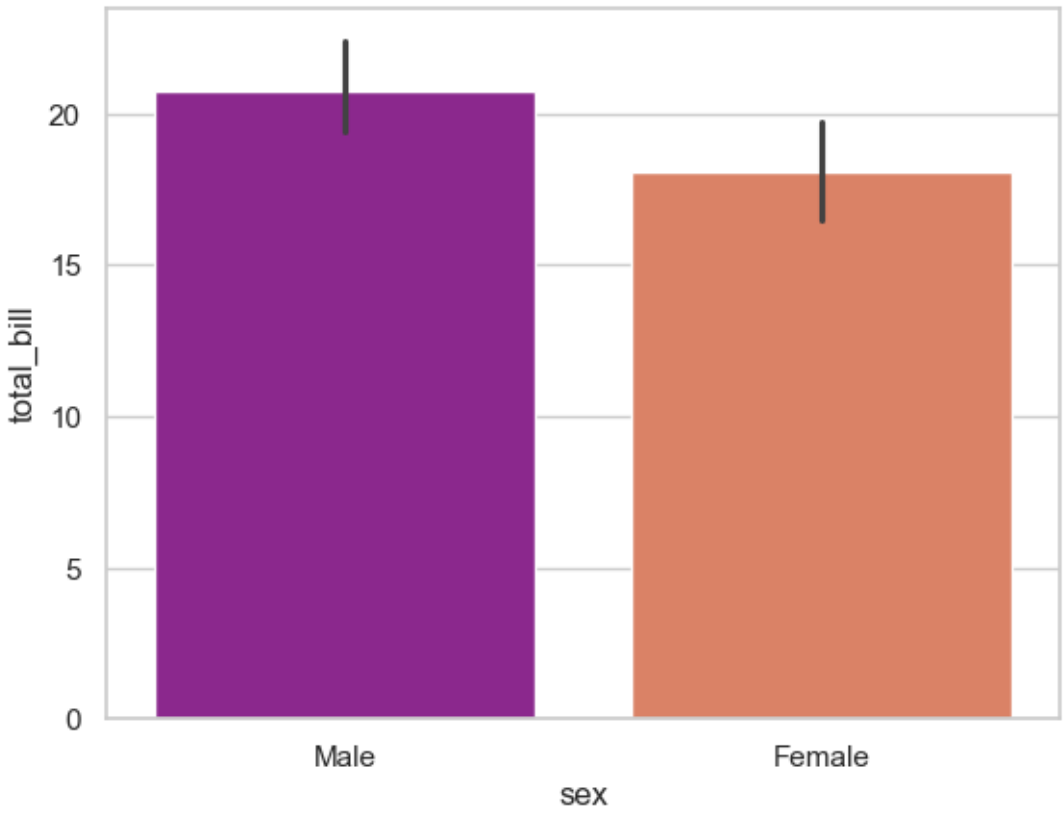


[14]:
```
# if we want to change the data on paricular axis
sns.set(style='whitegrid')
iris=sns.load_dataset("iris")
axis=sns.swarmplot(data=iris, x='sepal_length', y='species')
plt.title('Swarm Graph')
plt.show()
```

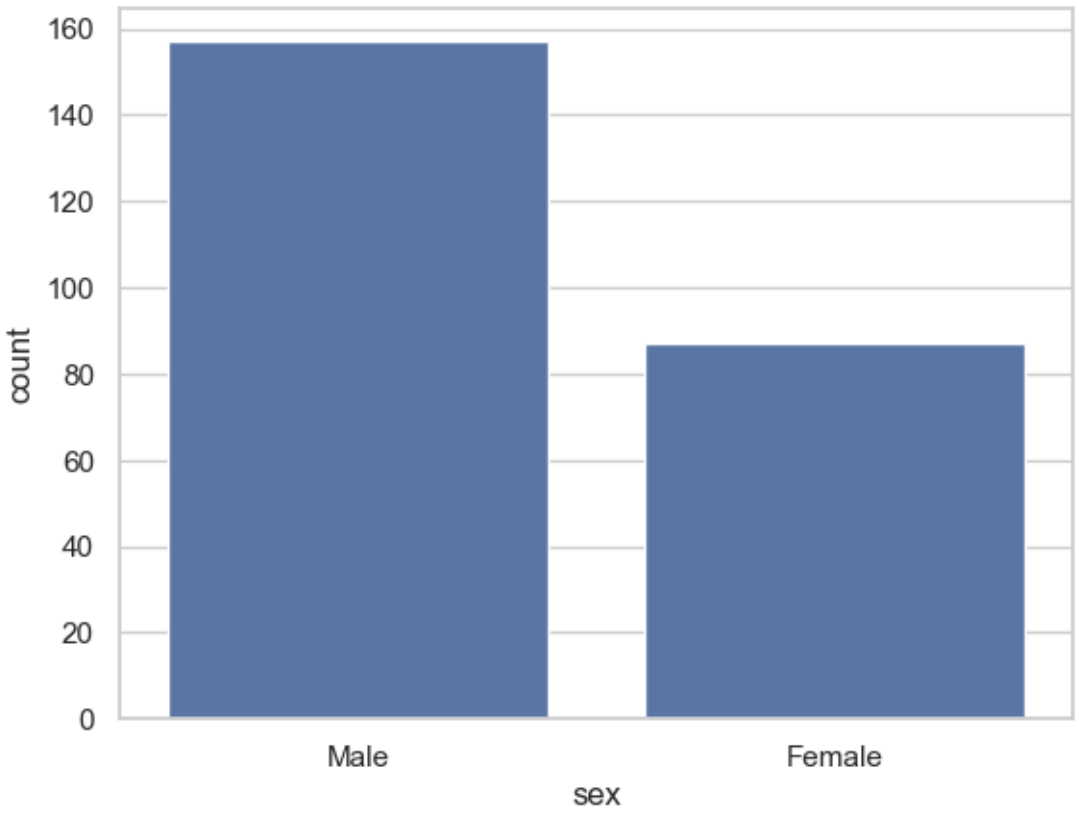


[15]:
```
# Bar graph
df=sns.load_dataset('tips')
sns.barplot(x='sex', y='total_bill',data=df, palette='plasma')
```

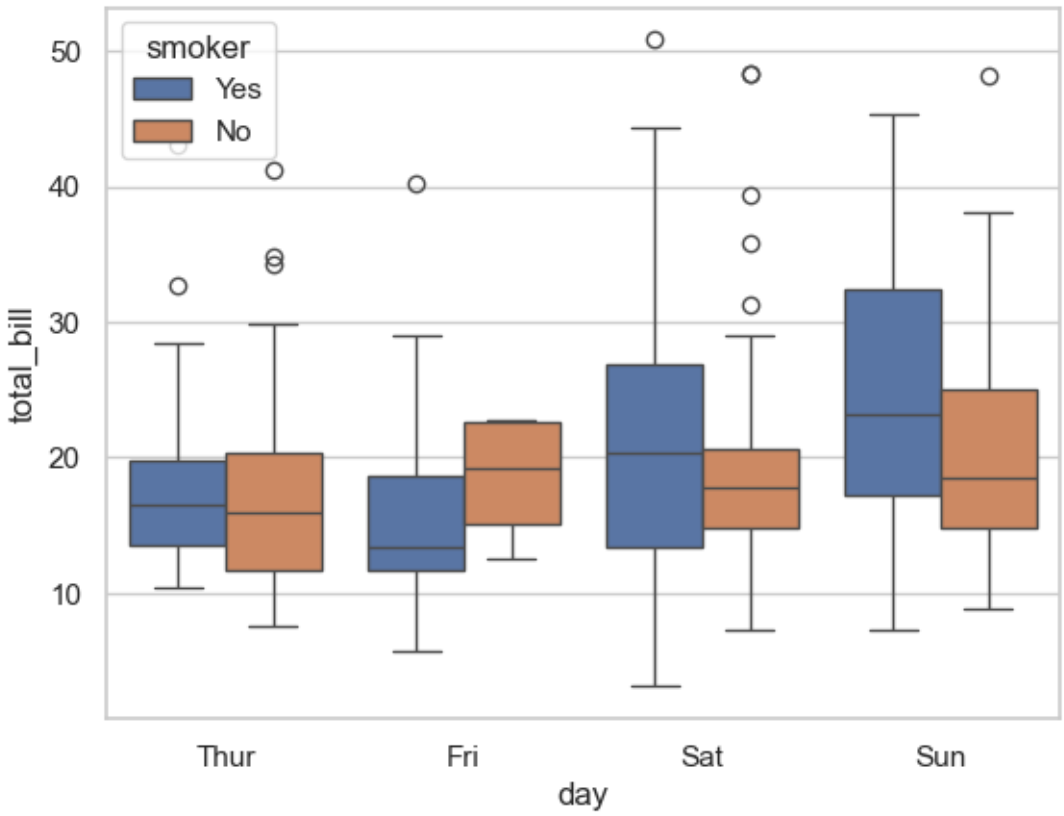
[15]: <Axes: xlabel='sex', ylabel='total_bill'>



[16]:
```
# countplot

df=sns.load_dataset('tips')
sns.countplot(x='sex', data=df)
plt.show()
```



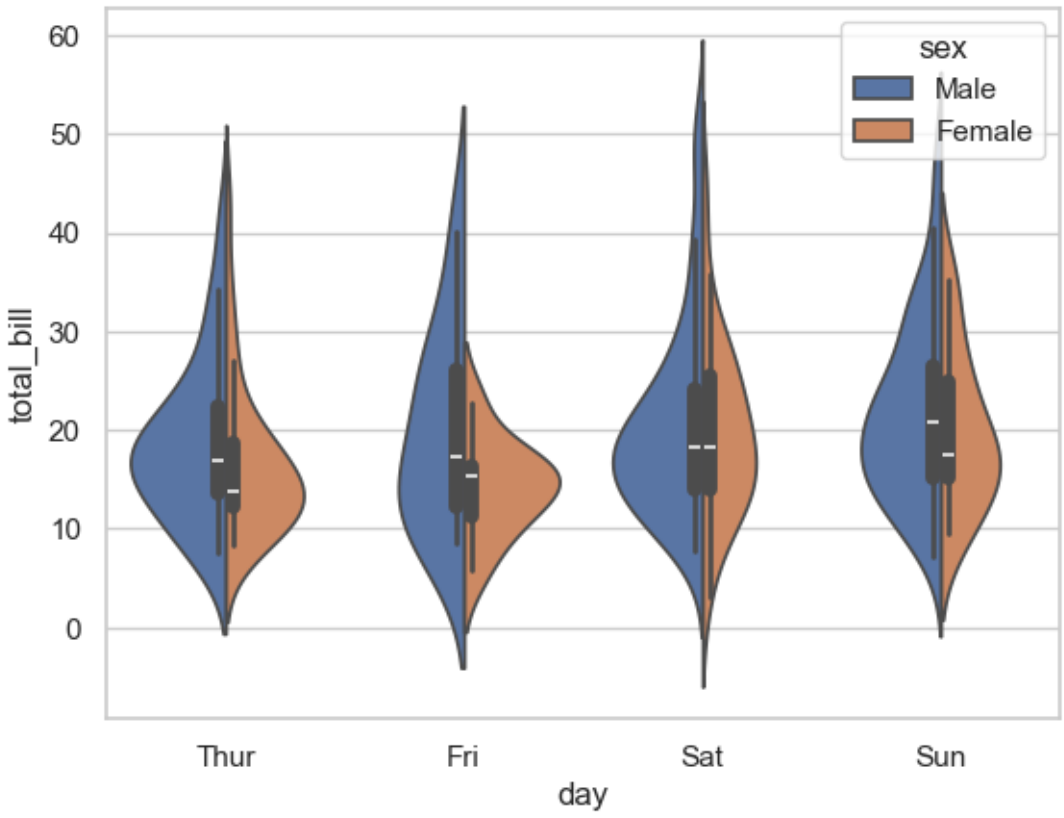
[17]:
```
df=sns.load_dataset('tips')
sns.boxplot(x='day', y='total_bill',data=df, hue='smoker')
plt.show()
```



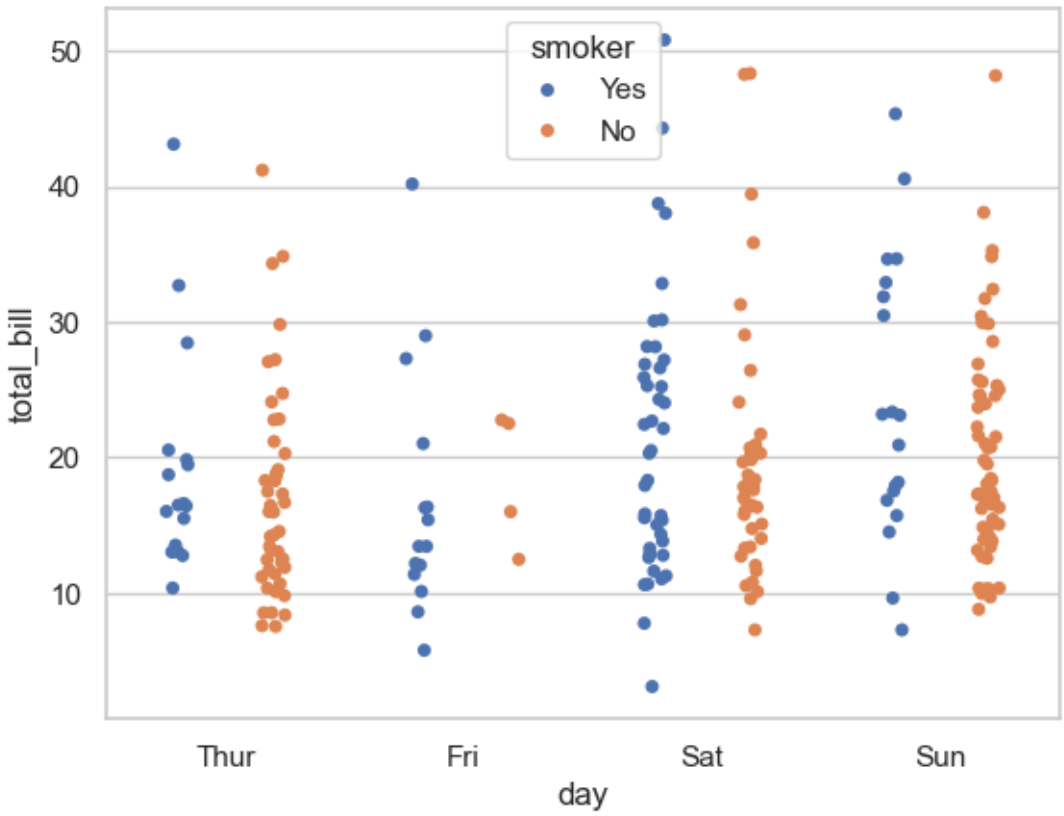
2 Violinplot

[ ]:

[19]:
```
df=sns.load_dataset("tips")
sns.violinplot(x='day', y='total_bill', data=df, hue='sex', split=True)
plt.show()
```



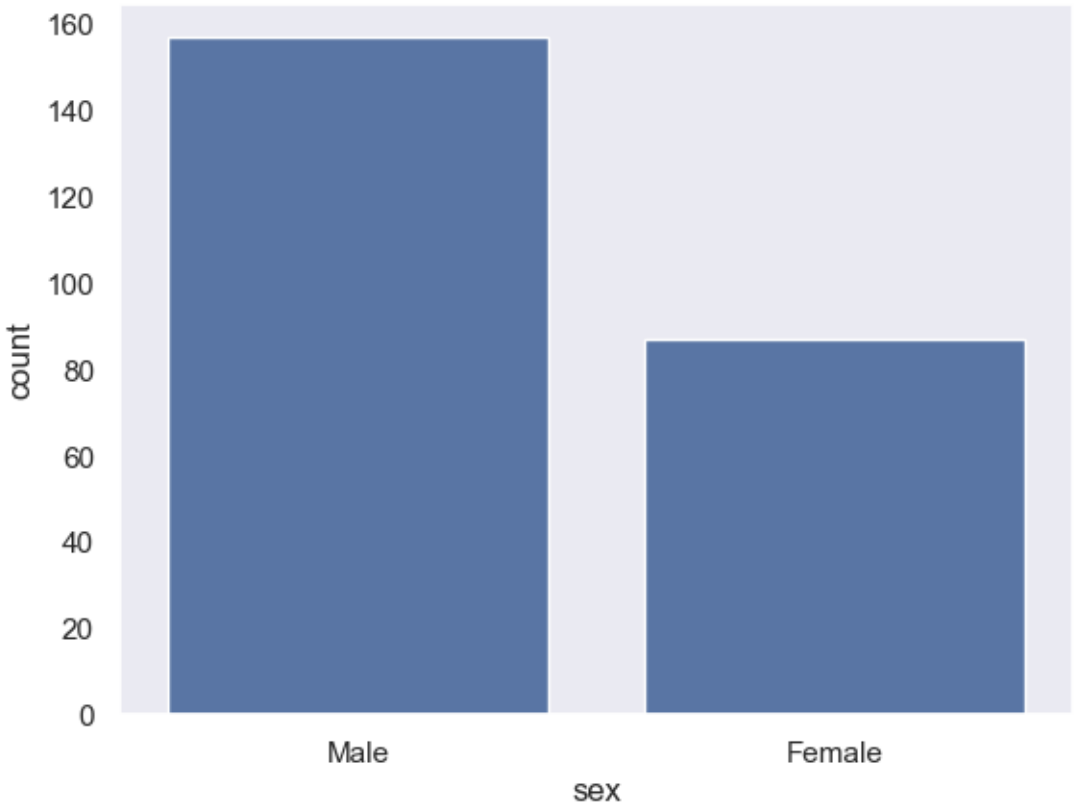
[20]:
```
df = sns.load_dataset('tips')
sns.stripplot(x='day', y='total_bill', data=df,
              jitter=True, hue='smoker', dodge=True)
plt.show()
```



3 Seaborn | Style and color

[22]:
```
# styling themes
#white, dark, whitegrid, darkgrid, ticks
tips=sns.load_dataset('tips')
sns.set_style('dark')
sns.countplot(x='sex', data=tips)
```

[22]:
```
<Axes: xlabel='sex', ylabel='count'>
```



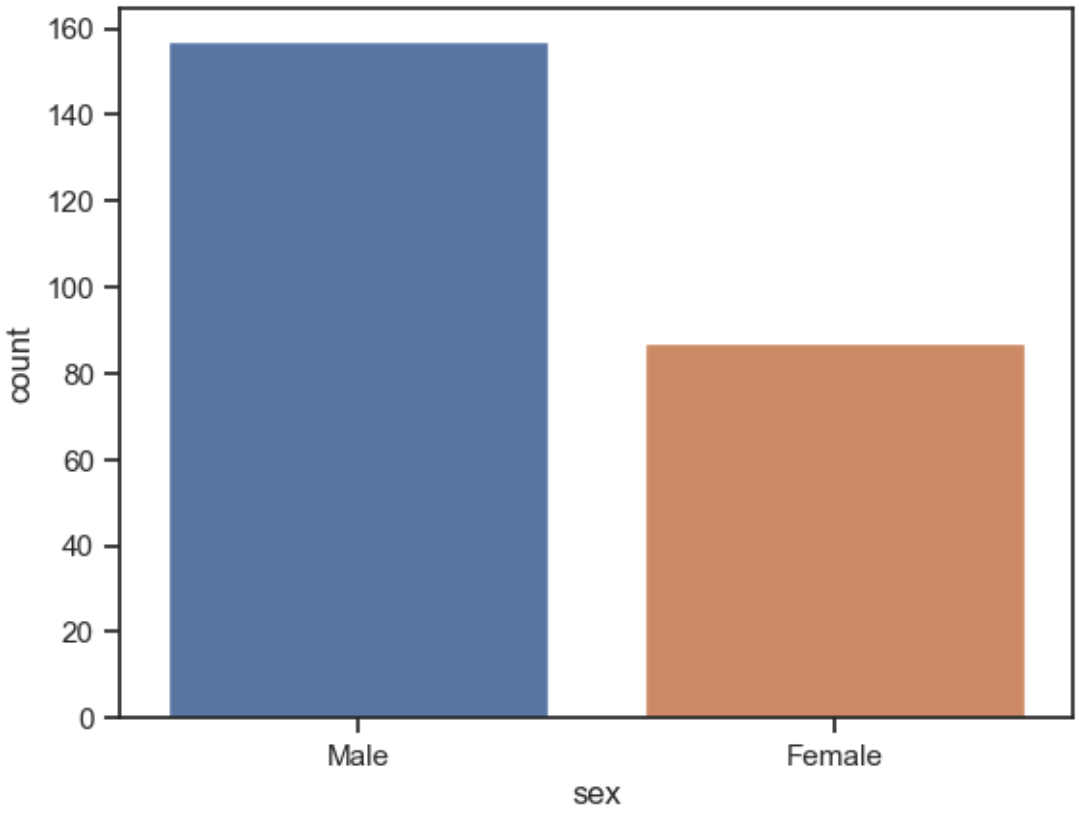
[23]:
```
import pandas as pd
data={
    'name':["kumar","mani","ravi","vasu","sai"],
    "age":[34,89,25,17,67],
    "city":["markapur","anantapur","kurnool","Guntur","prakasam"],
    "salary":[10000,20000,70000,30000,50000]
    }
df=pd.DataFrame(data)
df
```

[23]:

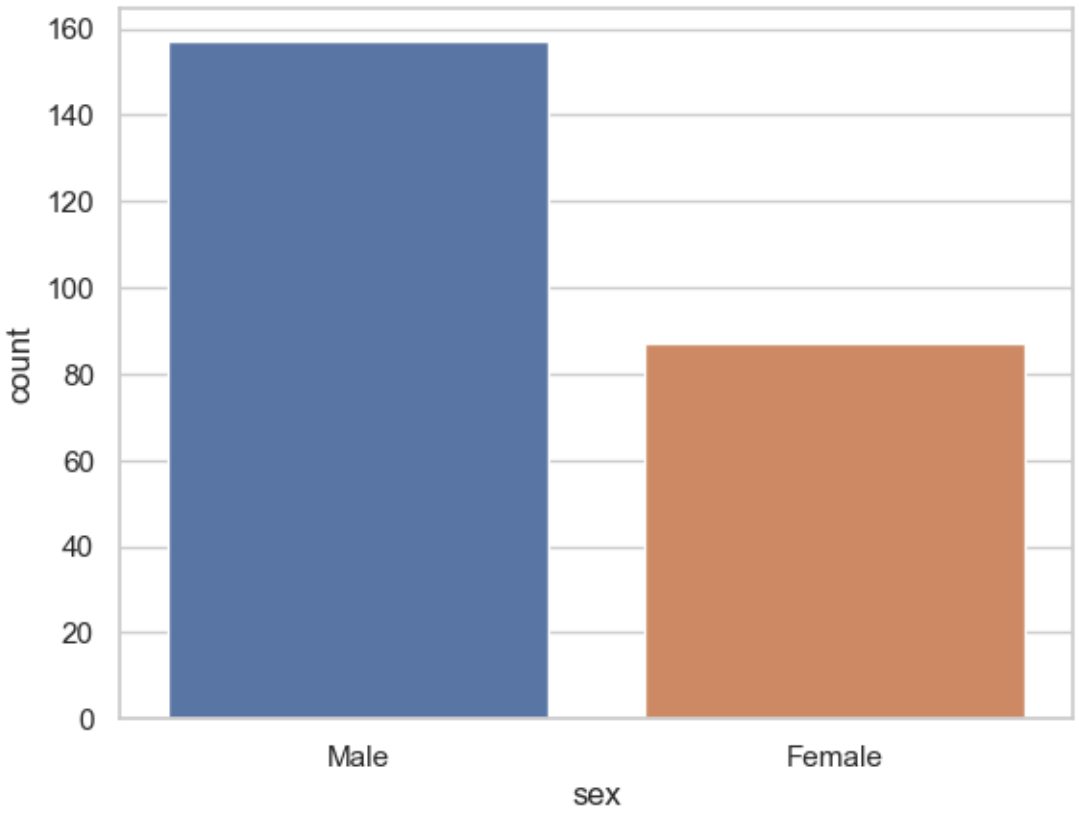
	name	age	city	salary
0	kumar	34	markapur	10000
1	mani	89	anantapur	20000
2	ravi	25	kurnool	70000
3	vasu	17	Guntur	30000
4	sai	67	prakasam	50000

[24]:
```
# ticks
tips = sns.load_dataset('tips')
sns.set_style('ticks')
sns.countplot(x ='sex', data = tips, palette = 'deep')
```

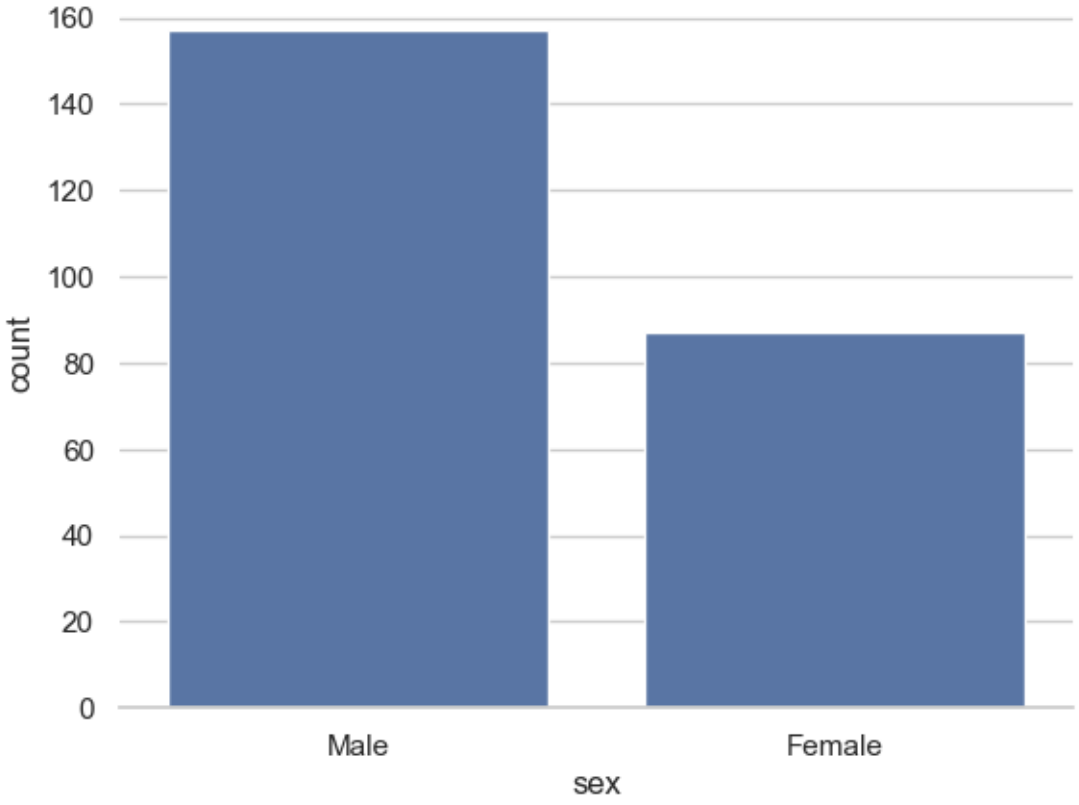
```
plt.show()
```



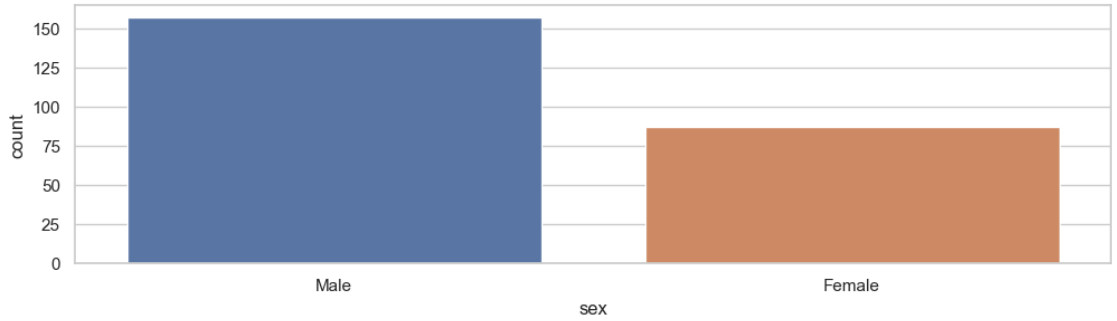
[25]:
```
# whitegrid
tips = sns.load_dataset('tips')
sns.set_style('whitegrid')
sns.countplot(x ='sex', data = tips, palette = 'deep')
plt.show()
```



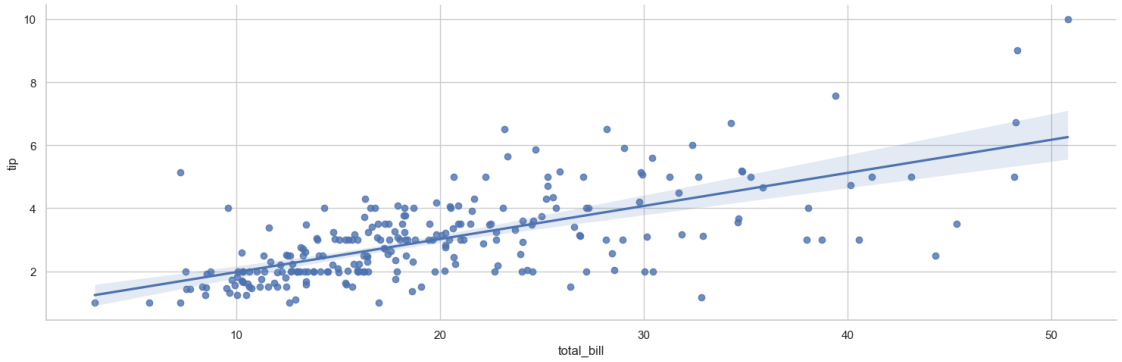
[26]:
```
# Removing Axes Spines
tips = sns.load_dataset('tips')
sns.set_style('whitegrid')
sns.countplot(x ='sex', data = tips)
sns.despine(left=True)
```



[27]:
```
# size and Aspect

tips=sns.load_dataset('tips')
plt.figure(figsize=(12,3))
sns.countplot(x='sex', data=tips, palette='deep')
plt.show()
```

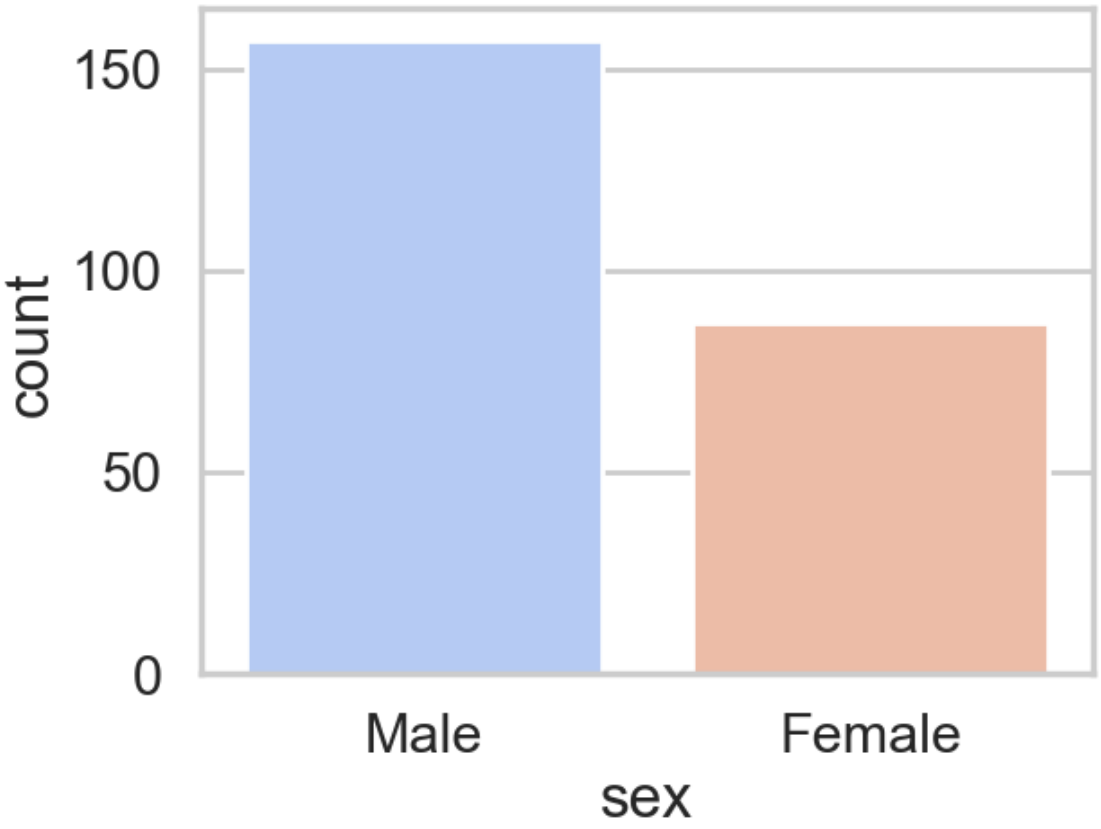


[28]:
```
# Grid type plot
tips= sns.load_dataset('tips')
sns.lmplot(x='total_bill', y='tip', aspect=3, data=tips)
plt.show()
```

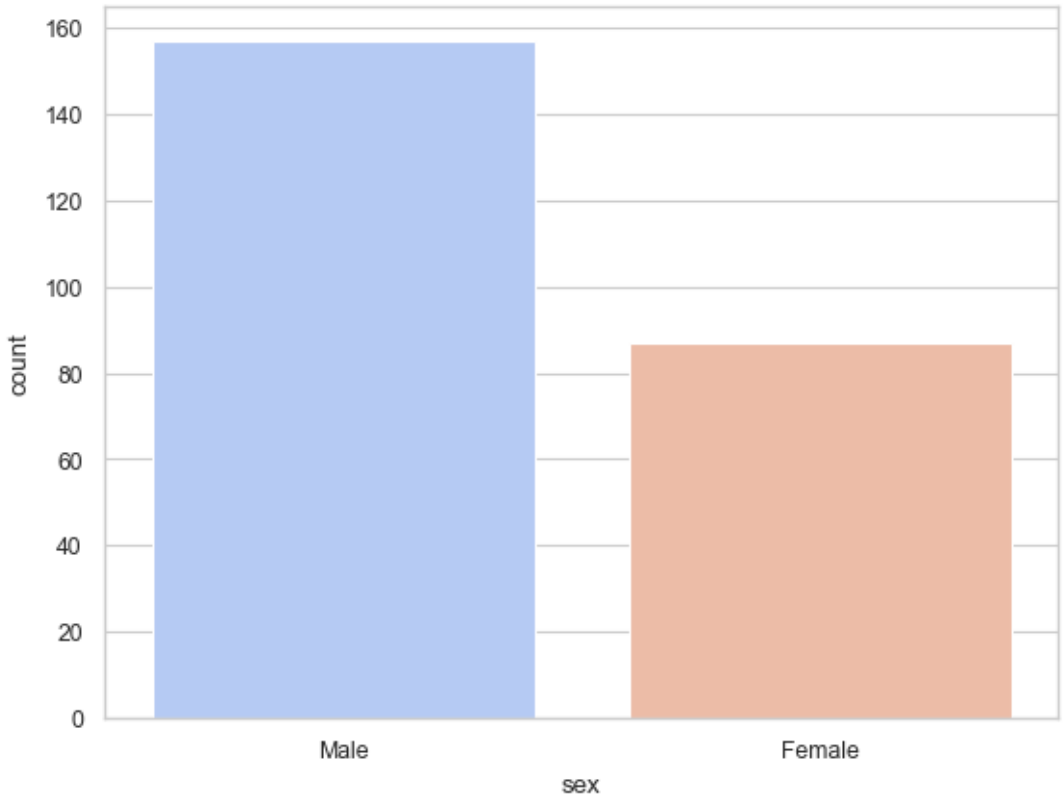


3.0.1 Scale and context

[30]:
```
# set_context()
# types------> poster, paper, notebook,talk

tips=sns.load_dataset('tips')
sns.set_context('poster',font_scale=1)
sns.countplot(x='sex', data=tips, palette='coolwarm')
plt.show()
```



[31]:
```
tips=sns.load_dataset('tips')
sns.set_context('paper',font_scale=1)
sns.countplot(x='sex', data=tips, palette='coolwarm')
plt.show()
```



3.0.2 –>color palette

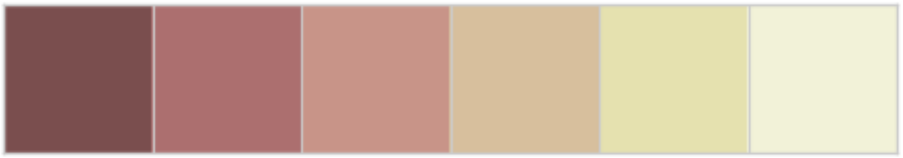
[33]:
```
# we are using color_palette(), which can be used for coloring the plot.
# using the palette we can generate the point with different colors.
'''syntax'''
#seaborn.color_palette(palette=None, n_colors=None, desat=None)
```

[33]: 'syntax'

[34]:
```
# types
# qualitative, Sequential, Diverging
from matplotlib import pyplot as plt
import seaborn as sns

current_color=sns.color_palette()
sns.palplot(current_color)
plt.show()
```

[35]:
```
#### sequential
```

[36]:
```
current_color=sns.color_palette()
sns.palplot(sns.color_palette("pink"))
plt.show()
```



[37]:
```
current_color=sns.color_palette()
sns.palplot(sns.color_palette("BrBG",7))
plt.show()
```

3.0.3 these are the possible value of the palette

[39]:
```
'''
'Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r', 'BuGn', 'BuGn_r',␣
↪'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r', 'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r',
'Greens', 'Greens_r', 'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges',␣
↪'Oranges_r',  'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1',␣
↪'Pastel1_r', 'Pastel2',
'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn', 'PuBuGn_r', 'PuBu_r', 'PuOr',␣
↪'PuOr_r', 'PuRd', 'PuRd_r', 'Purples', 'Purples_r', 'RdBu', 'RdBu_r',␣
↪'RdGy', 'RdGy_r',
```

```
'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r', 'Reds',␣
↪'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3', 'Set3_r', 'Spectral',␣
↪'Spectral_r', 'Wistia',
'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r', 'YlOrBr', 'YlOrBr_r',␣
↪'YlOrRd', 'YlOrRd_r', 'afmhot', 'afmhot_r', 'autumn', 'autumn_r', 'binary',␣
↪'binary_r',  'bone',
'bone_r', 'brg', 'brg_r', 'bwr', 'bwr_r', 'cividis', 'cividis_r', 'cool',␣
↪'cool_r',  'coolwarm', 'coolwarm_r', 'copper', 'copper_r', 'cubehelix',␣
↪'cubehelix_r', 'flag', 'flag_r',
'gist_earth', 'gist_earth_r', 'gist_gray', 'gist_gray_r', 'gist_heat',␣
↪'gist_heat_r', 'gist_ncar',  'gist_ncar_r', 'gist_rainbow',␣
↪'gist_rainbow_r', 'gist_stern', 'gist_stern_r', 'gist_yarg',
'gist_yarg_r', 'gnuplot', 'gnuplot2', 'gnuplot2_r', 'gnuplot_r', 'gray',␣
↪'gray_r', 'hot', 'hot_r',  'hsv', 'hsv_r', 'icefire', 'icefire_r',␣
↪'inferno', 'inferno_r', 'jet', 'jet_r', 'magma', 'magma_r',
'mako', 'mako_r', 'nipy_spectral', 'nipy_spectral_r', 'ocean', 'ocean_r',␣
↪'pink', 'pink_r',  'plasma', 'plasma_r', 'prism', 'prism_r', 'rainbow',␣
↪'rainbow_r', 'rocket', 'rocket_r',  'seismic', 'seismic_r',
'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r','tab20 ,␣
↪'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r', 'terrain',␣
↪'terrain_r', 'turbo',  'turbo_r', 'twilight', 'twilight_r',␣
↪'twilight_shifted',
'twilight_shifted_r', 'viridis',  'viridis_r', 'vlag', 'vlag_r', 'winter',␣
↪'winter_r' '''
```

[39]:
```
'\n'Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r', 'BuGn', 'BuGn_r',
'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r', 'Dark2', 'Dark2_r', 'GnBu',
'GnBu_r','\n'Greens', 'Greens_r', 'Greys', 'Greys_r', 'OrRd', 'OrRd_r',
'Oranges', 'Oranges_r',  'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1',
'Pastel1_r', 'Pastel2','\n'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn',
'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r', 'Purples',
'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r', \n'RdPu', 'RdPu_r', 'RdYlBu',
'RdYlBu_r', 'RdYlGn', 'RdYlGn_r', 'Reds', 'Reds_r', 'Set1', 'Set1_r', 'Set2',
'Set2_r', 'Set3', 'Set3_r', 'Spectral', 'Spectral_r', 'Wistia', \n'Wistia_r',
'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r', 'YlOrBr', 'YlOrBr_r', 'YlOrRd',
'YlOrRd_r', 'afmhot', 'afmhot_r', 'autumn', 'autumn_r', 'binary', 'binary_r',
'bone', \n'bone_r', 'brg', 'brg_r', 'bwr', 'bwr_r', 'cividis', 'cividis_r',
'cool', 'cool_r',  'coolwarm', 'coolwarm_r', 'copper', 'copper_r', 'cubehelix',
'cubehelix_r', 'flag', 'flag_r', \n'gist_earth', 'gist_earth_r', 'gist_gray',
'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_ncar',  'gist_ncar_r',
'gist_rainbow', 'gist_rainbow_r', 'gist_stern', 'gist_stern_r', 'gist_yarg',
\n'gist_yarg_r', 'gnuplot', 'gnuplot2', 'gnuplot2_r', 'gnuplot_r', 'gray',
'gray_r', 'hot', 'hot_r',  'hsv', 'hsv_r', 'icefire', 'icefire_r', 'inferno',
'inferno_r', 'jet', 'jet_r', 'magma', 'magma_r',  \n'mako', 'mako_r',
'nipy_spectral', 'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
'plasma', 'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r', 'rocket',
```

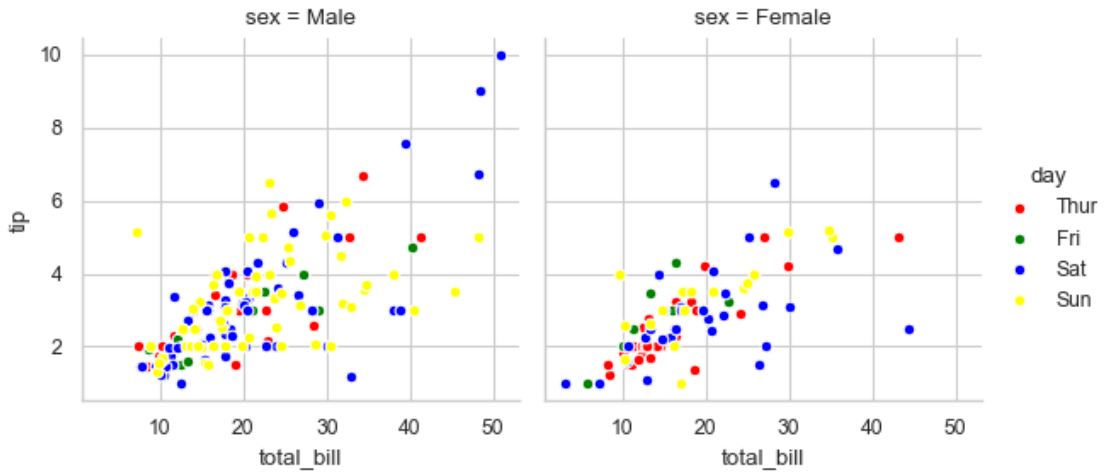
```
'rocket_r',  'seismic', 'seismic_r',\n'spring', 'spring_r', 'summer',
'summer_r', 'tab10', 'tab10_r','tab20 , 'tab20_r', 'tab20b', 'tab20b_r',
'tab20c', 'tab20c_r', 'terrain', 'terrain_r', 'turbo',  'turbo_r', 'twilight',
'twilight_r', 'twilight_shifted', \n'twilight_shifted_r', 'viridis',
'viridis_r', 'vlag', 'vlag_r', 'winter', 'winter_r' '
```

[40]:
```
sns.palplot(sns.color_palette("gnuplot2",5))
```

[41]:
```
# set the color

color=['red','green','blue','yellow','white']
sns.set_palette(color)
sns.palplot(sns.color_palette())
```

4 Seaborn.FacetGrid() method

[43]:
```
data=sns.load_dataset('tips')
#facetgrid
graph=sns.FacetGrid(data, col='sex' ,hue='day')
graph.map(plt.scatter, 'total_bill', 'tip', edgecolor='w').add_legend()
plt.show()
```



[44]:
```
print(data.head())
data.shape
```

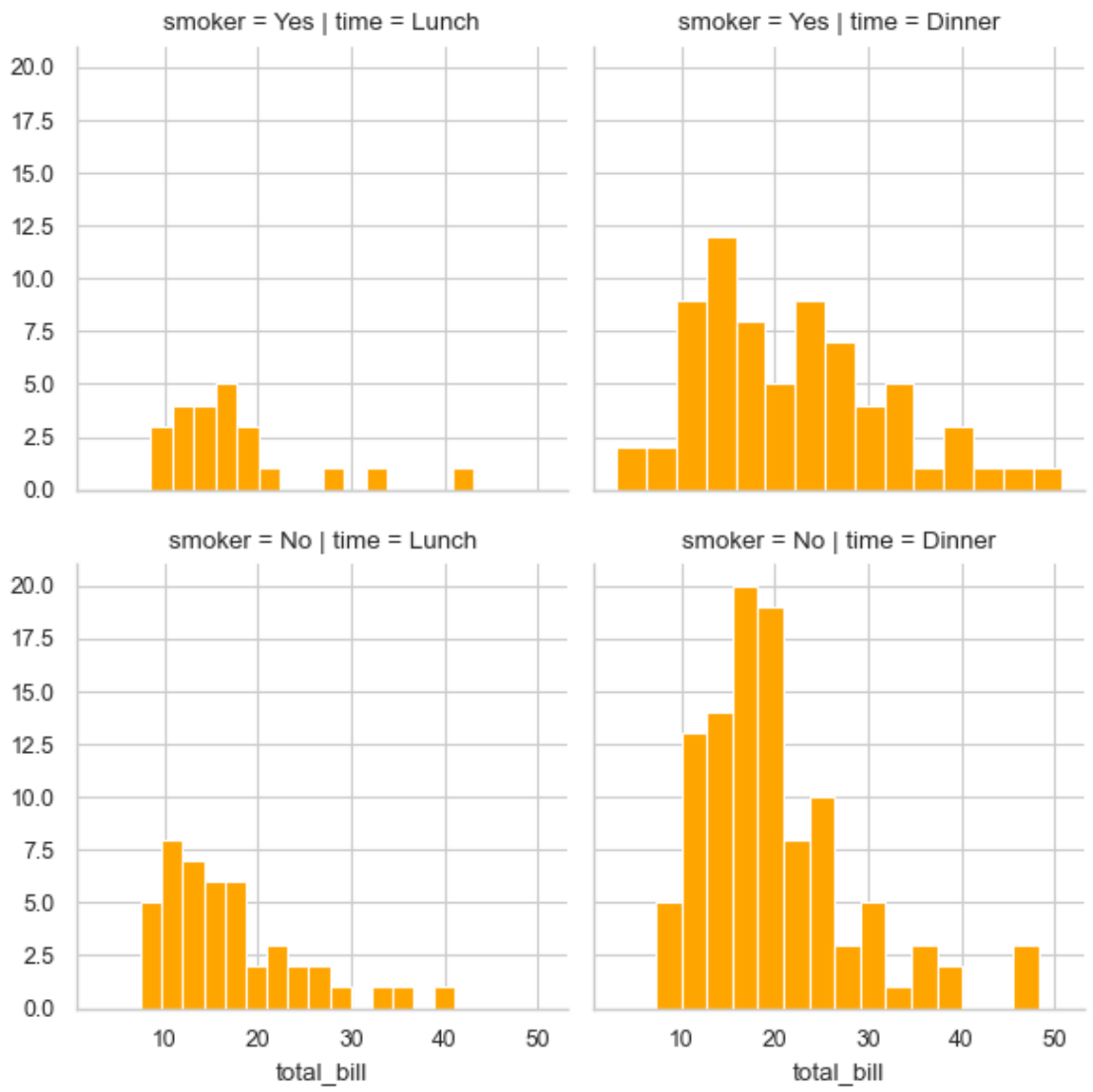
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female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

[44]:
```
(244, 7)
```

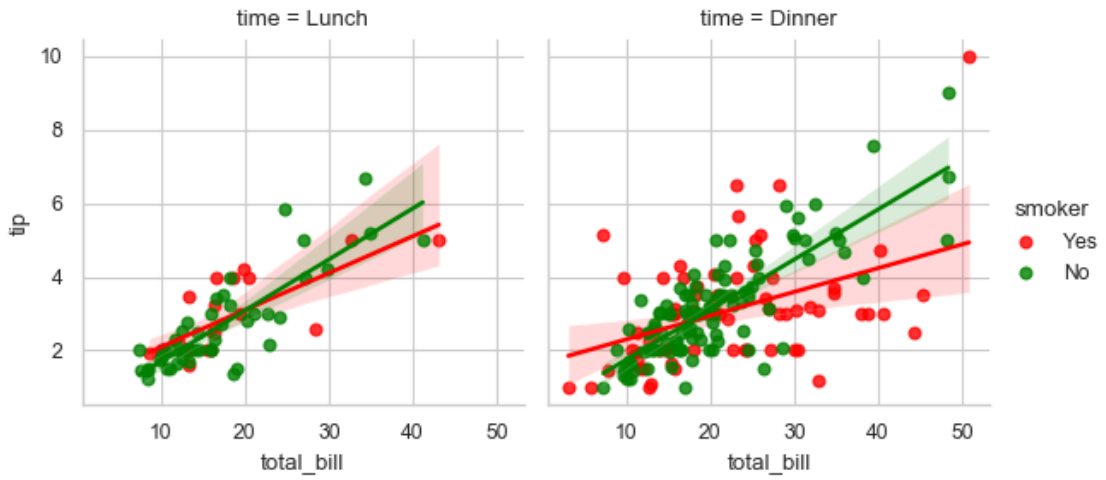
[45]:
```
data['time'].unique()
```

[45]:
```
['Dinner', 'Lunch']
Categories (2, object): ['Lunch', 'Dinner']
```

[46]:
```
data=sns.load_dataset('tips')
graph=sns.FacetGrid(data, row='smoker',col='time')
graph.map(plt.hist, 'total_bill', bins=15, color='orange')
plt.show()
```



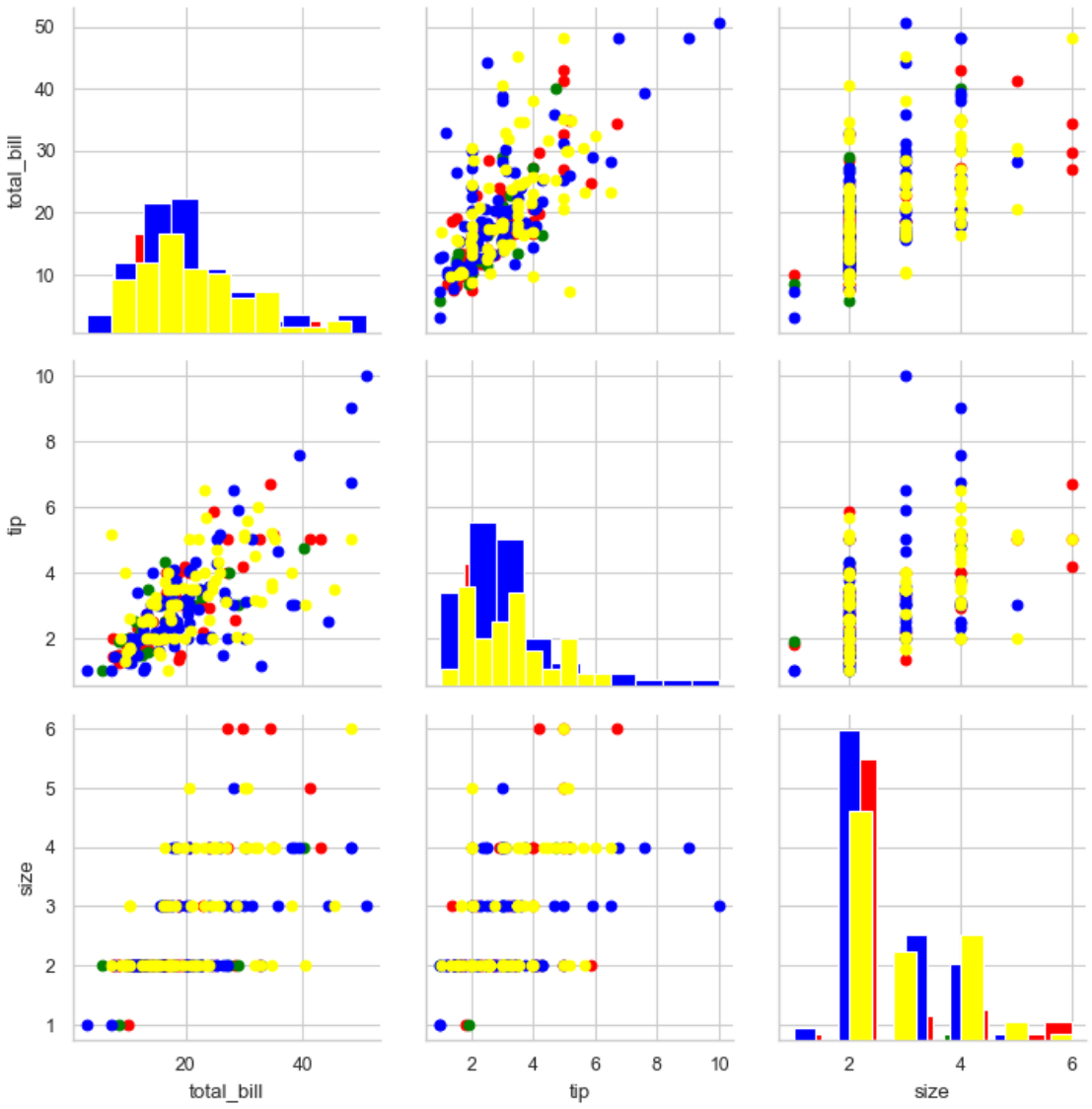
[47]:
```
graph = sns.FacetGrid(data, col ='time', hue ='smoker')
graph.map(sns.regplot, "total_bill", "tip").add_legend()
plt.show()
```



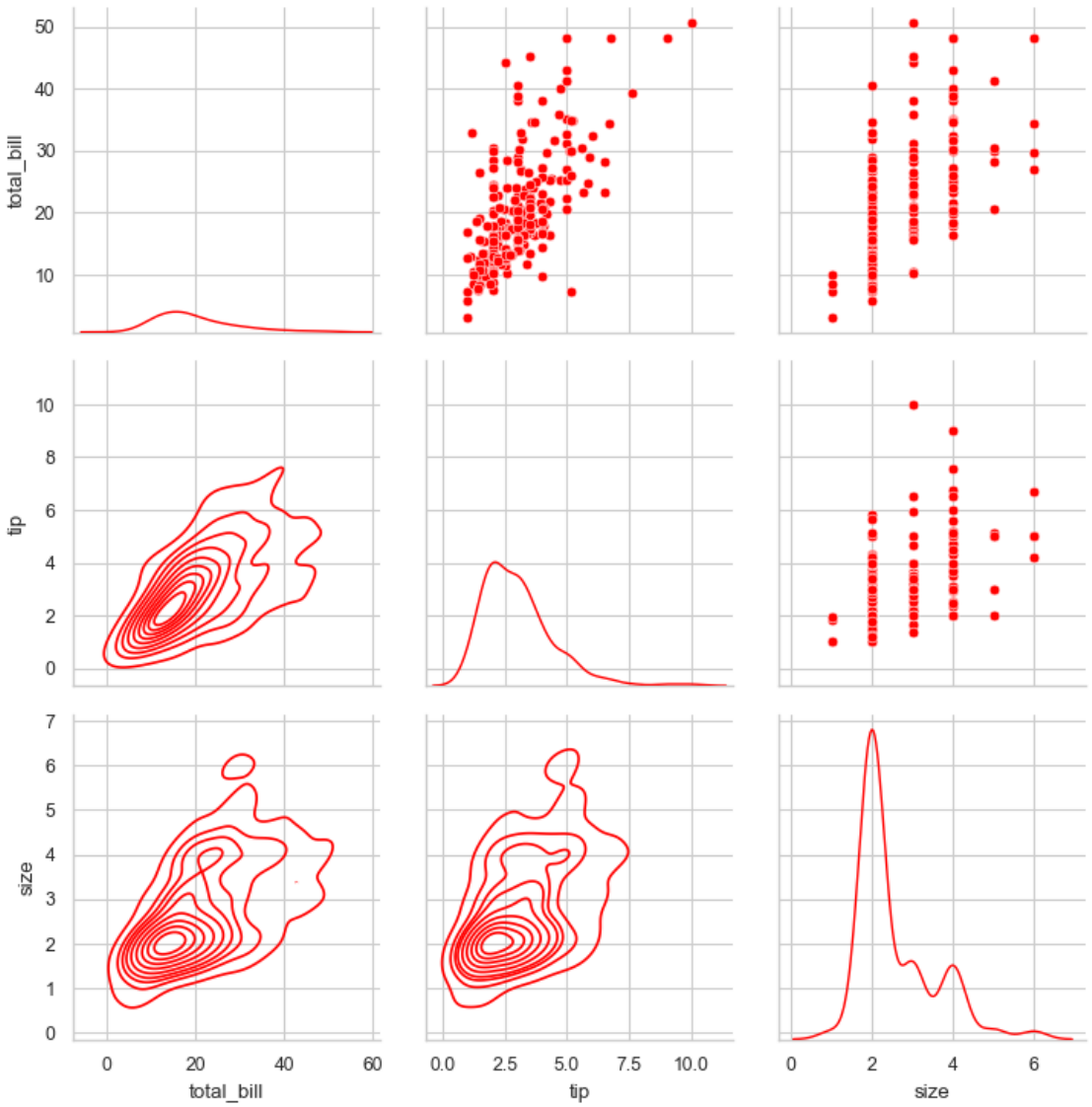
4.0.1 seaborn.pairGride() method

[49]:
```
#syntax: sns.PairGrid(data, hue,palette, vars, dropna)
```

[50]:
```
graph=sns.PairGrid(data, hue='day')
graph=graph.map_diag(plt.hist)
grapg=graph.map_offdiag(plt.scatter)
plt.show()
```



[51]:
```
graph=sns.PairGrid(data)
graph=graph.map_upper(sns.scatterplot)
graph=graph.map_lower(sns.kdeplot)
graph=graph.map_diag(sns.kdeplot, lw=1)
plt.show()
```



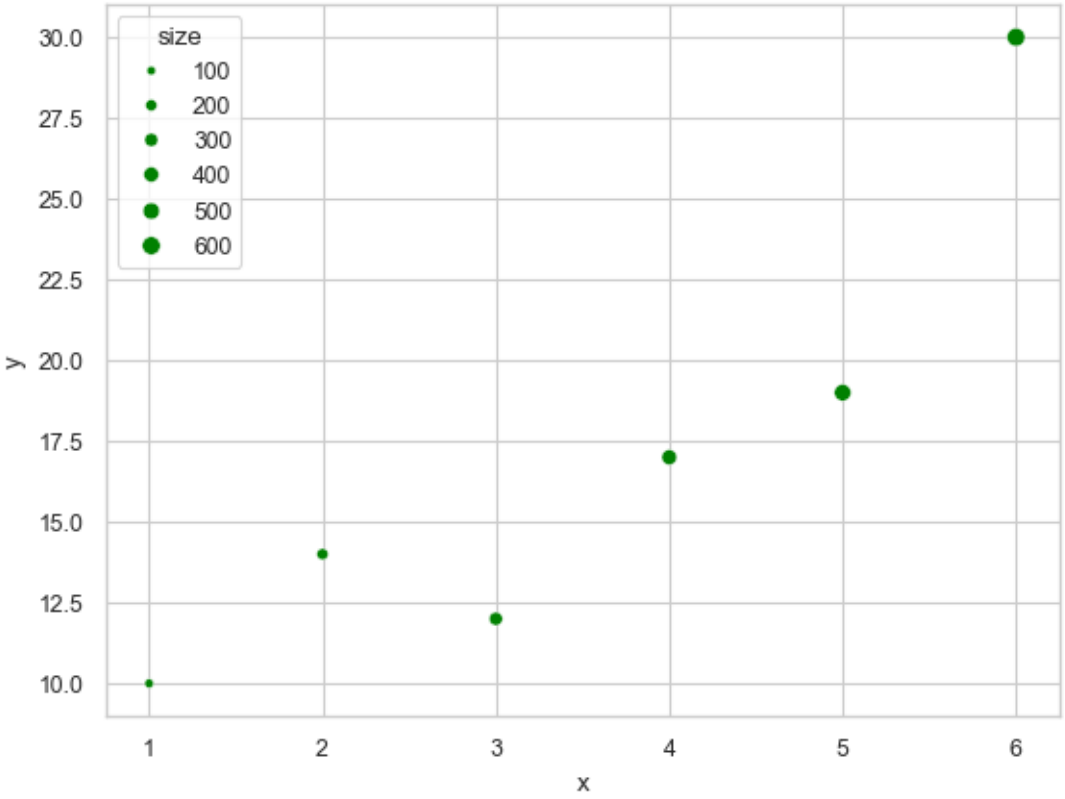
5 Scatterplots

[53]:
```
# syntax: sns.scatterplot(data, x, y, size)
# Method 1: Using the size Parameter in scatterplot()
# Method 2: Using Matplotlib's scatter Function with the s Parameter
# Method 3: Using s Parameter in scatterplot() with Fixed Size
# Method 4: Using the scatter_kws Parameter in Seaborn's relplot()
```

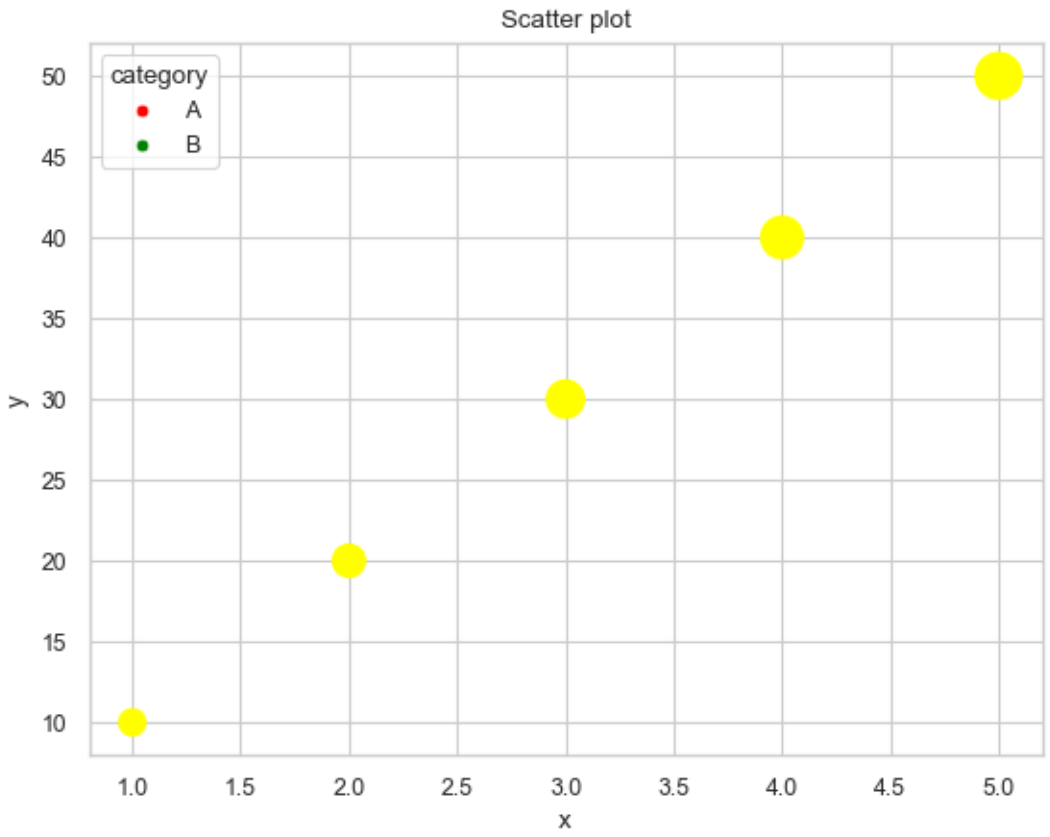
[54]:
```
# Method 1: Using the size Parameter in scatterplot()

data={'x':[1,2,3,4,5,6],
```

```
      'y':[10,14,12,17,19,30],
      'size':[100, 200, 300, 400, 500, 600]}
sns.scatterplot(x='x', y='y', size='size', data=data, color='green')
```

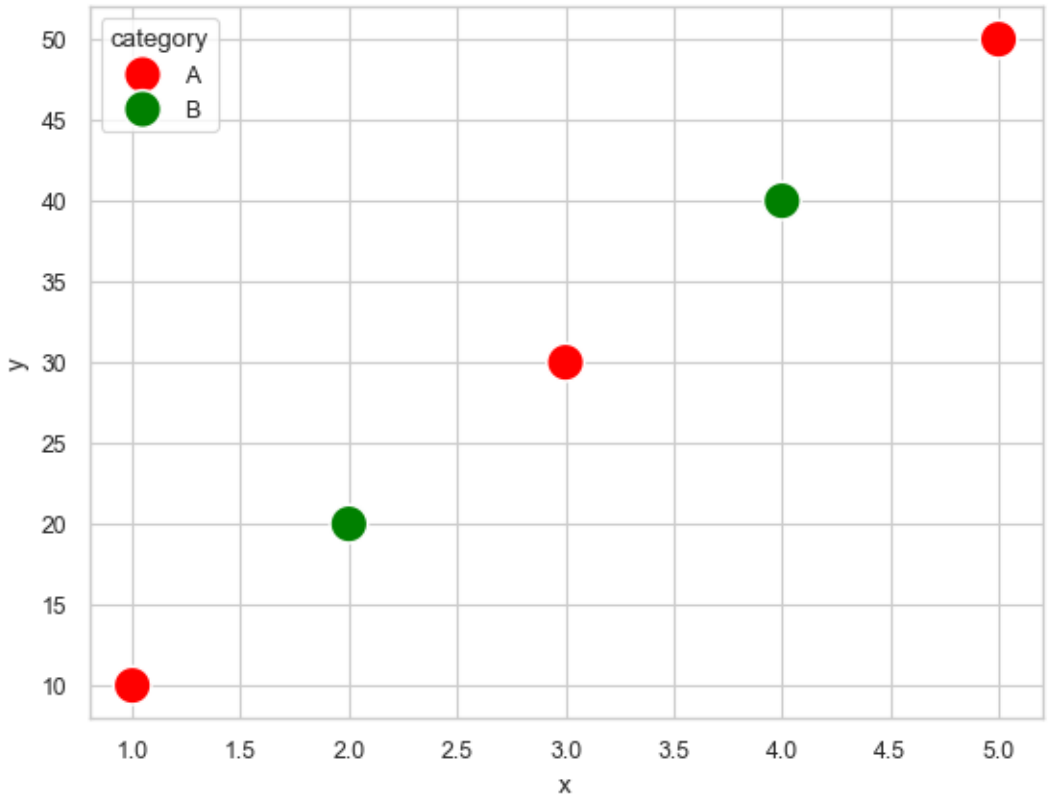
[54]: <Axes: xlabel='x', ylabel='y'>



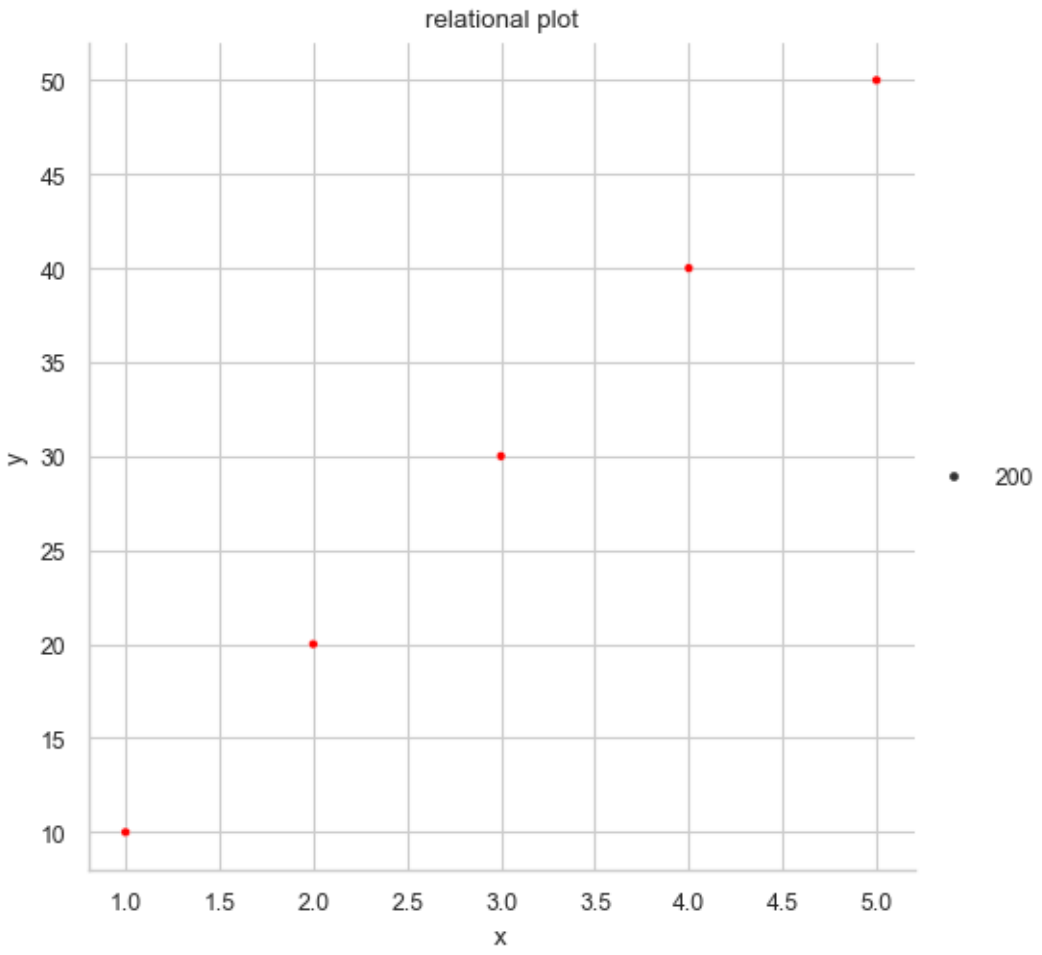
[55]:
```
# Method 2: Using Matplotlib's scatter Function with the s Parameter
data1 = {
    'x': [1, 2, 3, 4, 5],
    'y': [10, 20, 30, 40, 50],
    'category': ['A', 'B', 'A', 'B', 'A'],
    'size': [100, 150, 200, 250, 300]
}
sns.scatterplot(data=data1, x='x', y='y', hue='category')
plt.scatter(data1['x'], data1['y'], s=data1['size'], marker='o', color='yellow')
plt.title('Scatter plot')
plt.show()
```



[56]:
```
# Method 3: Using s Parameter in scatterplot() with Fixed Size
data1 = {
    'x': [1, 2, 3, 4, 5],
    'y': [10, 20, 30, 40, 50],
    'category': ['A', 'B', 'A', 'B', 'A']

}
sns.scatterplot(x='x', y='y', s=200, data=data1, hue='category')
plt.show()
```



[57]:
```
# Method 4: Using the scatter_kws Parameter in Seaborn's relplot()
data1 = {
    'x': [1, 2, 3, 4, 5],
    'y': [10, 20, 30, 40, 50],
    'category': ['A', 'B', 'A', 'B', 'A']
    }
sns.relplot(x='x', y='y', data=data1, kind='scatter', size=200)
plt.title('relational plot')
plt.show()
```



5.0.1 visualizing Relationship between variables with scatter plots

[59]:
```
import seaborn as sns
sns.set(style='darkgrid')
data=pd.read_csv(r"C:\Users\Windows\Downloads\Walmart.xls")
data.head()
```

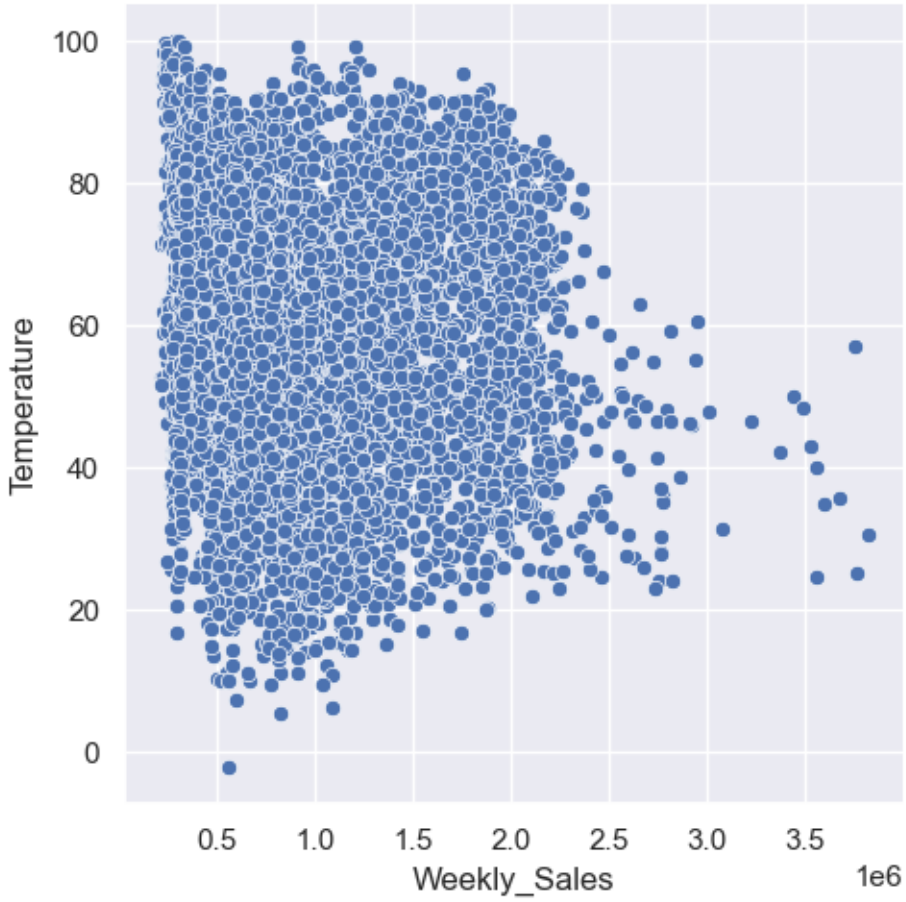
[59]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	\
0	1	05-02-2010	1643690.90	0	42.31	2.572	
1	1	12-02-2010	1641957.44	1	38.51	2.548	
2	1	19-02-2010	1611968.17	0	39.93	2.514	
3	1	26-02-2010	1409727.59	0	46.63	2.561	
4	1	05-03-2010	1554806.68	0	46.50	2.625	

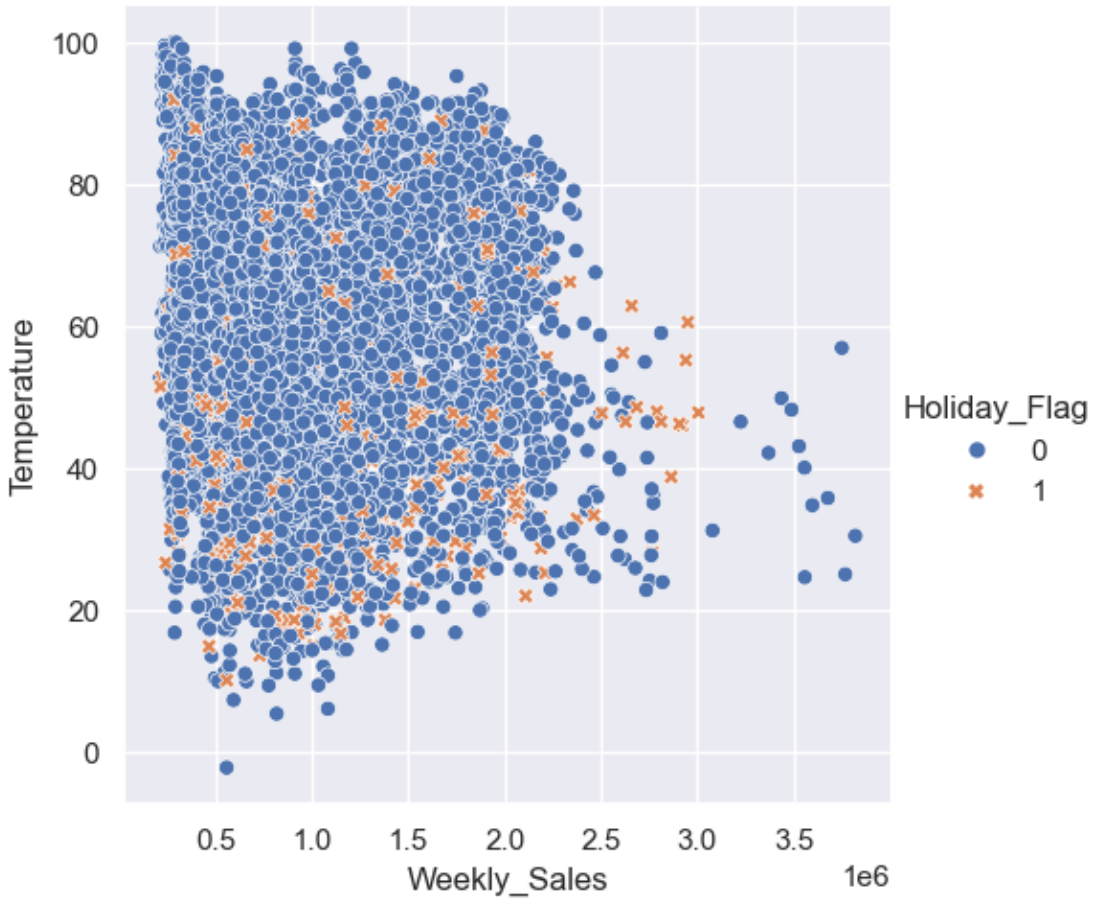
	CPI	Unemployment
0	211.096358	8.106
1	211.242170	8.106

```
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

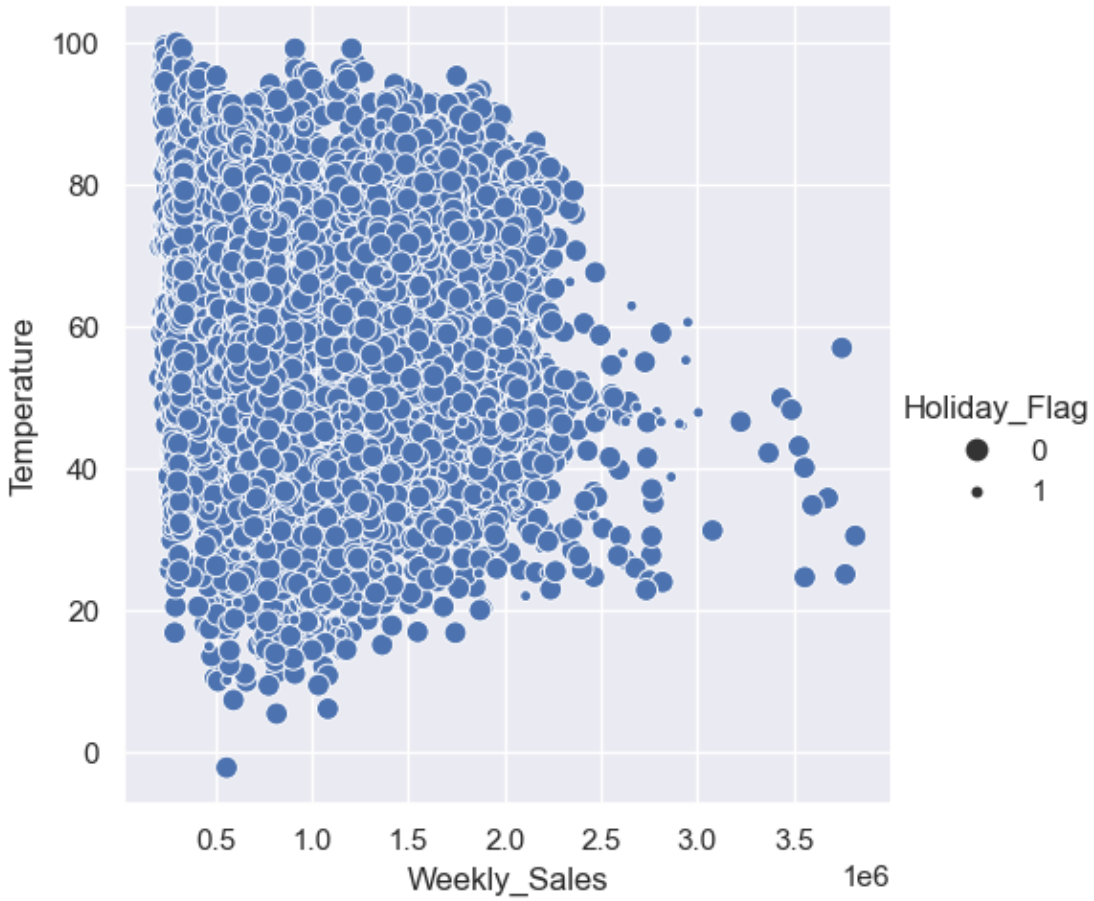
[60]:
```
sns.set(style='darkgrid')
sns.relplot(x="Weekly_Sales", y="Temperature", data=data)
plt.show()
```



[61]:
```
sns.relplot(x="Weekly_Sales", y="Temperature", data=data, hue="Holiday_Flag",␣
  ↪style="Holiday_Flag")
plt.show()
```

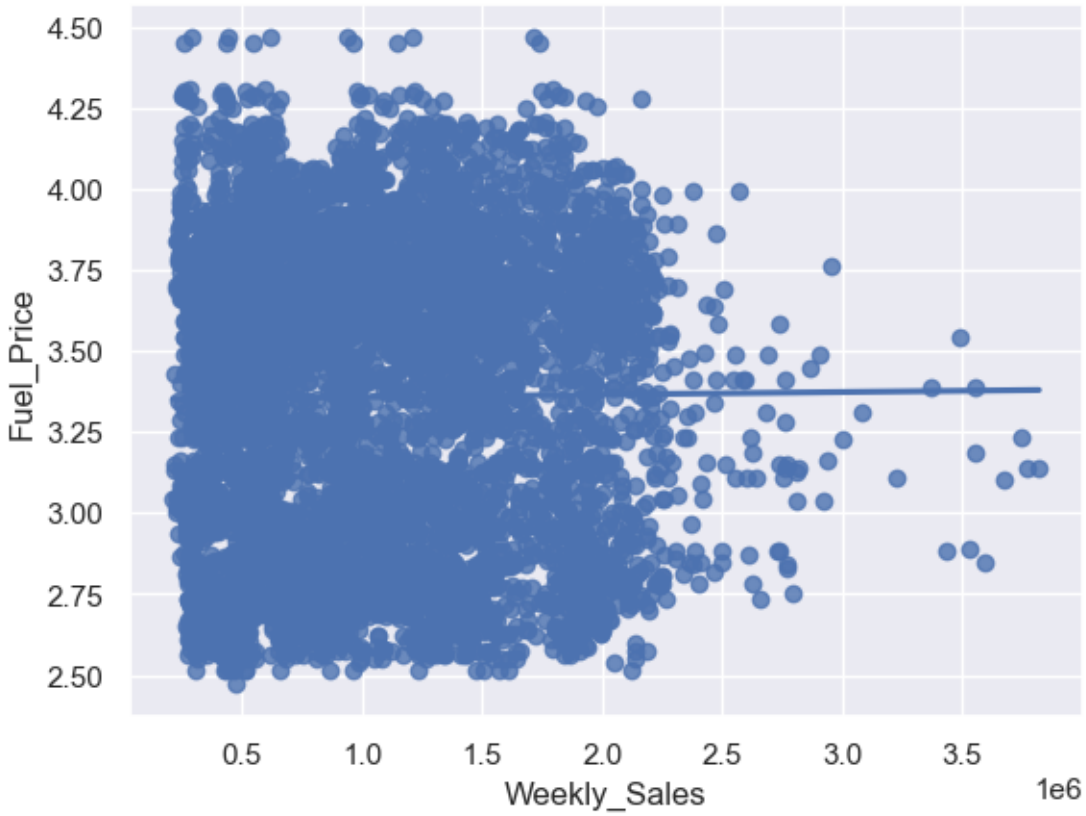


[62]:
```
sns.relplot(x="Weekly_Sales", y="Temperature", data=data,size="Holiday_Flag")
plt.show()
```

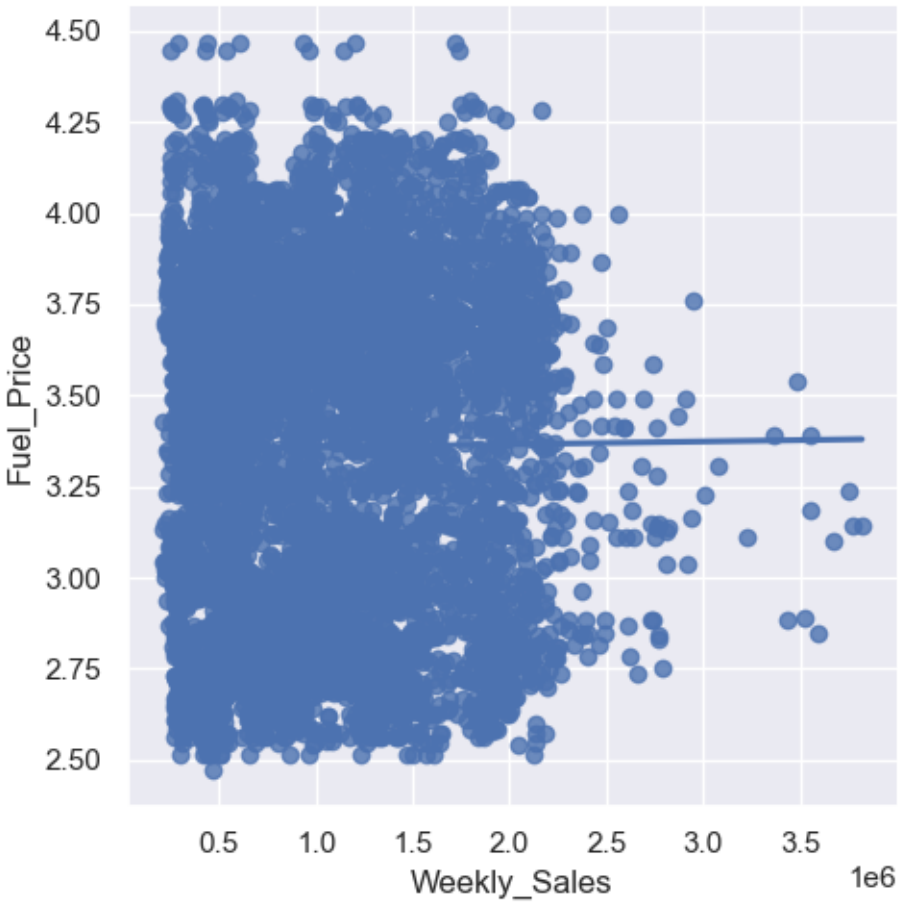


scatter with regplot and lmplot

[64]:
```
sns.regplot(x="Weekly_Sales", y="Fuel_Price", ci=None, data=data)
plt.show()
```



[65]:
```
sns.lmplot(x="Weekly_Sales", y="Fuel_Price", ci=None, data=data)
plt.show()
```

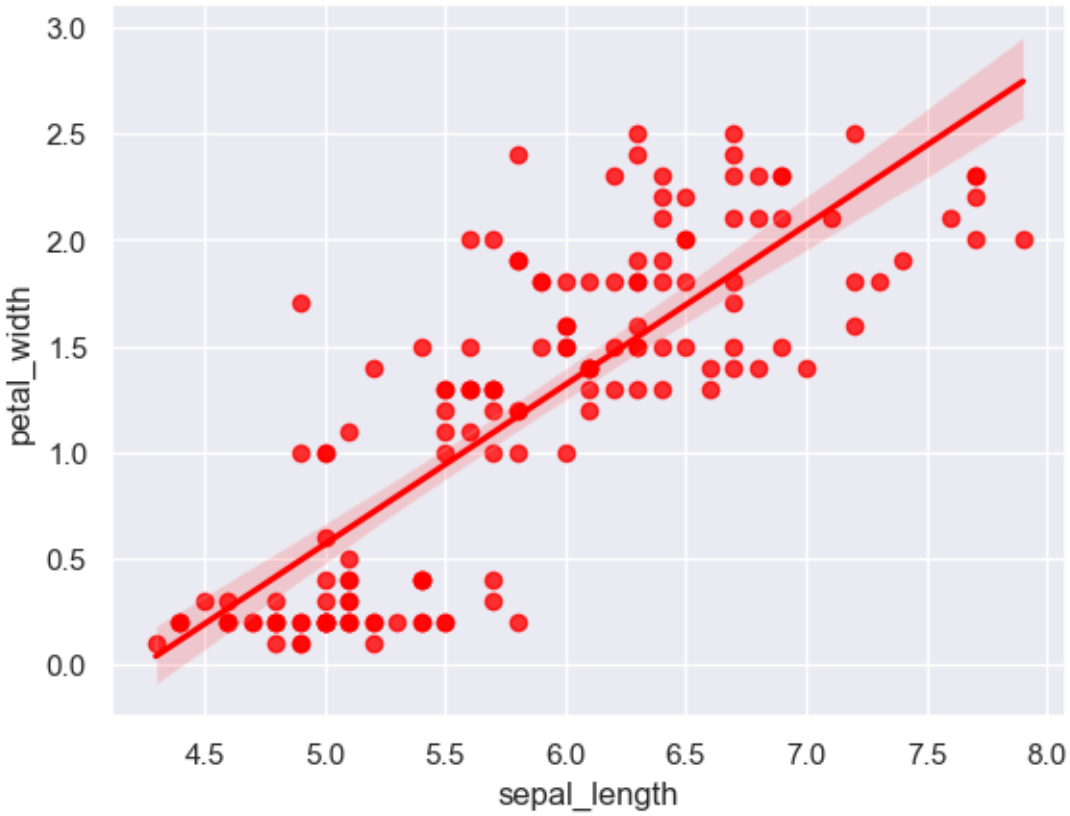


[66]:
```
dataset=sns.load_dataset("iris")
dataset.head()
```

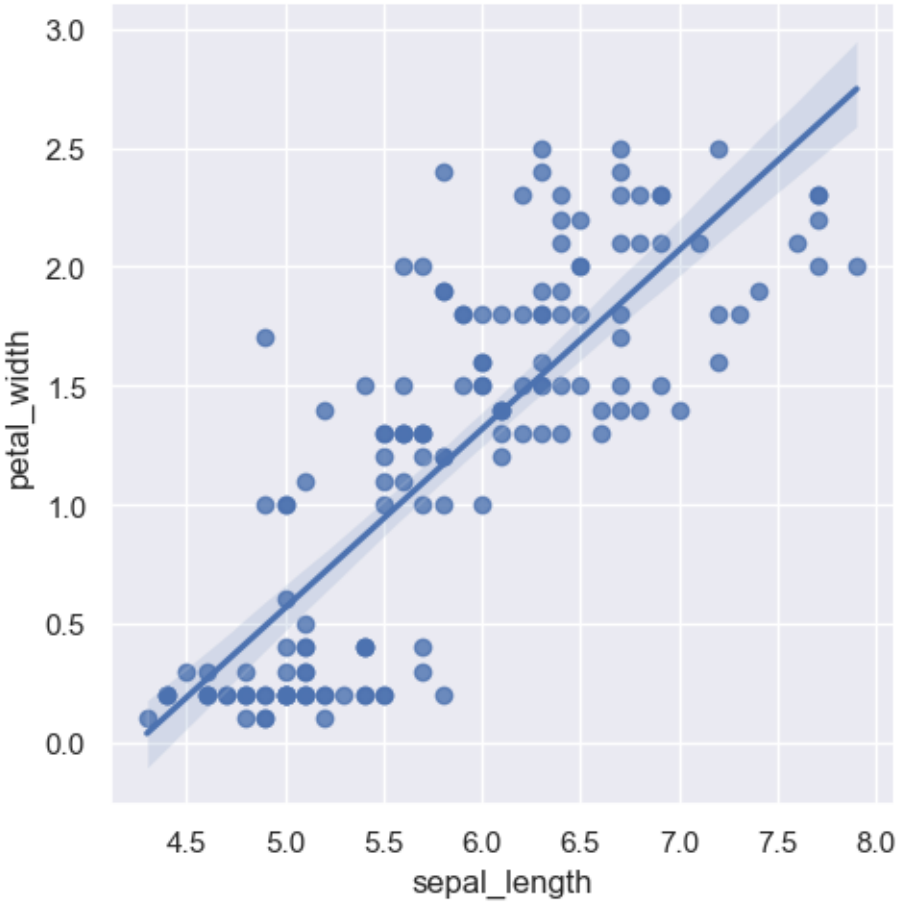
[66]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[67]:
```
sns.regplot(x="sepal_length", y="petal_width", data=dataset, color='red')
plt.show()
```

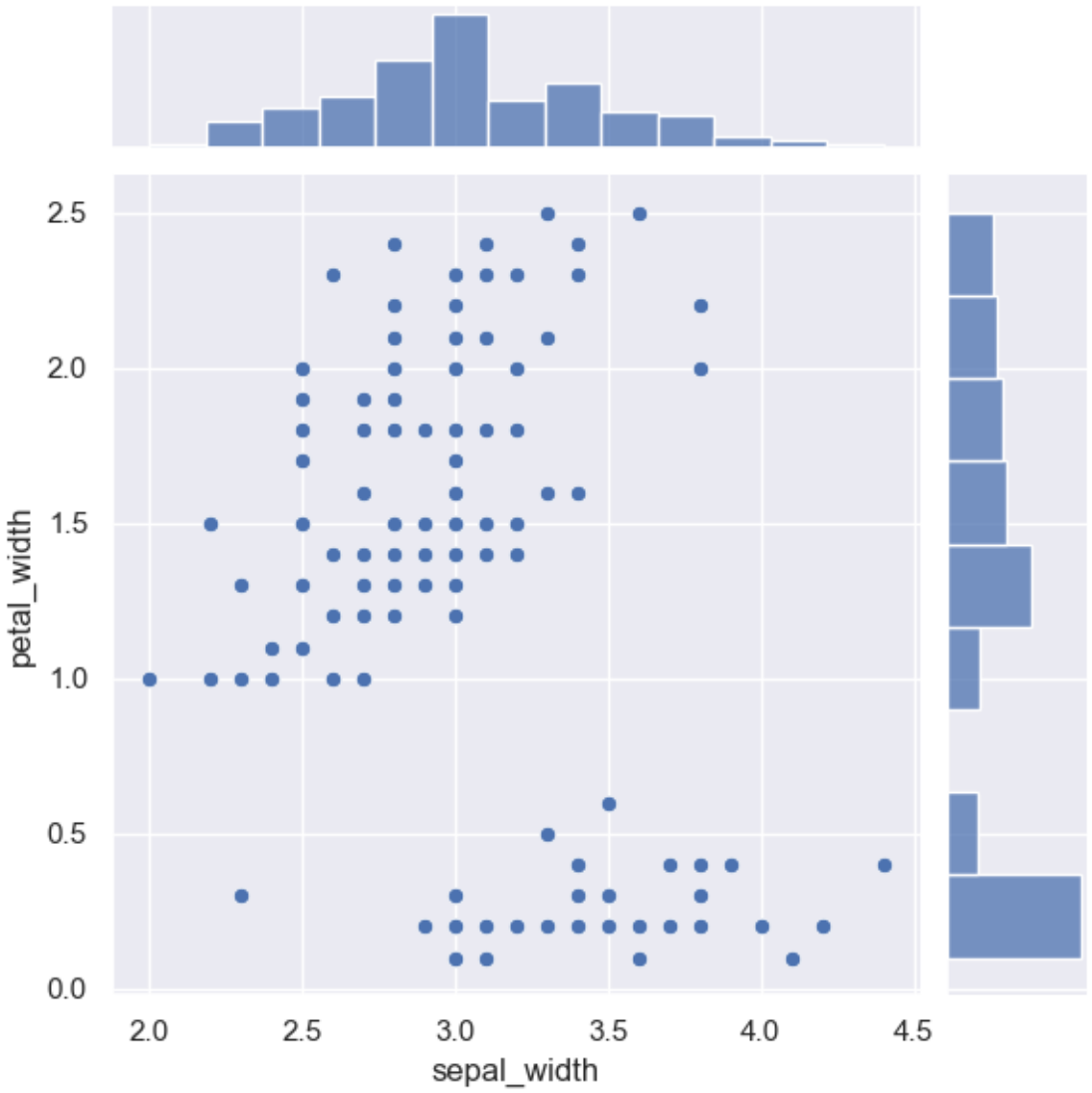


[68]:
```
sns.lmplot(x="sepal_length", y="petal_width", data=dataset)
plt.show()
```

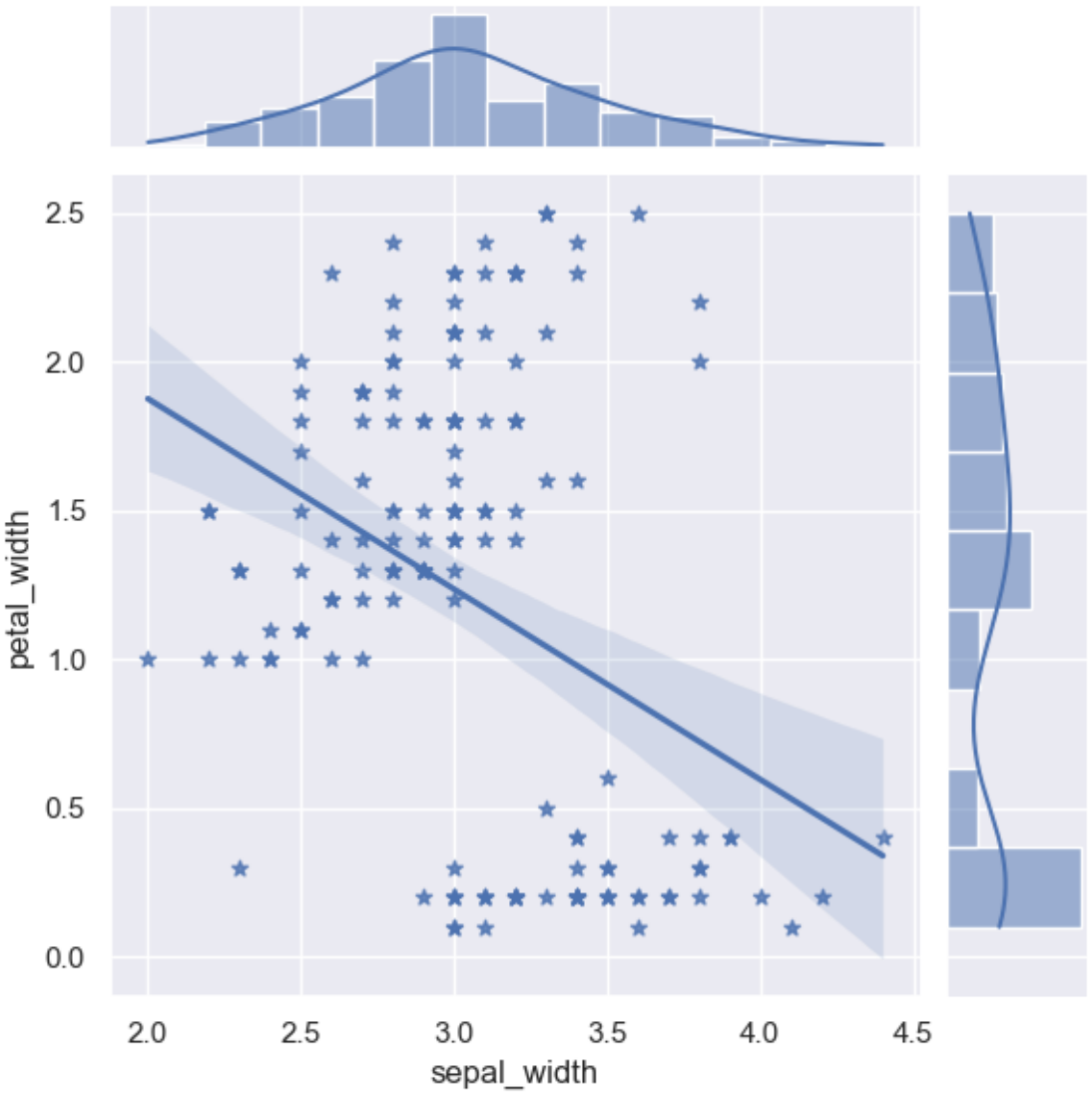


[69]:
```
#### Marginal Histogrmas in pyhton
```

[70]:
```
sns.jointplot(data=dataset, x="sepal_width", y="petal_width")
plt.show()
```

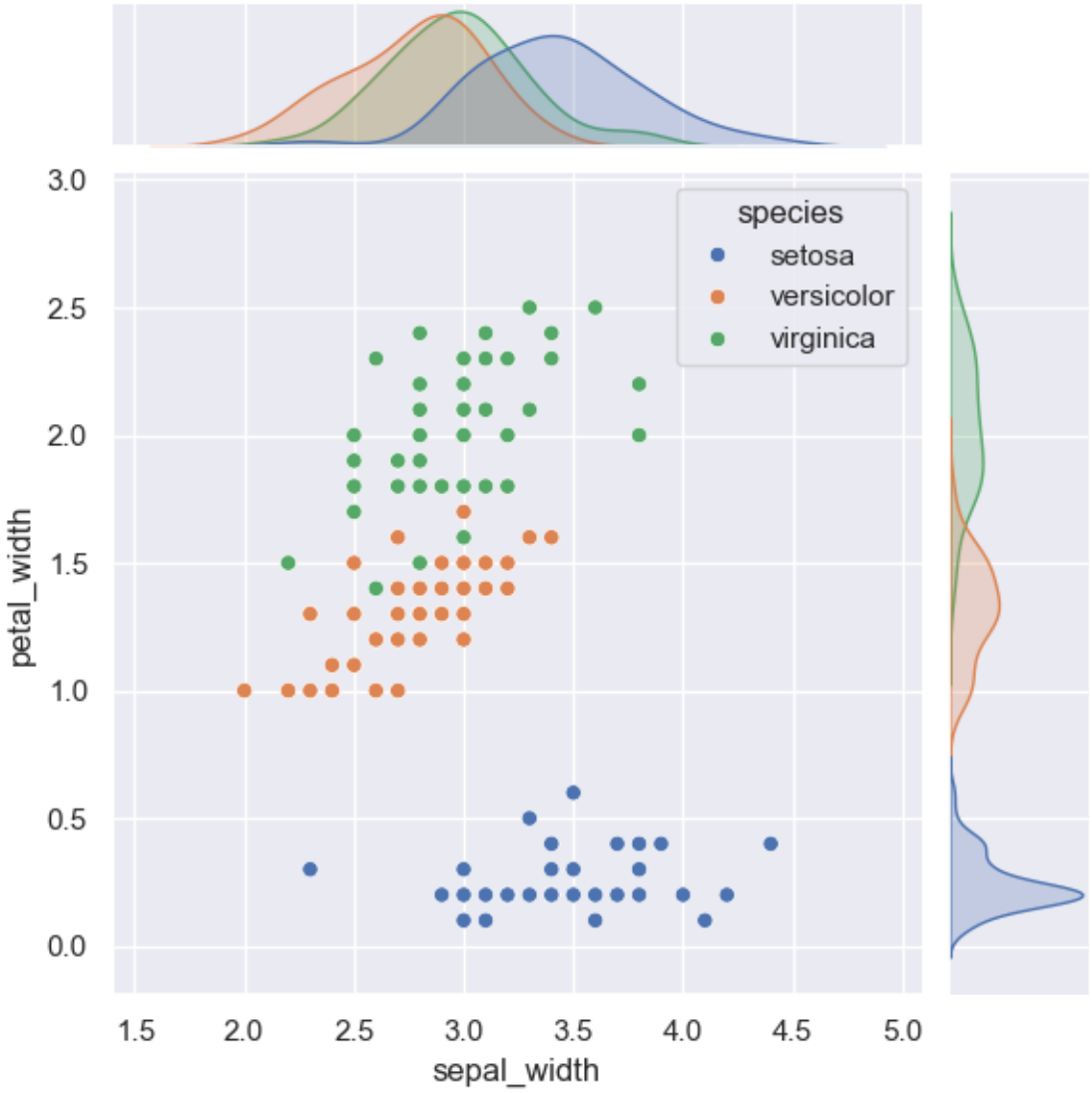


[71]:
```
# here "*" is used as a marker for scatterplot
sns.jointplot(data=dataset, x="sepal_width", y="petal_width" ,kind="reg",␣
↪marker="*")
plt.show()
```



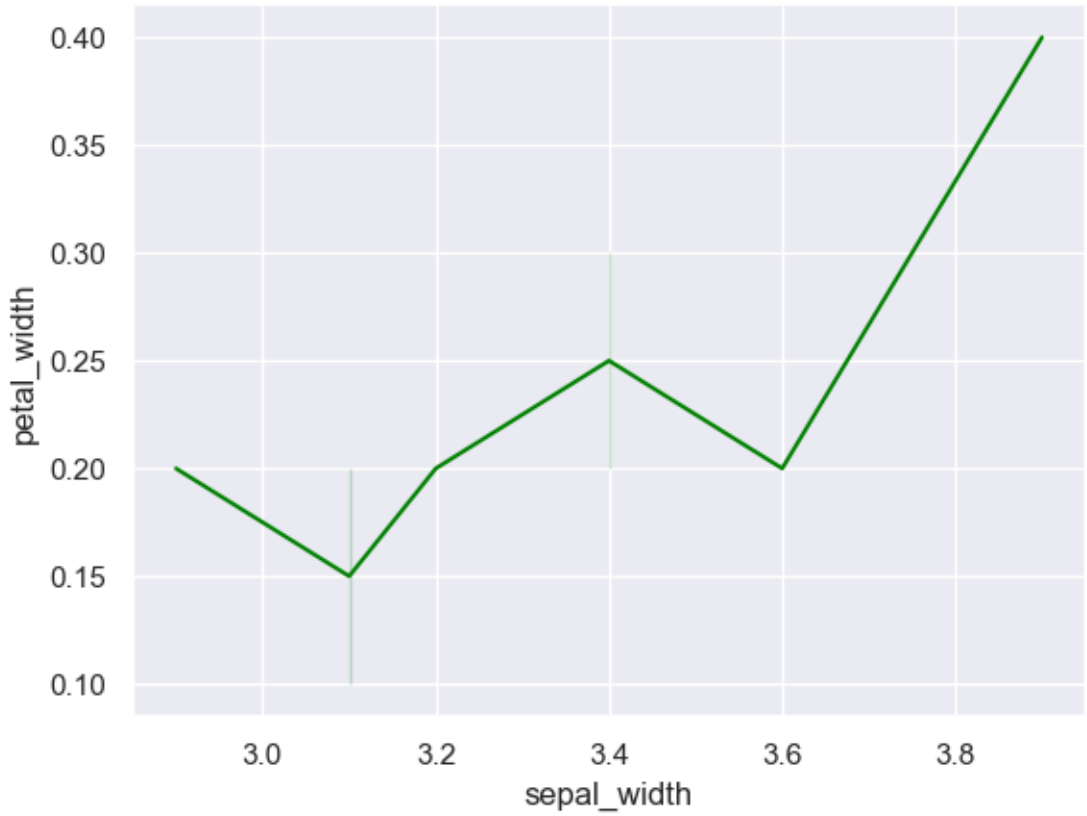
[72]:
```
sns.jointplot(data=dataset, x="sepal_width", y="petal_width", hue="species")
```

[72]:
```
<seaborn.axisgrid.JointGrid at 0x1d3b7c34410>
```



6 Line plot

[74]:
```
dataset=dataset.iloc[2:10, :]
sns.lineplot( x="sepal_width", y="petal_width", data=dataset, color='green')
plt.show()
```



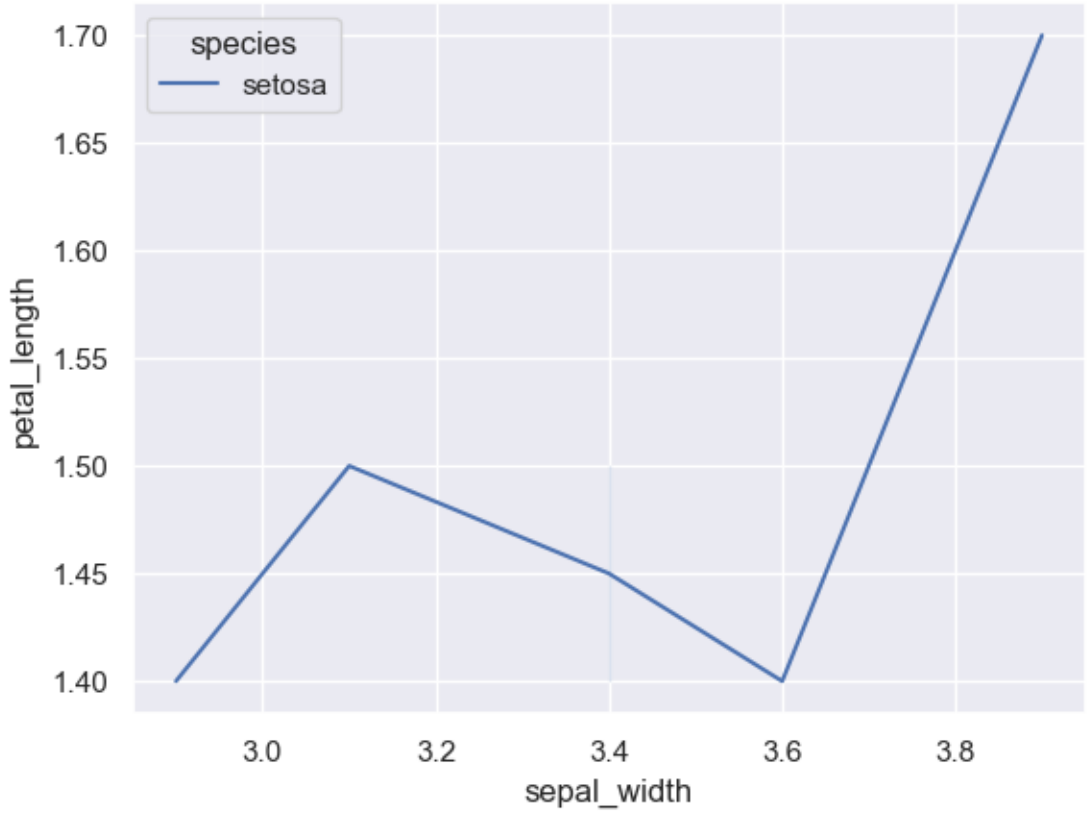
[75]:
```
# syntax: set()
# Attributes:
# Context: plotting context parameters
# Style: defines style
# Palette: sets a color palette
# Font
# Font_scale: sets font size
# Color_codes: If set True then palette is activated, short hand notations for␣
 ↪colors can be remapped from the palette.
# rc: parameter to over-ride the above parameters
```

[76]:
```
dataset.head()
```

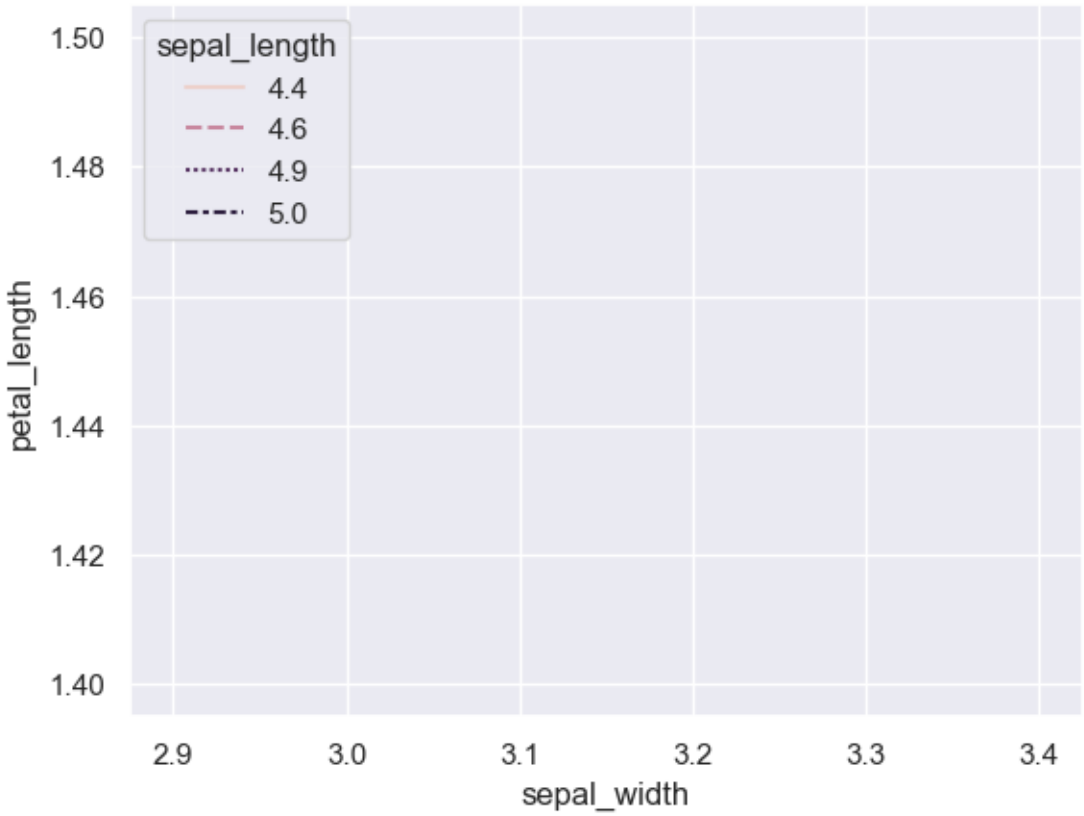
[76]:

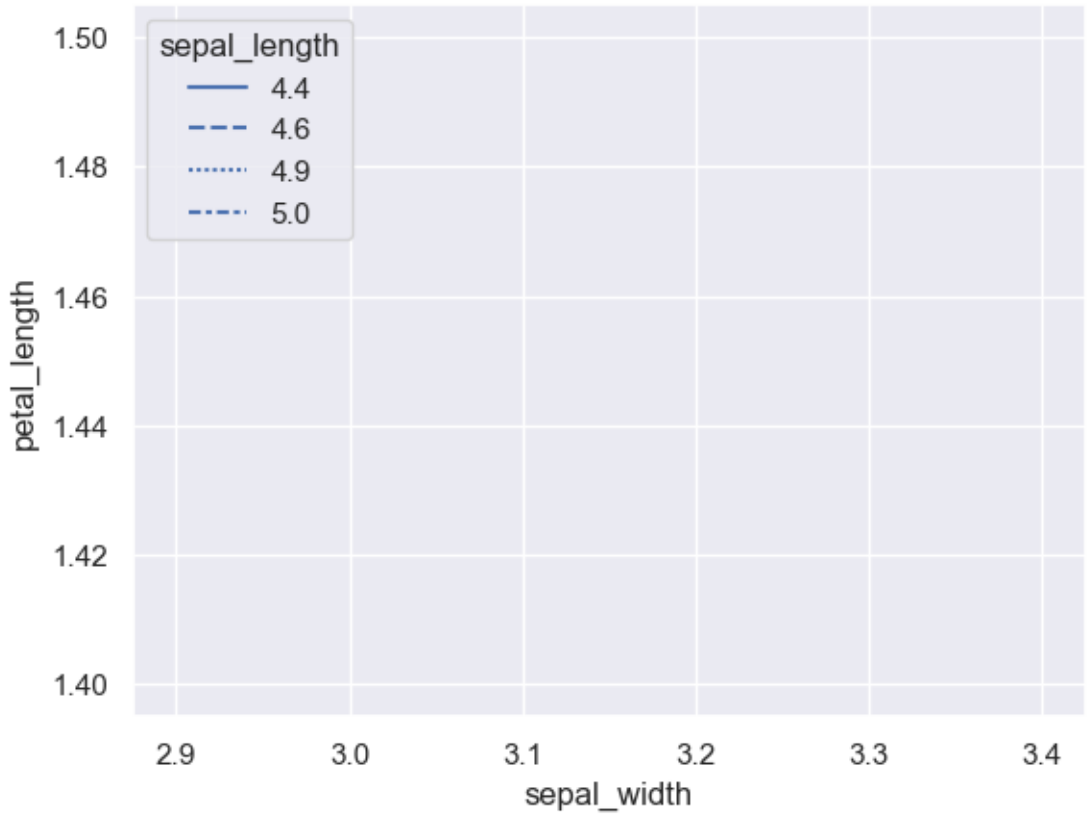
	sepal_length	sepal_width	petal_length	petal_width	species
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa

[77]:
```
dataset=dataset.iloc[2:10, :]
sns.lineplot(x="sepal_width", y="petal_length", data=dataset, hue="species")
sns.set(style="darkgrid")
plt.show()
```

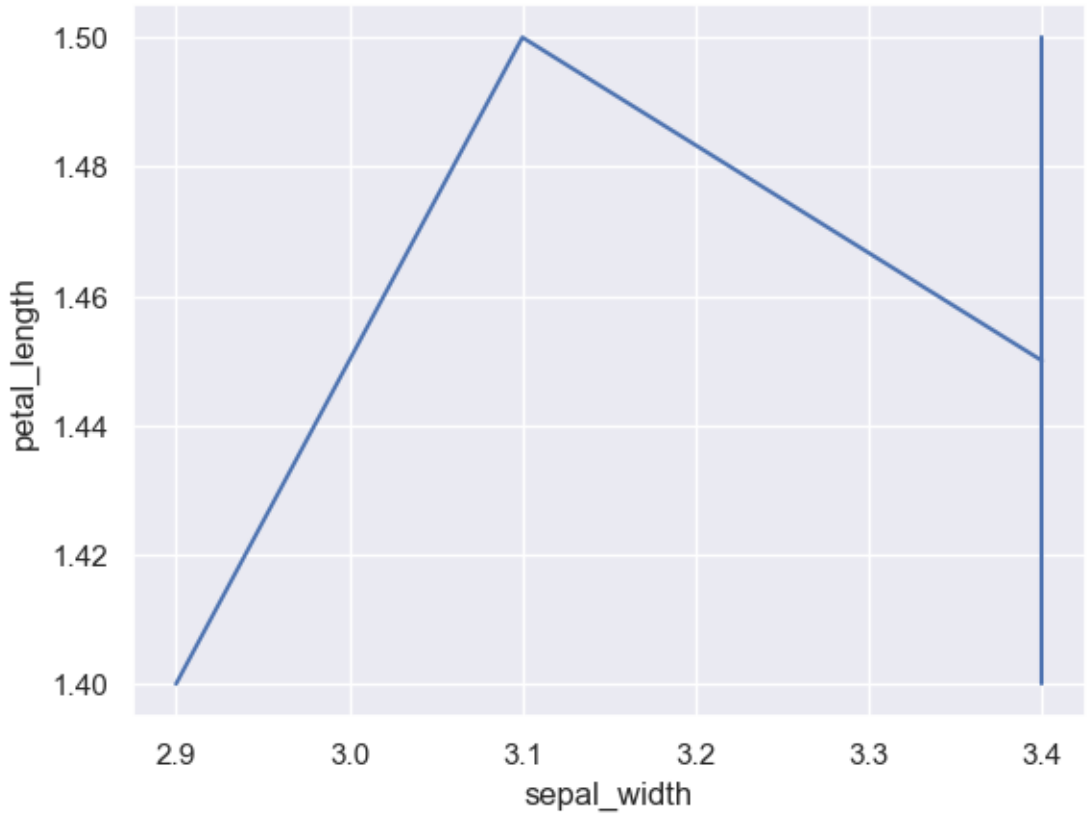


[78]:
```
dataset=dataset.iloc[2:10, :]

sns.lineplot(x="sepal_width", y="petal_length", data=dataset,␣
 ↪hue="sepal_length", style="sepal_length")
plt.show()

sns.lineplot(x="sepal_width", y="petal_length", data=dataset,␣
 ↪style="sepal_length")
plt.show()
```

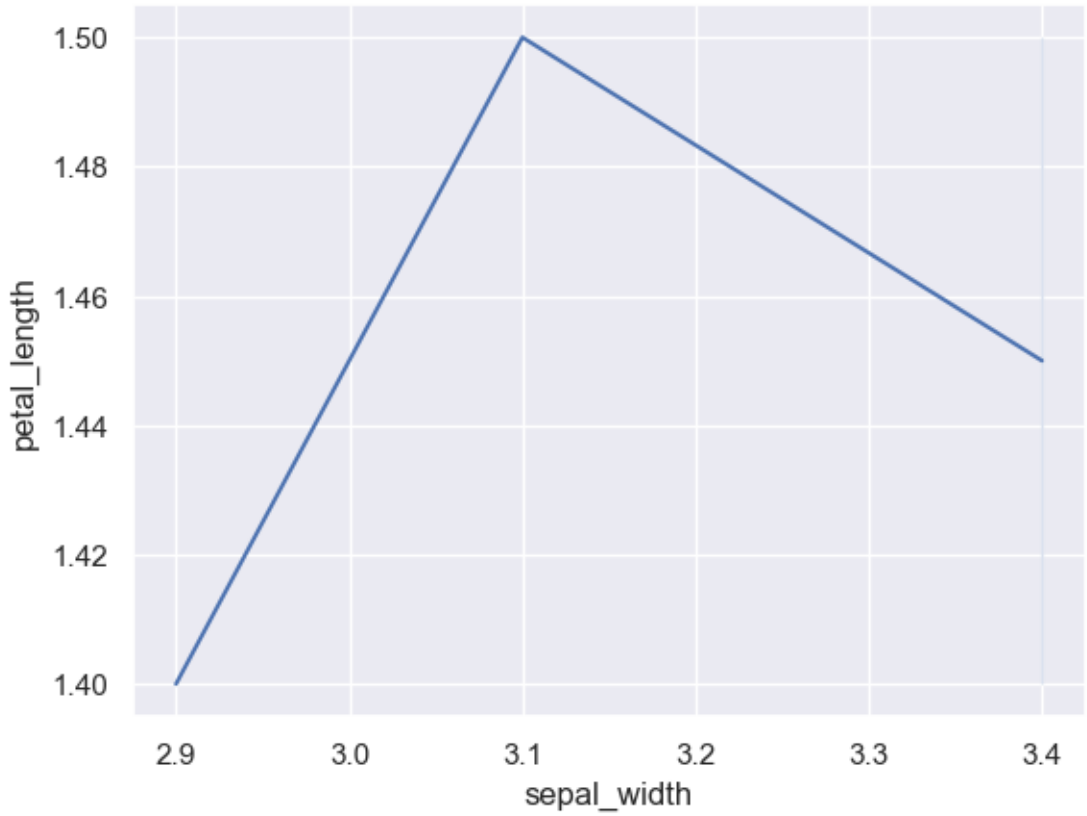

sepal_length
4.4
4.6
4.9
5.0
1.50
1.48
1.46
1.44
1.42
1.40
petal_length
2.9
3.0
3.1
3.2
3.3
3.4
sepal_width



[79]:
```
sns.lineplot(x="sepal_width", y="petal_length", data=dataset, err_style="bars")
plt.show()
```



[80]:
```
sns.lineplot(x="sepal_width", y="petal_length", data=dataset,palette="pastel")
plt.show()
```



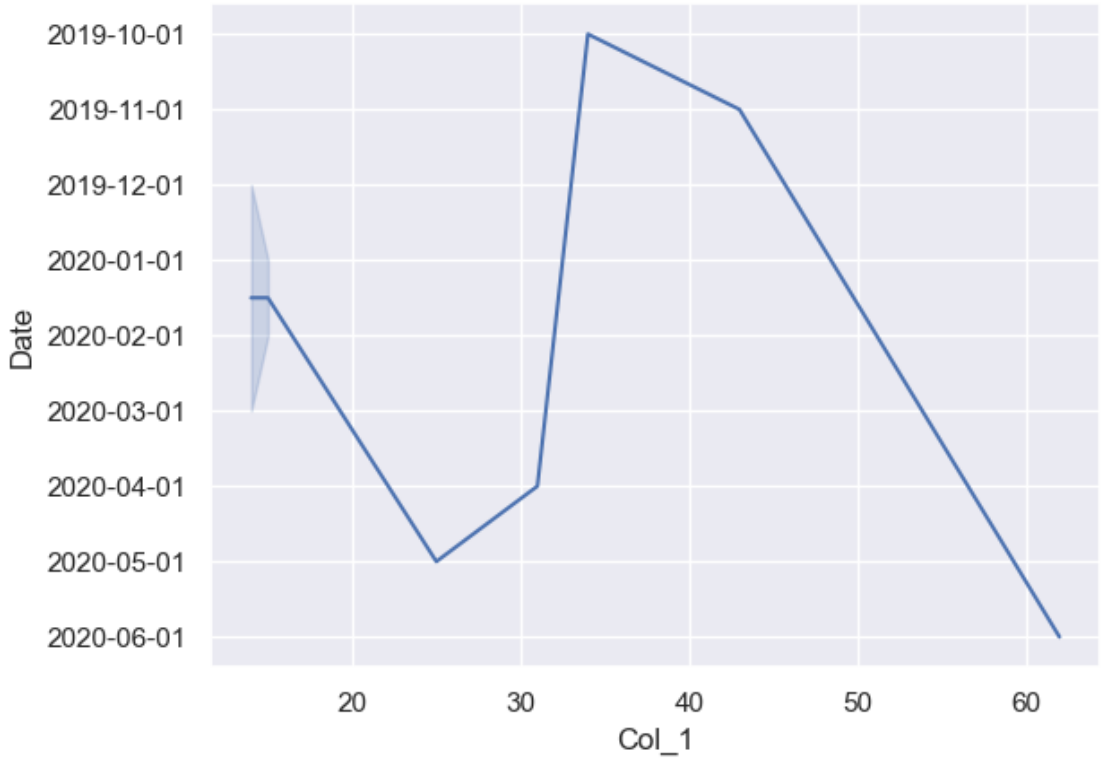
6.0.1 –> Time series plot

[82]:
```
# creating data
df = pd.DataFrame({'Date': ['2019-10-01', '2019-11-01',
                            '2019-12-01','2020-01-01',
                            '2020-02-01', '2020-03-01',
                            '2020-04-01', '2020-05-01',
                            '2020-06-01'],

                   'Col_1': [34, 43, 14, 15,
                             15, 14, 31, 25, 62],

                   'Col_2': [52, 66, 78, 15, 15,
                             5, 25, 25, 86],

                   'Col_3': [13, 73, 82, 58, 52,
                             87, 26, 5, 56],

                   'Col_4': [44, 75, 26, 15, 15,
                             14, 54, 25, 24]})
```

```
display(df)
```

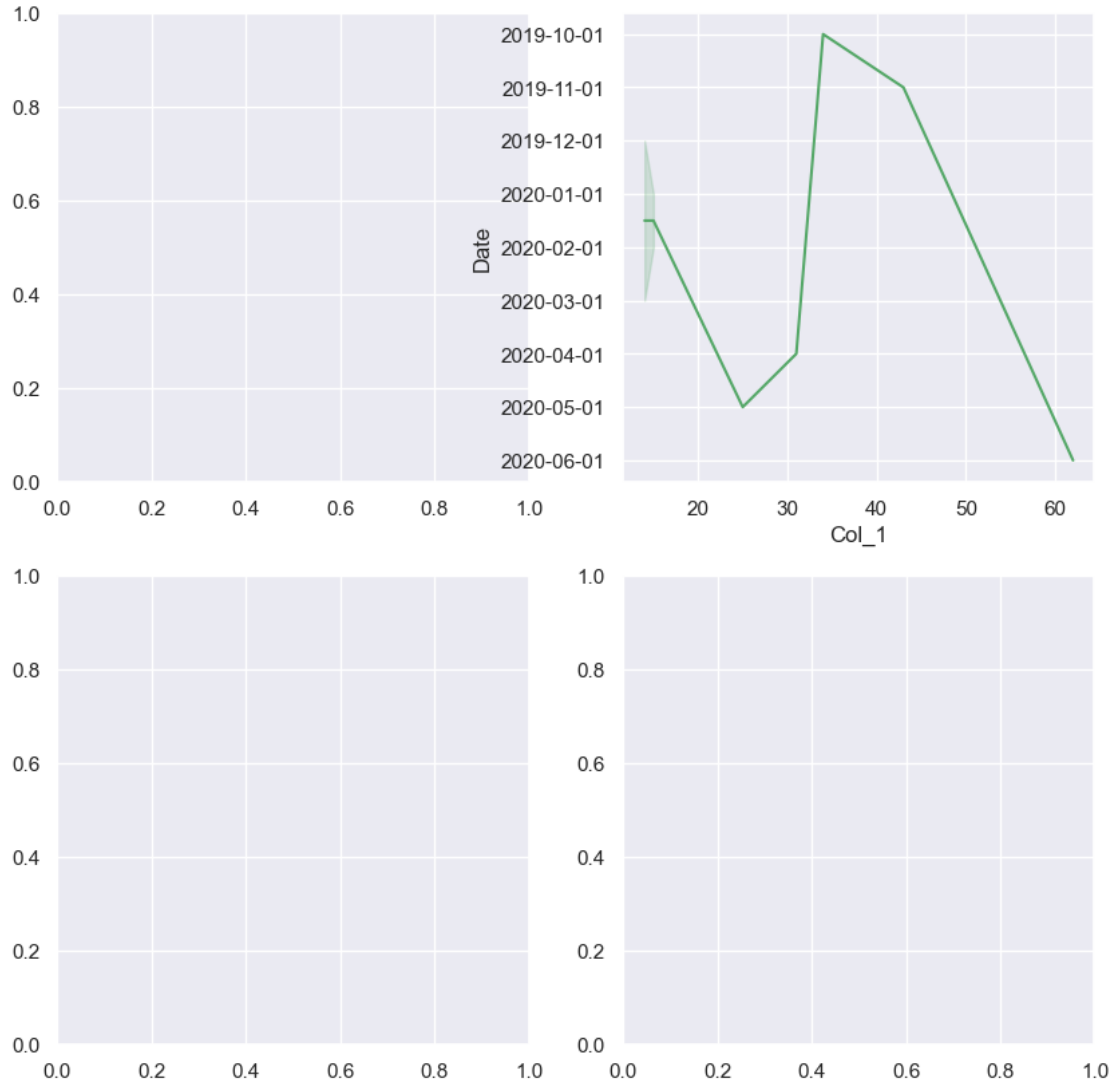
```
         Date  Col_1  Col_2  Col_3  Col_4
0  2019-10-01     34     52     13     44
1  2019-11-01     43     66     73     75
2  2019-12-01     14     78     82     26
3  2020-01-01     15     15     58     15
4  2020-02-01     15     15     52     15
5  2020-03-01     14      5     87     14
6  2020-04-01     31     25     26     54
7  2020-05-01     25     25      5     25
8  2020-06-01     62     86     56     24
```

[83]:
```
sns.lineplot(x="Col_1", y="Date", data=df)
plt.show()
```



[84]:
```
fig,ax=plt.subplots(2,2, figsize=(10,10))
sns.lineplot(x="Col_1", y="Date", data=df, color='g', ax=ax[0][1])
```

[84]:
```
<Axes: xlabel='Col_1', ylabel='Date'>
```

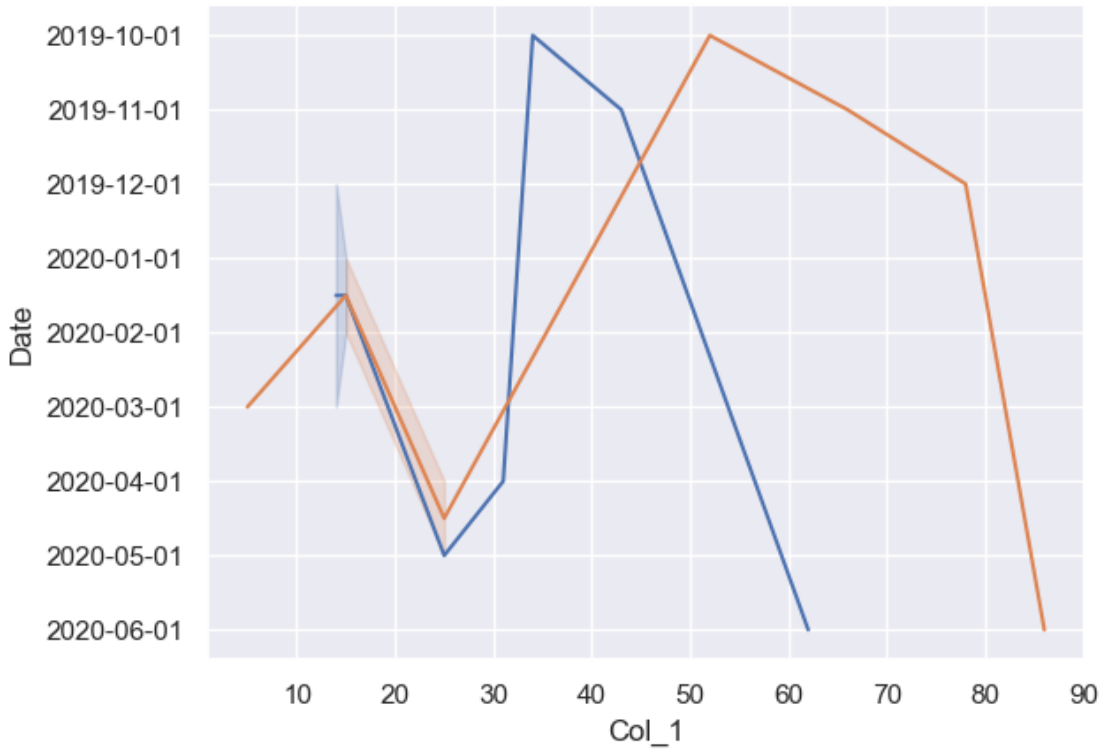


[ ]:

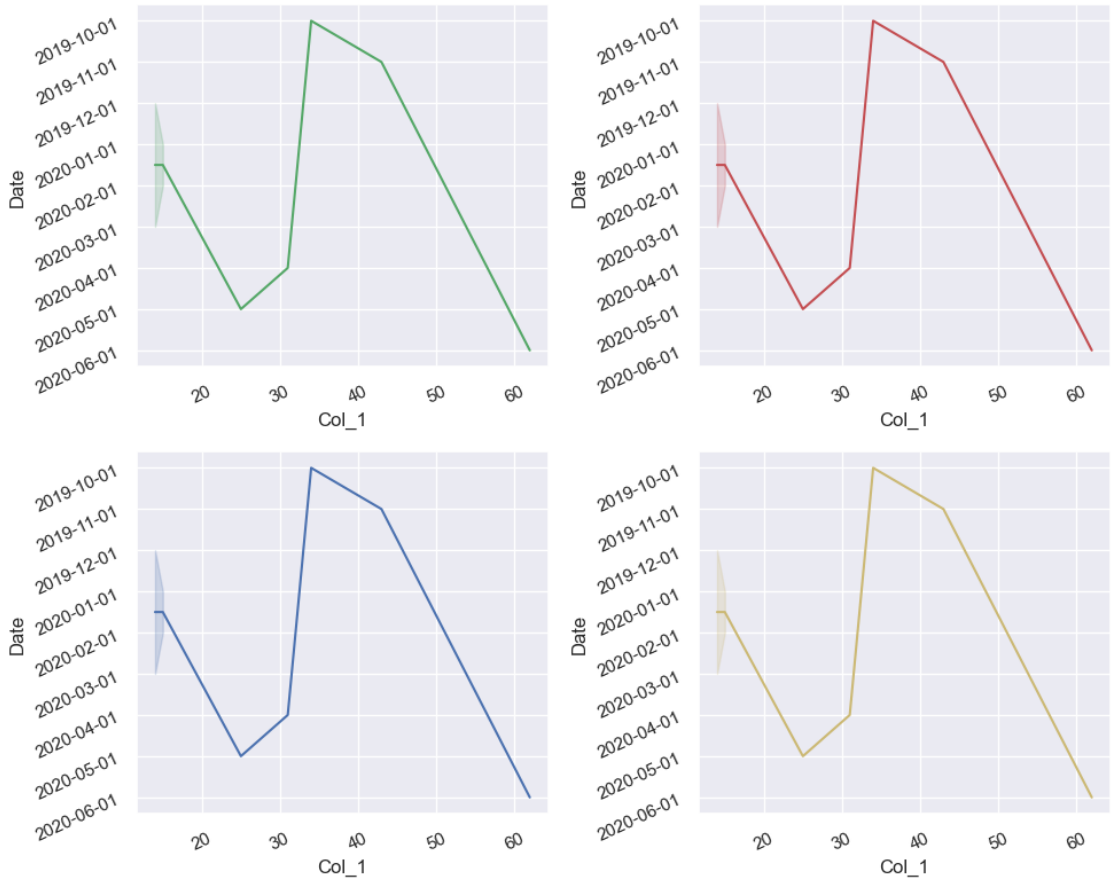
[ ]:

[ ]:

[85]:
```
sns.lineplot(x="Col_1", y="Date", data=df)
sns.lineplot(x="Col_2", y="Date", data=df)

plt.show()
```

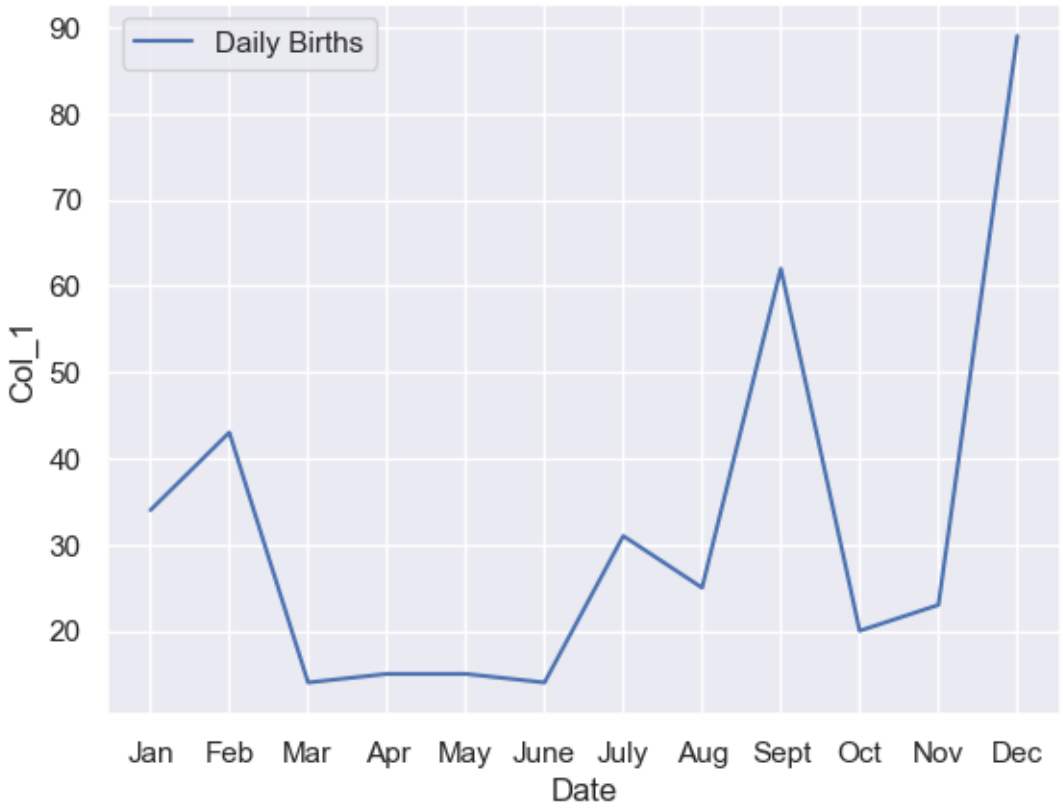


[86]:
```
fig,ax =  plt.subplots( 2, 2, figsize = ( 10, 8))
sns.lineplot(x="Col_1", y="Date", data=df, color='g',ax=ax[0][0])
ax[0][0].tick_params(labelrotation=25)

sns.lineplot(x="Col_1", y="Date", data=df, color='r',ax=ax[0][1])
ax[0][1].tick_params(labelrotation=25)

sns.lineplot(x="Col_1", y="Date", data=df, color='b',ax=ax[1][0])
ax[1][0].tick_params(labelrotation=25)

sns.lineplot(x="Col_1", y="Date", data=df, color='y',ax=ax[1][1])
ax[1][1].tick_params(labelrotation=25)
fig.tight_layout(pad=1.2)
```



6.0.2 –> Time series with Rolling average

[88]:
```
df = pd.DataFrame({'Date':[ '1959-01-01', '1959-02-01', '1959-03-01',␣
 ↪'1959-04-01',
       '1959-05-01', '1959-06-01', '1959-07-01', '1959-08-01',
       '1959-09-01', '1959-10-01', '1959-11-01', '1959-12-01'],
                   'Col_1': [34, 43, 14, 15, 15, 14, 31, 25, 62, 20, 23,89] })

sns.lineplot(x="Date", y="Col_1", data=df,label="Daily Births")
pos=[ '1959-01-01', '1959-02-01', '1959-03-01', '1959-04-01',
       '1959-05-01', '1959-06-01', '1959-07-01', '1959-08-01',
       '1959-09-01', '1959-10-01', '1959-11-01', '1959-12-01']
lab=[ 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'June',
       'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec']
plt.xticks(pos, lab)
plt.show()
```

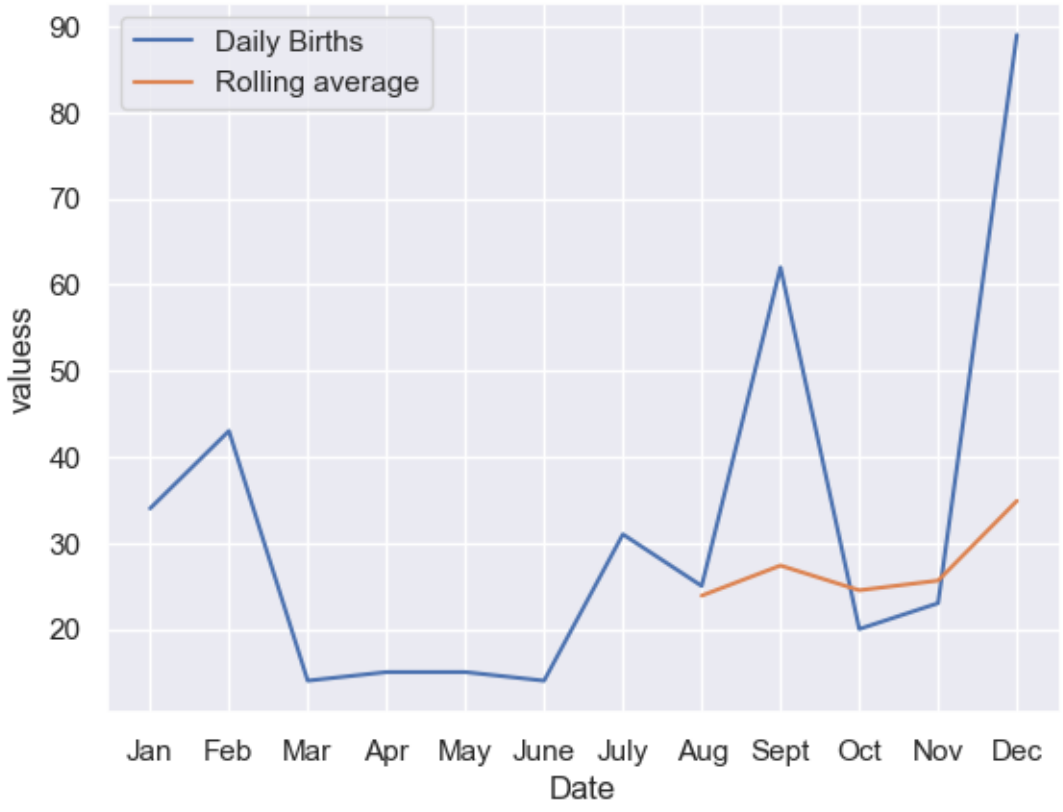


[89]:
```
# computing a 8 days rolling average
df["8_Days_rolling_avg"]=df.Col_1.rolling(8).mean()
# viewing the dataset
display(df.head(12))
```

	Date	Col_1	8_Days_rolling_avg
0	1959-01-01	34	NaN
1	1959-02-01	43	NaN
2	1959-03-01	14	NaN
3	1959-04-01	15	NaN
4	1959-05-01	15	NaN
5	1959-06-01	14	NaN
6	1959-07-01	31	NaN
7	1959-08-01	25	23.875
8	1959-09-01	62	27.375
9	1959-10-01	20	24.500
10	1959-11-01	23	25.625
11	1959-12-01	89	34.875

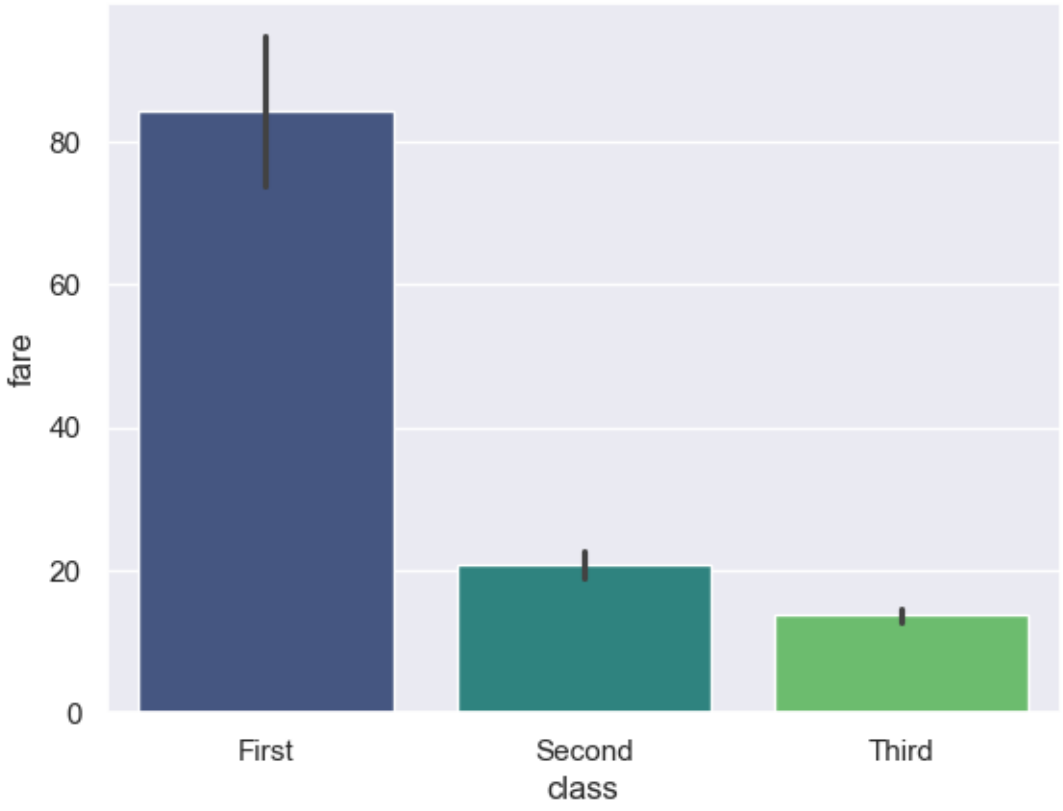
[90]:
```
df = pd.DataFrame({'Date':[ '1959-01-01', '1959-02-01', '1959-03-01',␣
↪'1959-04-01',
```

```
       '1959-05-01', '1959-06-01', '1959-07-01', '1959-08-01',
       '1959-09-01', '1959-10-01', '1959-11-01', '1959-12-01'],
        'Col_1': [34, 43, 14, 15, 15, 14, 31, 25, 62, 20, 23,89] })
df["8_Days_rolling_avg"]=df.Col_1.rolling(8).mean()
pos=[ '1959-01-01', '1959-02-01', '1959-03-01', '1959-04-01',
       '1959-05-01', '1959-06-01', '1959-07-01', '1959-08-01',
       '1959-09-01', '1959-10-01', '1959-11-01', '1959-12-01']
lab=[ 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'June',
       'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec']

sns.lineplot(x="Date", y="Col_1", data=df,label="Daily Births")
sns.lineplot(x="Date", y="8_Days_rolling_avg", data=df,label="Rolling average")
plt.xticks(pos, lab)
plt.xlabel("Date")
plt.ylabel("valuess")
plt.show()
```



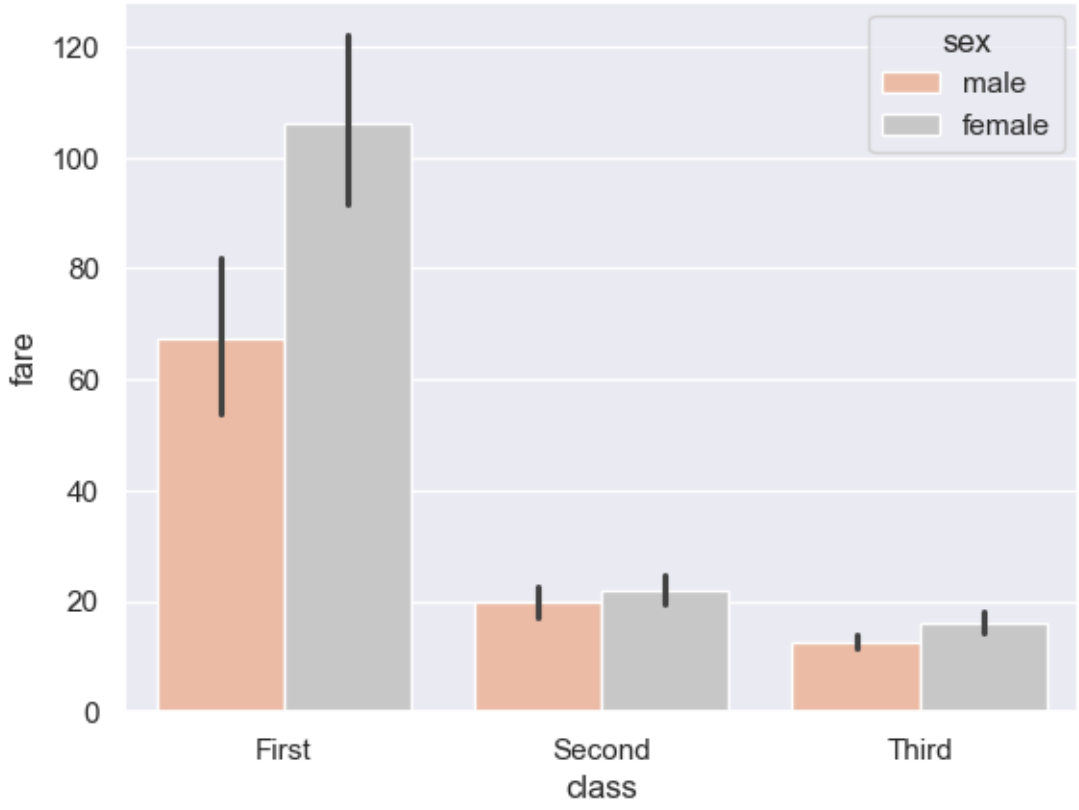
7 Barplot with seaborn

[92]:
```
# 1: Draw a set of vertical bar plots grouped by a categorical variable.
dff=sns.load_dataset('titanic')
sns.barplot(x="class", y="fare", data=dff, palette="viridis")
plt.show()
```

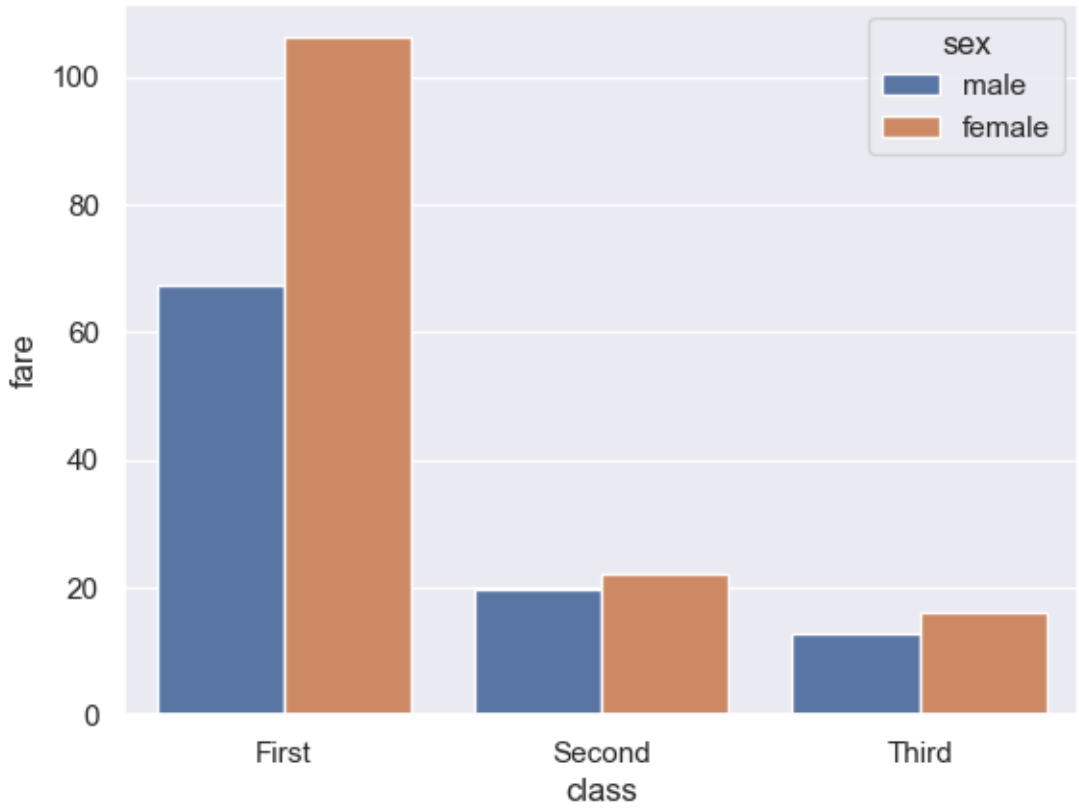


[93]:
```
sns.barplot(x="class", y="fare", hue="sex",data=dff, palette="RdGy")
plt.show()

# color
# Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r,␣
 ↪CMRmap, CMRmap_r, Dark2, Dark2_r,
# GnBu, GnBu_r, Greens, Greens_r, Greys, Greys_r, OrRd, OrRd_r, Oranges,␣
 ↪Oranges_r, PRGn, PRGn_r, Paired, Paired_r,
# Pastel1, Pastel1_r, Pastel2, Pastel2_r, PiYG, PiYG_r, PuBu, PuBuGn, PuBuGn_r,␣
 ↪PuBu_r, PuOr, PuOr_r, PuRd, PuRd_r,
# Purples, Purples_r, RdBu, RdBu_r, RdGy, RdGy_r, RdPu, RdPu_r, RdYlBu,␣
 ↪RdYlBu_r, RdYlGn, RdYlGn_r, Reds, Reds_r, Set1,
```

```
# Set1_r, Set2, Set2_r, Set3, Set3_r, Spectral, Spectral_r, Wistia, Wistia_r,␣
 ↪YlGn, YlGnBu, YlGnBu_r, YlGn_r, YlOrBr,
# YlOrBr_r, YlOrRd, YlOrRd_r, afmhot, afmhot_r, autumn, autumn_r, binary,␣
 ↪binary_r, bone, bone_r, brg, brg_r, bwr, bwr_r,
# cividis, cividis_r, cool, cool_r, coolwarm, coolwarm_r, copper, copper_r,␣
 ↪cubehelix, cubehelix_r, flag, flag_r, gist_earth,
# gist_earth_r, gist_gray, gist_gray_r, gist_heat, gist_heat_r, gist_ncar,␣
 ↪gist_ncar_r, gist_rainbow, gist_rainbow_r, gist_stern,
```

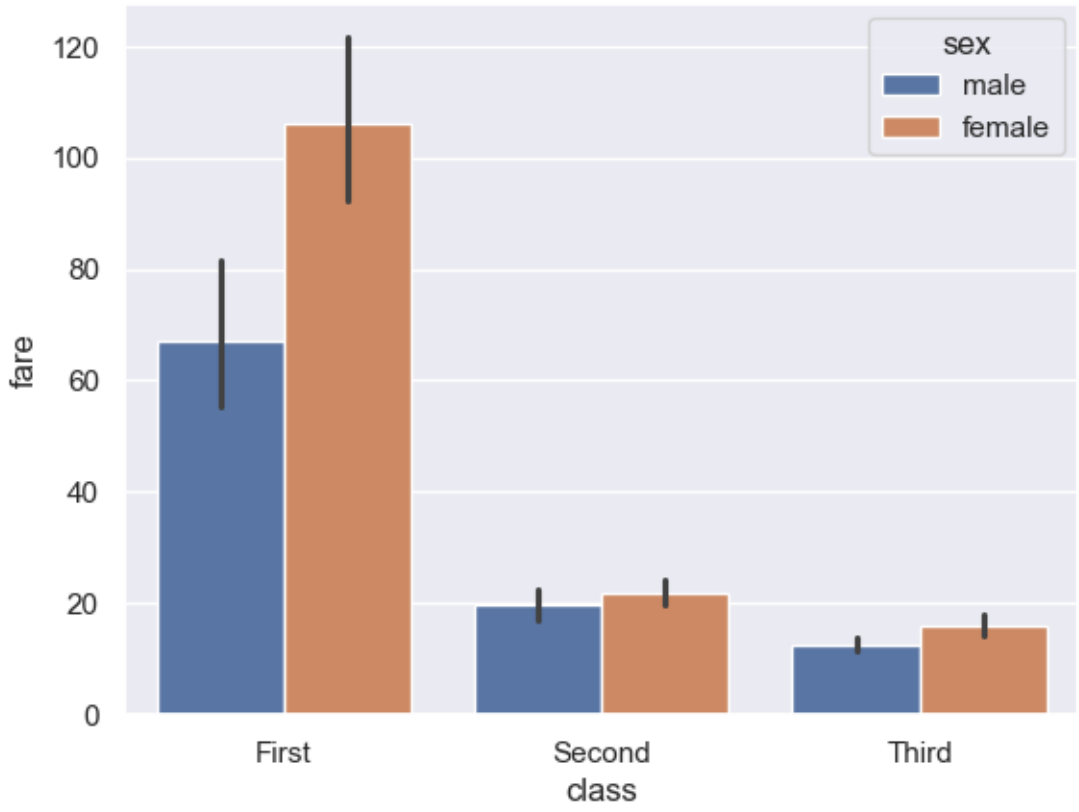


[94]:
```
# error won't be show
sns.barplot(x="class", y="fare", hue="sex",data=dff, ci=None)
plt.show()
```

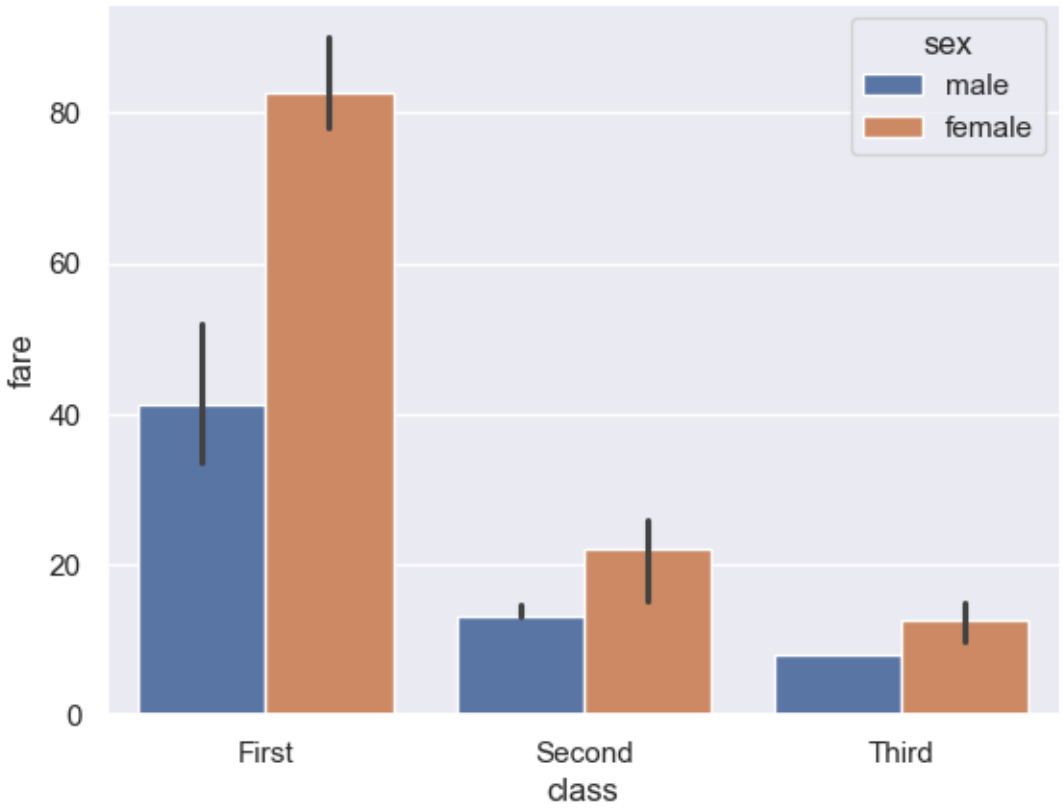


[95]:
```
# Using the statistical function NumPy.median and NumPy.mean to estimate within␣
↪each categorical bin.
```

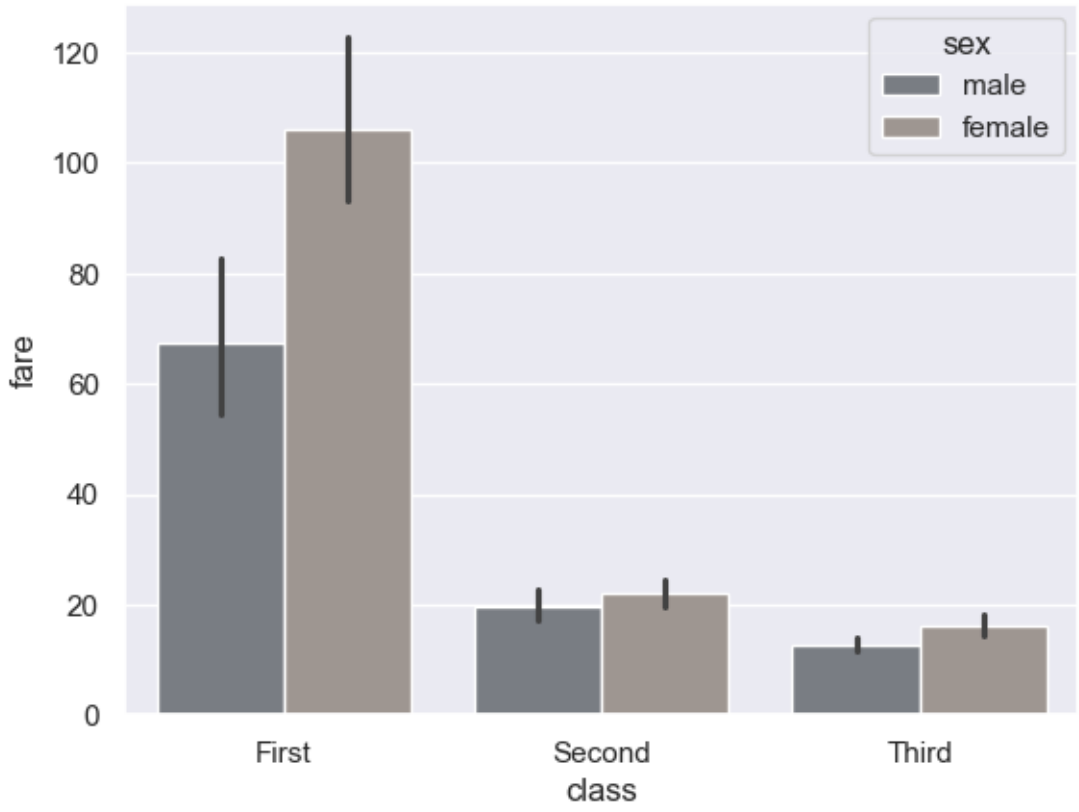
[96]:
```
from numpy import mean
from numpy import median

sns.barplot(x="class", y="fare", hue="sex",data=dff, estimator=mean)
plt.show()
```



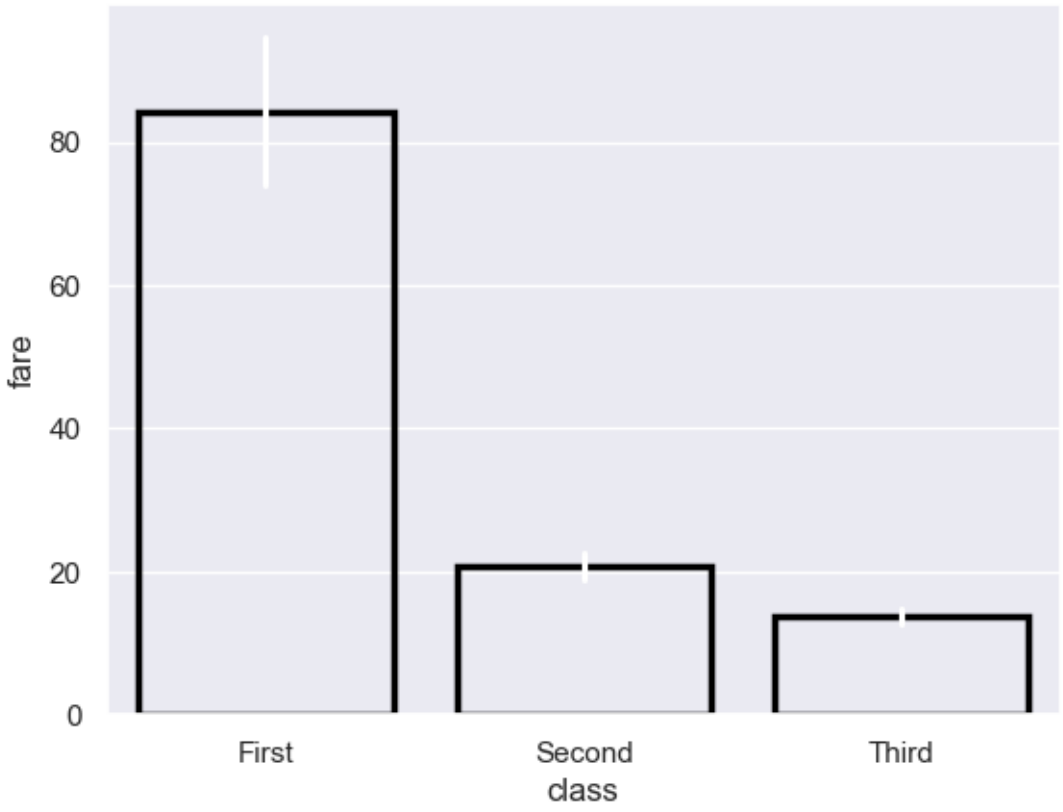
[97]:
```
sns.barplot(x="class", y="fare", hue="sex",data=dff, estimator=median)
plt.show()
```



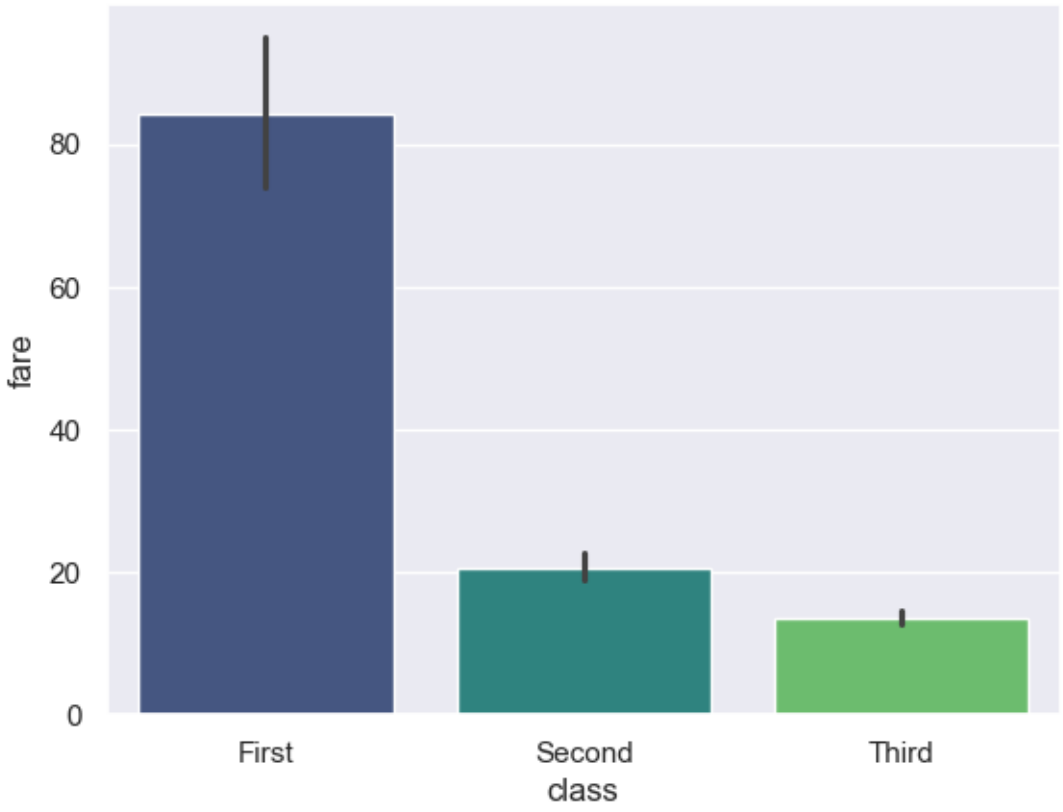
[98]:
```
# Using the saturation parameter.
sns.barplot(x="class", y="fare", hue="sex",data=dff, saturation=0.1)
plt.show()
```



[99]:
```
# Use matplotlib.axes.Axes.bar() parameters to control the style.
sns.barplot(x="class", y="fare", data=dff,linewidth=2.5, facecolor=(1, 0, 1,␣
↪0),errcolor="1", edgecolor="0")
plt.show()
```



[100]:
```
sns.barplot(data=dff, x="class", y="fare", palette="viridis")
plt.show()
```



[101]:
```
sort=dff.sort_values("fare")
sort
```

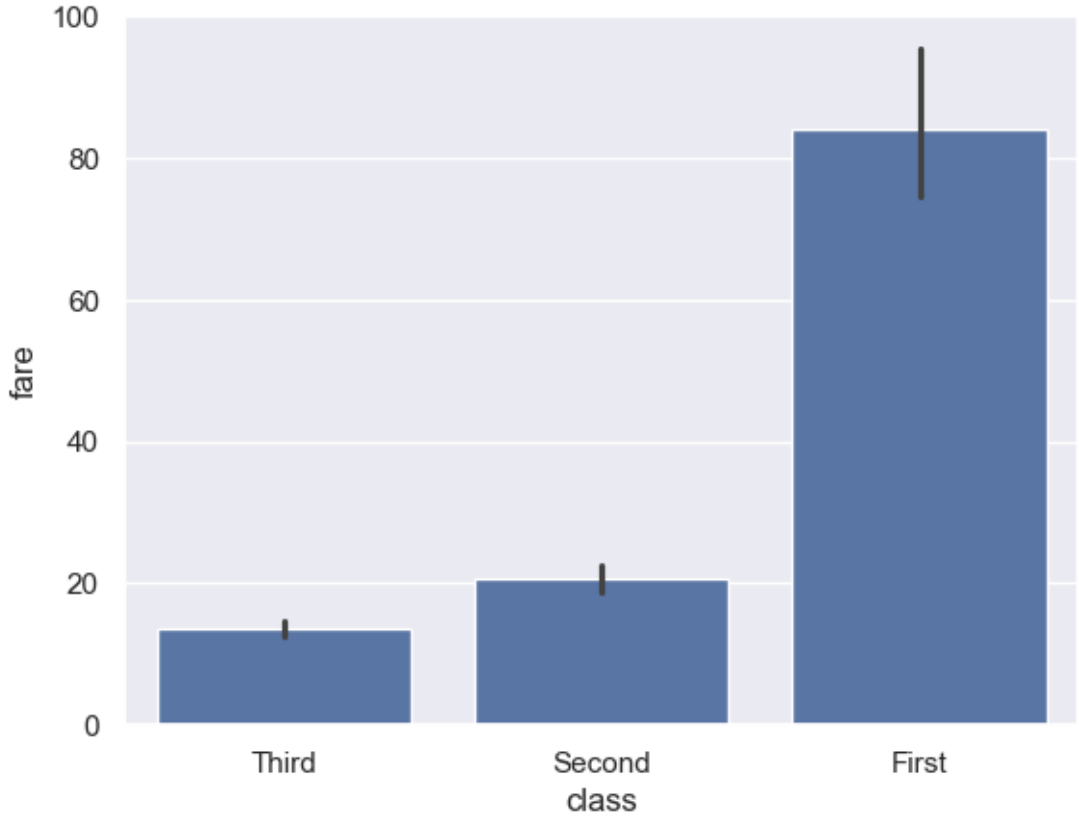
[101]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
271	1	3	male	25.0	0	0	0.0000	S	Third	
597	0	3	male	49.0	0	0	0.0000	S	Third	
302	0	3	male	19.0	0	0	0.0000	S	Third	
633	0	1	male	NaN	0	0	0.0000	S	First	
277	0	2	male	NaN	0	0	0.0000	S	Second	
..	…	…	…	…	…	…	…	…	…	
438	0	1	male	64.0	1	4	263.0000	S	First	
341	1	1	female	24.0	3	2	263.0000	S	First	
737	1	1	male	35.0	0	0	512.3292	C	First	
258	1	1	female	35.0	0	0	512.3292	C	First	
679	1	1	male	36.0	0	1	512.3292	C	First	

	who	adult_male	deck	embark_town	alive	alone
271	man	True	NaN	Southampton	yes	True
597	man	True	NaN	Southampton	no	True
302	man	True	NaN	Southampton	no	True
633	man	True	NaN	Southampton	no	True

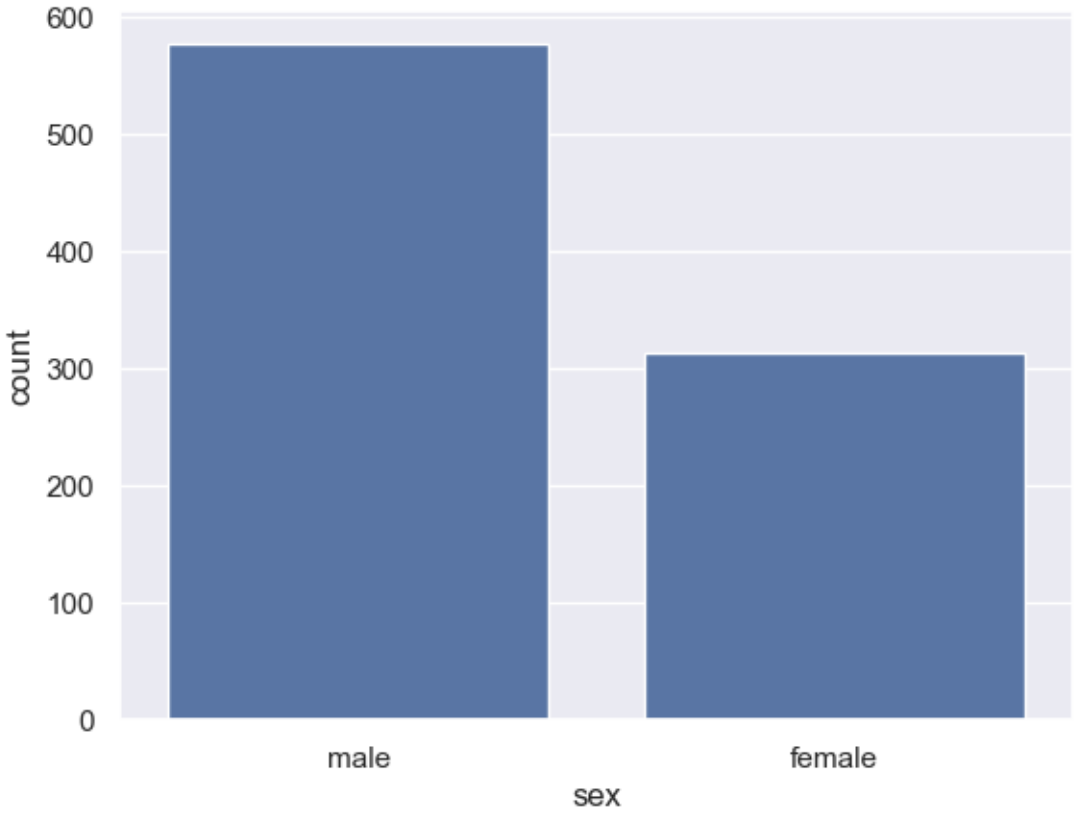
```
277    man        True   NaN  Southampton    no   True
..     …          …   …          …    …    …
438    man        True     C  Southampton    no  False
341  woman       False     C  Southampton   yes  False
737    man        True     B    Cherbourg   yes   True
258  woman       False   NaN    Cherbourg   yes   True
679    man        True     B    Cherbourg   yes  False

[891 rows x 15 columns]
```

[102]:
```
# sorting

sort=dff.groupby("class")["fare"].mean().sort_values(ascending=True).index
sns.barplot(data=dff, x="class", y="fare",order=sort)
plt.show()
```

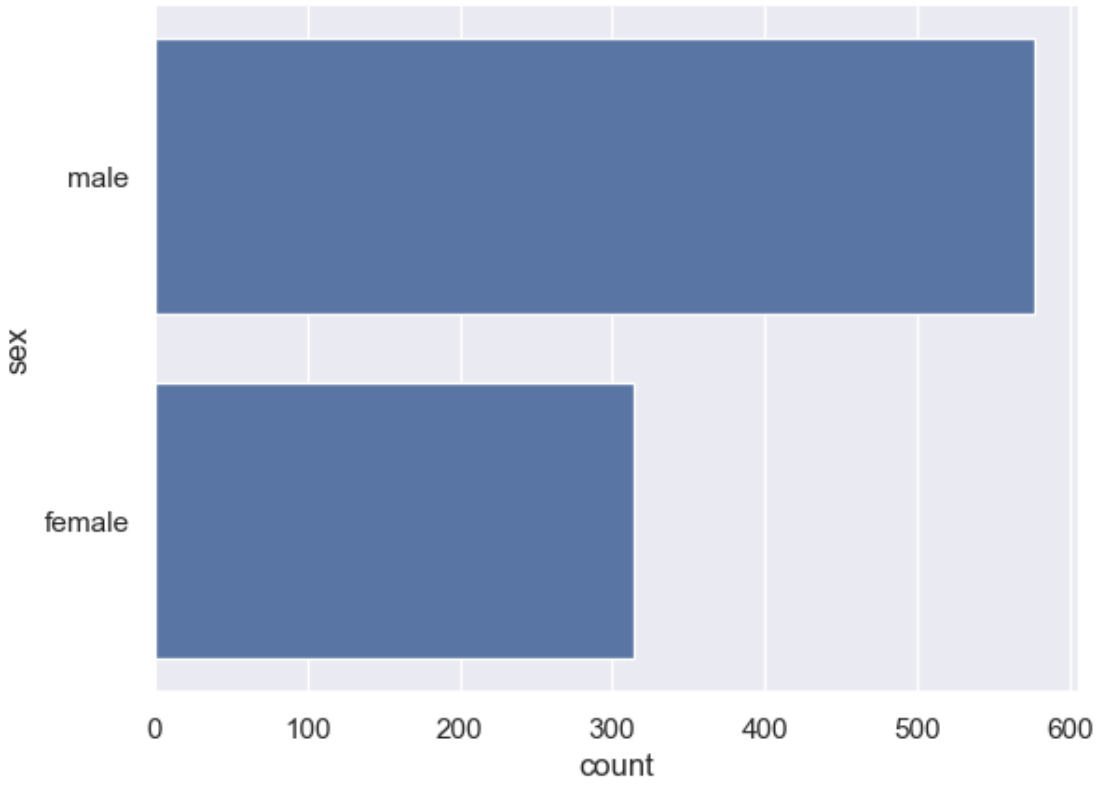


8 Countplot

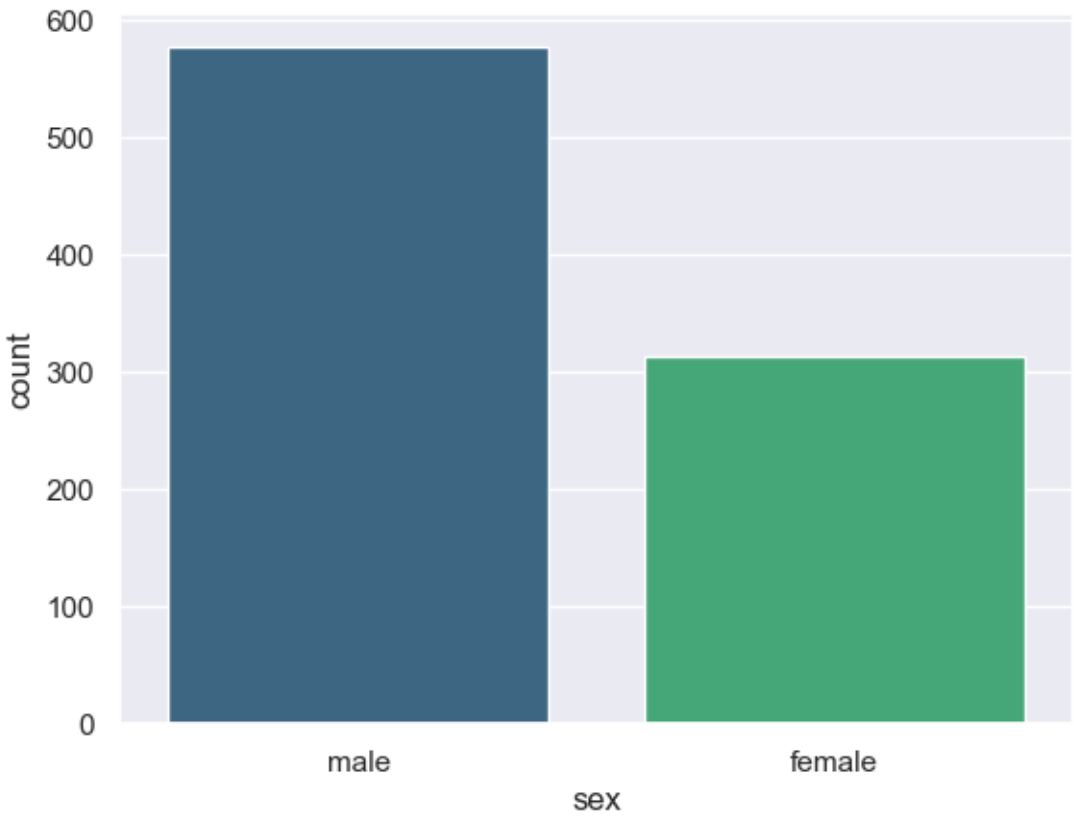
[104]:
```
# verically
sns.countplot(x="sex", data=dff)
plt.show()
```



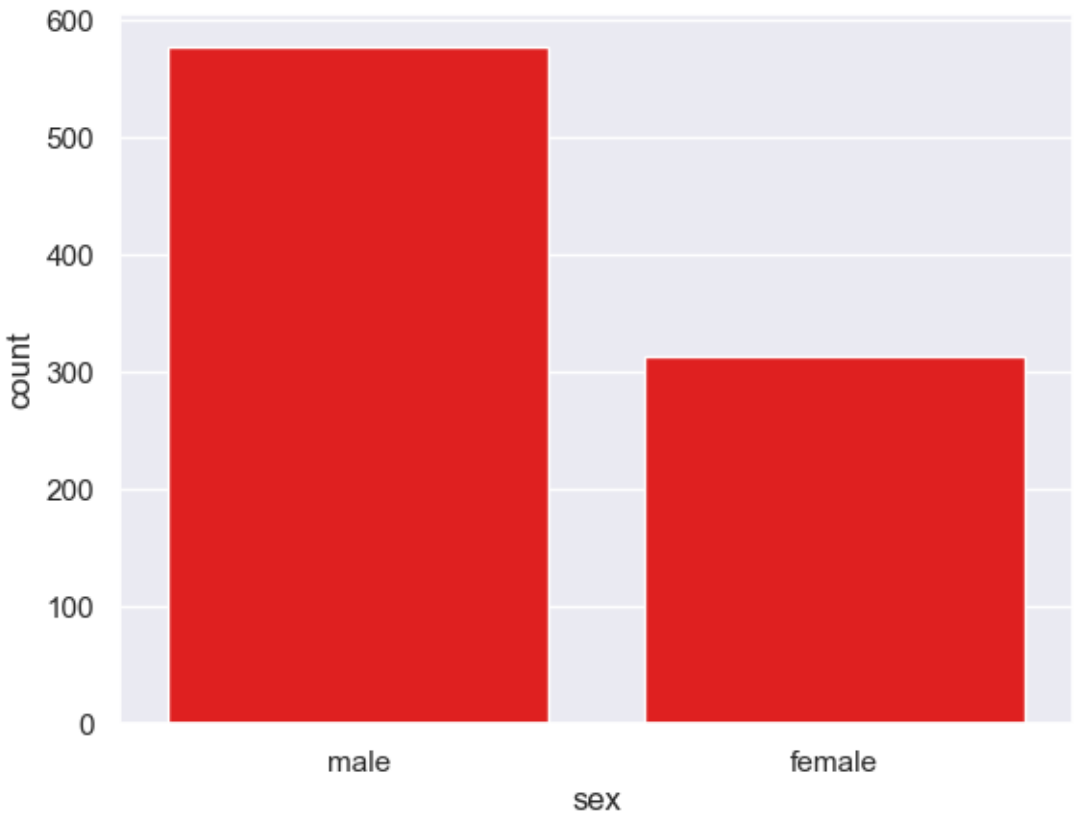
[105]:
```
# horizontally
sns.countplot(y="sex", data=dff)
plt.show()
```



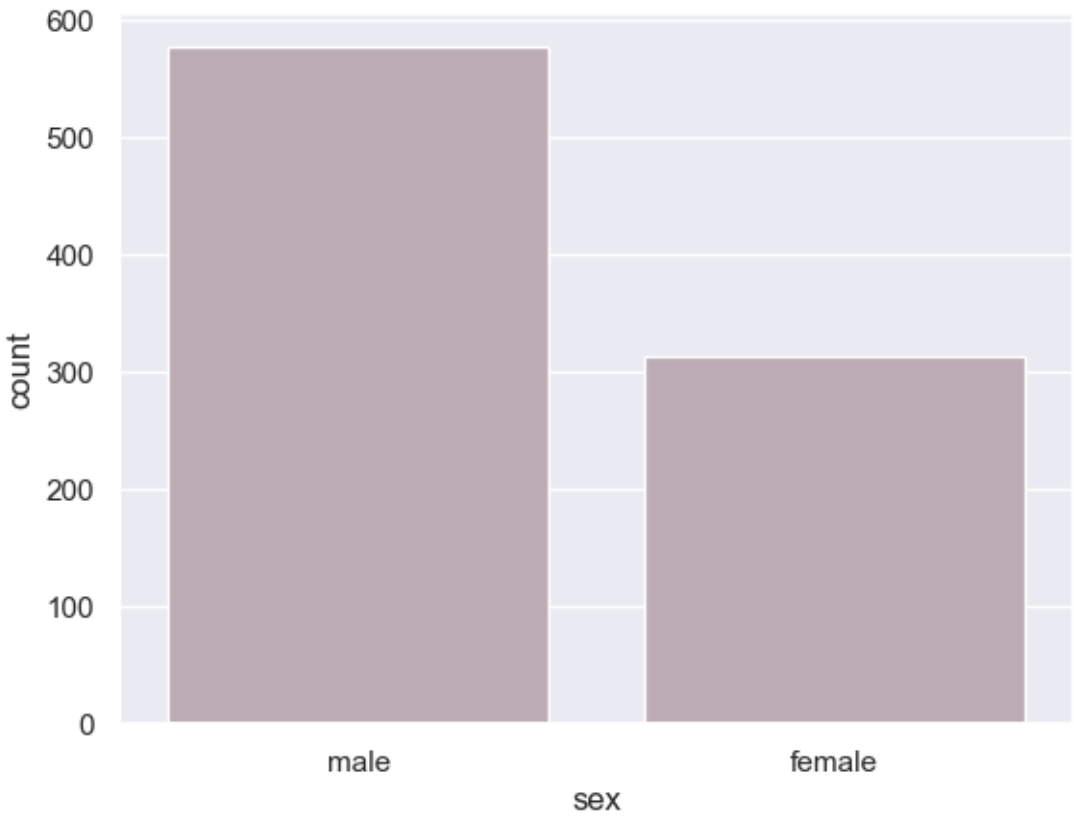
[106]:
```
# use different colors
sns.countplot(x="sex", data=dff, palette="viridis")
plt.show()
```



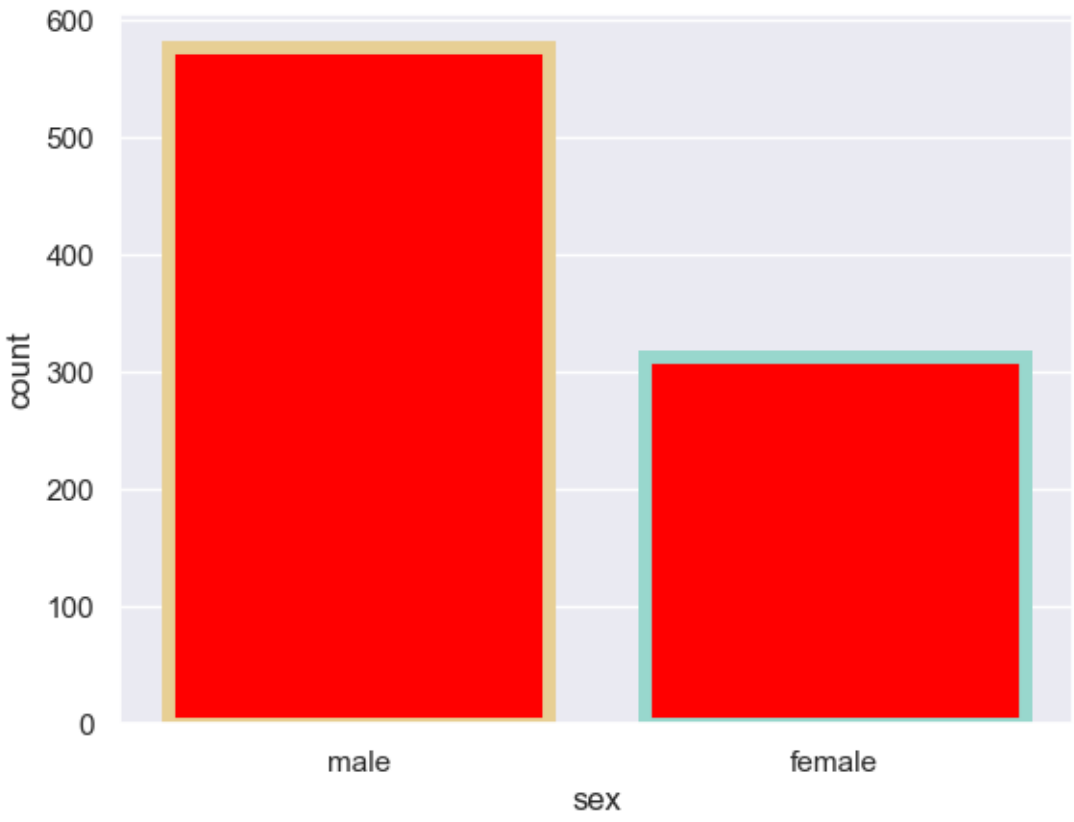
[107]:
```
sns.countplot(x="sex", data=dff, color='red')
plt.show()
```



[108]:
```
sns.countplot(x="sex", data=dff, color="hotpink", saturation=0.1)
plt.show()
```



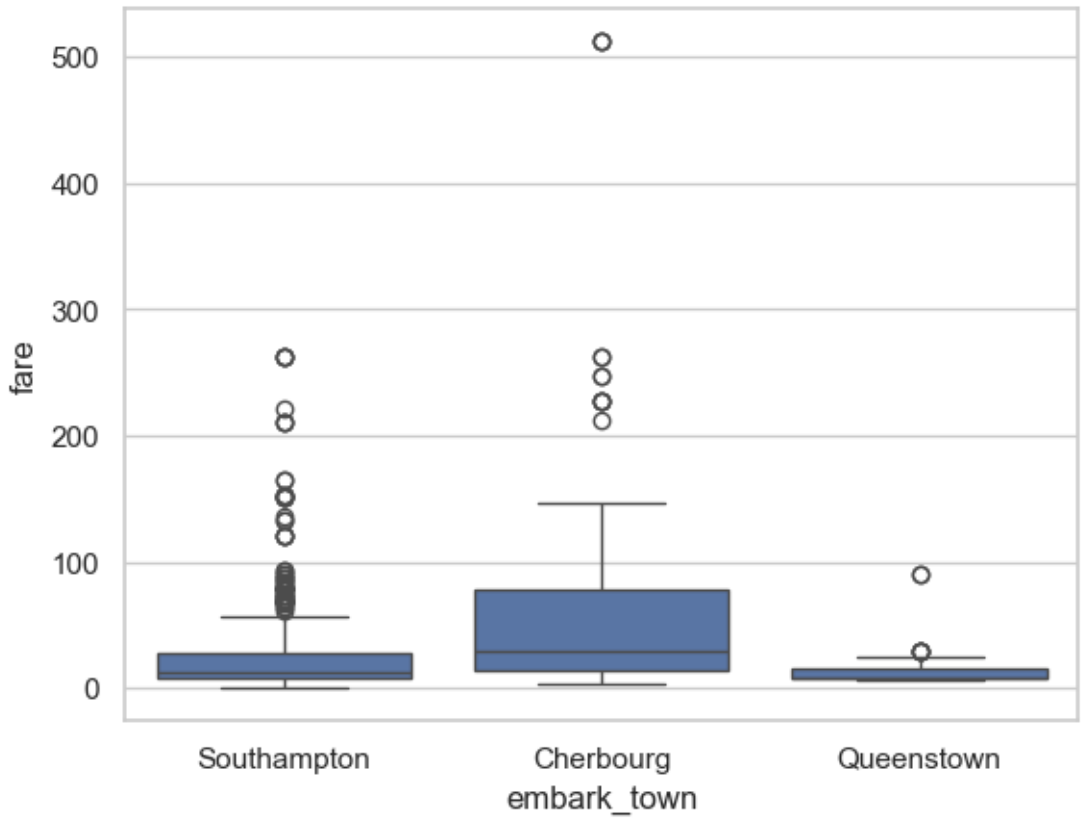
[109]:
```
# use matplotlib.Axes.bar() parameters to control the style
sns.countplot(x="sex", data=dff, color="yellow", facecolor=(1,0,0,1),␣
 ↪linewidth=5, edgecolor=sns.color_palette("BrBG",2))
plt.show()
```



9 Boxplot

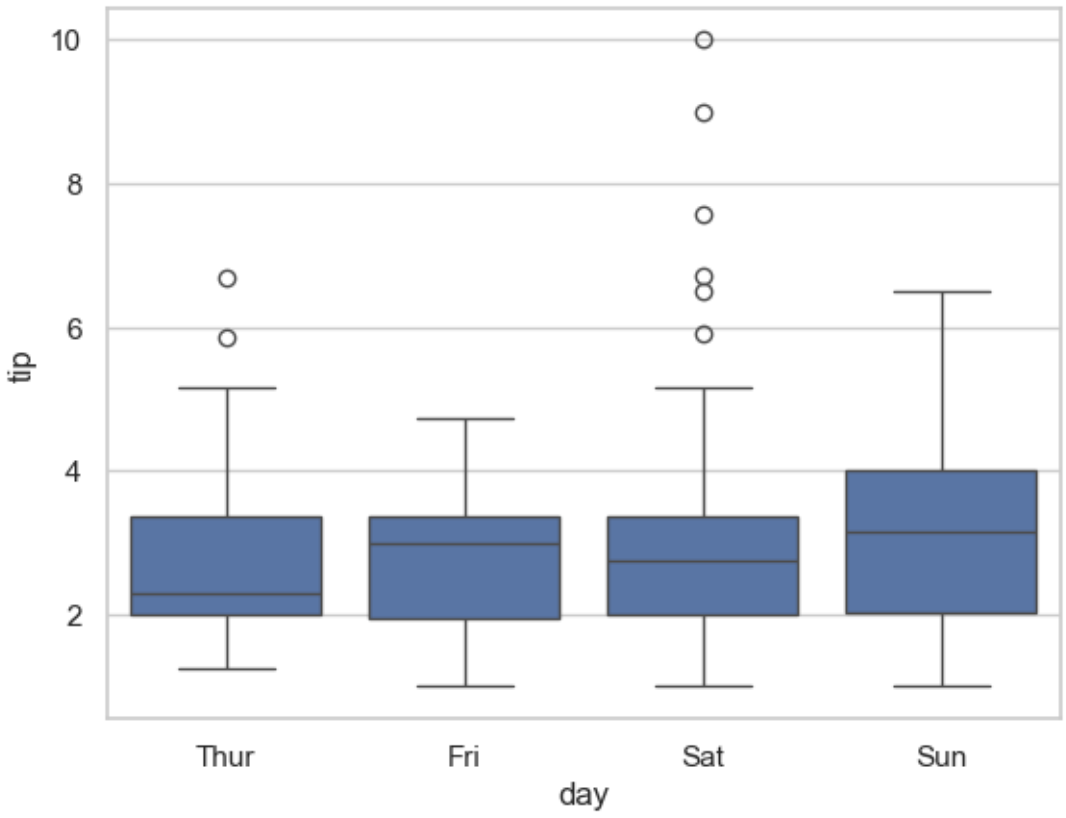
[111]:
```
# Syntax: seaborn.boxplot(x=None, y=None, hue=None, data=None, order=None,␣
 ↪hue_order=None, orient=None, color=None, palette=None, saturation=0.75,␣
 ↪width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, ax=None,␣
 ↪**kwargs)
# median: the middle value of data. marked as Q2, portrays the 50th percentile.
# first quartile: the middle value between "minimum non-outlier" and median.␣
 ↪marked as Q1, portrays the 25th percentile.
# third quartile: the middle value between "maximum non-outlier" and median.␣
 ↪marked as Q3, portrays the 75th percentile.
# "maximum non-outlier": calculated by (Q3 + 1.5*IQR). All values above this␣
 ↪are considered outliers.
# "minimum non-outlier": calculated by (Q1 - 1.5*IQR). All values below this␣
 ↪are considered outliers.
```

[112]:
```
sns.set(style='whitegrid')
sns.boxplot(x="embark_town", y="fare", data=dff)
plt.show()
```



[113]:
```
tip=sns.load_dataset("tips")
sns.boxplot(x="day", y="tip", data=tip, )
```

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

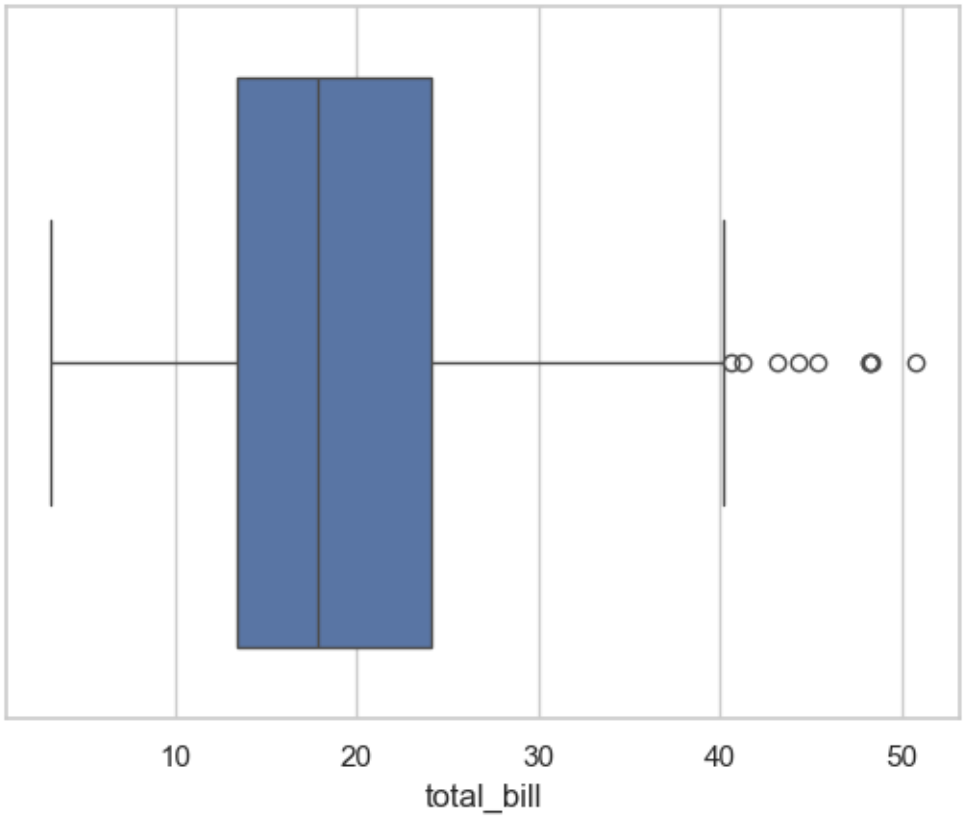


[114]:
```
tip.head()
```

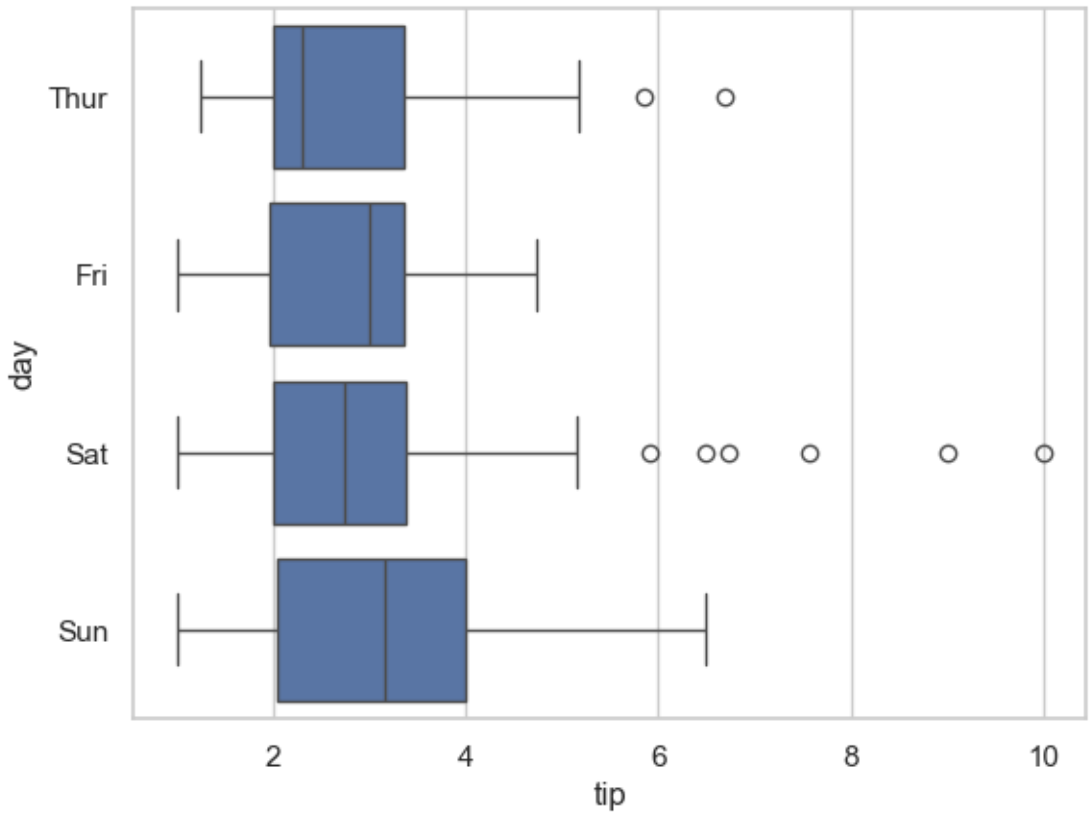
[114]:

	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

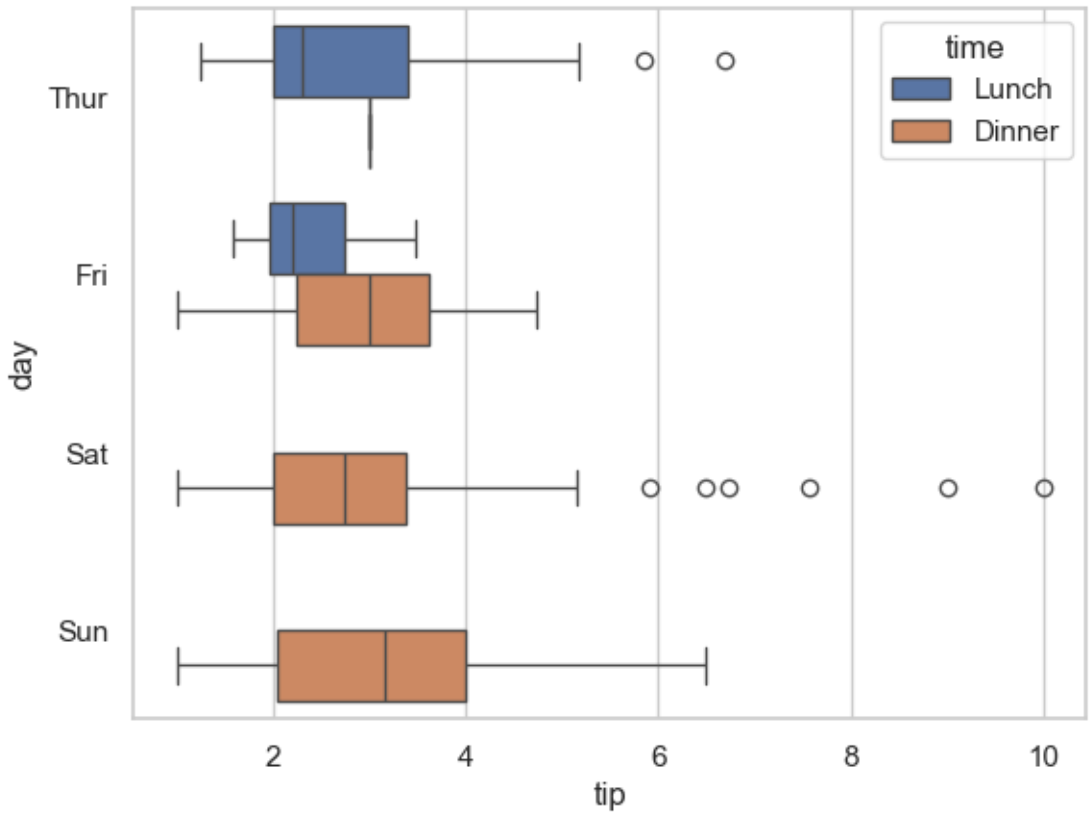
[115]:
```
# Draw single horizontal box plot using only one axis
sns.boxplot(x=tip["total_bill"])
plt.show()
```



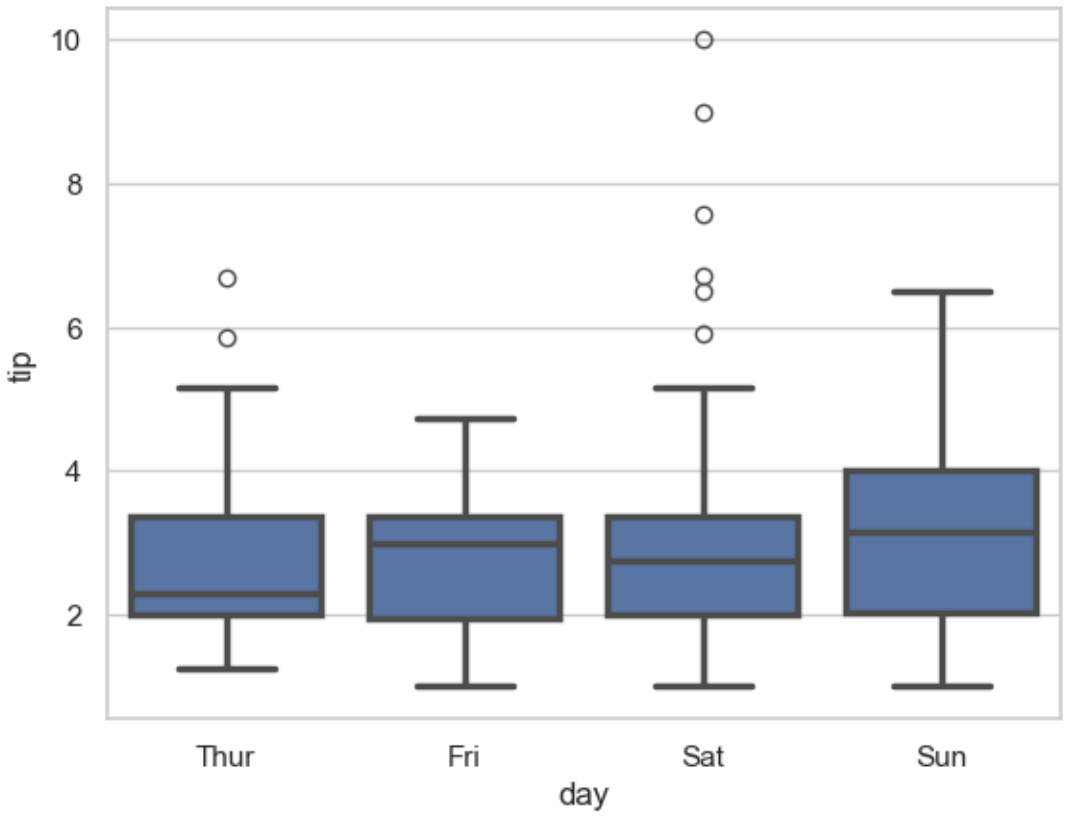
[116]:
```
sns.boxplot(x='tip', y='day', data=tip)
plt.show()
```



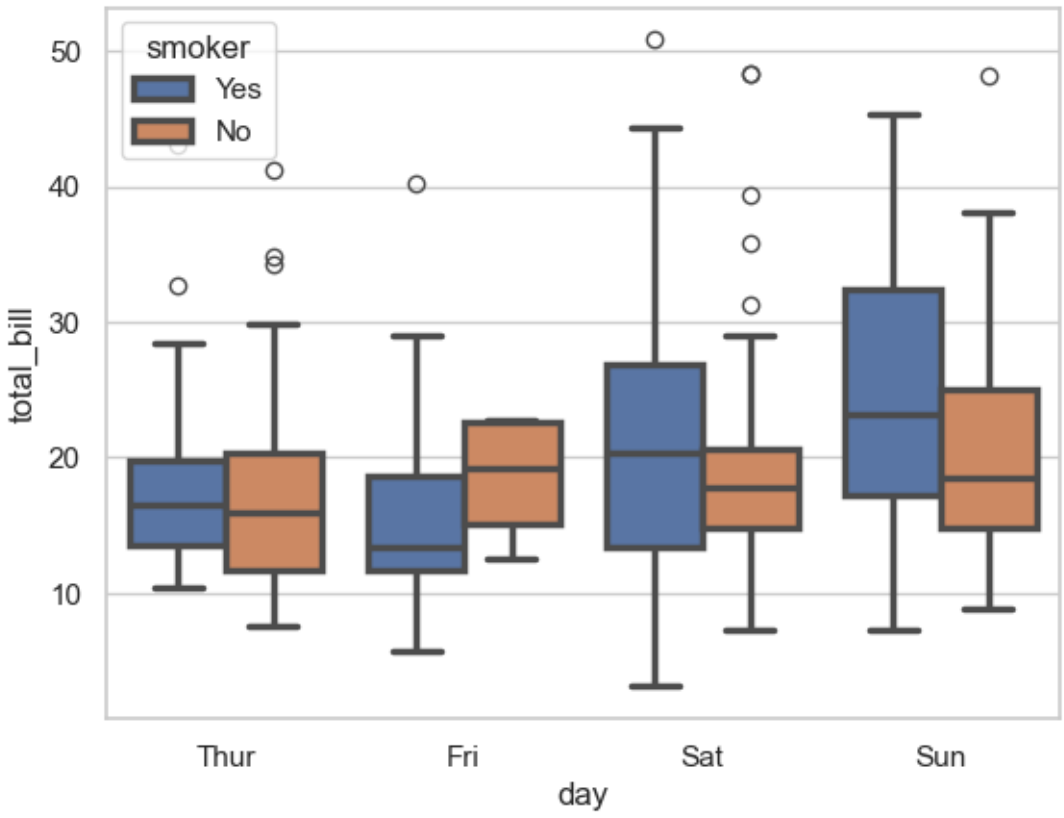
[117]:
```
# using hue value
sns.boxplot(x='tip', y='day', data=tip, hue="time")
plt.show()
```



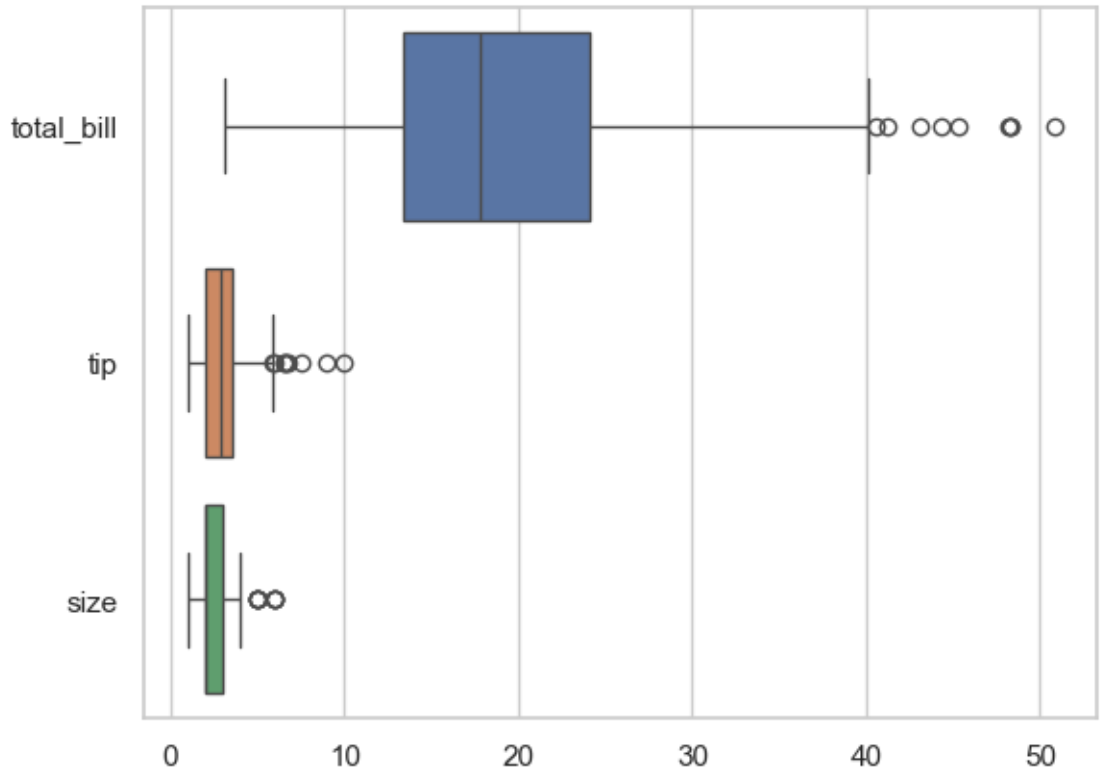
[118]:
```
# draw outlines aroound the data points using linewidth
sns.set(style='whitegrid')
sns.boxplot(x='day', y='tip', data=tip, linewidth=2.5)
plt.show()
```



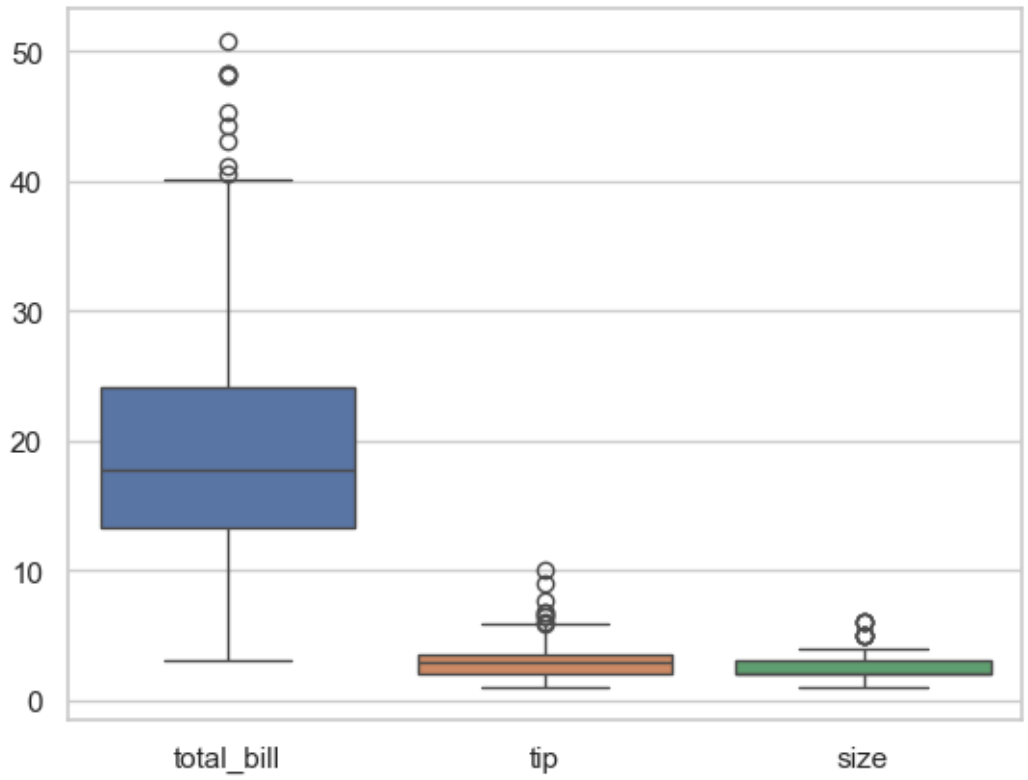
[119]:
```
# Draw the each levl of the hue variable at different locations on the major␣
 ↪categotical axis
sns.set(style='whitegrid')
sns.boxplot(x='day', y='total_bill', data=tip, linewidth=2.5, hue="smoker")
plt.show()
```



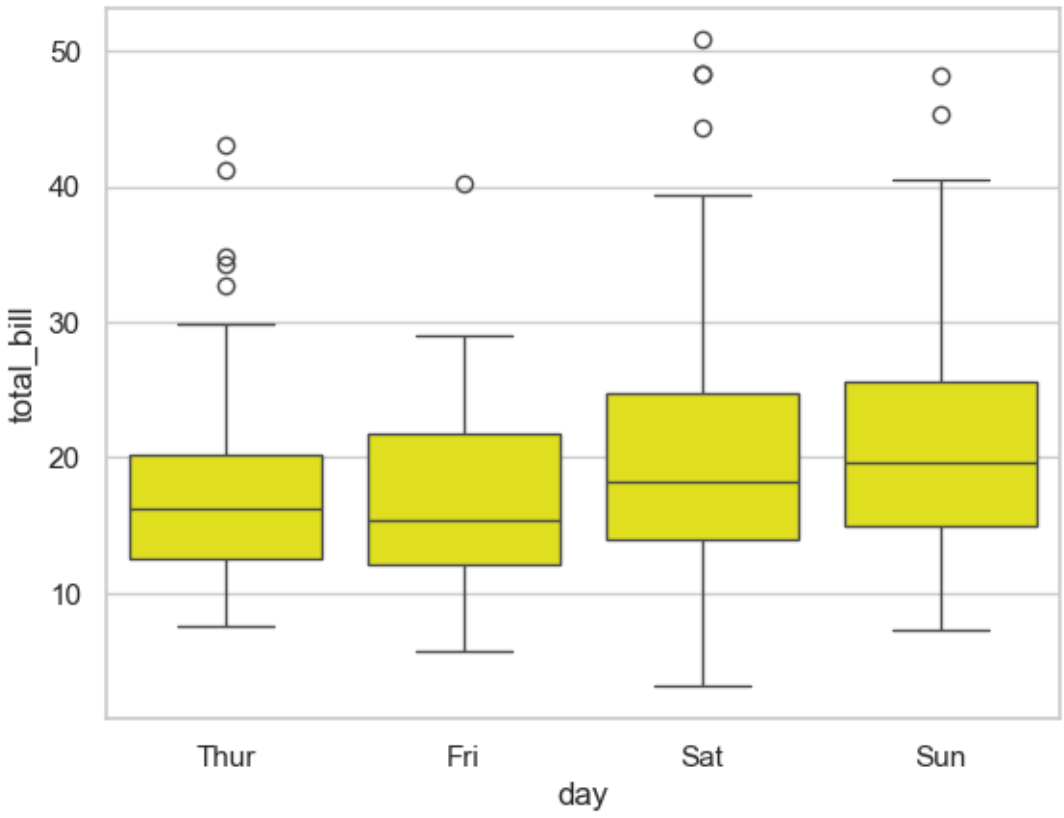
[120]:
```
# control orientation of the plot (vertical or horizontal)
sns.set(style='whitegrid')
sns.boxplot( data=tip, orient="h")
plt.show()
```



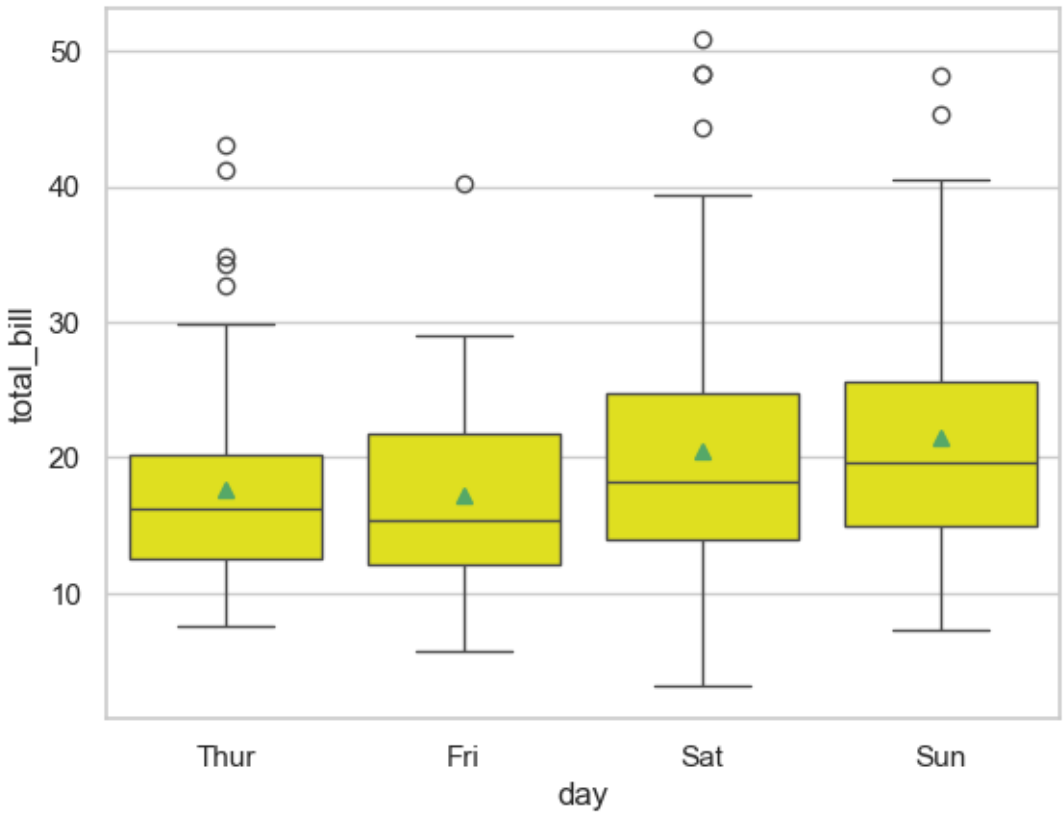
[121]:
```
sns.set(style='whitegrid')
sns.boxplot(data=tip, orient="v")
plt.show()
```



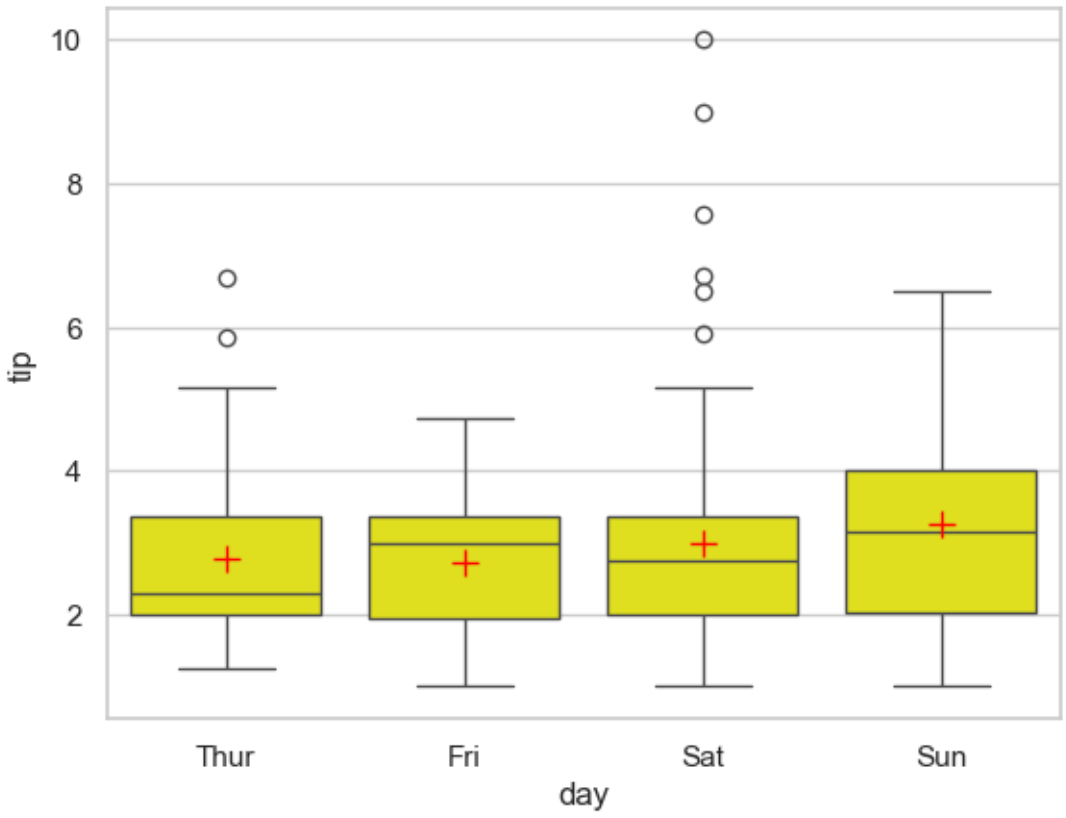
[122]:
```
# using color attributes for color for all the elements
sns.set(style='whitegrid')
sns.boxplot(x='day', y='total_bill', data=tip,color="yellow")
plt.show()
```



[123]:
```
# to show the mean
sns.set(style='whitegrid')
sns.boxplot(x='day', y='total_bill', data=tip,color="yellow", showmeans=True)
plt.show()
```

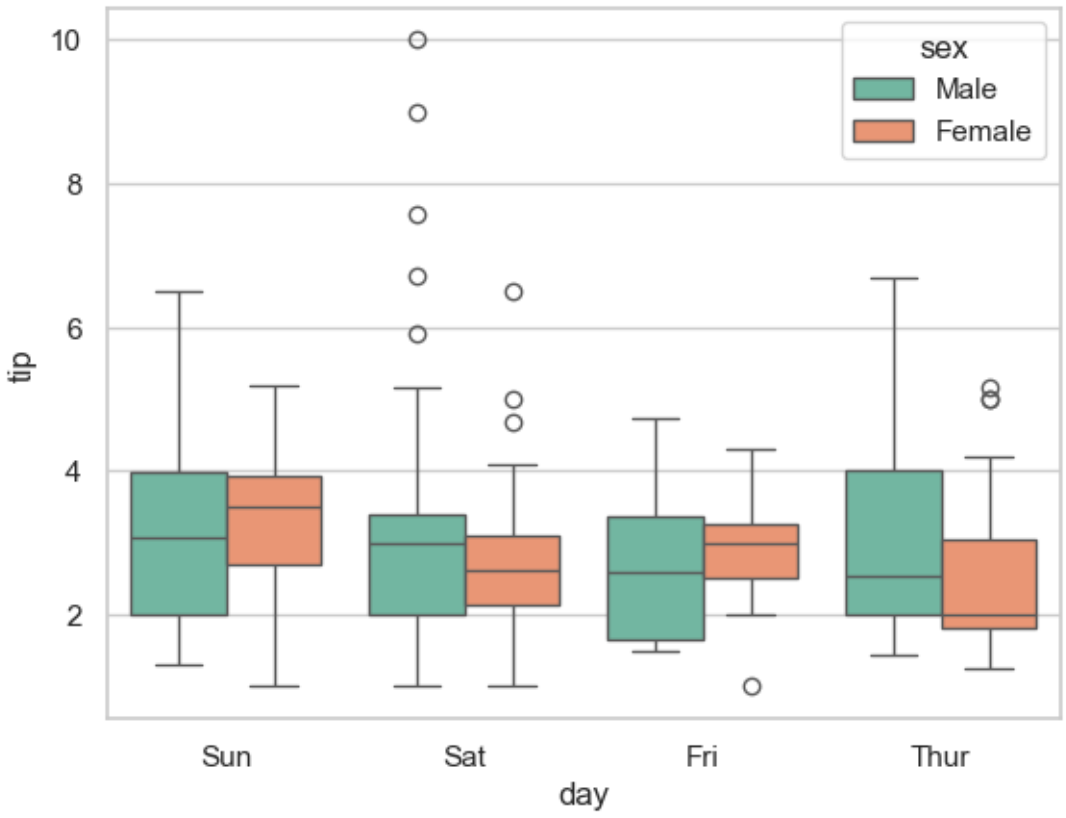


[124]:
```
sns.set(style='whitegrid')
sns.boxplot(x='day', y='tip', data=tip,color="yellow", showmeans=True,␣
 ↪meanprops={"marker":"+","markeredgecolor":"red", "markersize":"10"})
plt.show()
```

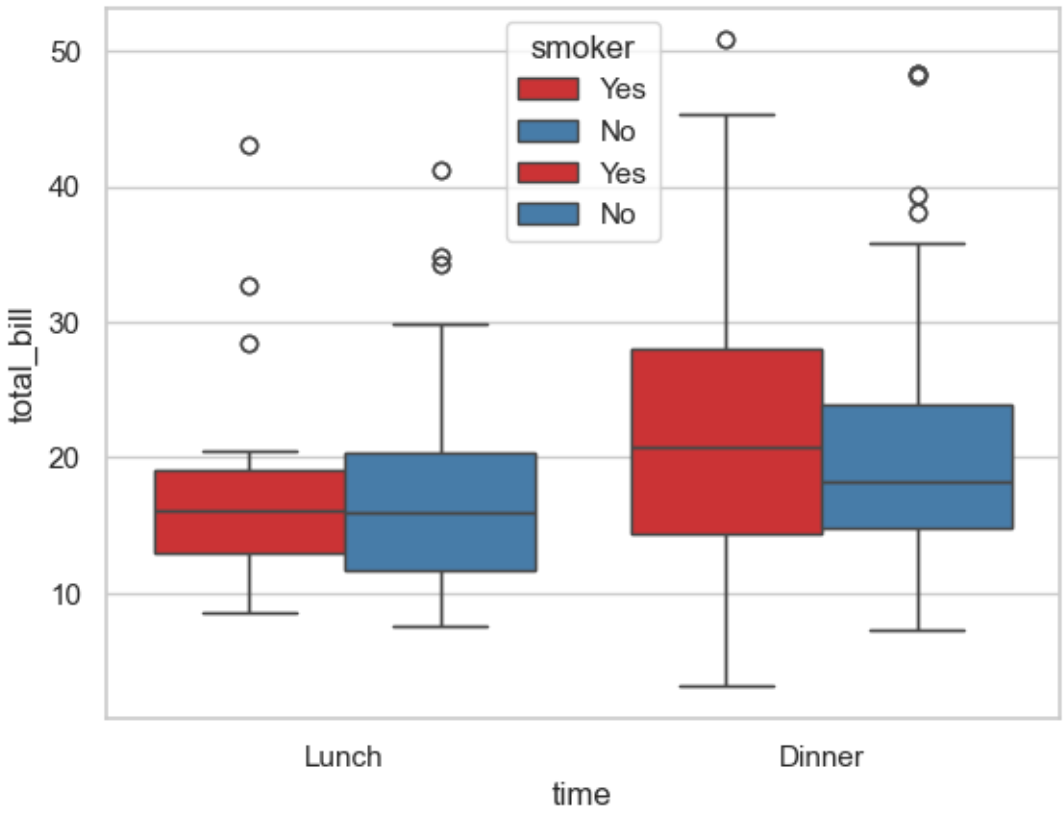


9.0.1 How to manually order Boxplot

[126]:
```
sns.boxplot(x='day', y='tip', data=tip, order=['Sun','Sat','Fri', 'Thur'],␣
↪hue='sex', palette='Set2')
plt.show()
```

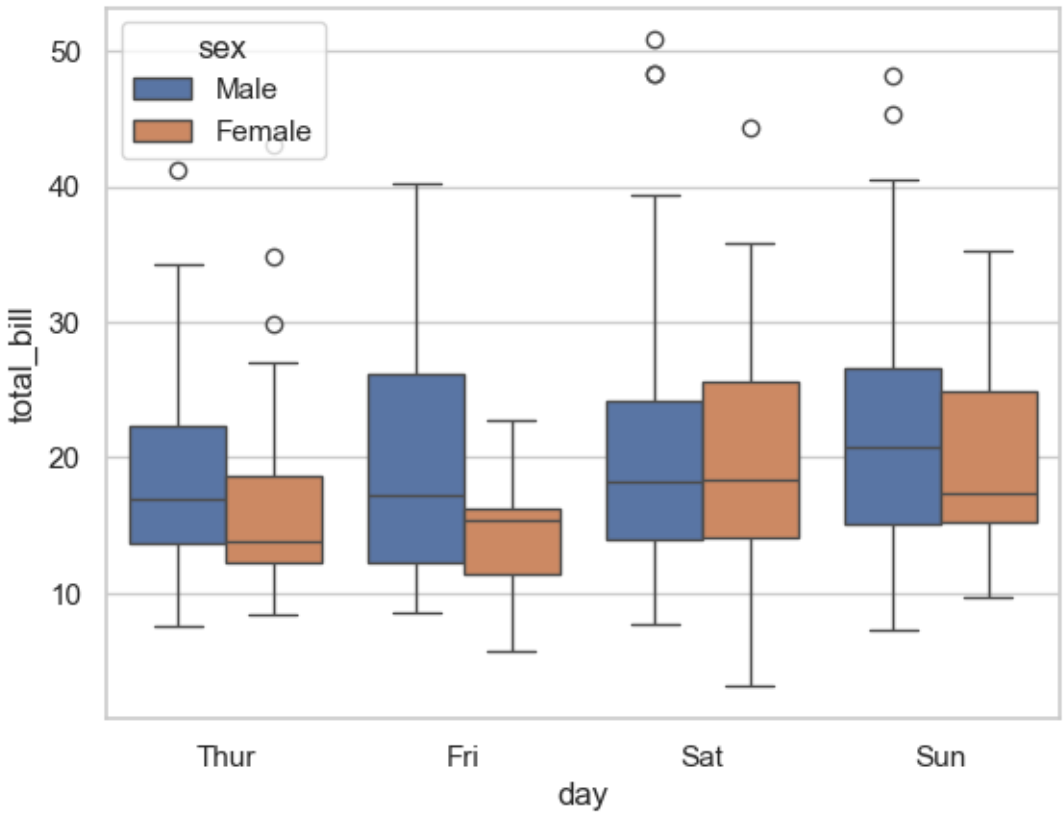


[127]:
```
before=sns.boxplot(x="time", y="total_bill", hue="smoker", data=tips,␣
↪palette="Set1")
after=sns.boxplot(x="time", y="total_bill", hue="smoker", order=['Dinner',␣
↪'Lunch'],
                  data=tips, palette="Set1")
```

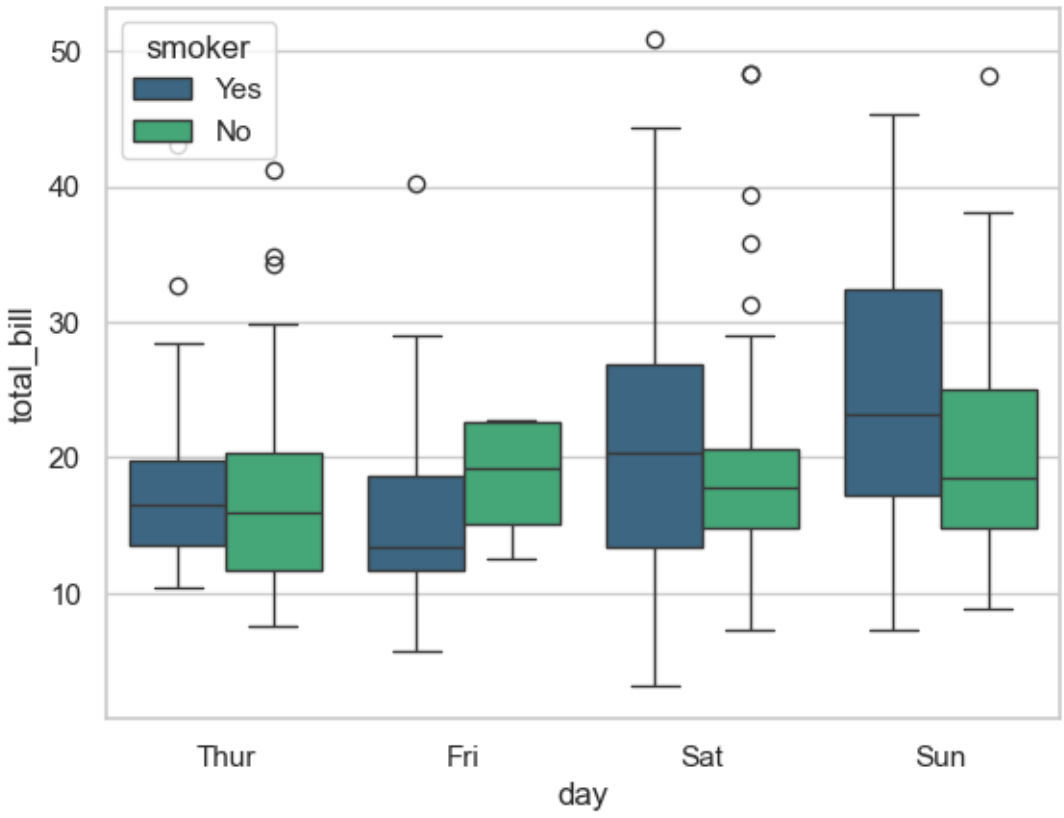


9.0.2 –>Group Boxplots

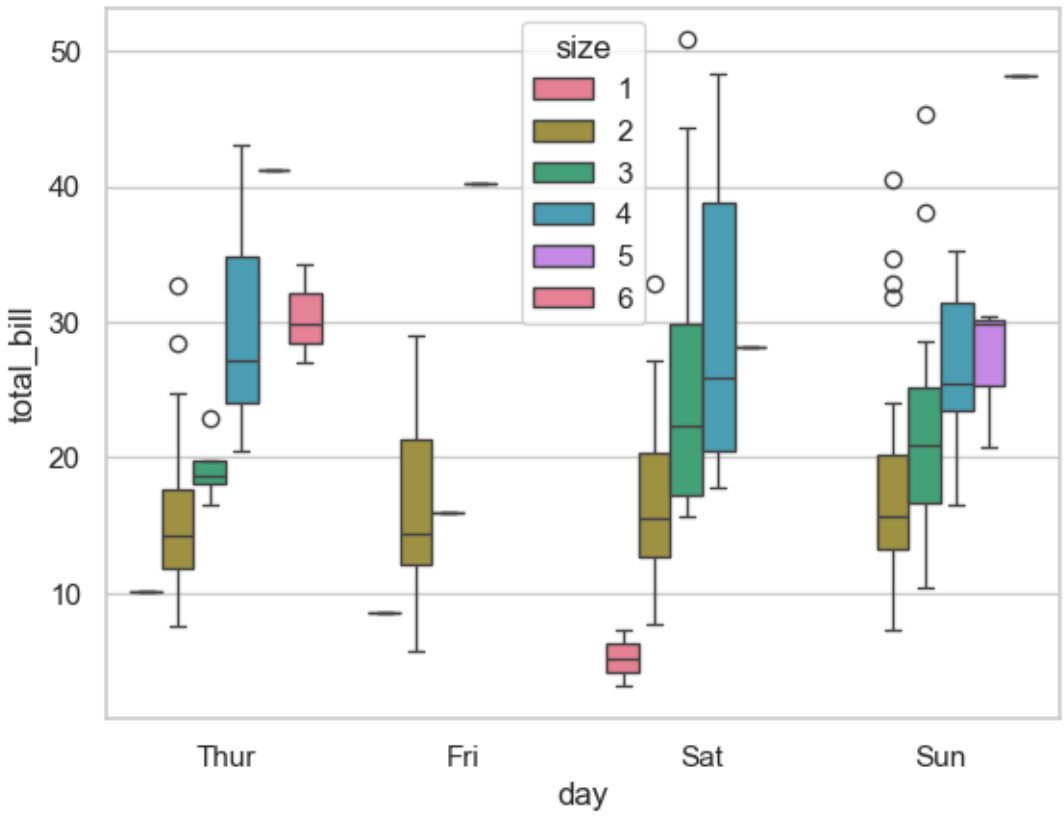
[129]:
```
# Create grouped boxplot
sns.boxplot(x=tip['day'], y=tip["total_bill"], hue=tip["sex"])
plt.show()
```



[130]:
```
# palette adding
sns.boxplot(x=tip['day'], y=tip["total_bill"],␣
  ↪hue=tip["smoker"],palette='viridis')
plt.show()
```

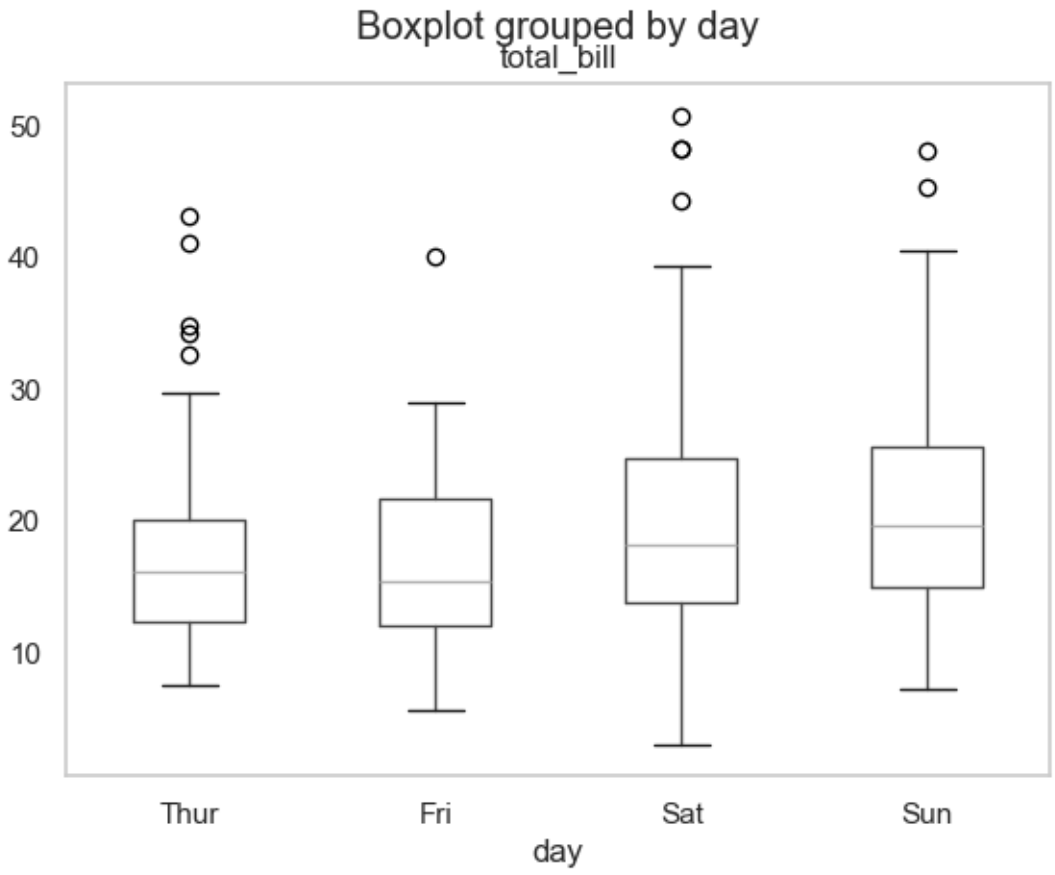


[131]:
```
sns.boxplot(x=tip['day'], y=tip["total_bill"], hue=tip["size"], palette='husl')
plt.show()
```



9.0.3 Box plot

[133]:
```
# A box plot consist of 5 things.
# Minimum
# First Quartile or 25%
# Median (Second Quartile) or 50%
# Third Quartile or 75%
# Maximum


tip.boxplot(by='day', column=['total_bill'], grid=False)
plt.show()
```

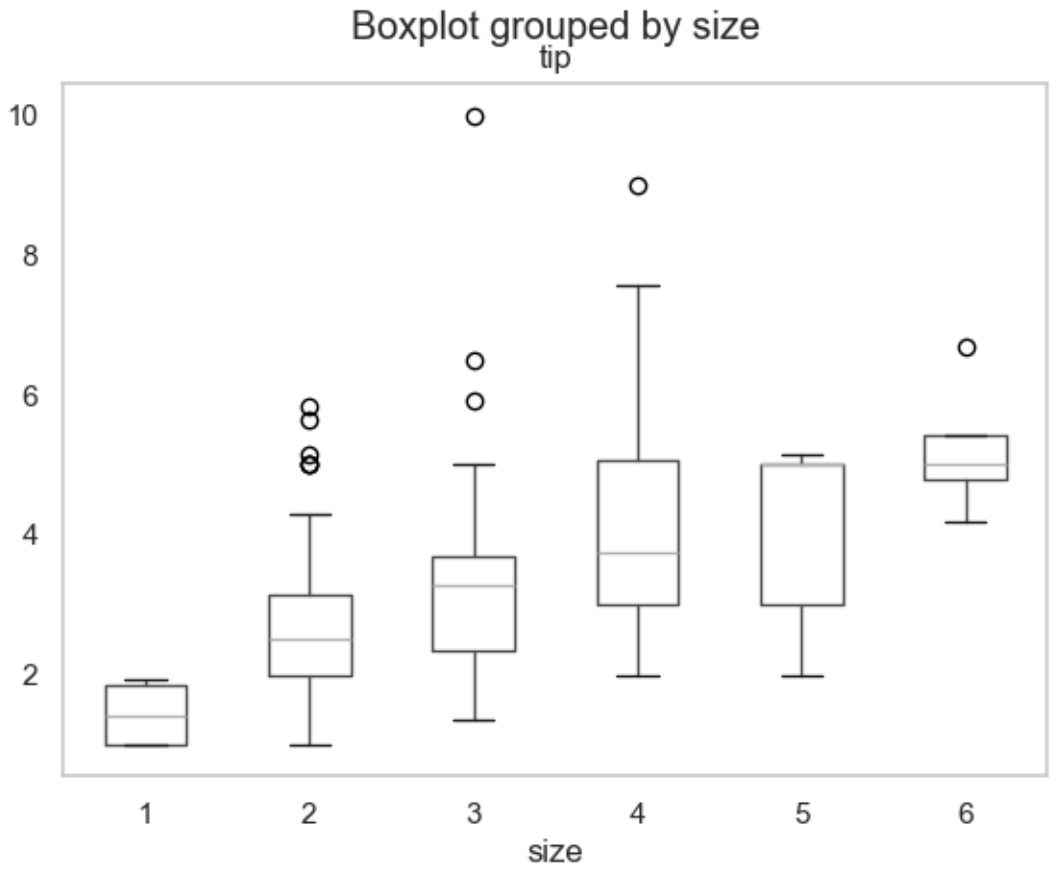


[134]:
```
tip.head()
```

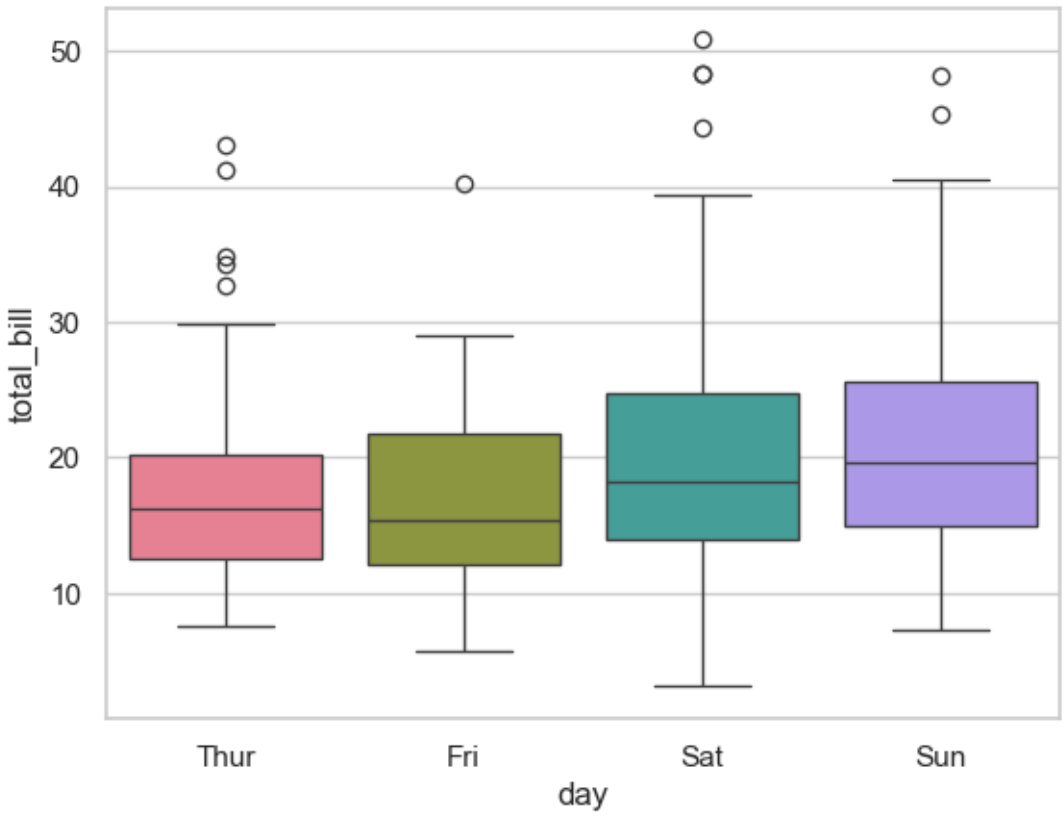
[134]:

	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

[135]:
```
tip.boxplot(by='size', column=['tip'], grid=False)
plt.show()
```



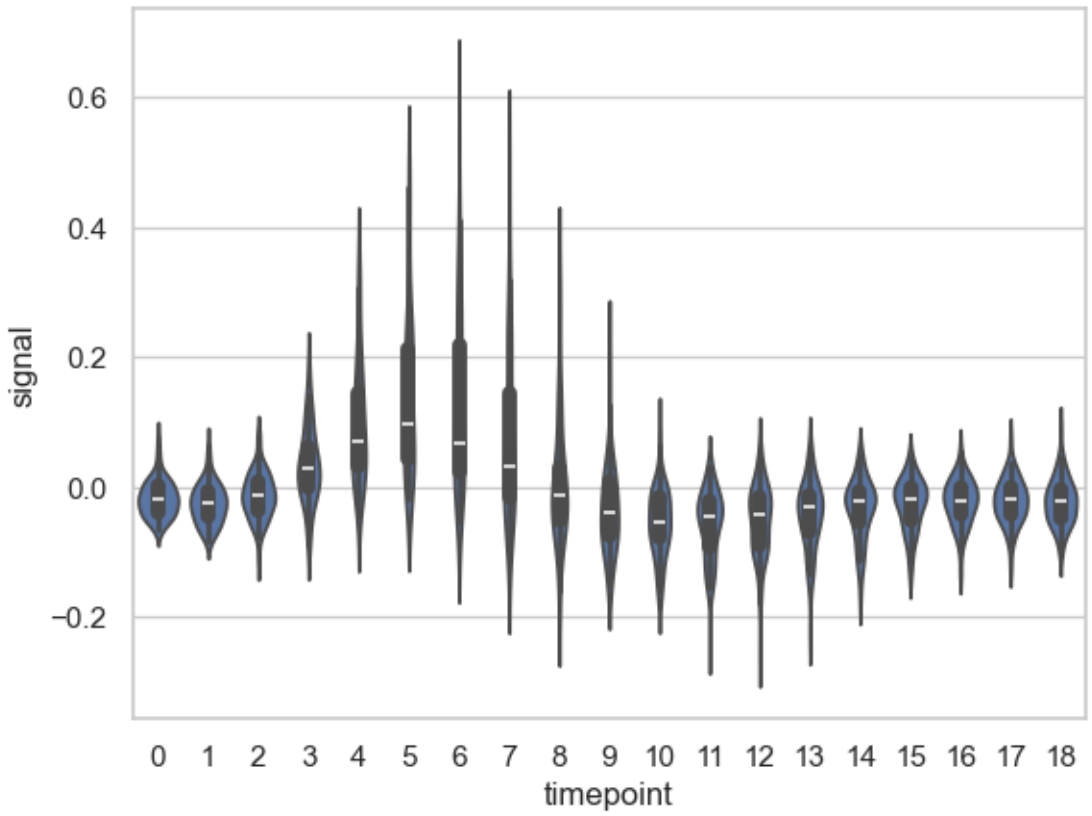
[136]:
```
# Draw a verical boxplot grouped
# by a categorical variable
sns.set_style("whitegrid")
sns.boxplot(x='day', y='total_bill',data=tip, palette='husl')
plt.show()
```



10 Violin plot

[138]:
```
# Syntax: seaborn.violinplot(x=None, y=None, hue=None, data=None, order=None,␣
↪hue_order=None, bw='scott', cut=2, scale='area', scale_hue=True,␣
↪gridsize=100, width=0.8, inner='box', split=False, dodge=True, orient=None,␣
↪linewidth=None, color=None, palette=None, saturation=0.75, ax=None, **kwargs)
```

[139]:
```
# Basic visualization
sns.set(style='whitegrid')
fmri=sns.load_dataset('fmri')
sns.violinplot(x='timepoint', y='signal', data=fmri)
plt.show()
```



[140]:
```
fmri['subject'].unique()
```

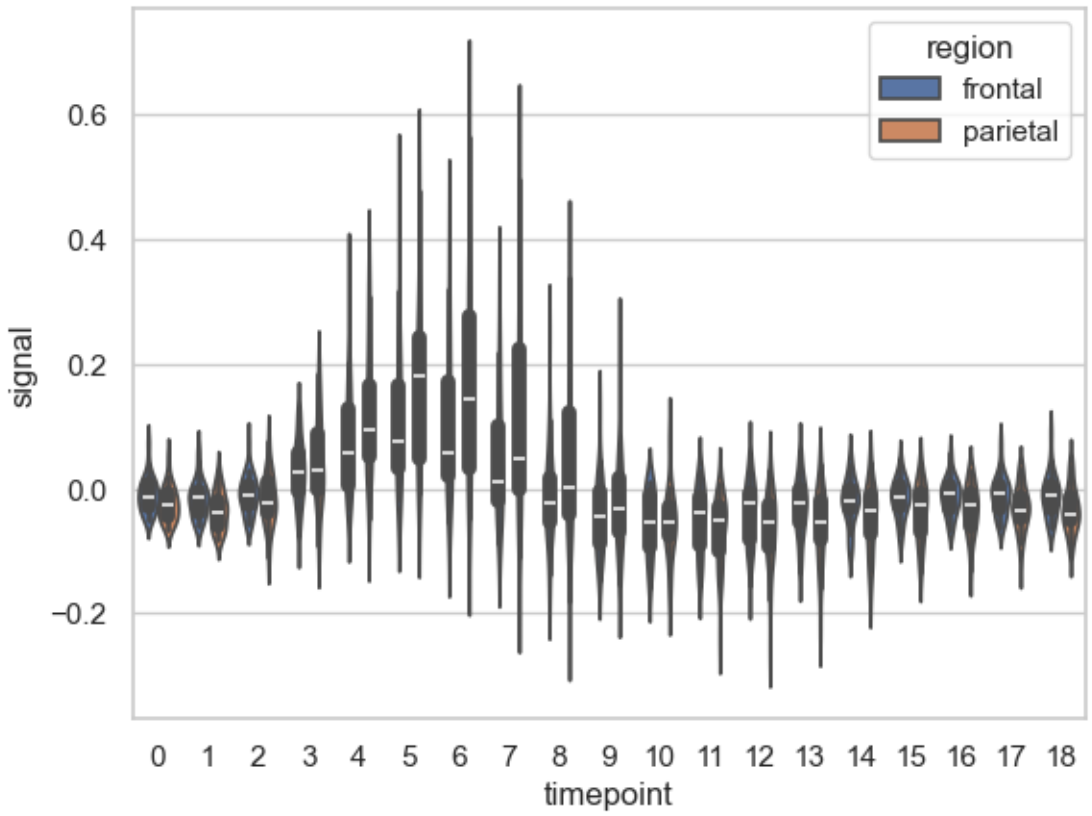
[140]:
```
array(['s13', 's5', 's12', 's11', 's10', 's9', 's8', 's7', 's6', 's4',
       's3', 's2', 's1', 's0'], dtype=object)
```

[141]:
```
fmri.head()
```

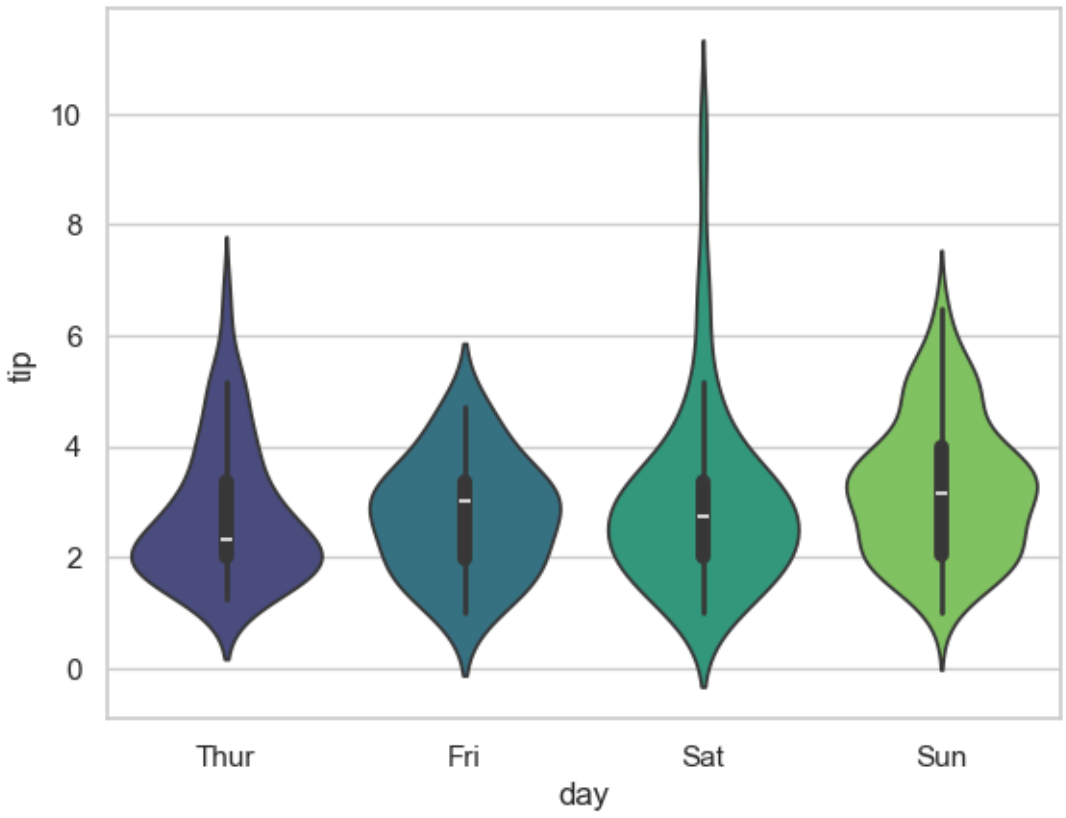
[141]:

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

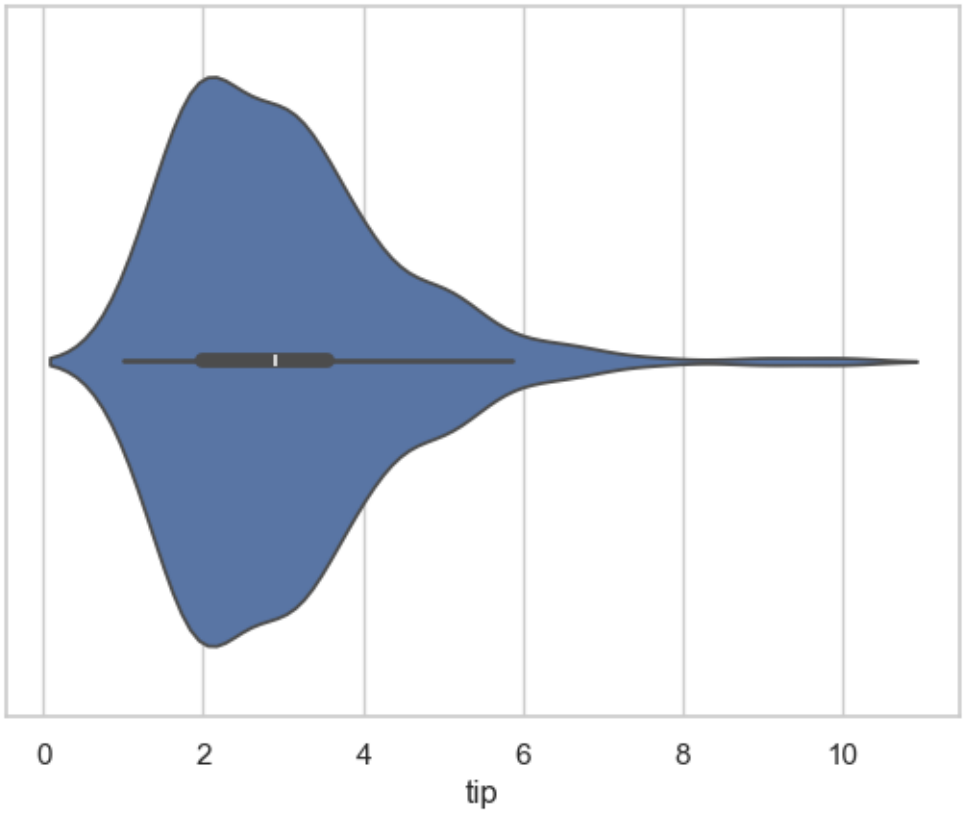
[142]:
```
sns.set(style='whitegrid')
sns.violinplot(x='timepoint', y='signal', hue='region', data=fmri)
plt.show()
```



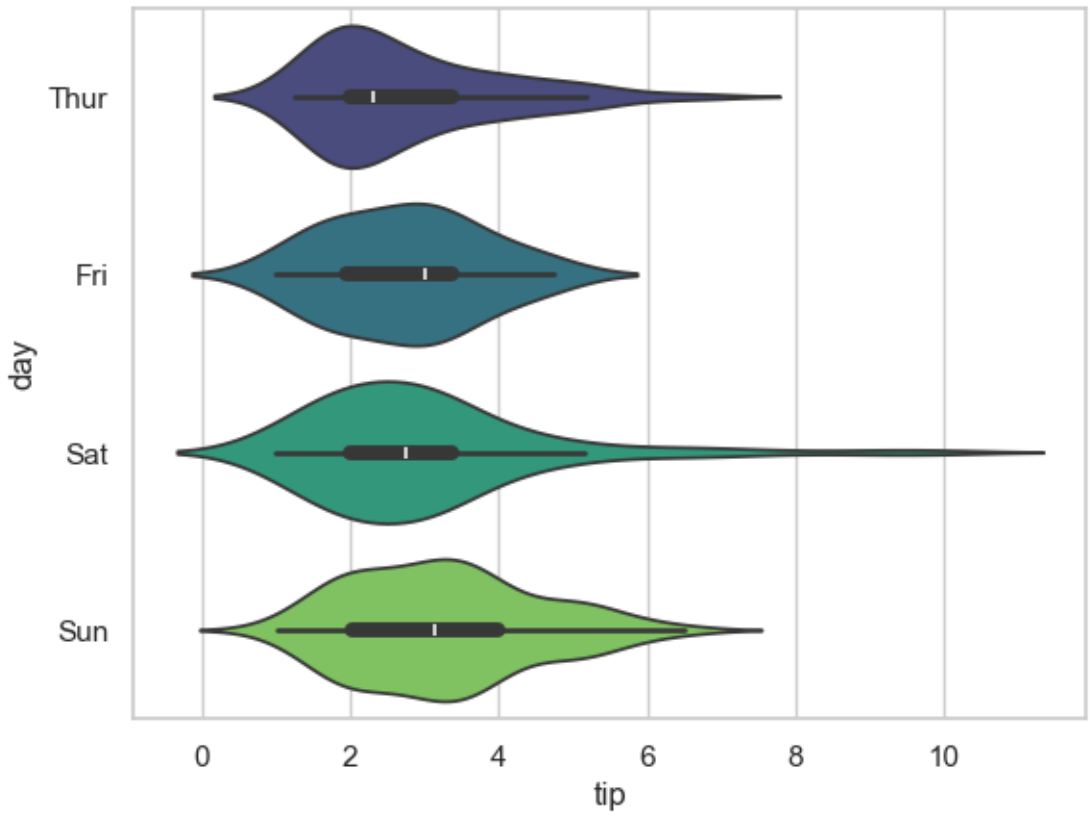
[143]:
```
sns.set(style='whitegrid')
sns.violinplot(x='day', y='tip', data=tip, palette='viridis')
plt.show()
```



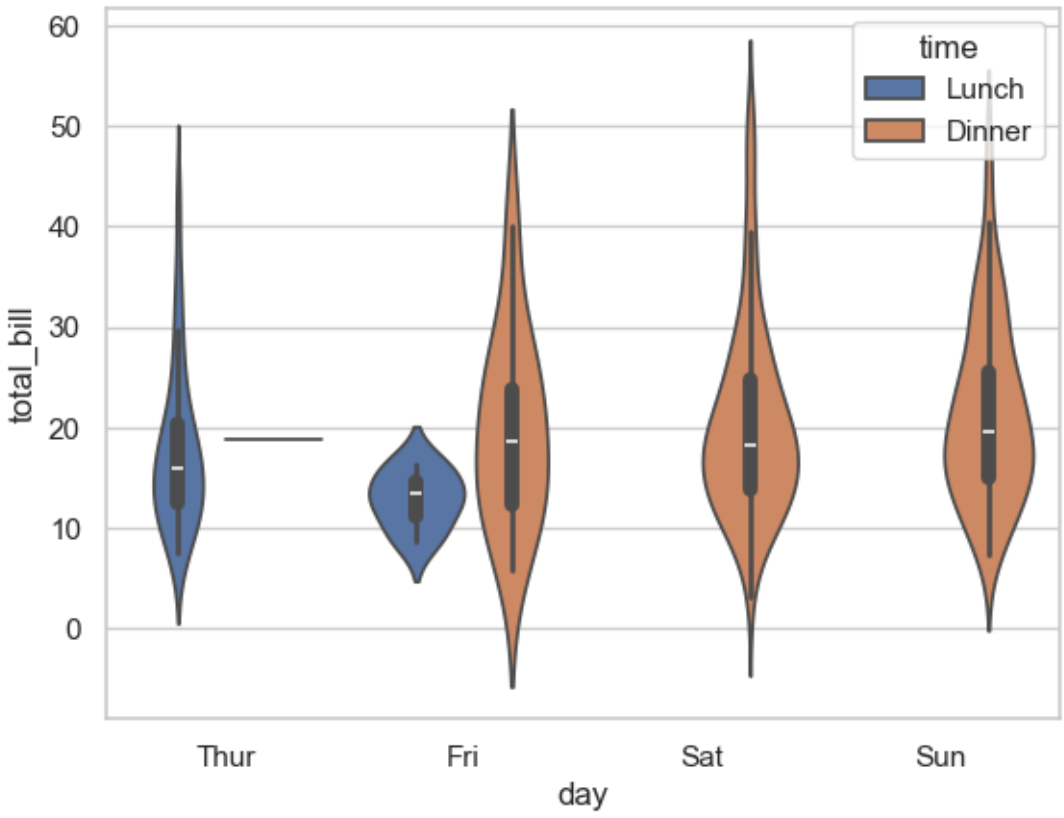
[144]:
```
# draw single horizontal swarm plot using only one axis
sns.violinplot(x=tip['tip'])
plt.show()
```



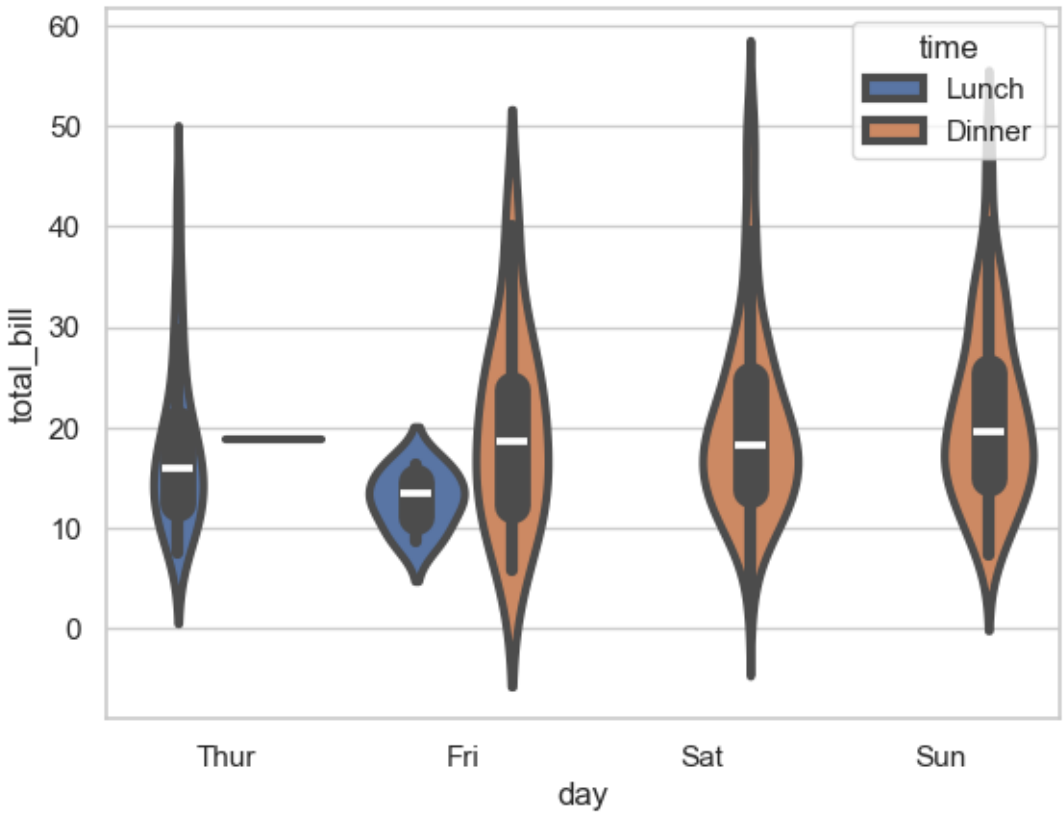
[145]:
```
# if you want to horizontal
sns.violinplot(x='tip', y='day', data=tip, palette='viridis')
plt.show()
```



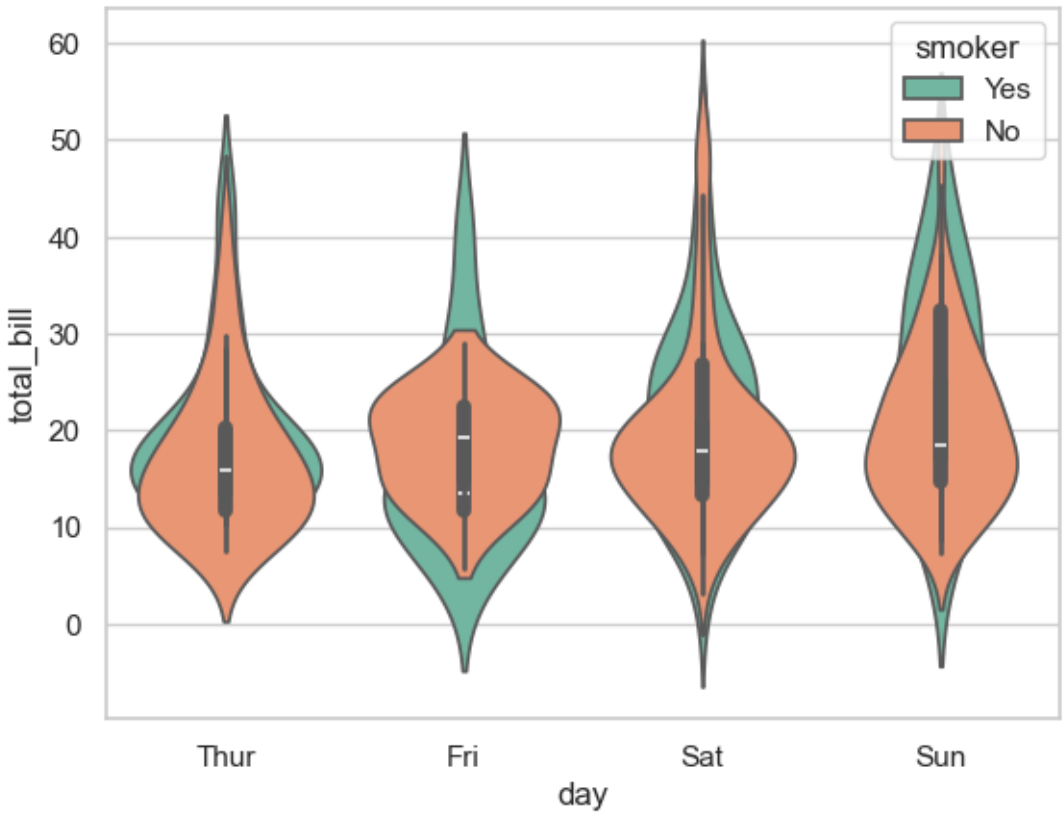
[146]:
```
# using "hue" parameter
sns.violinplot(x='day', y='total_bill', hue='time', data=tip)
plt.show()
```



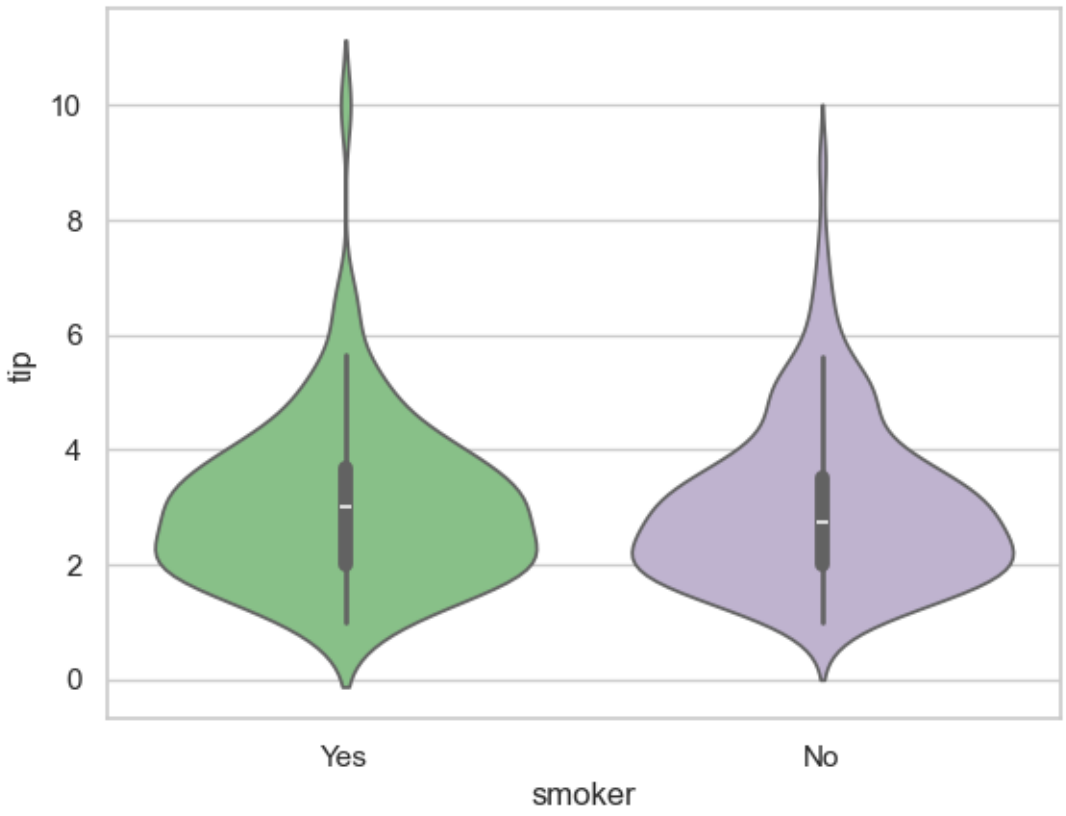
[147]:
```
# Draw outlines around the data points using "linewidth"
sns.violinplot(x='day', y='total_bill', hue='time', data=tip, linewidth=3)
plt.show()
```



[148]:
```
# Draw each level of the hue variable at different locations on the major␣
 ↪categorical axis
# once "dodge" argument we are using seperate the point for diff hue levels
# True, False
sns.violinplot(x='day', y='total_bill', hue='smoker', data=tip, dodge=False,␣
 ↪palette='Set2')
plt.show()
```

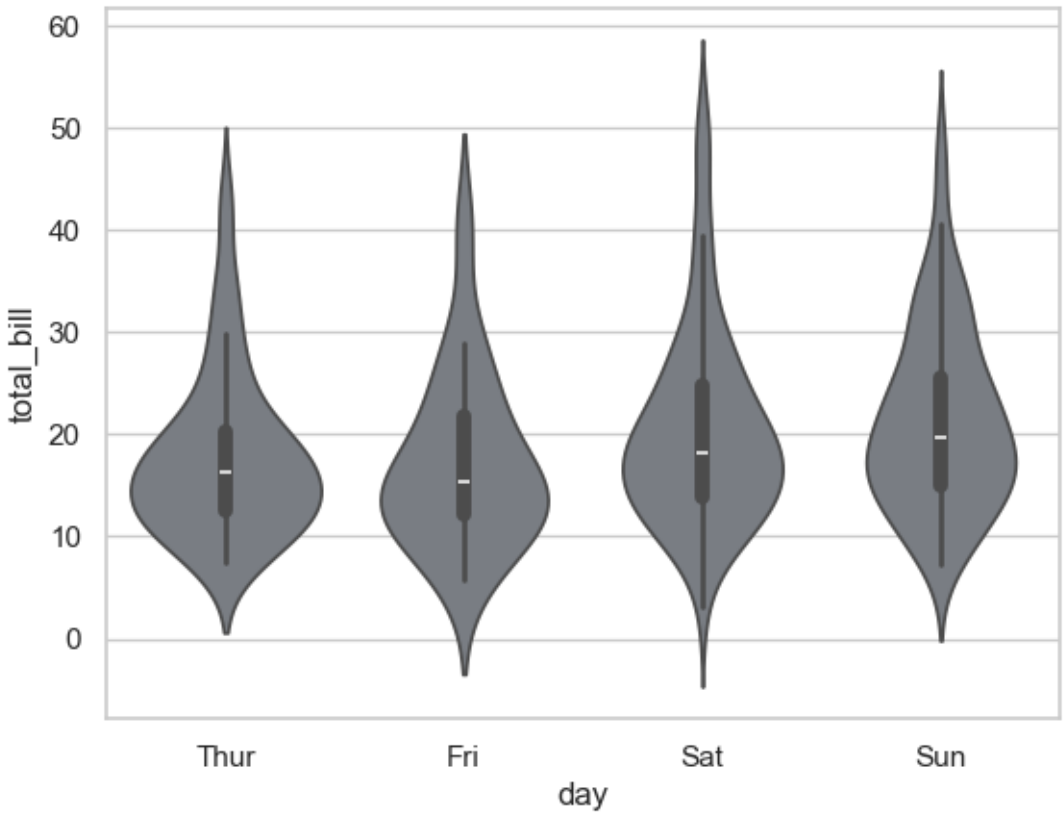


[149]:
```
# vionlin order by passing an expplicit order
sns.violinplot(x='smoker', y='tip', data=tip, order=['Yes', 'No'],␣
 ↪palette='Accent')
plt.show()
```



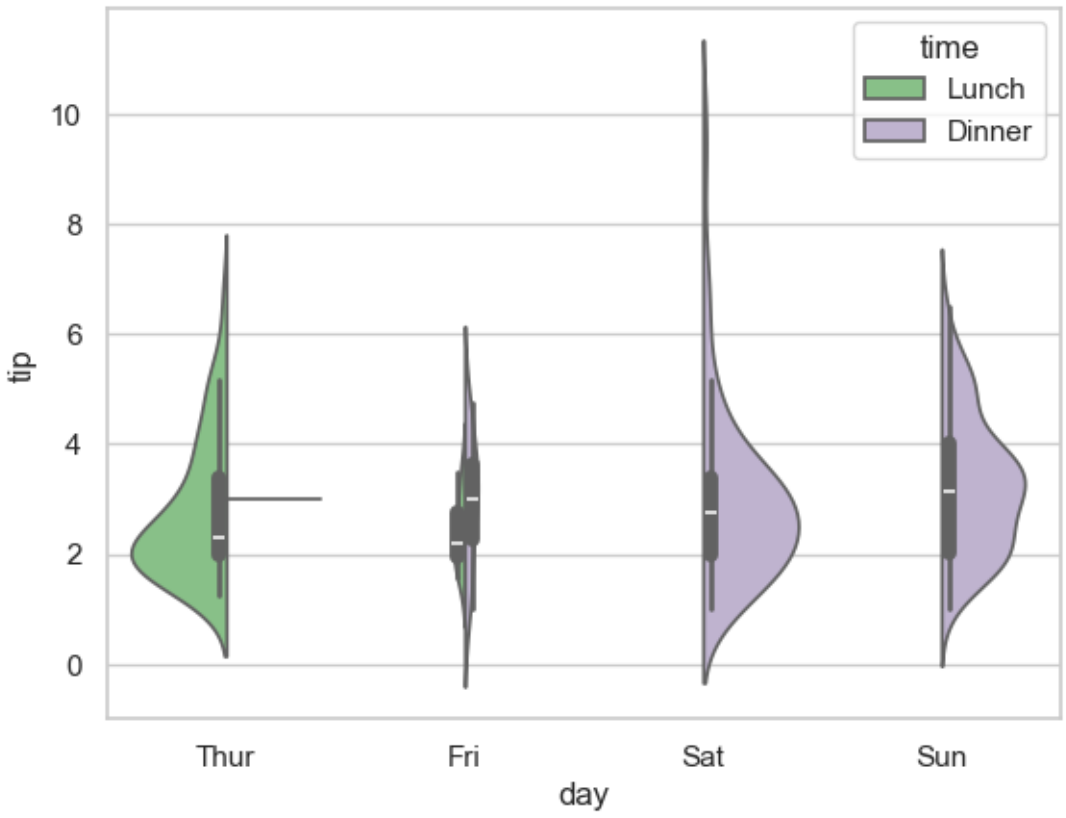
[150]:
```
# Adding palette attribute
```

[151]:
```
# adding Saturated parameter
sns.violinplot(x='day', y='total_bill',saturation=0.1, data=tip)
plt.show()
```



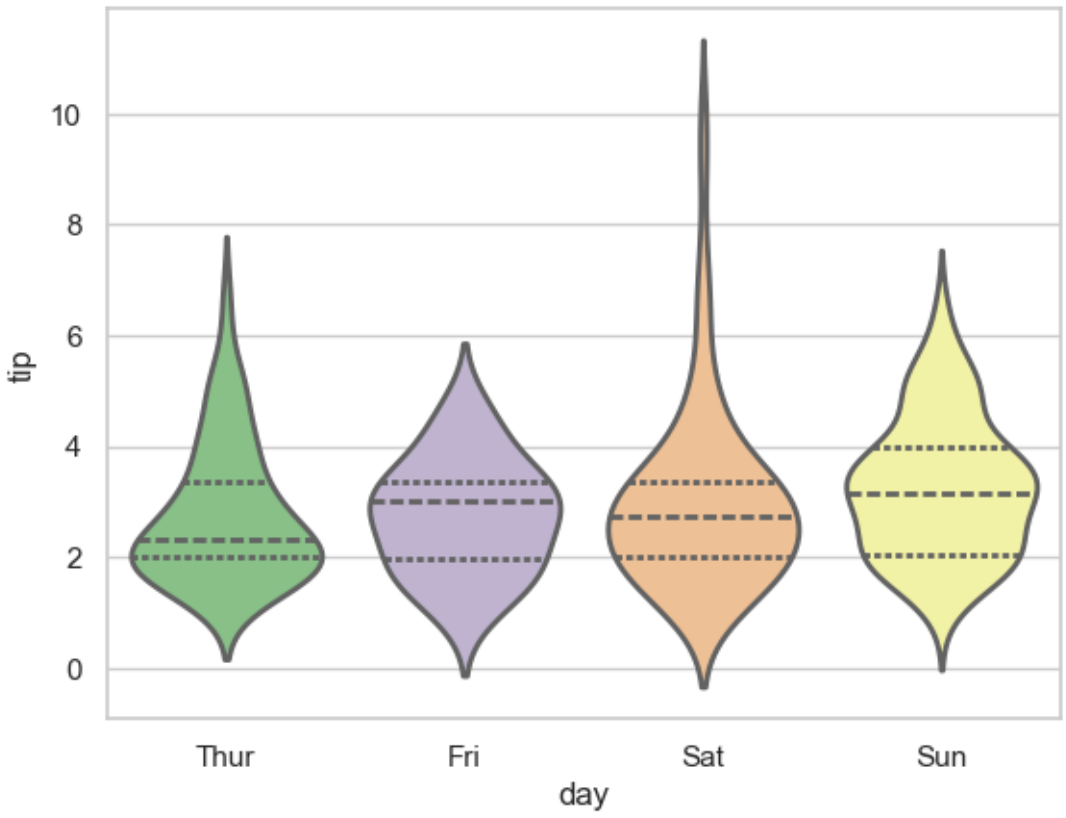
[152]:
```
# we can use color parameter
```

[153]:
```
# Scale the violin width by the number of obseravtions in each bin
# scale='area' or default, 'count', 'width'.
sns.violinplot(x='day', y='tip', hue='time', data=tip, palette='Accent',␣
 ↪scale='count', split=True)
plt.show()
```



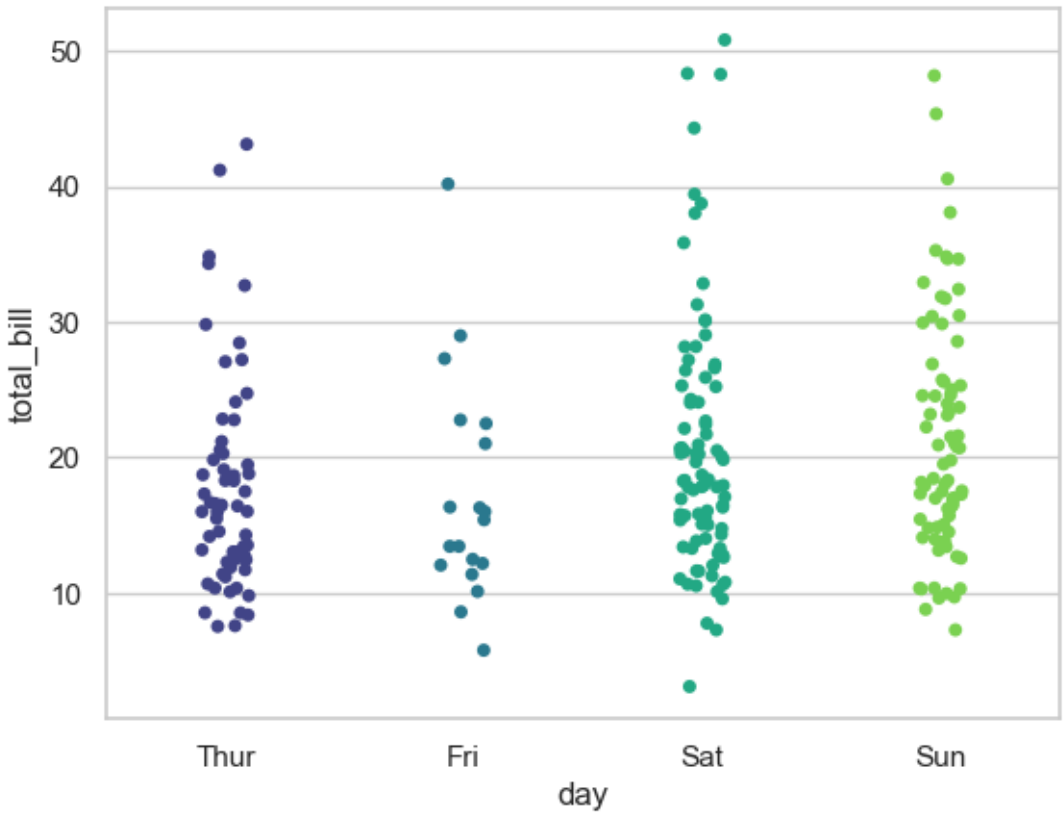
[154]:
```
# how to make grouped violinpot
# Syntax: seaborn.violinplot(x, y, hue, data, inner,linewidth,…)
 # inner: {"box", "quartile", "point", "stick", None}, (optional) : Represents␣
 ↪datapoints in the violin interior.
```

[155]:
```
sns.violinplot(x='day', y='tip', inner='quart', linewidth=2, data=tip,␣
 ↪palette='Accent')
plt.show()
```

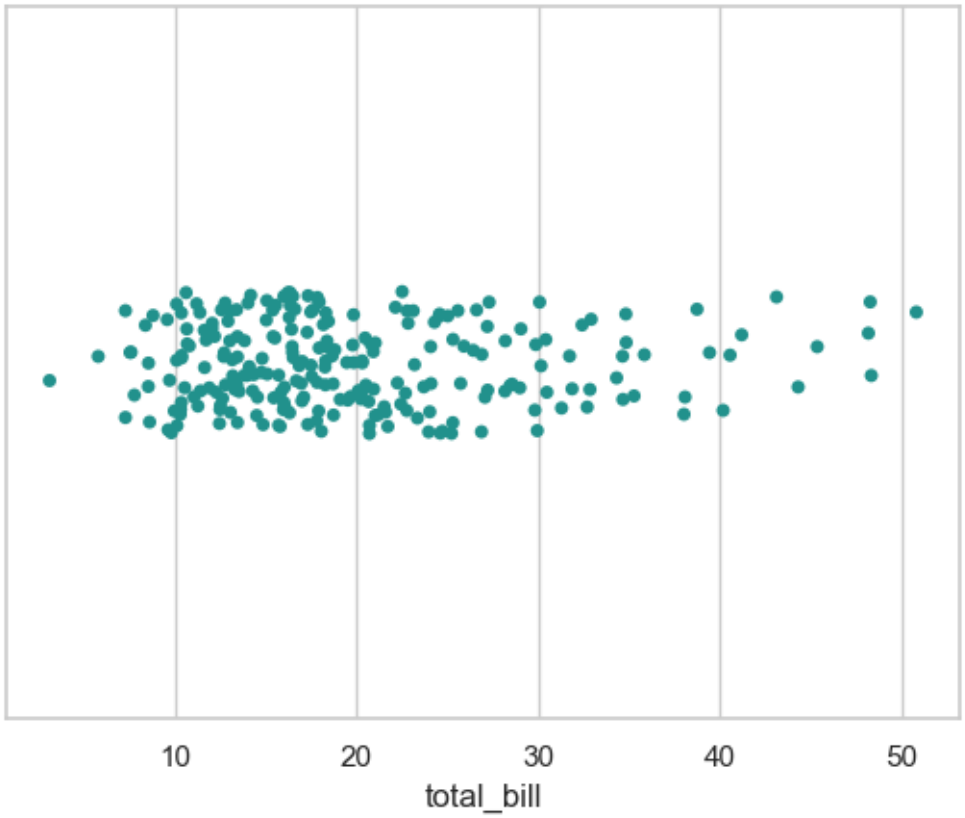


11 Strip plot

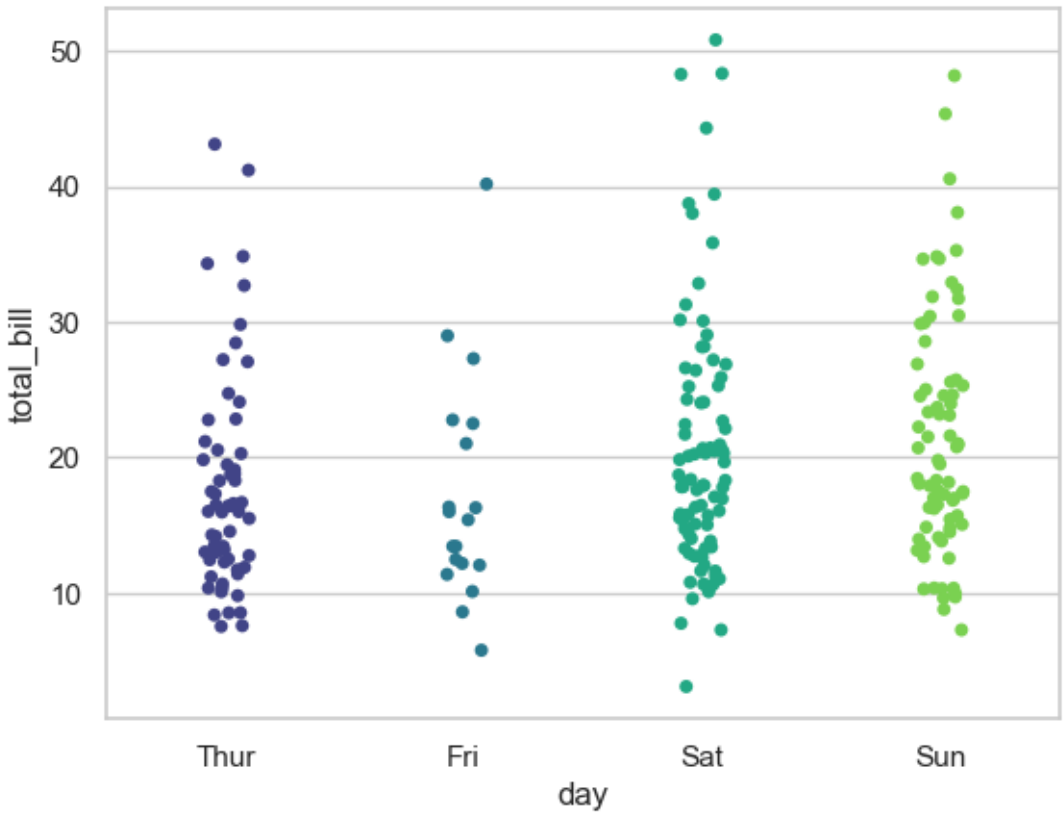
[157]:
```
# A strip plot is drawn on its own.
# It is a good complement to a boxplot or violinplot in cases where all␣
 ↪observations are shown along with some representation of the underlying␣
 ↪distribution.
# It is used to draw a scatter plot based on the category.

sns.set(style='whitegrid')
sns.stripplot(x='day', y='total_bill', data=tip, palette='viridis')
plt.show()
```



[158]:
```
# Draw a single horizontal strip plot using Stripplot
sns.stripplot(x='total_bill',  data=tip, palette='viridis')
plt.show()
```

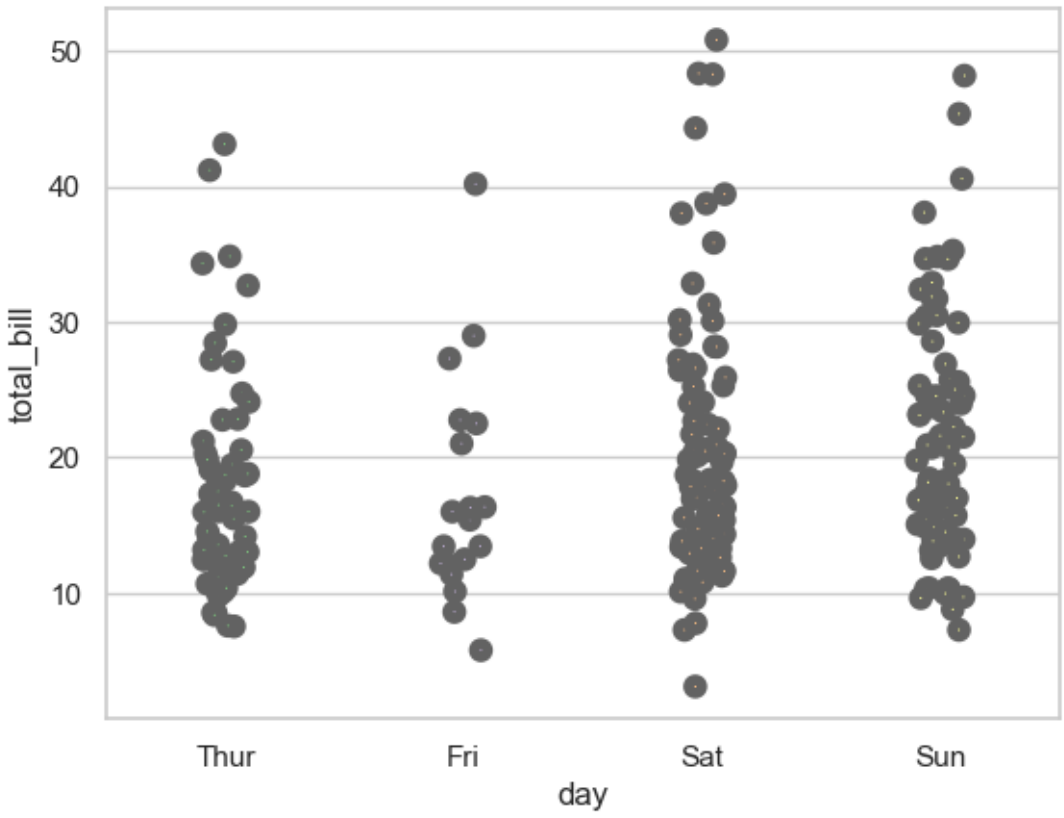


[159]:
```
# draw the strip plot using jitter parameter
# jitter can be used to provide displacements along the horizontal axis,
# which is useful when there are large clusters of data points.
# You can specify the amount of jitter (half the width of the uniform random␣
 ↪variable support),
# or just use True for a good default.

sns.stripplot(x='day', y='total_bill', jitter=0.1,data=tip, palette='viridis')
plt.show()
```



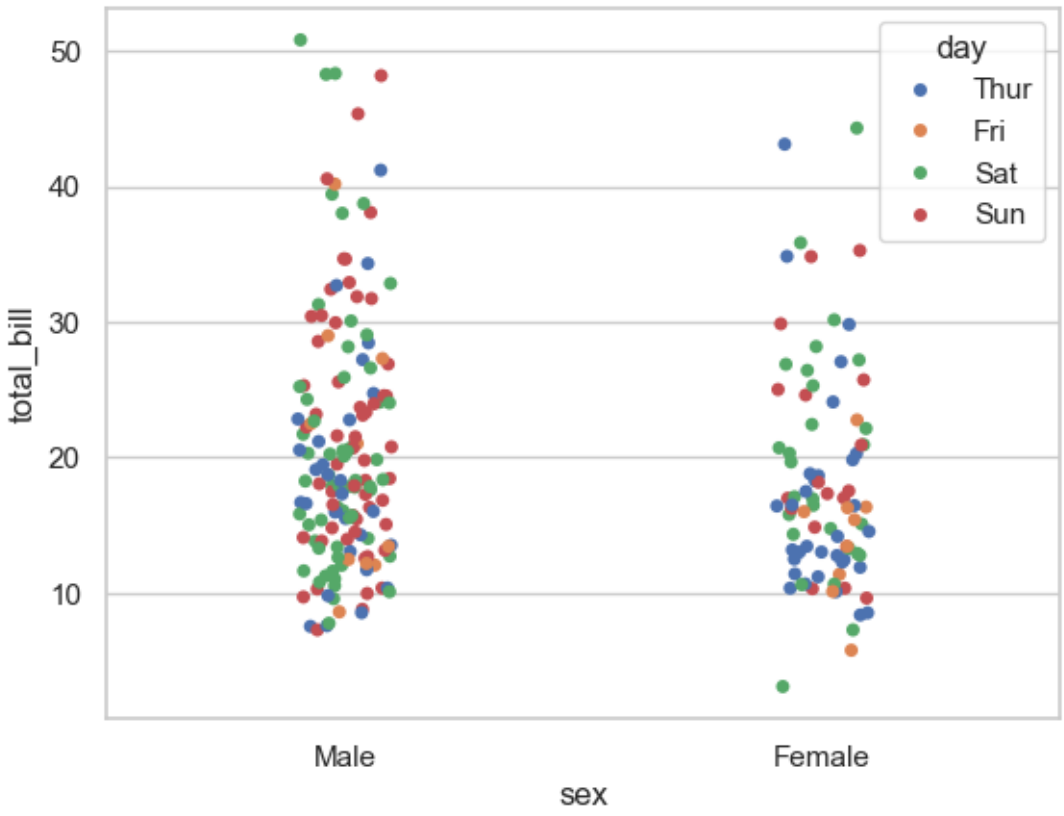
[160]:
```
# Draw outlines aroound the data points using linewidth
sns.stripplot(y="total_bill", x="day", data=tips,linewidth=4, palette='Accent')
```

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



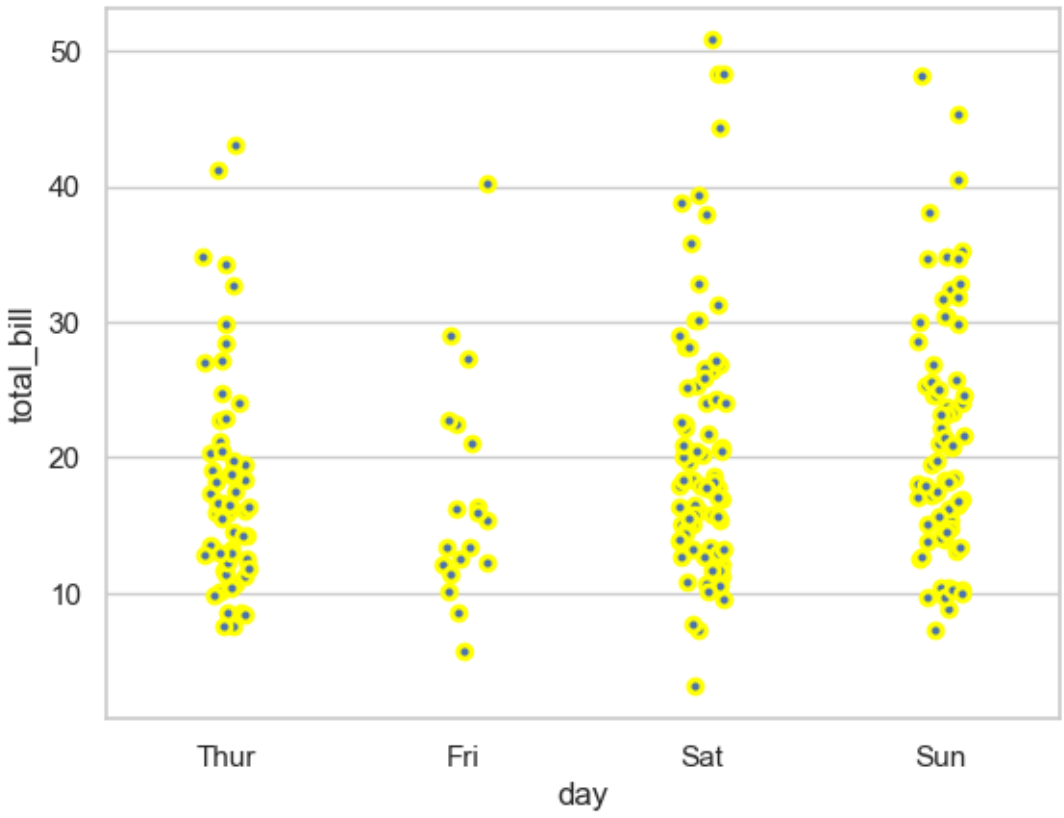
[161]:
```
sns.stripplot(y="total_bill", x="sex", data=tips, hue='day')
```

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

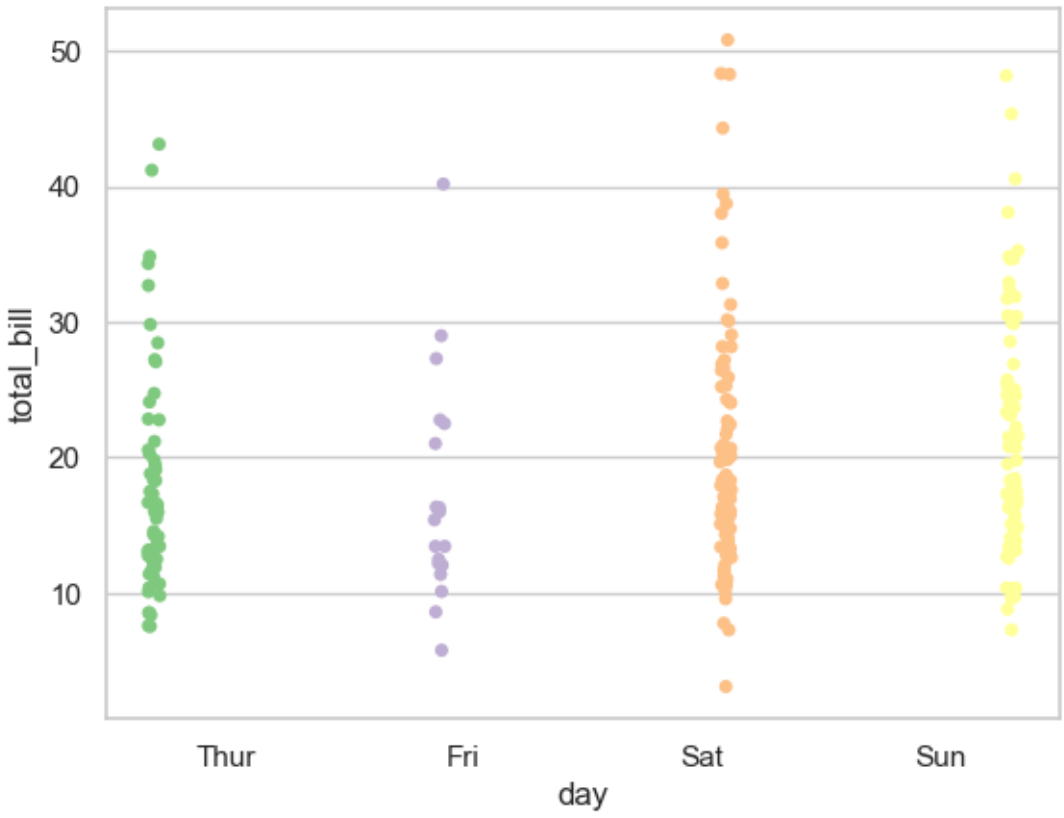


[162]:
```
# edge color
sns.stripplot(y="total_bill", x="day", data=tip, linewidth=2,edgecolor='yellow')
```

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

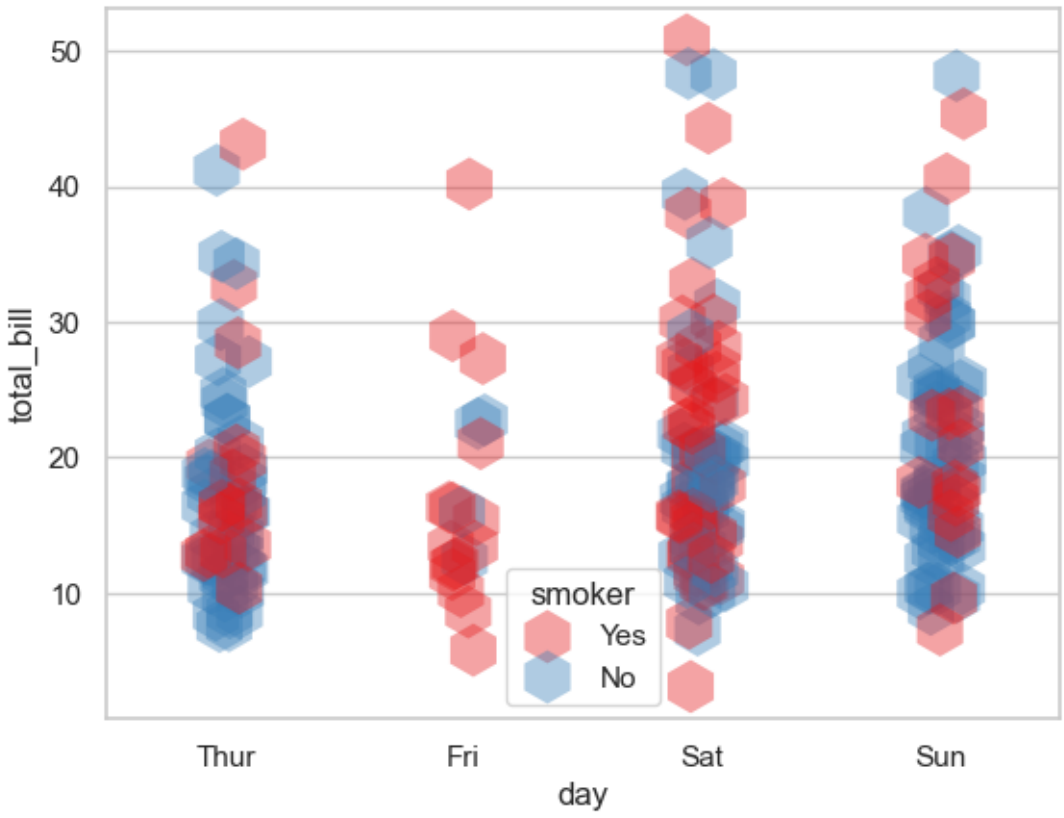


[163]:
```
# Draaw each level pf the hue variable at different locations on the major␣
 ↪categorical axis
# When using hue nesting, setting dodge should be True will separate the strips
# for different hue levels along the categorical axis. And Palette is used for␣
 ↪the
# different levels of the hue variable.
sns.stripplot(y="total_bill", x="day", data=tip, dodge=True, palette='Accent')
plt.show()
```



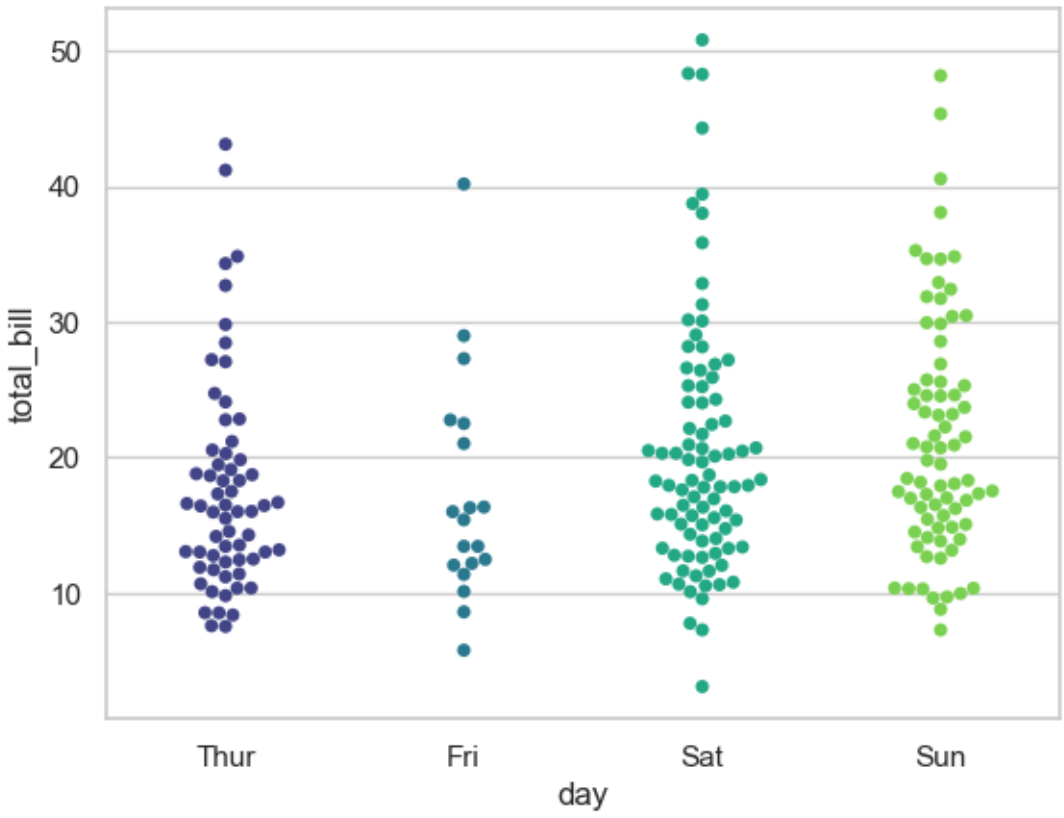
[164]:
```
# plotting large points and differnt aesthetics with marker and alpha parameter
# we will uses alpha to manage transparency of the data point ,
# and use marker for to customize the data point

sns.stripplot(x="day", y="total_bill", hue="smoker", data=tip, palette="Set1",␣
↪size=20, marker="h", alpha=0.4)
```

[164]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```

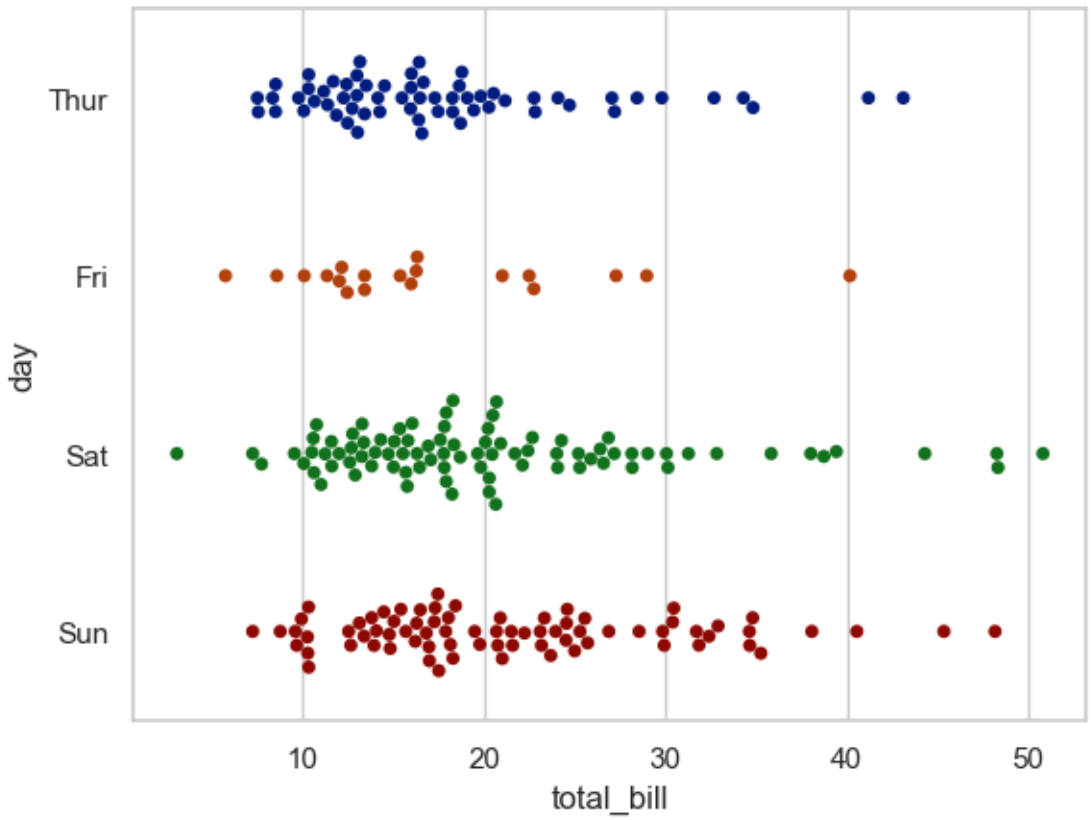


12 Swarm plot

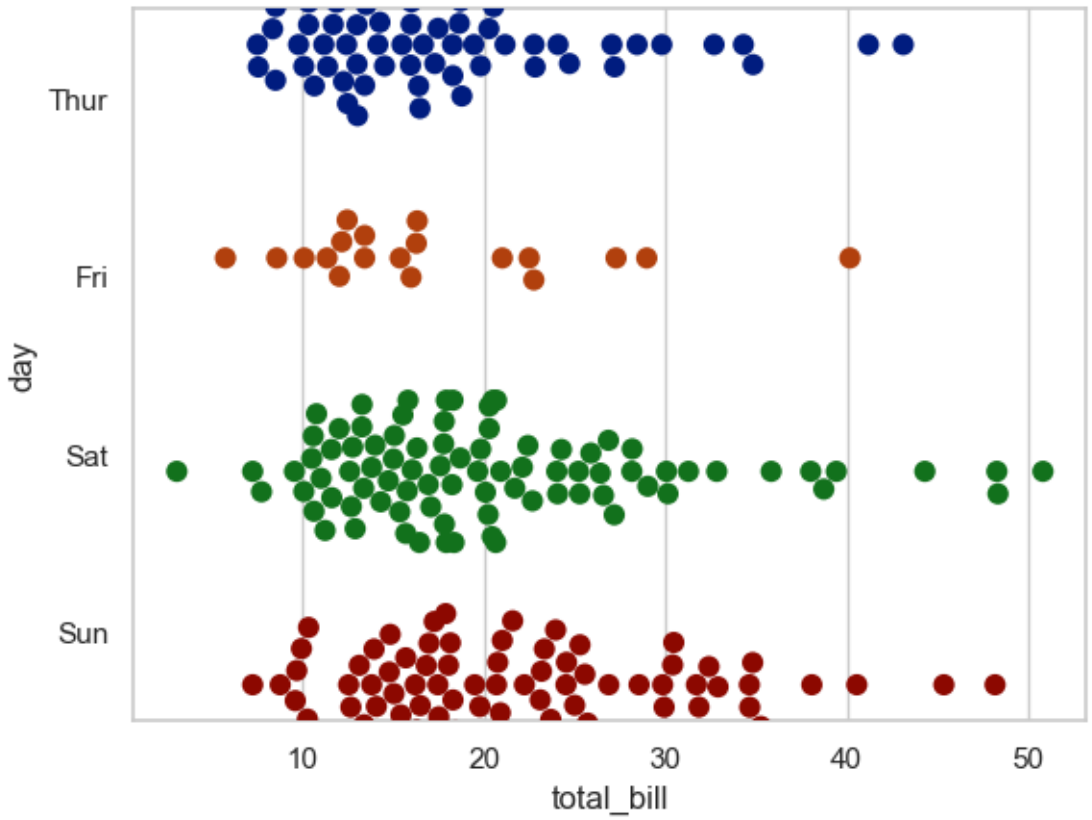
[166]:
```
# Syntax : seaborn.swarmplot(parameters)
# Draw a categorical scatterplot with non-overlapping points.
# A swarm plot can be drawn on its own, but it is also a good complement to a␣
 ↪box or violin plot
# in cases where you want to show all observations along with some␣
 ↪representation of
# the underlying distribution. Arranging the points properly requires an␣
 ↪accurate
# transformation between data and point coordinates. This means that␣
 ↪non-default axis
# limits must be set *before* drawing the plot.

sns.swarmplot(x ="day", y = "total_bill",  data =tip, size = 5,␣
 ↪palette='viridis')
plt.show()
```



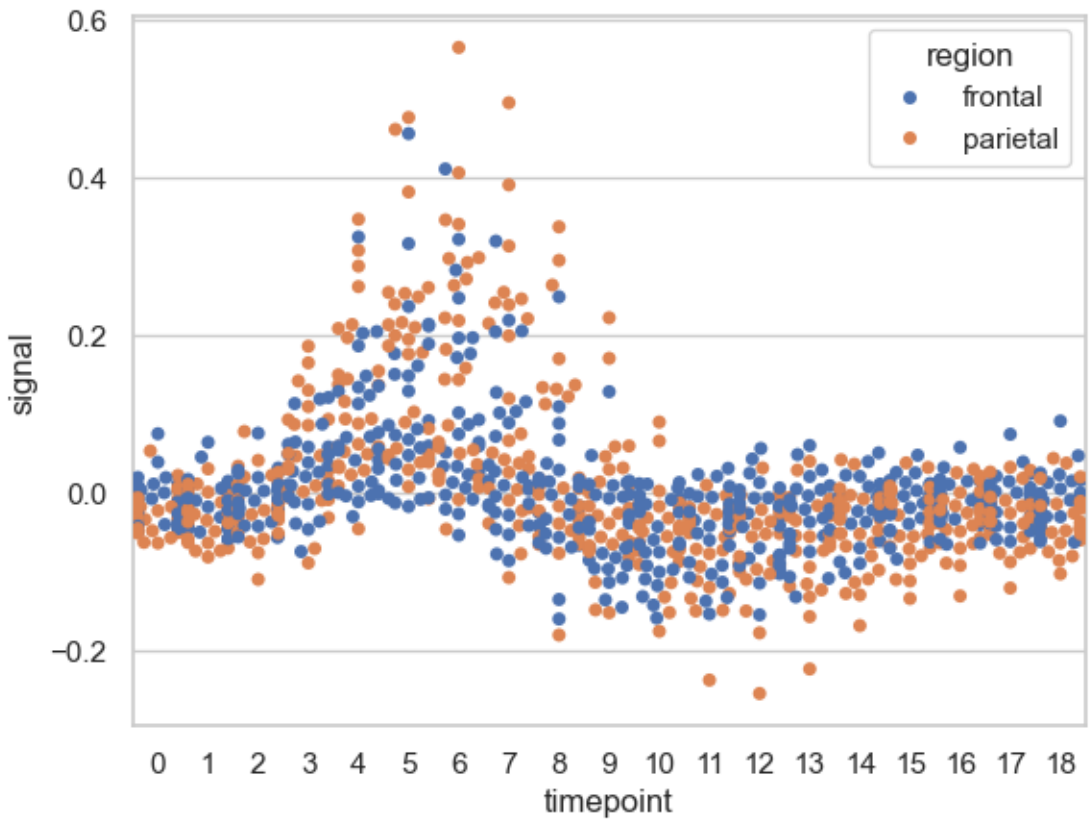
[167]:
```
sns.swarmplot(x ="total_bill", y = "day", orient='h', data =tip, size = 5,␣
↪palette='dark')
plt.show()
```



[168]:
```
sns.swarmplot(x ="total_bill", y = "day", orient='h', data =tip, size = 8,␣
↪dodge=True,palette='dark')
plt.show()
```



[169]:
```
sns.swarmplot(x="timepoint",y="signal",hue="region",data=fmri)
plt.show()
```



[170]:
```
# we Can perform color, edgecolor, palette, alpha, size,order,dodge, linewidth
```

13 Histogram plots

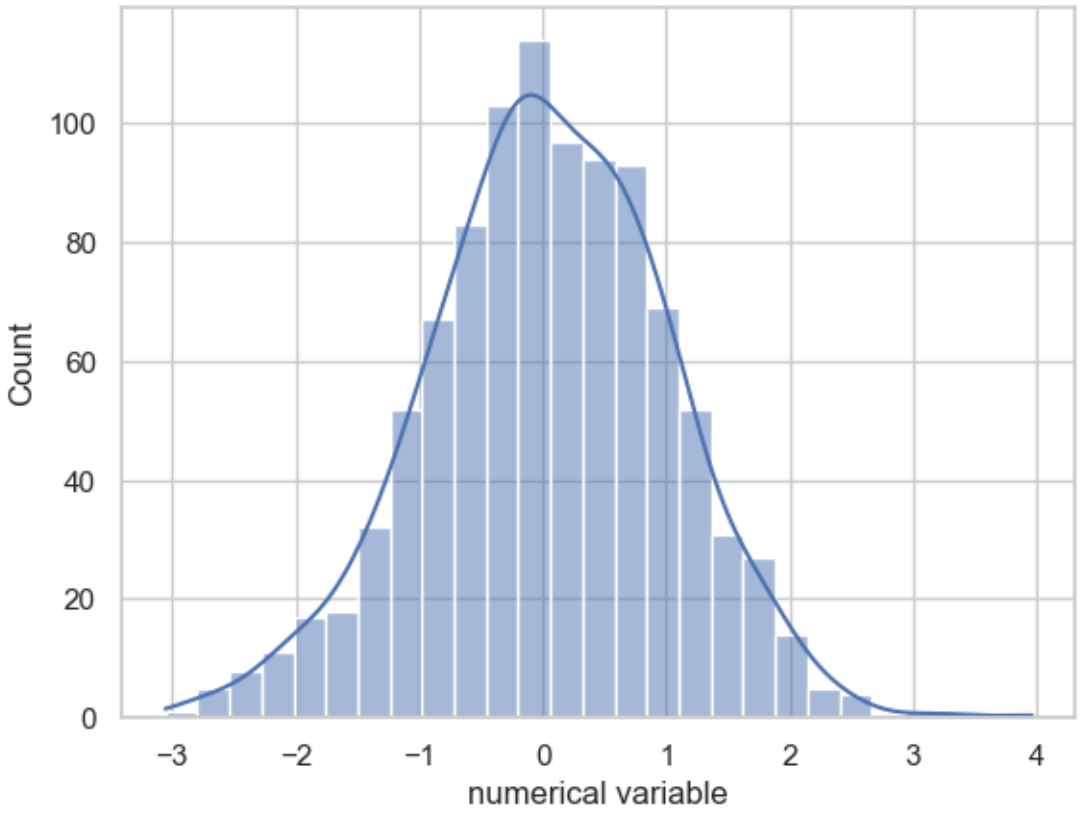
[172]:
```
# Histograms are visualization tools that represent the distribution of a set␣
 ↪of continuous data.
# In a histogram, the data is divided into a set of intervals or bins (usually␣
 ↪on the x-axis) and
# the count of data points that fall into each bin corresponding to the height␣
 ↪of the bar above that bin.
# These bins may or may not be equal in width but are adjacent (with no gaps).

# syntax= sns.histplot(data, x, y, hue, stat, bins, binwidth, discrete, kde,␣
 ↪log_scale)
```

[337]:
```
''' Kernel Density Estimation (KDE) '''
# A density plot (also known as kernel density plot) is another visualization␣
 ↪tool for evaluating data distributions.
```

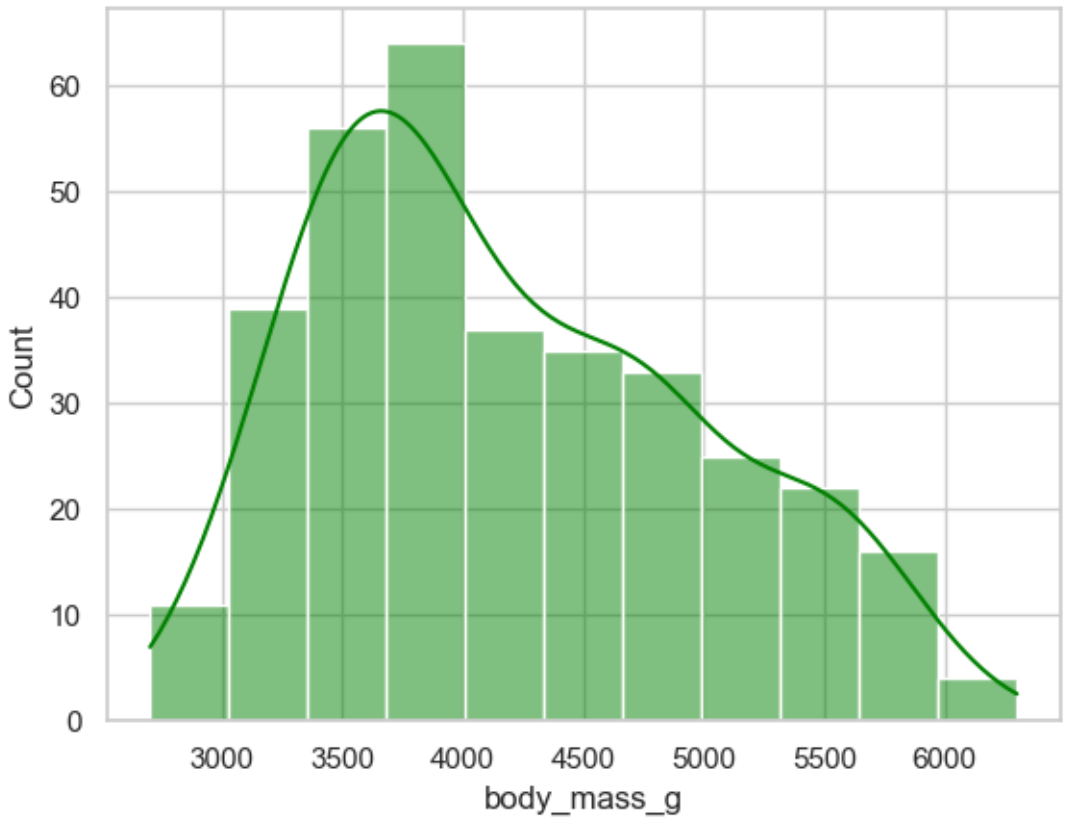
```
#  It can be considered as a smoothed histogram. The peaks of a density plot␣
 ↪help display where values are concentrated over the interval.
# There are a variety of smoothing techniques. Kernel Density Estimation (KDE)␣
 ↪is one of the techniques used to smooth a histogram.
```

[337]:
```
' Kernel Density Estimation (KDE) '
```

[341]:
```
np.random.seed(1)
num_var=np.random.randn(1000)
num_var=pd.Series(num_var, name="numerical variable")
sns.histplot(data=num_var, kde=True)
plt.show()
```



[349]:
```
pegn=sns.load_dataset("penguins")
sns.histplot(data=pegn, kde=True, x="body_mass_g", color='green')
plt.show()
```

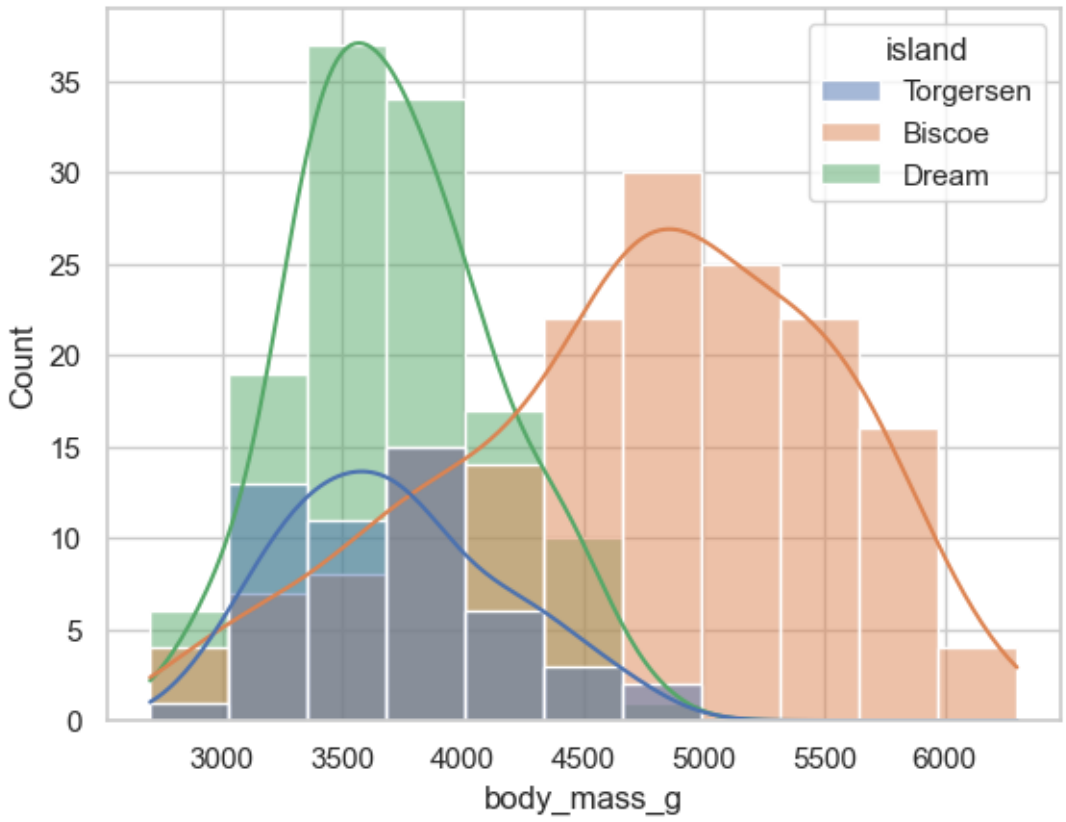


[351]:
```
pegn.head()
```

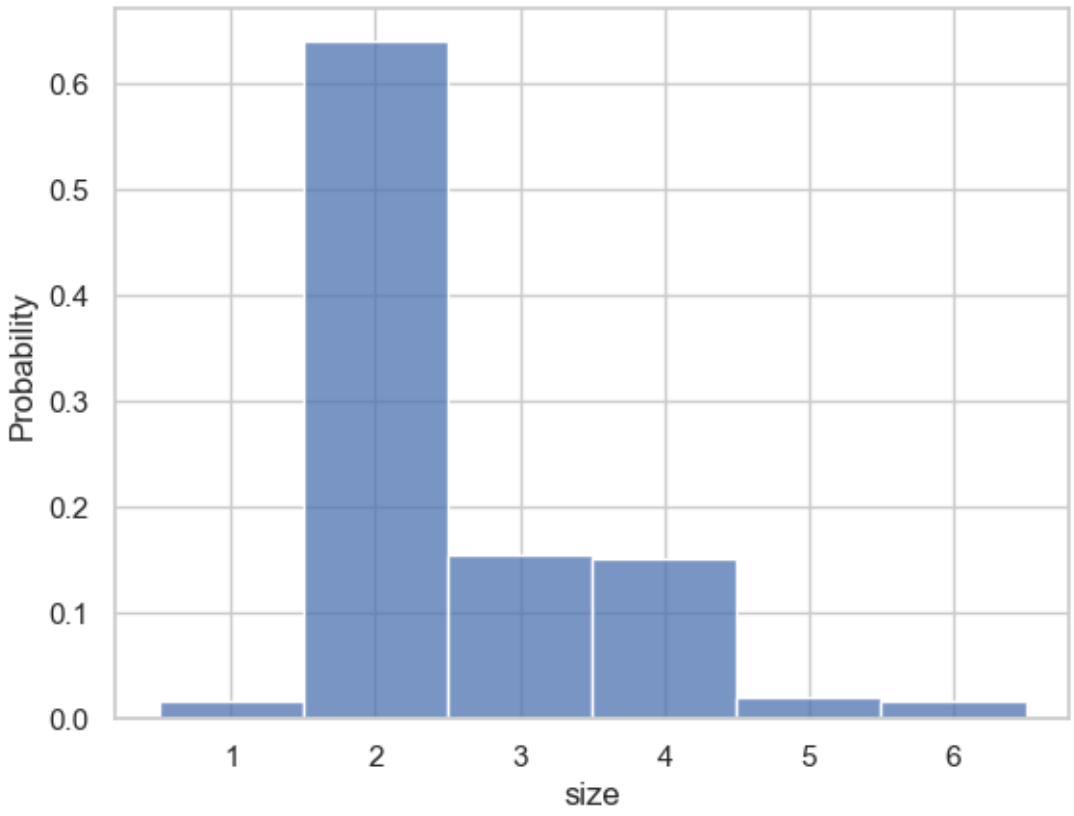
[351]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

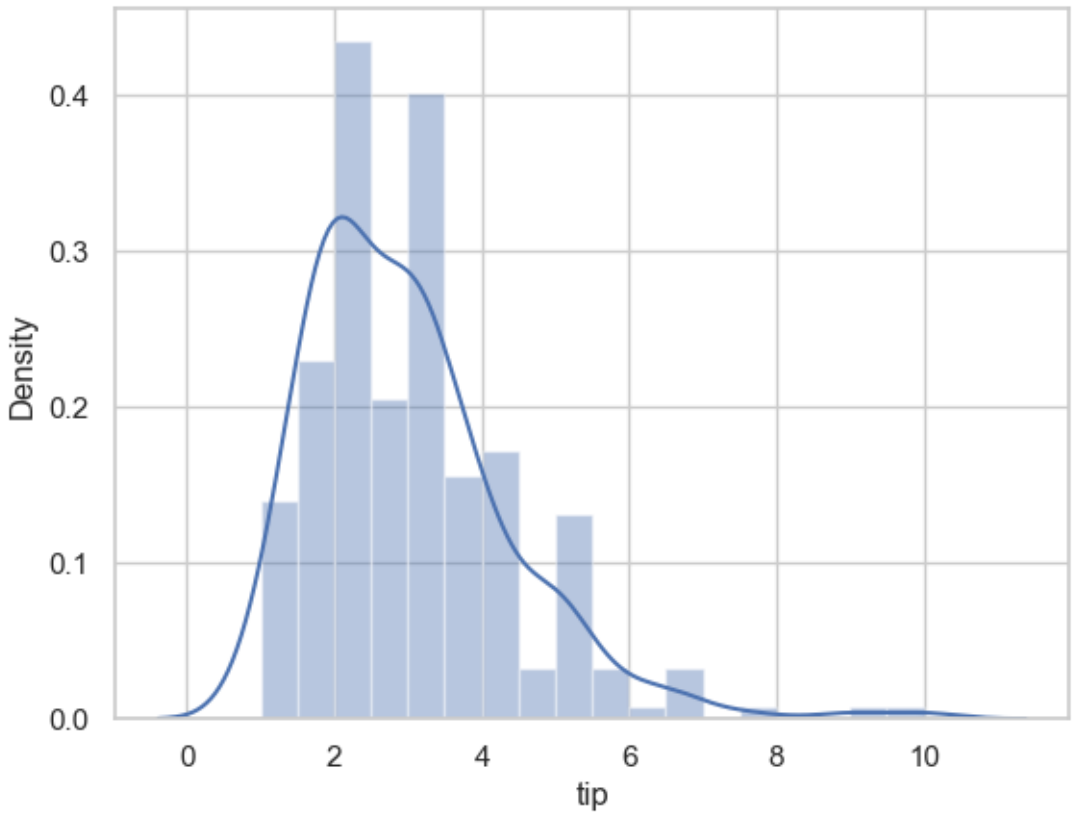
[359]:
```
sns.histplot(data=pegn, x='body_mass_g', hue='island', kde=True)
plt.show()
```



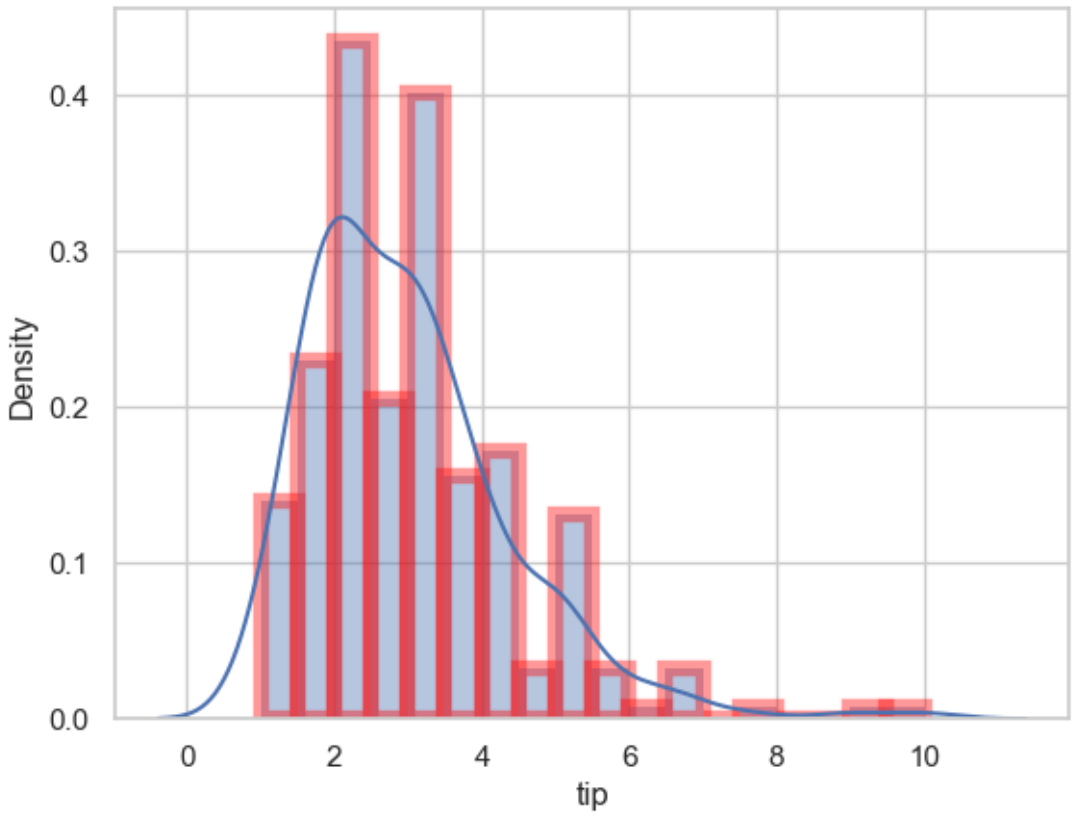
[361]:
```
sns.histplot(data=tip, x='size', stat='probability', discrete=True)
plt.show()
```



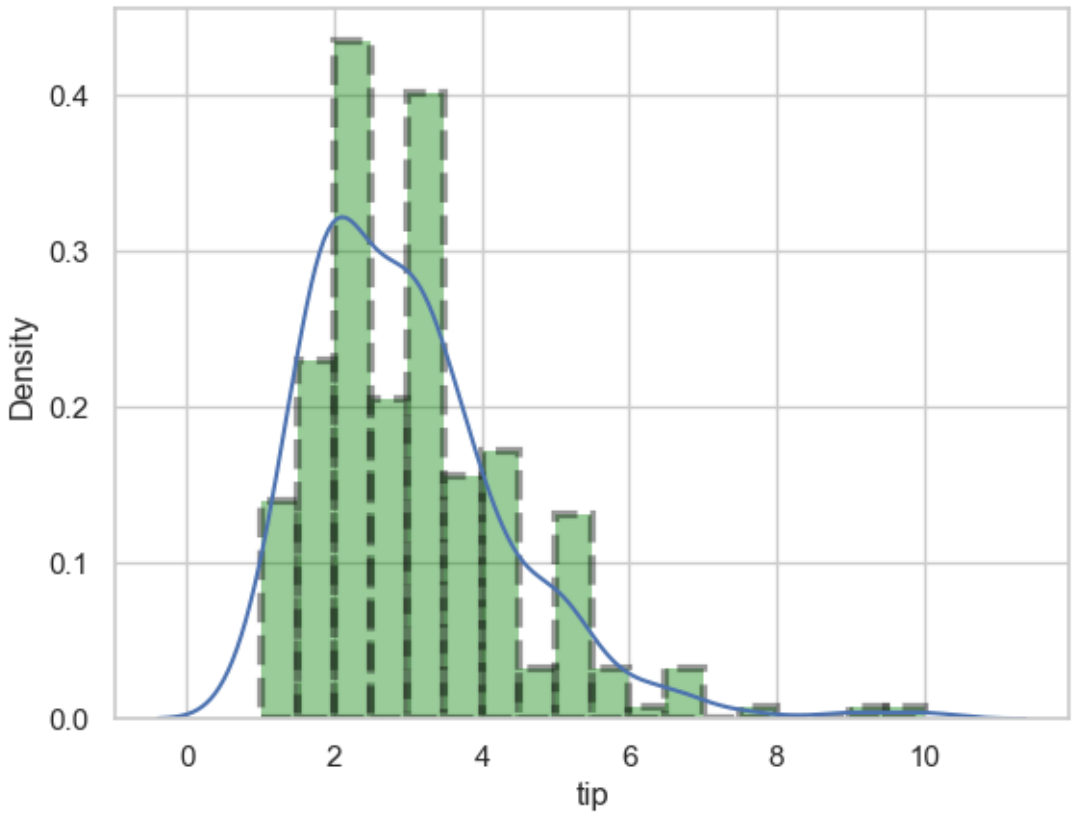
[371]:
```
# How to add outline or Edge color to histogram

sns.distplot(tip['tip'])
plt.show()
```



[381]:
```
# edgecolor and linewidth
sns.distplot(tip['tip'],hist_kws=dict(edgecolor='red', linewidth=6))
plt.show()
```

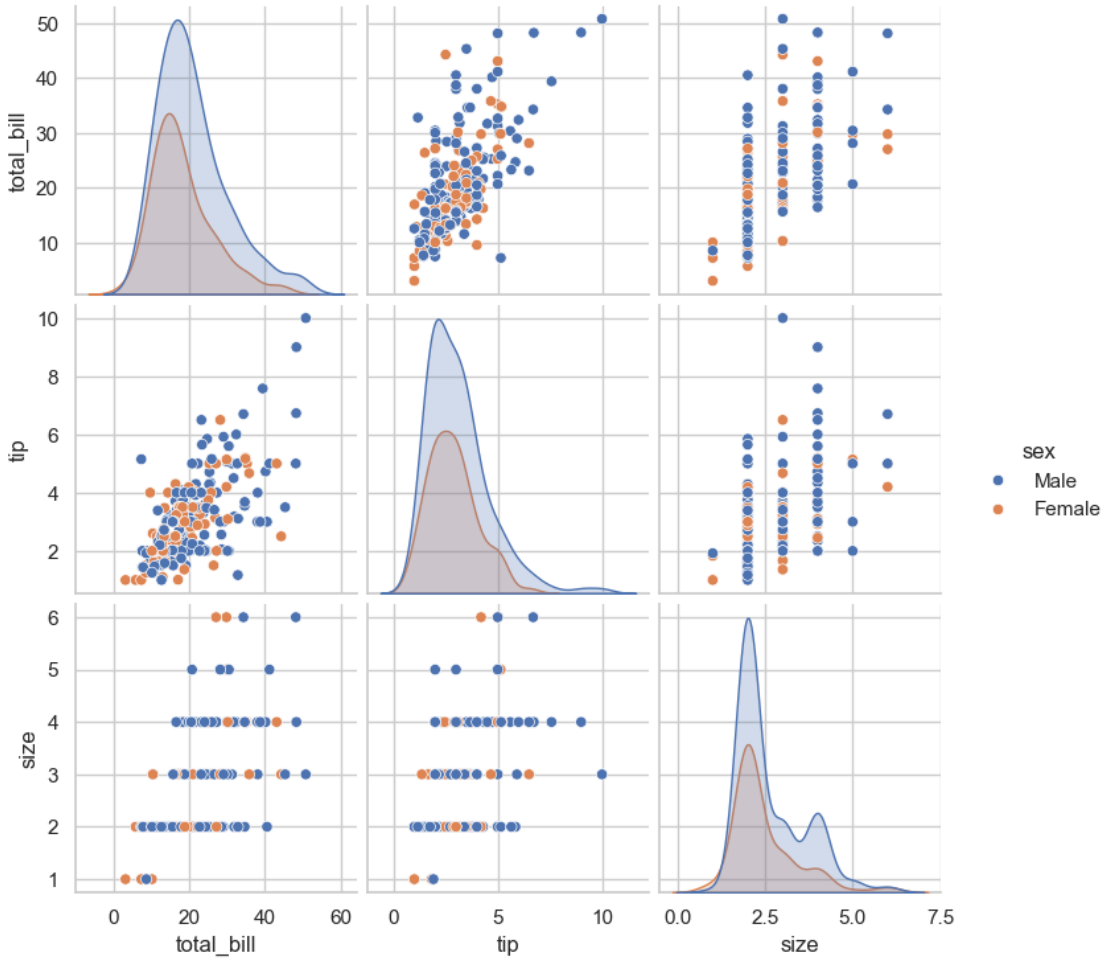


[385]:
```
# linestyle
sns.distplot(tip['tip'],hist_kws={'color':'green', 'edgecolor':'black',␣
 ↪'linewidth':3, 'linestyle':'--'})
plt.show()
```

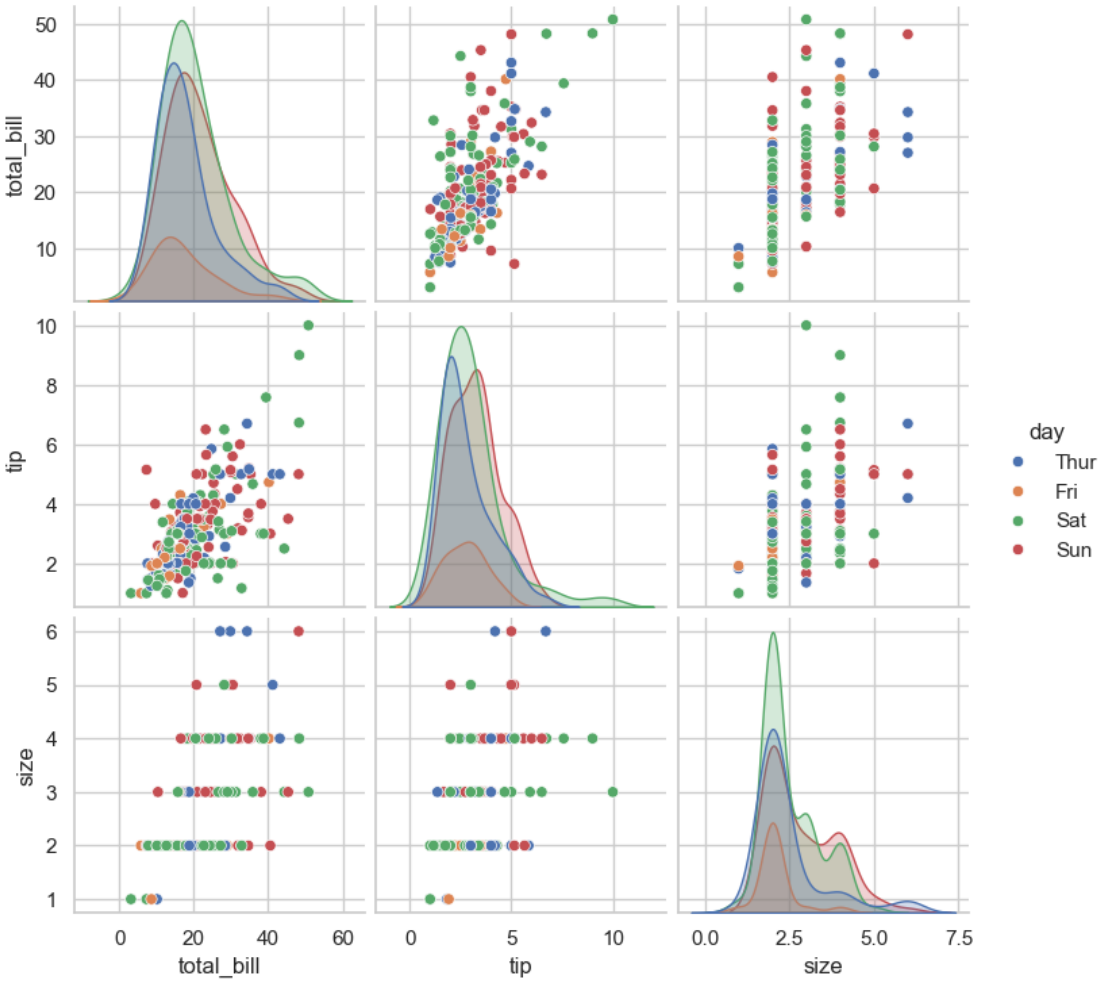


14 pair plot

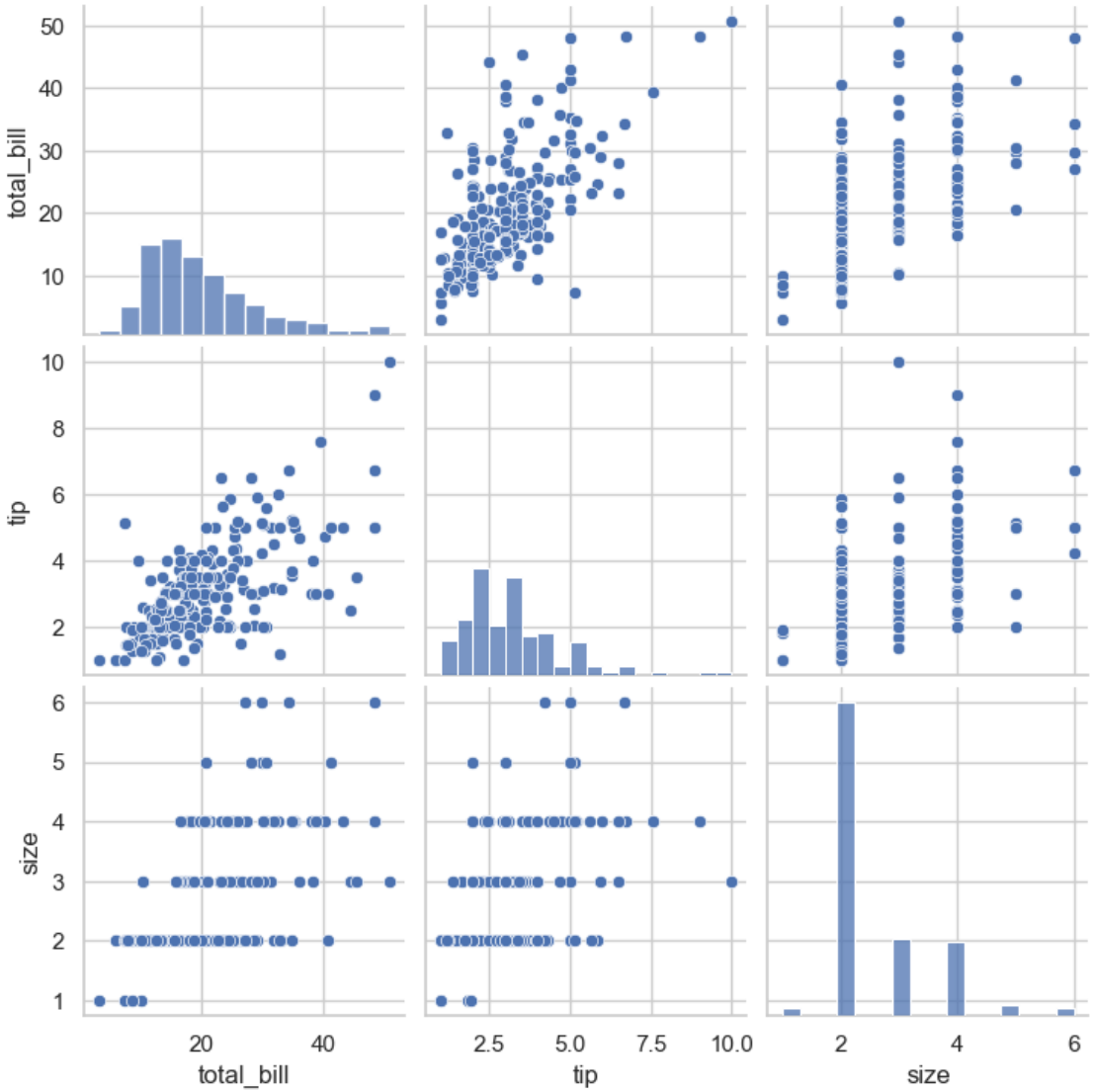
[394]:
```
# To plot multiple pairwise bivariate distributions in a dataset,
# you can use the .pairplot() function.

# The diagonal plots are the univariate plots, and this displays the␣
 ↪relationship
# for the (n, 2) combination of variables in a DataFrame as a matrix of plots.

sns.pairplot(data=tip, hue='sex')
plt.show()
```



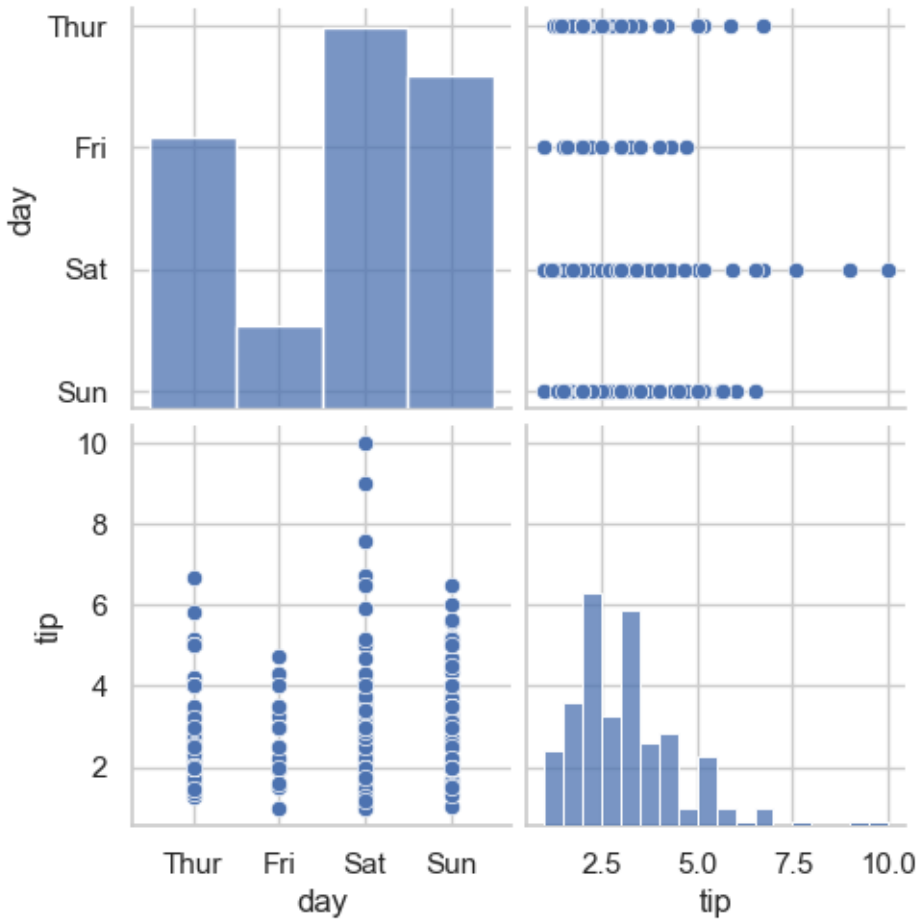
[398]:
```
sns.pairplot(data=tip, hue ='day')
# to show
plt.show()
```



[404]:
```
sns.pairplot(tip)
plt.show()
```

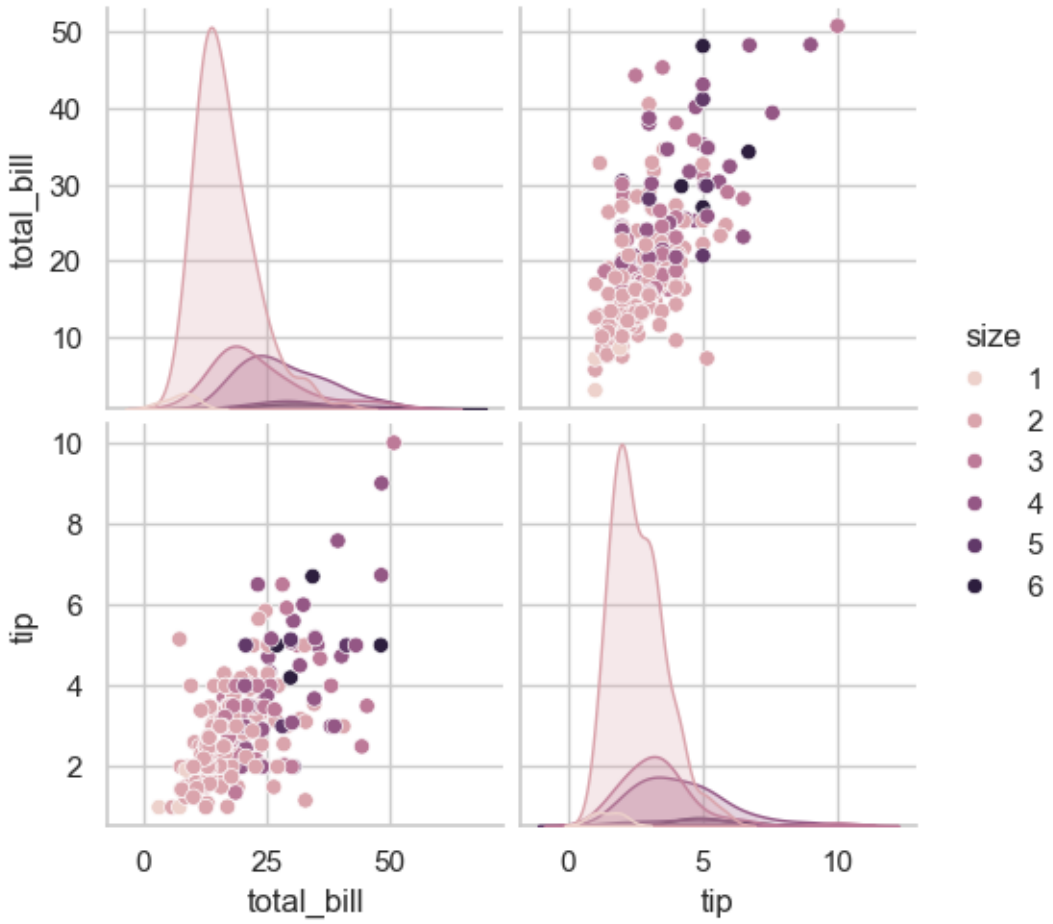


[425]:
```
# Pairplot Seaborn: Plotting Selected Variables
selected_var=['day', 'tip']
sns.pairplot(tip, vars=selected_var)
plt.show()
```

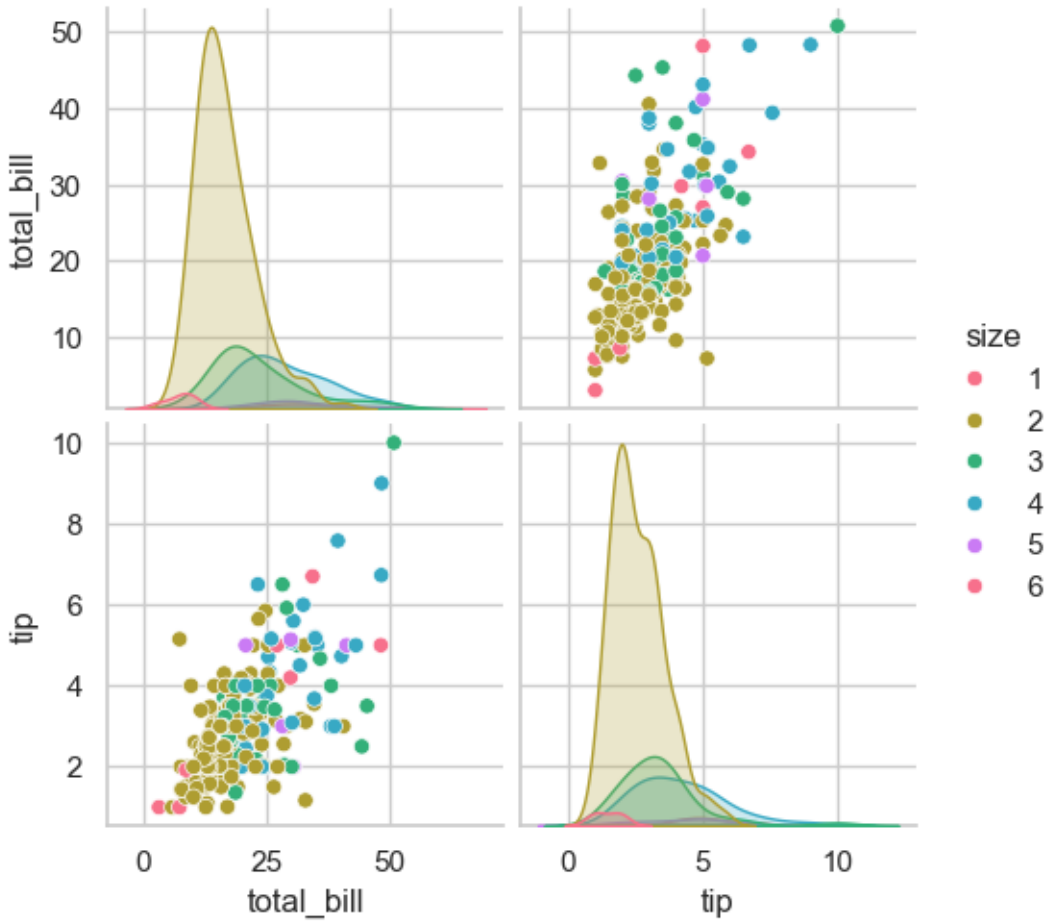


[ ]:
```
# Pairplot Seaborn: Adding a Hue Color to a Seaborn Pairplot
```

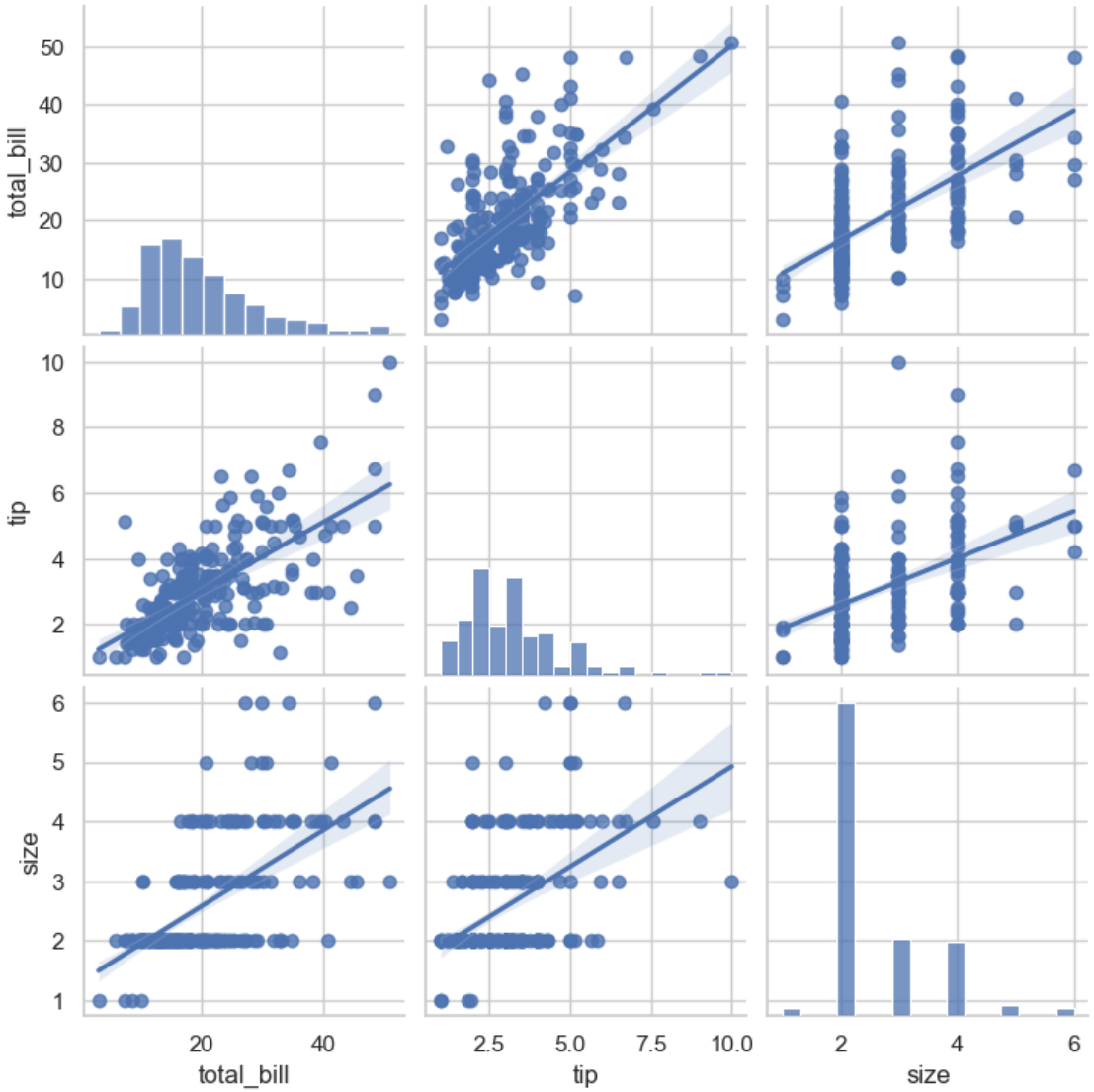
[427]:
```
sns.pairplot(tip, hue='size')
plt.show()
```



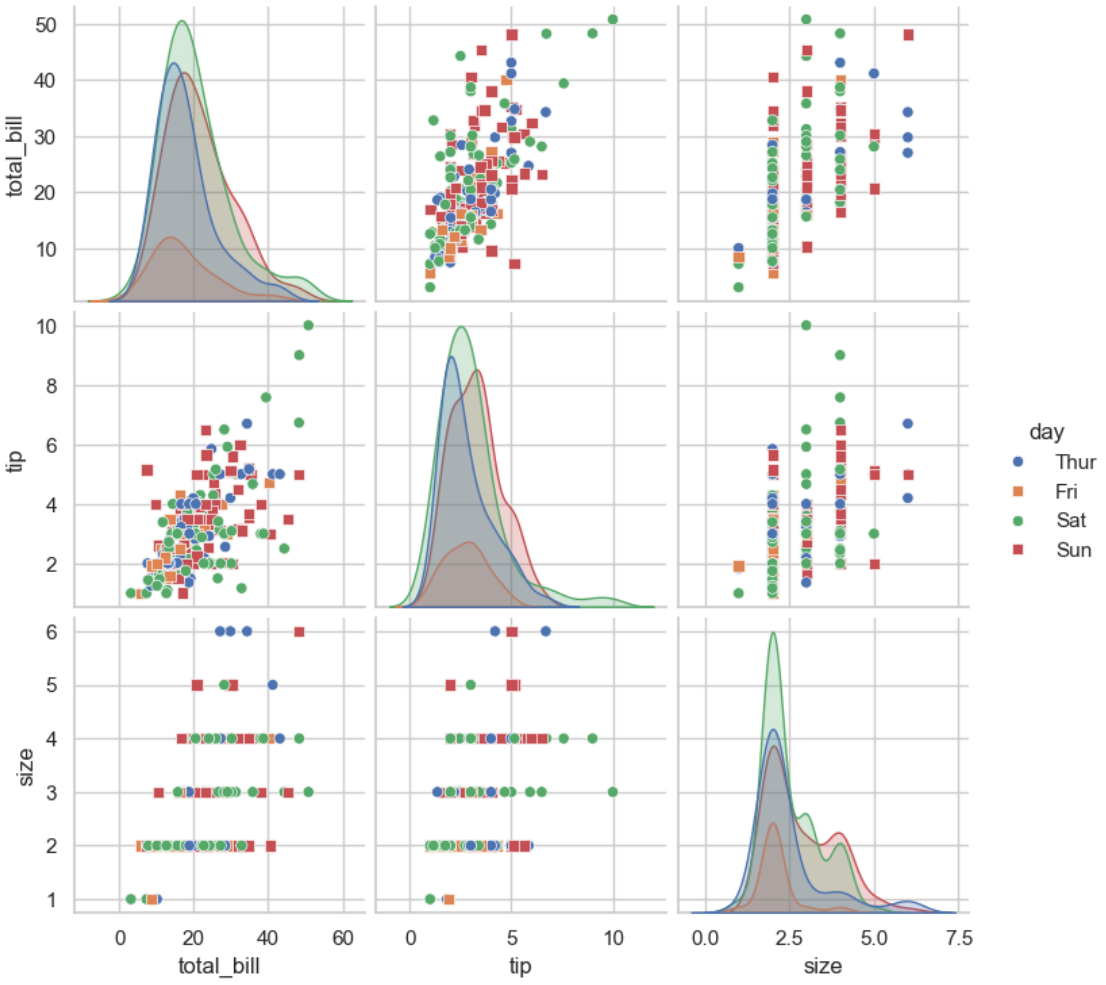
[429]:
```
# modify color palette
sns.pairplot(tip, hue='size', palette='husl')
plt.show()
```



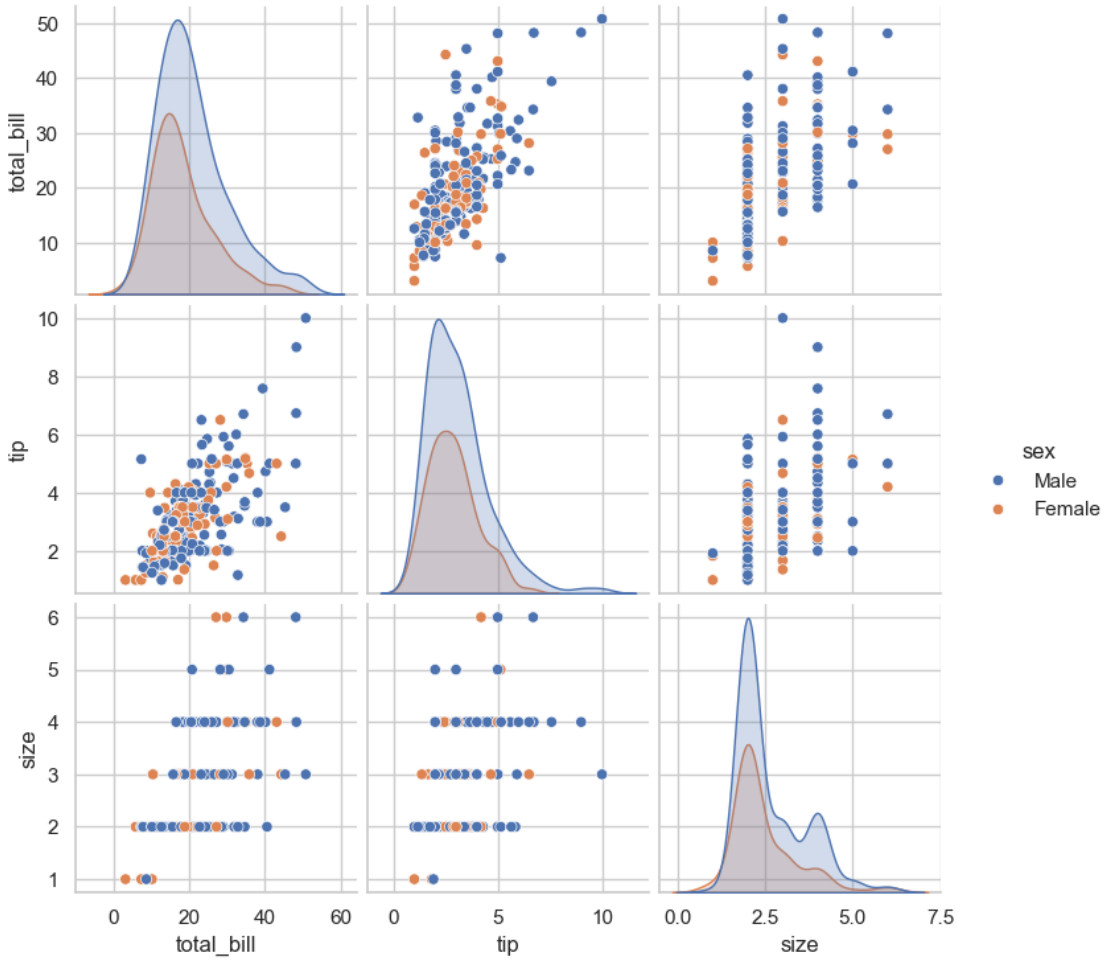
[463]:
```
# Adjusting plot kind
# type of plot . you can choose any like scatter, kde, reg(regression)
sns.pairplot(tip, kind='reg')
plt.show()
```



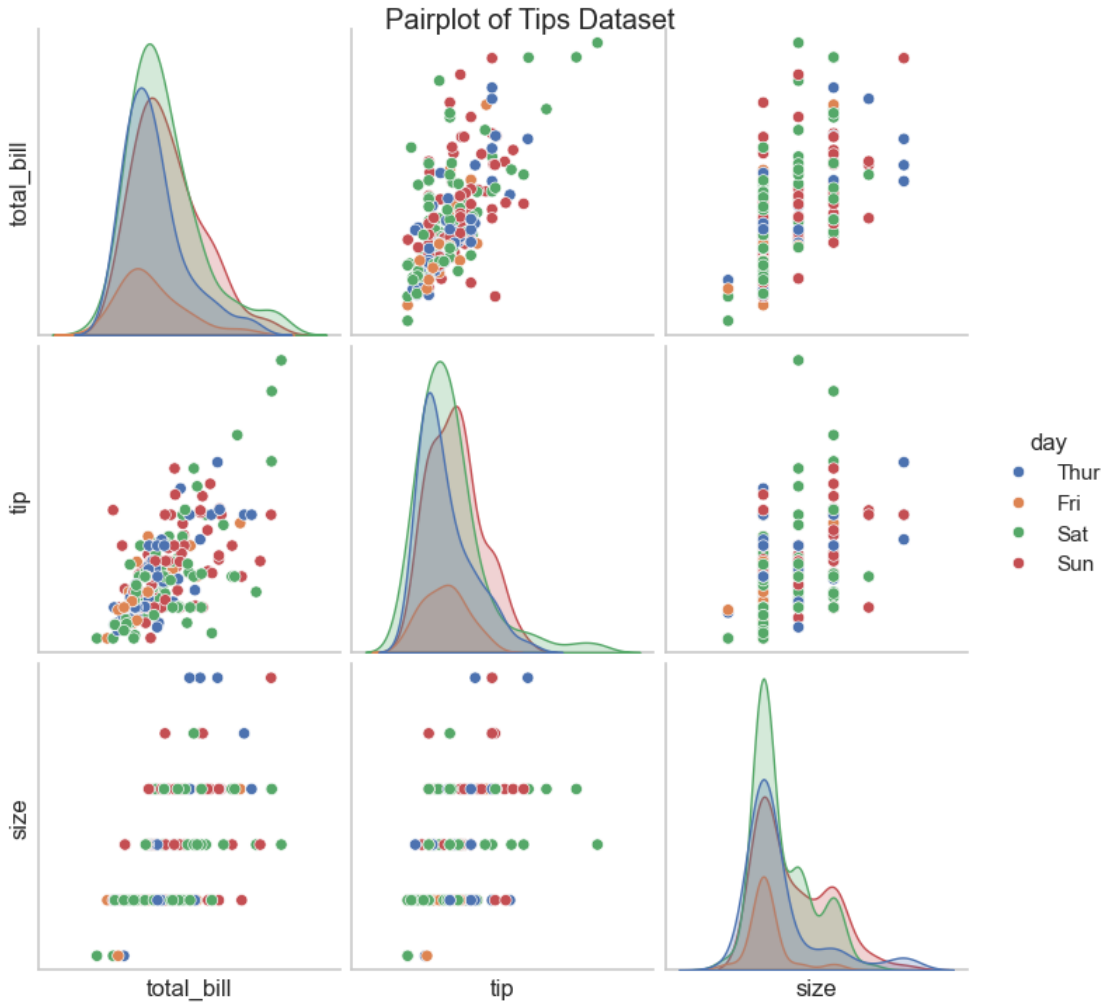
[435]:
```
# controlling the markers
# specifty the diff markers
sns.pairplot(tip, hue='day', markers=['o','s'])
plt.show()
```



[441]:
```
# limiting the variables
# if you are interested in only a subset of hte variable . you can specify them␣
  ↪using the vars parameter
sns.pairplot(tip, hue='sex', vars=['total_bill', 'tip', 'size'])
plt.show()
```



[451]:
```
g = sns.pairplot(tip, hue='day')
g.fig.suptitle("Pairplot of Tips Dataset", y=1.0)  # Add a title
g.set(xticks=[], yticks=[])  # Remove tick labels
plt.show()
```

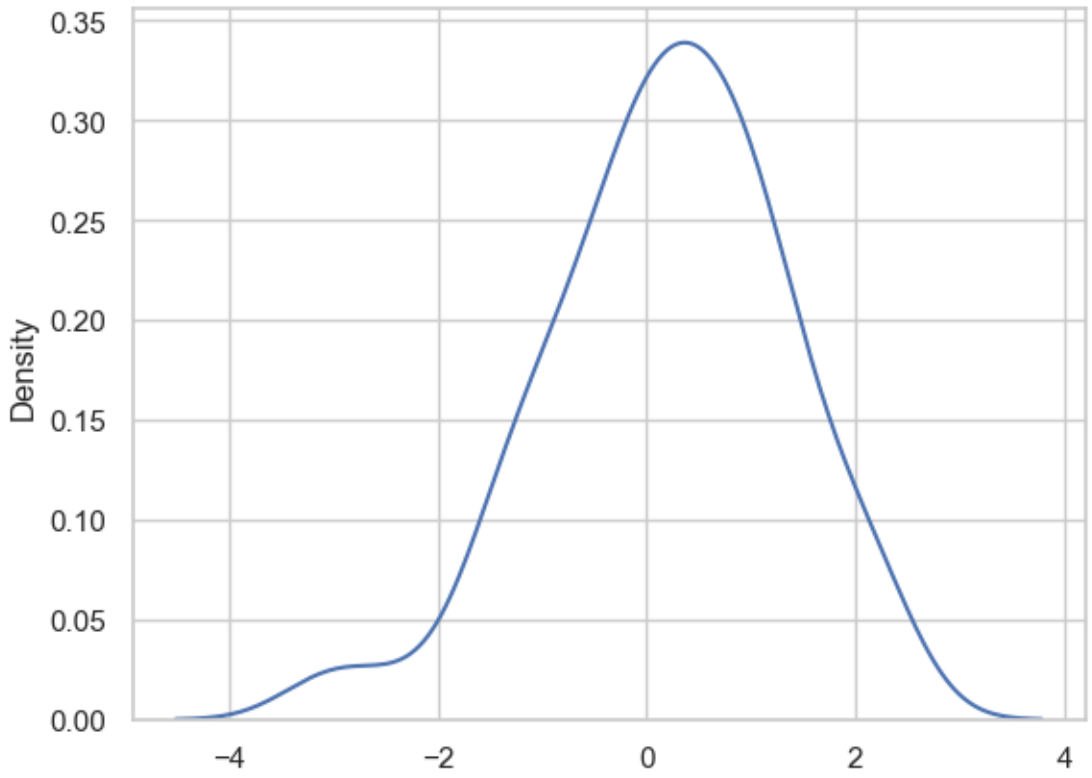


15 KDE plot(Kernel density Estimation)

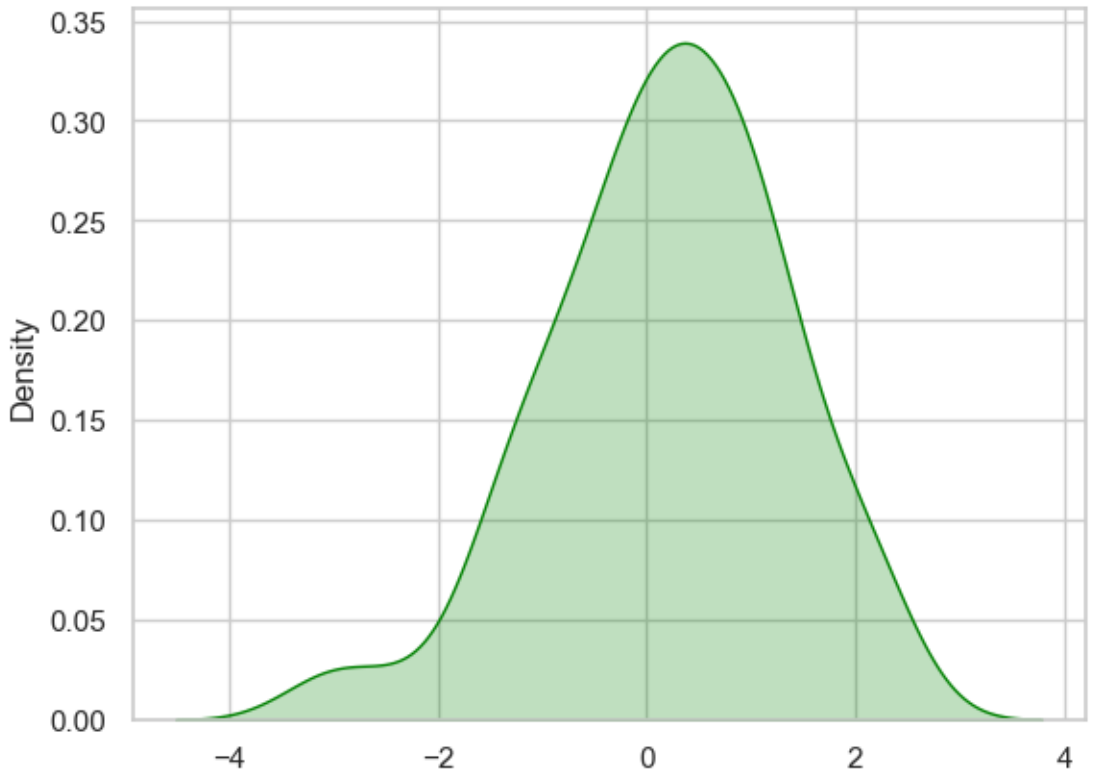
[ ]:
```
#Kernel Density Estimate (KDE) Plot allows to estimate the probability density␣
↪function of the continuous or non-parametric from our data set curve in one␣
↪or more dimensions it means we can create plot a single graph for multiple␣
↪samples which helps in more efficient data visualization.
```

[472]:
```
from matplotlib import pyplot as plt
%matplotlib inline

x=np.random.randn(100)
y=np.random.randn(100)
sns.kdeplot(x)
```

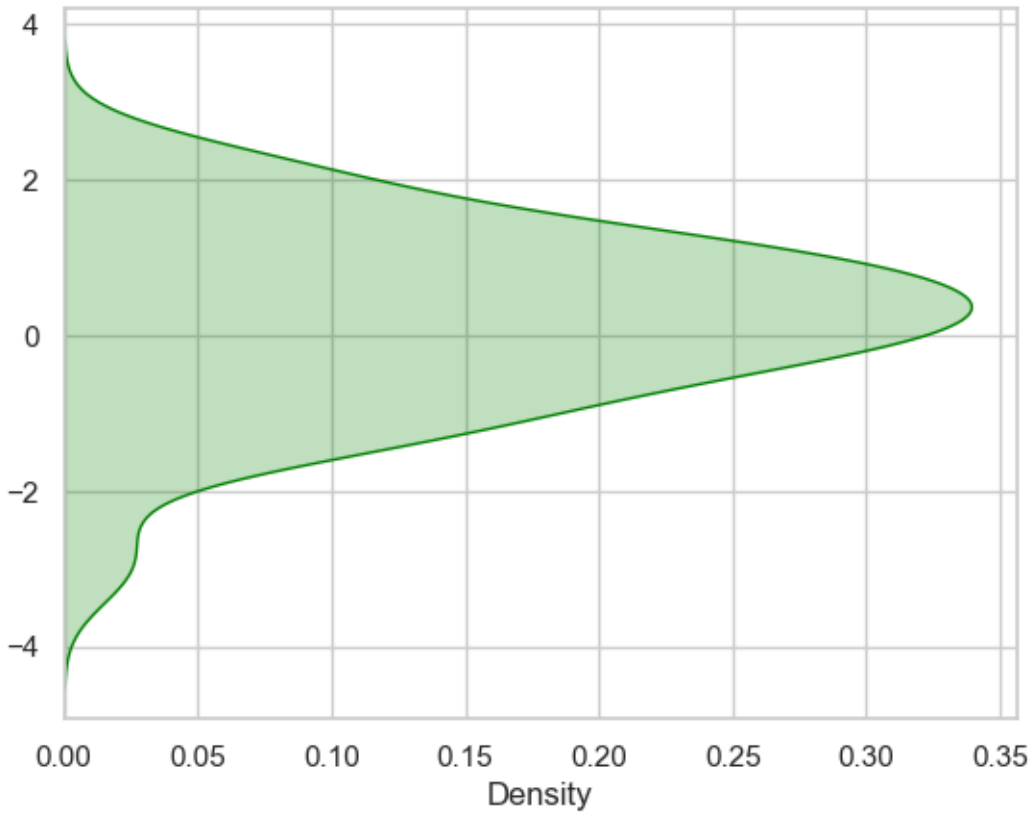
[472]:
```
<Axes: ylabel='Density'>
```



[480]:
```
# change the shade color with color attributes
sns.kdeplot(x, shade=True, color='green')
plt.show()
```



[482]:
```
sns.kdeplot(x, shade=True, color='green', vertical=True)
plt.show()
```



[549]:
```
# Bivariate Kdeplot for two variables
# simple pass two vaaraibles into the seaborn.kdeplot()
# sns.kdeplot(x,y ,fill=True)
```

16 Heat Map

[ ]:
```
# Heatmap is defined as a graphical representation of data using colors to␣
↪visualize the value of the matrix. In this, to represent more common values␣
↪or higher activities brighter colors basically reddish colors are used and␣
↪to represent less common or activity values, darker colors are preferred.␣
↪Heatmap is also defined by the name of the shading matrix. Heatmaps in␣
↪Seaborn can be plotted by using the seaborn.heatmap() function.
```

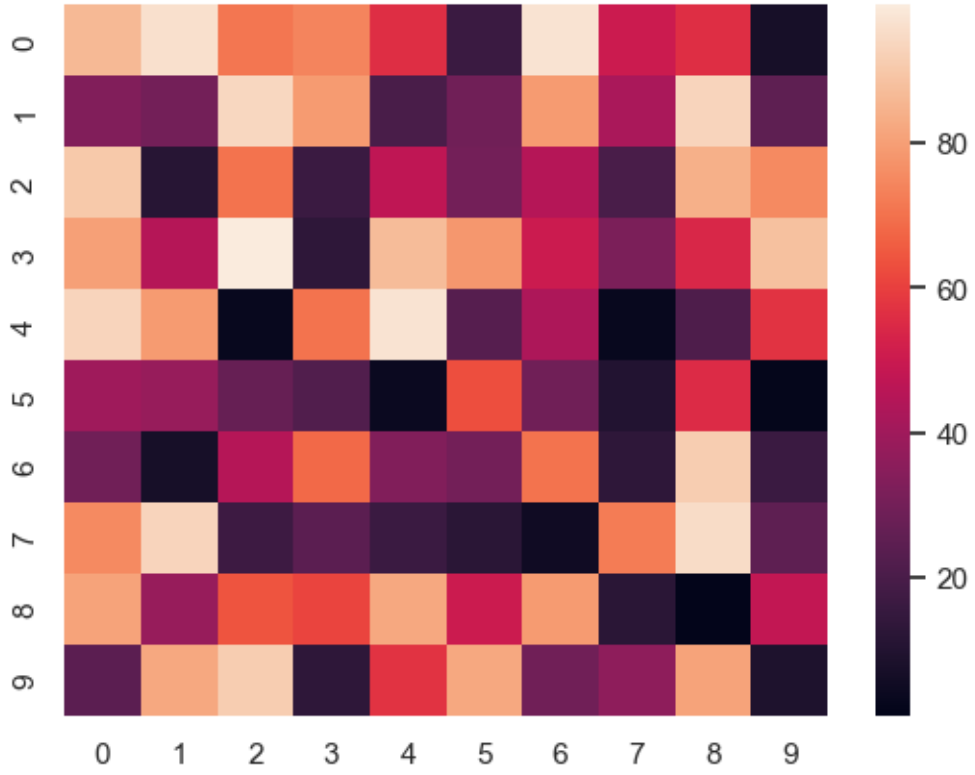
[ ]:
```
#Syntax: seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None,␣
↪annot_kws=None, linewidths=0, linecolor='white', cbar=True, **kwargs)
```

[568]:
```
data=np.random.randint(low=1, high=100, size=(10,10))
print(data)
```

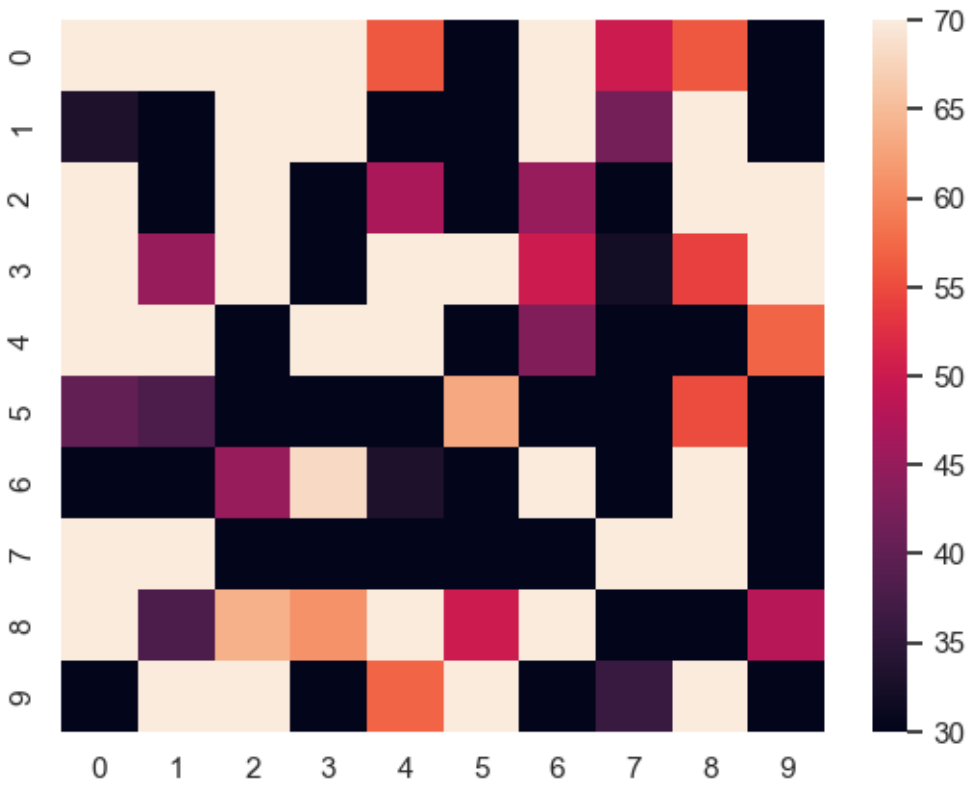
```
[[86 96 71 74 56 16 97 50 56  7]
```

```
[33 30 94 79 20 29 79 42 93 25]
[90 11 70 16 47 30 45 20 84 75]
[80 45 99 13 87 78 50 32 54 88]
[93 79  3 70 97 23 43  3 21 57]
[40 38 27 22  4 63 29 10 55  2]
[29  7 45 68 33 30 70 13 91 16]
[75 93 17 24 16 12  5 72 95 25]
[81 38 64 61 82 50 79 12  1 48]
[24 82 91 13 57 82 29 36 81  9]]
```

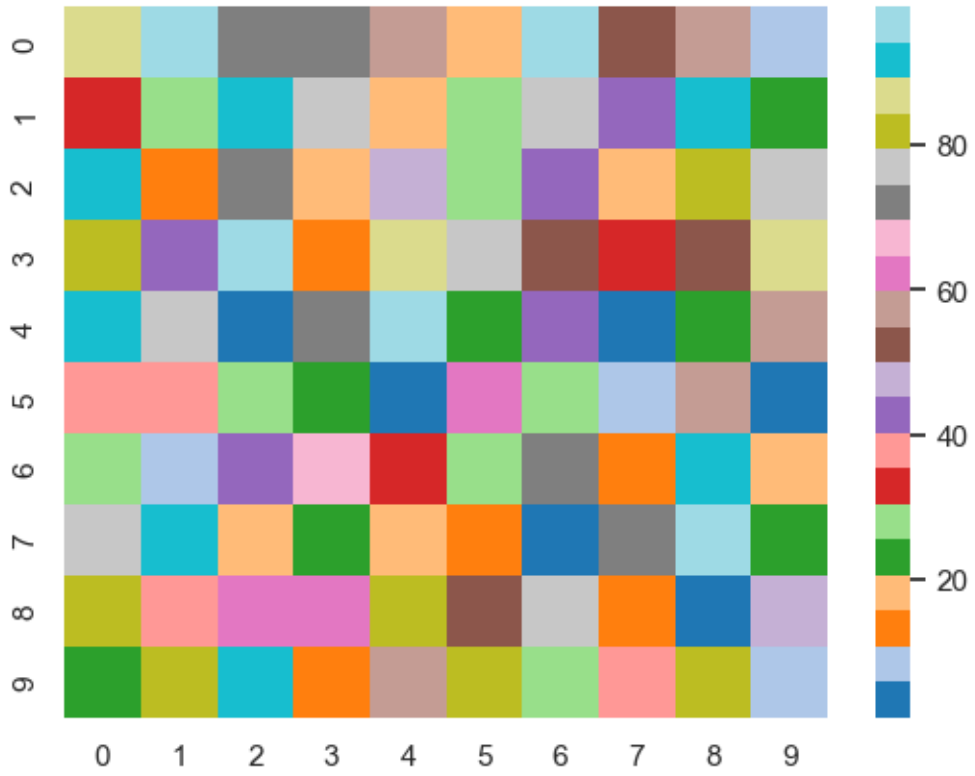
[570]:
```
sns.heatmap(data=data)
plt.show()
```



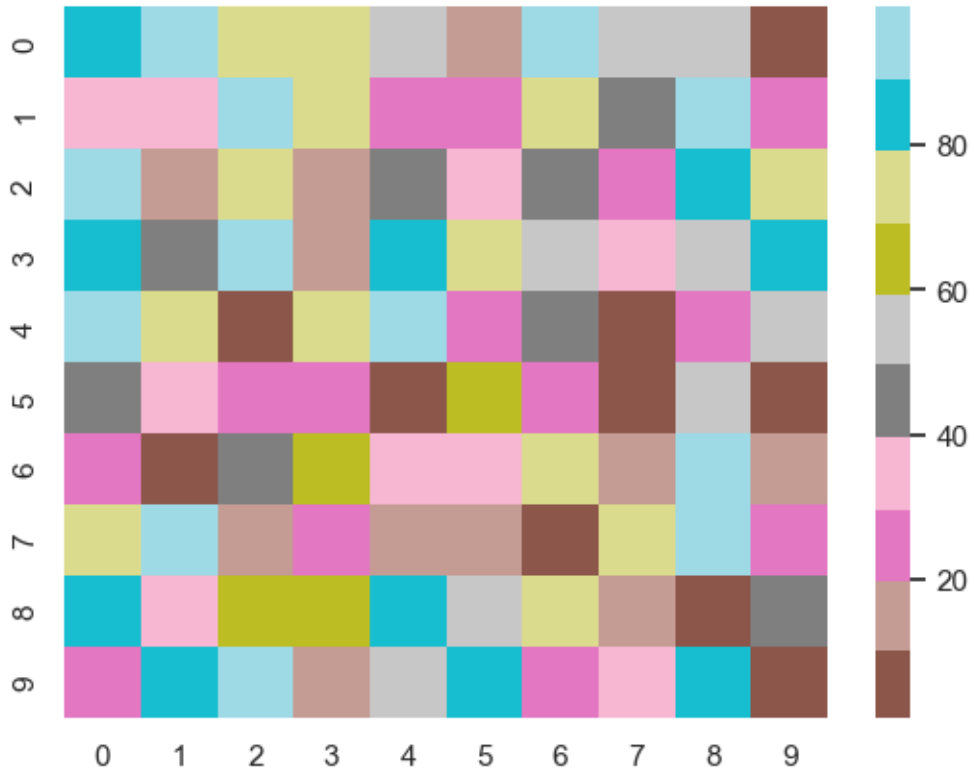
[572]:
```
# Anchoring the colormap
# If we set the vmin value to 30 and the vmax value to 70, then only the cells␣
 ↪with values between 30 and 70 will be displayed. This is called anchoring␣
 ↪the colormap.

vmin=30
vmax=70
sns.heatmap(data=data, vmin=30, vmax=70)
plt.show()
```



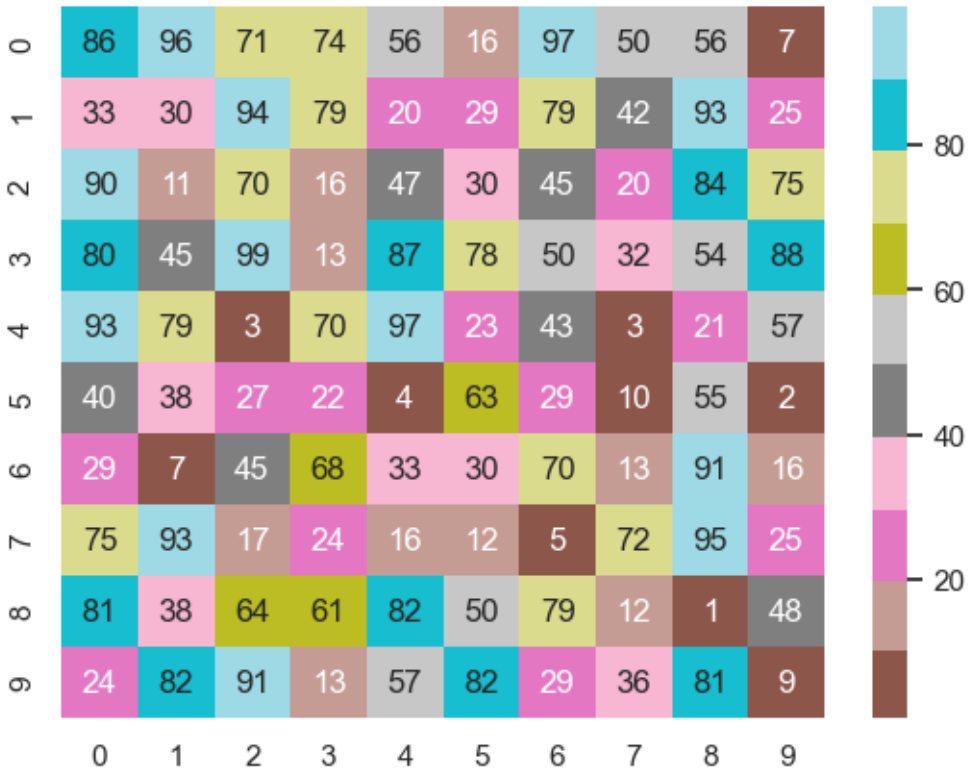
[578]:
```
# choosing color map
sns.heatmap(data=data, cmap="tab20")
plt.show()
```



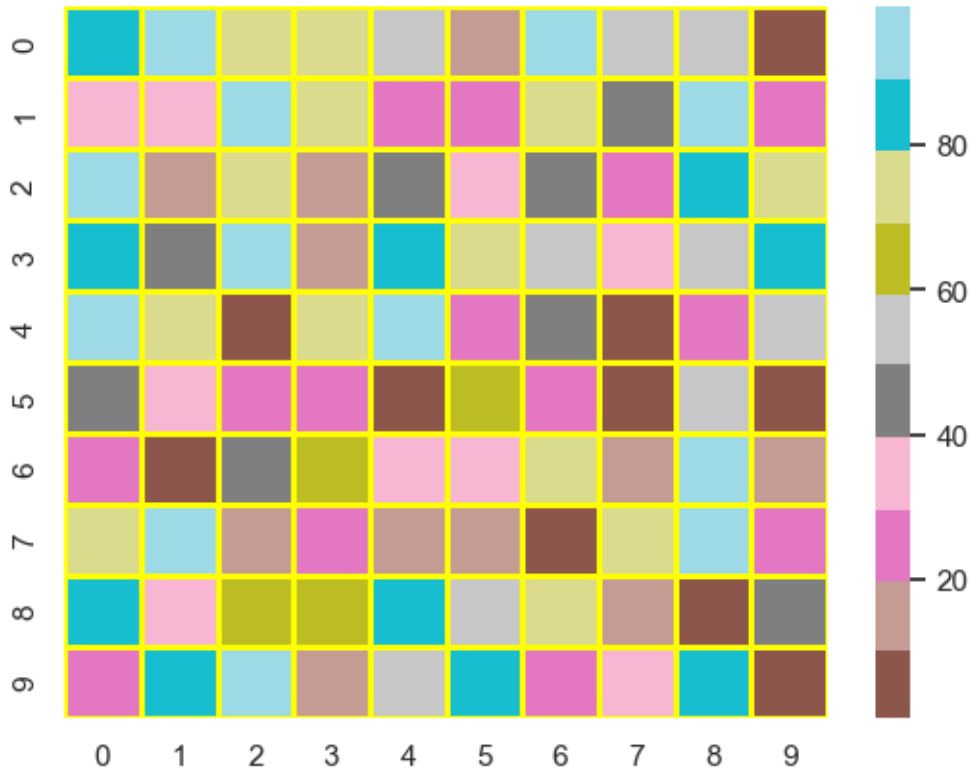
[580]:
```
# centering the colormap
sns.heatmap(data=data, cmap='tab20',center=0)
plt.show()
```



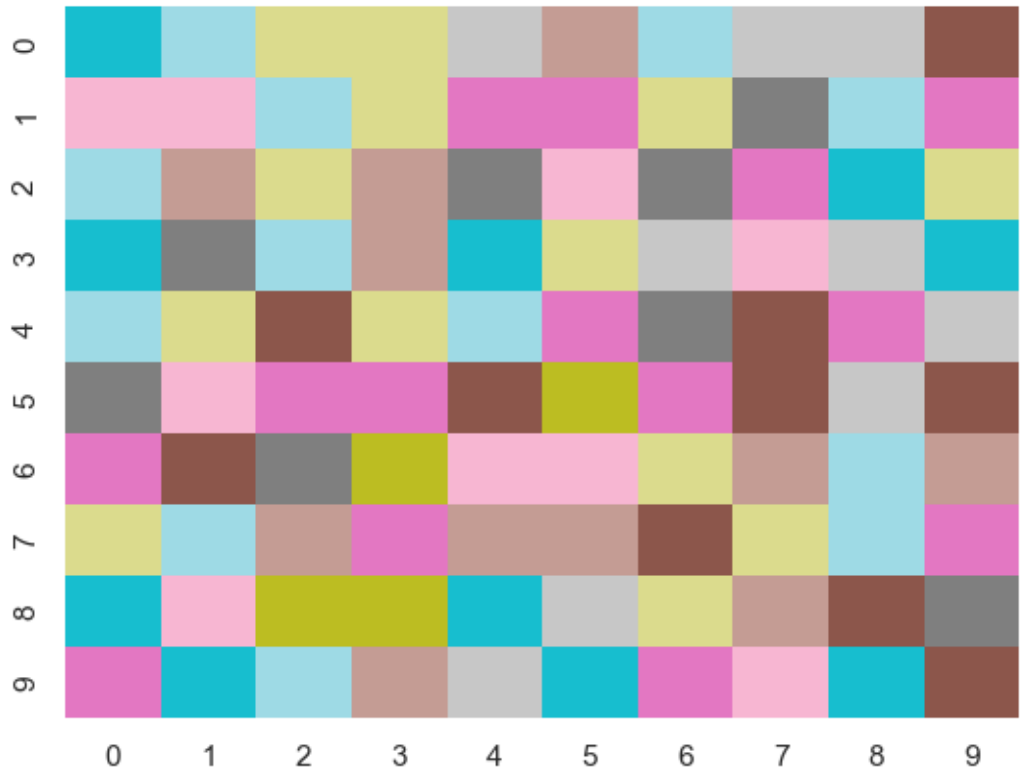
[582]:
```
# display cells
# annot
sns.heatmap(data=data, cmap='tab20',center=0, annot=True)
plt.show()
```



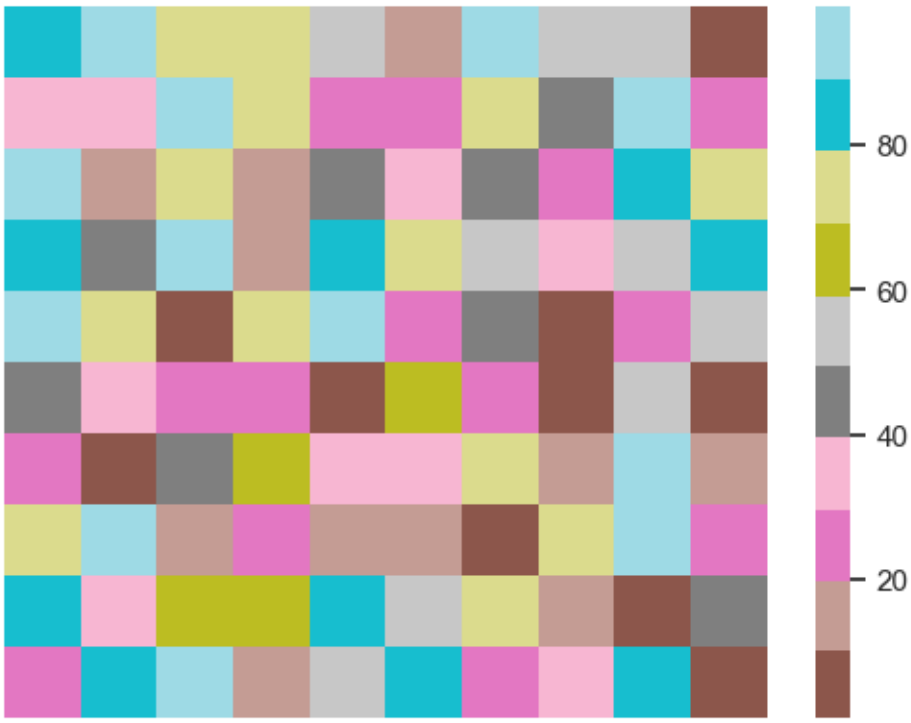
[584]:
```
# customizing the seperating values
sns.heatmap(data=data, cmap='tab20',center=0, linewidths=2, linecolor='yellow')
plt.show()
```



[586]:
```
# hiding the color bar
sns.heatmap(data=data, cmap='tab20',center=0, cbar=False)
plt.show()
```



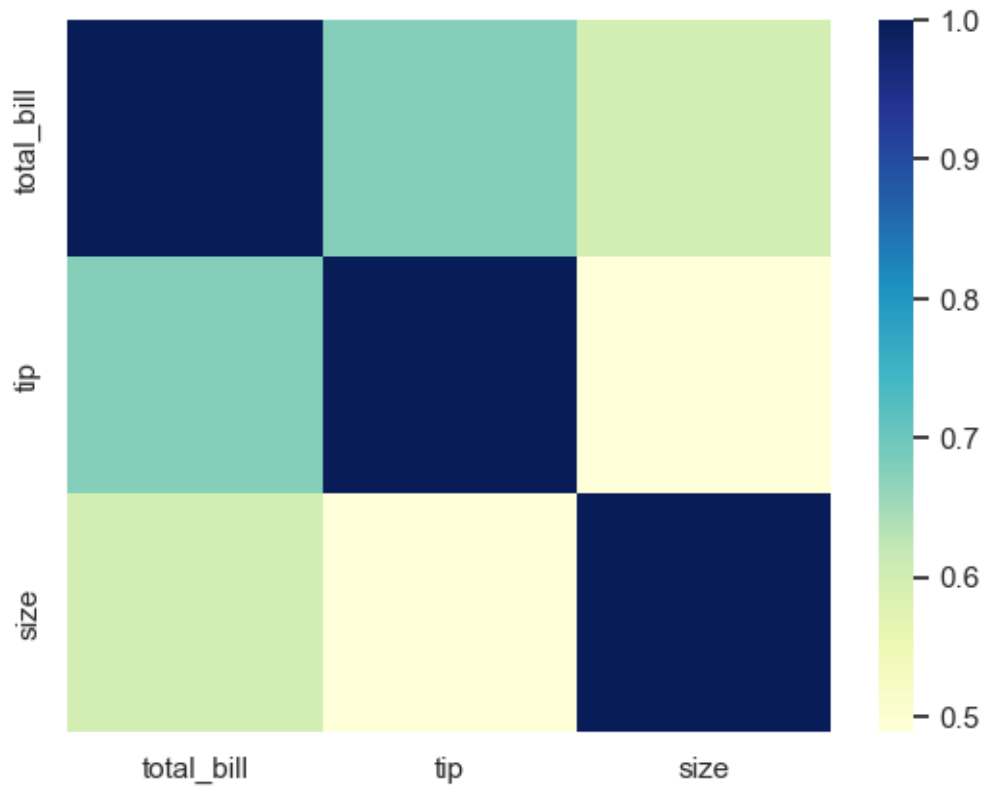
[588]:
```
sns.heatmap(data=data, cmap='tab20',center=0, xticklabels=False,␣
↪yticklabels=False)
plt.show()
```



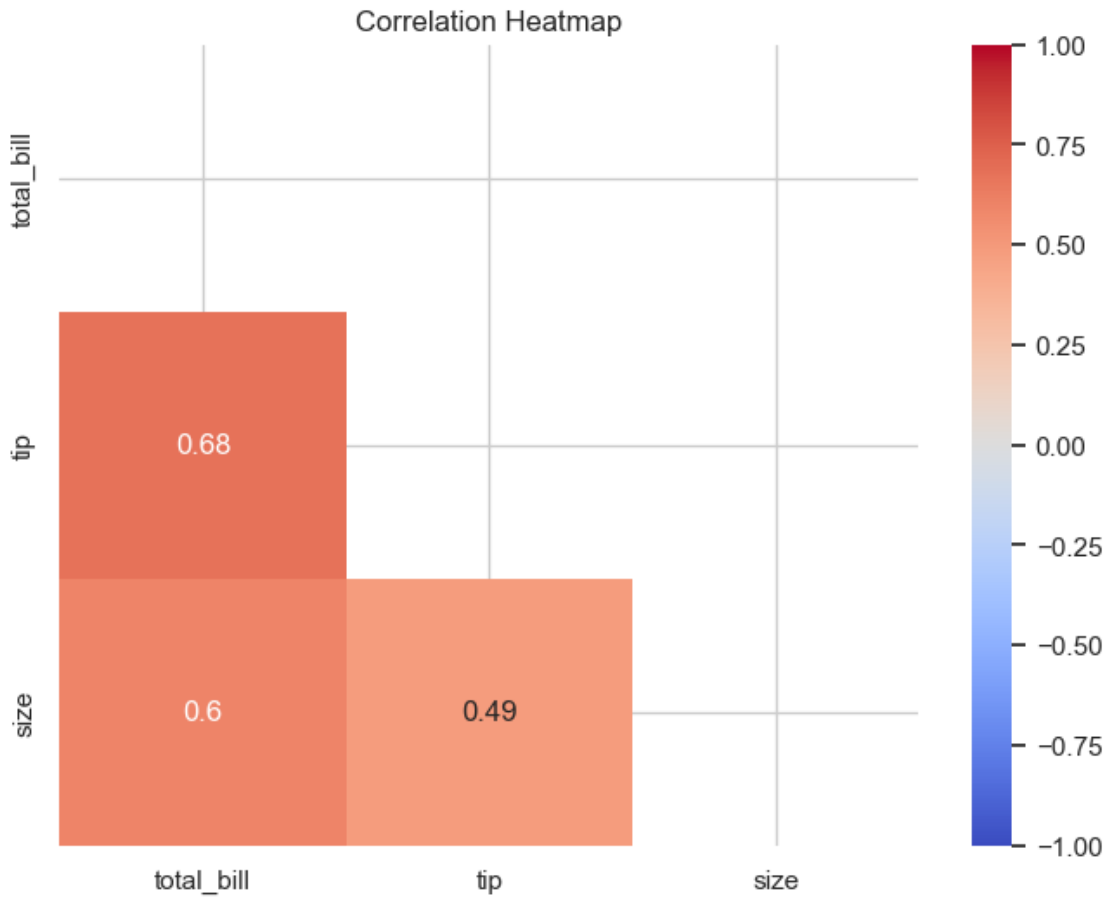
16.0.1 Correlation heatmap

[ ]:
```
# A correlation map (also known as a correlation matrix heatmap) is a visual␣
 ↪representation of the
# correlation coefficients between variables in a dataset. It helps show the␣
 ↪relationships
# between pairs of variables, typically represented in a matrix format with␣
 ↪color gradients
# to indicate the strength and direction of each correlation.
```

[595]:
```
sns.heatmap(tip.corr(numeric_only=True), cmap="YlGnBu", annot=False)
plt.show()
```

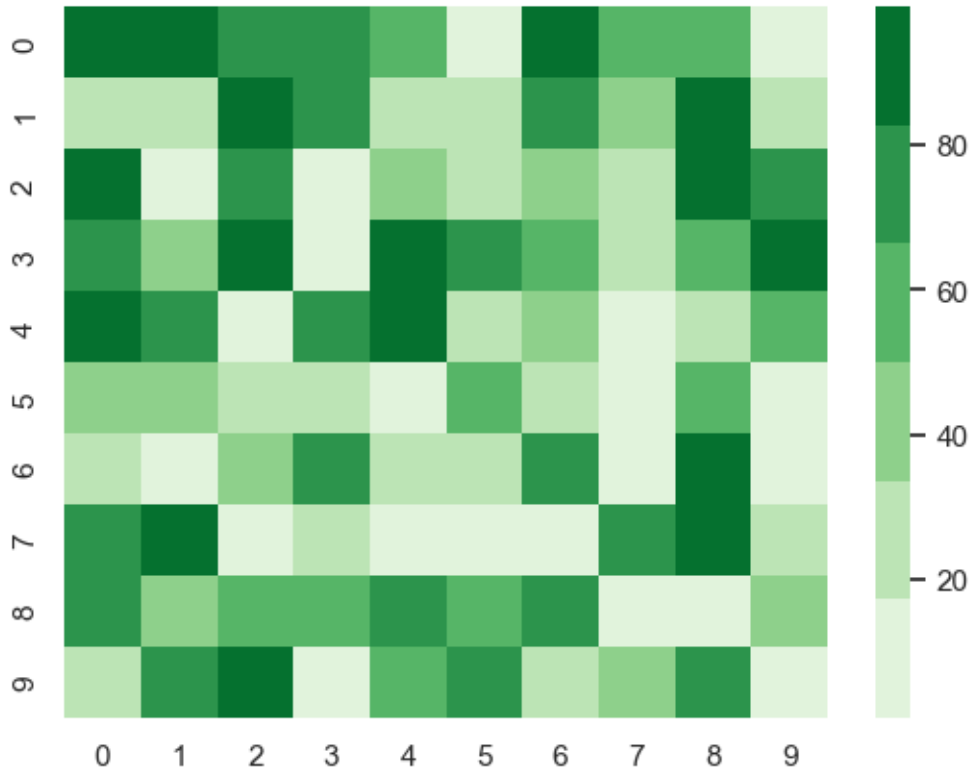


[609]:
```
# Calculate the correlation matrix
corr_matrix = tip.corr(numeric_only=True)
# Create a mask for the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
# Plot the heatmap with the mask
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, mask=mask, cmap="coolwarm", vmin=-1,␣
 ↪vmax=1)
plt.title("Correlation Heatmap")
plt.show()
```

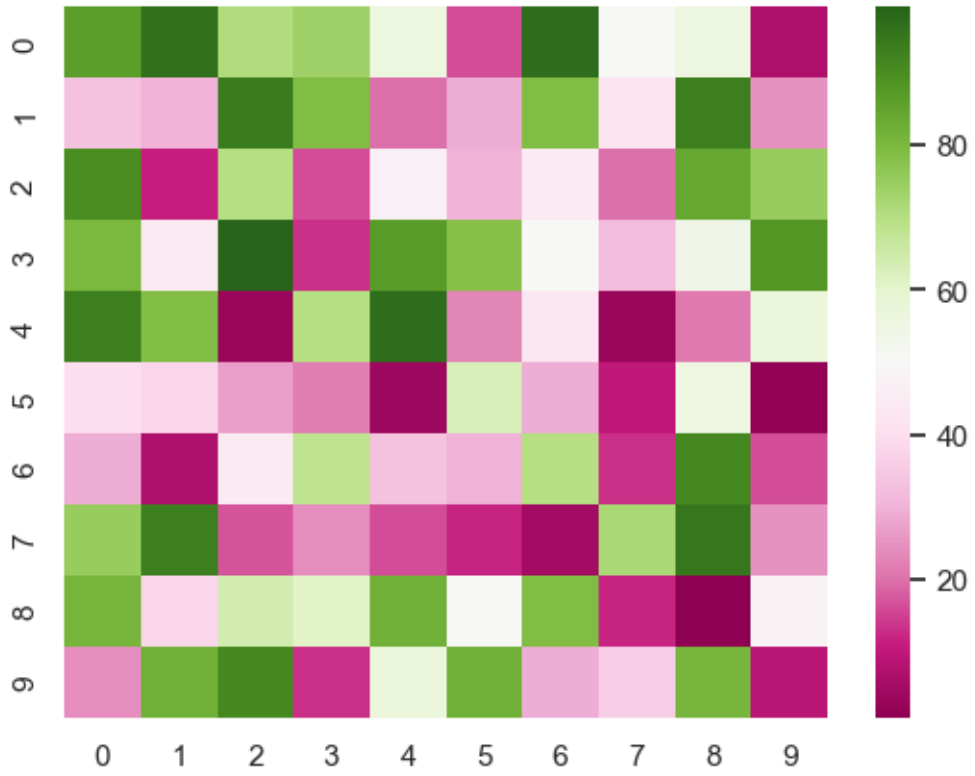


16.0.2 –> color map

[618]:
```
clrmap= sns.color_palette('Greens')
sns.heatmap(data, cmap=clrmap)
plt.show()
```



[622]:
```
# Diverging colormaps
# they are usedd to represent numeric values that go from high toloe and both␣
  ↪high and low are off interest.
sns.heatmap(data, cmap="PiYG")
plt.show()
```

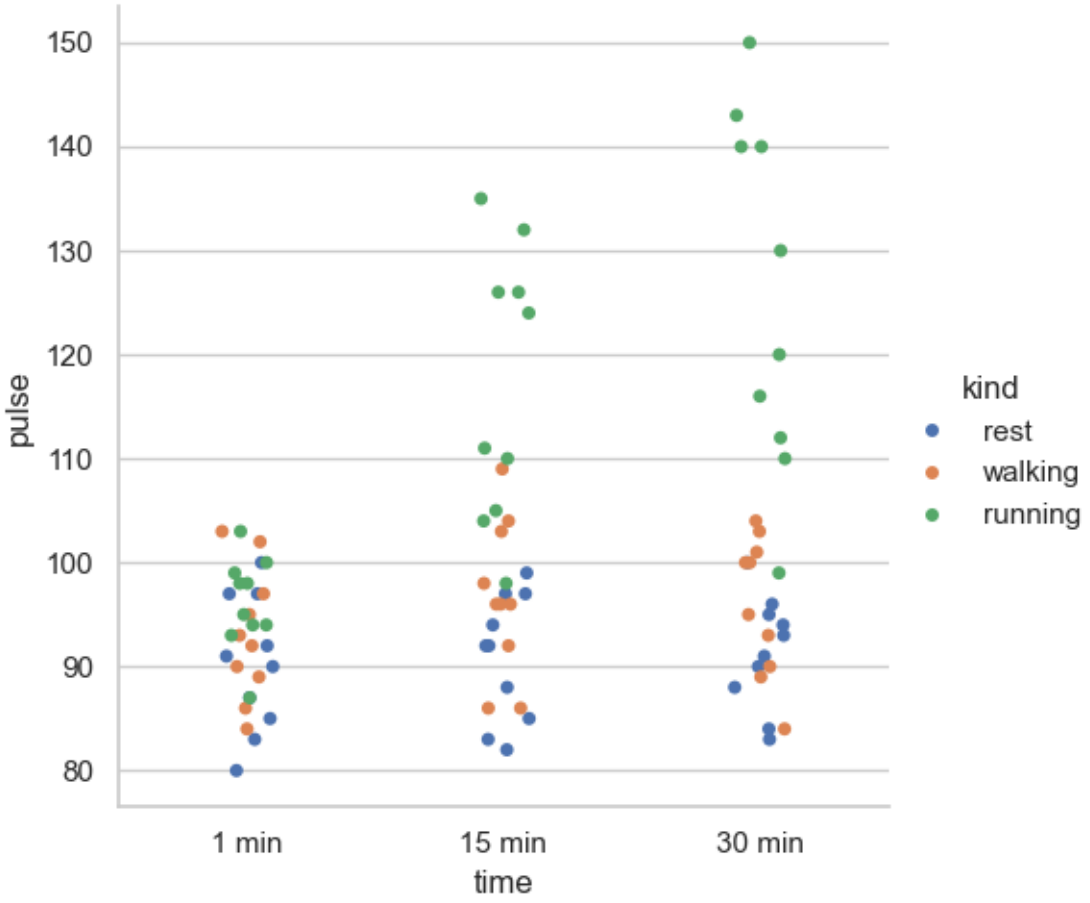


17 Catplot

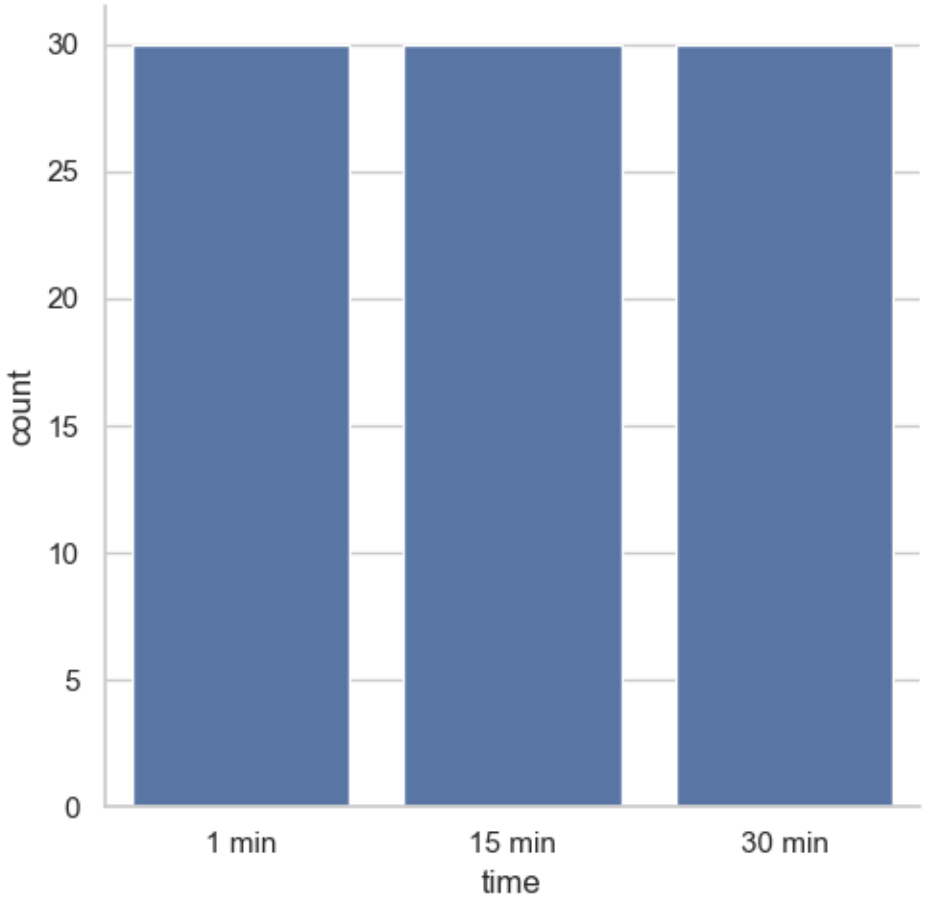
[ ]:

[ ]:
```
# Syntax: seaborn.catplot(*, x=None, y=None, hue=None, data=None, row=None,␣
↪col=None, kind='strip', color=None, palette=None, **kwargs)
```

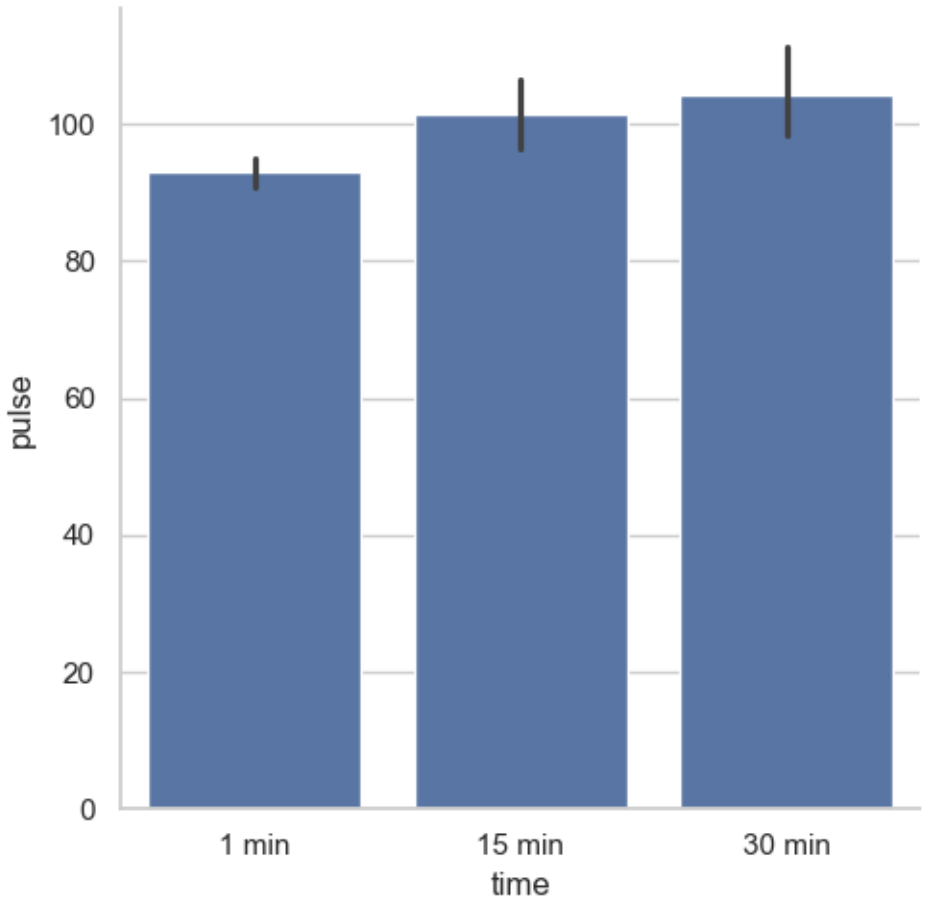
[656]:
```
exercise = sns.load_dataset("exercise")
sns.catplot(x="time", y="pulse",
                hue="kind",
                data=exercise)
plt.show()
```



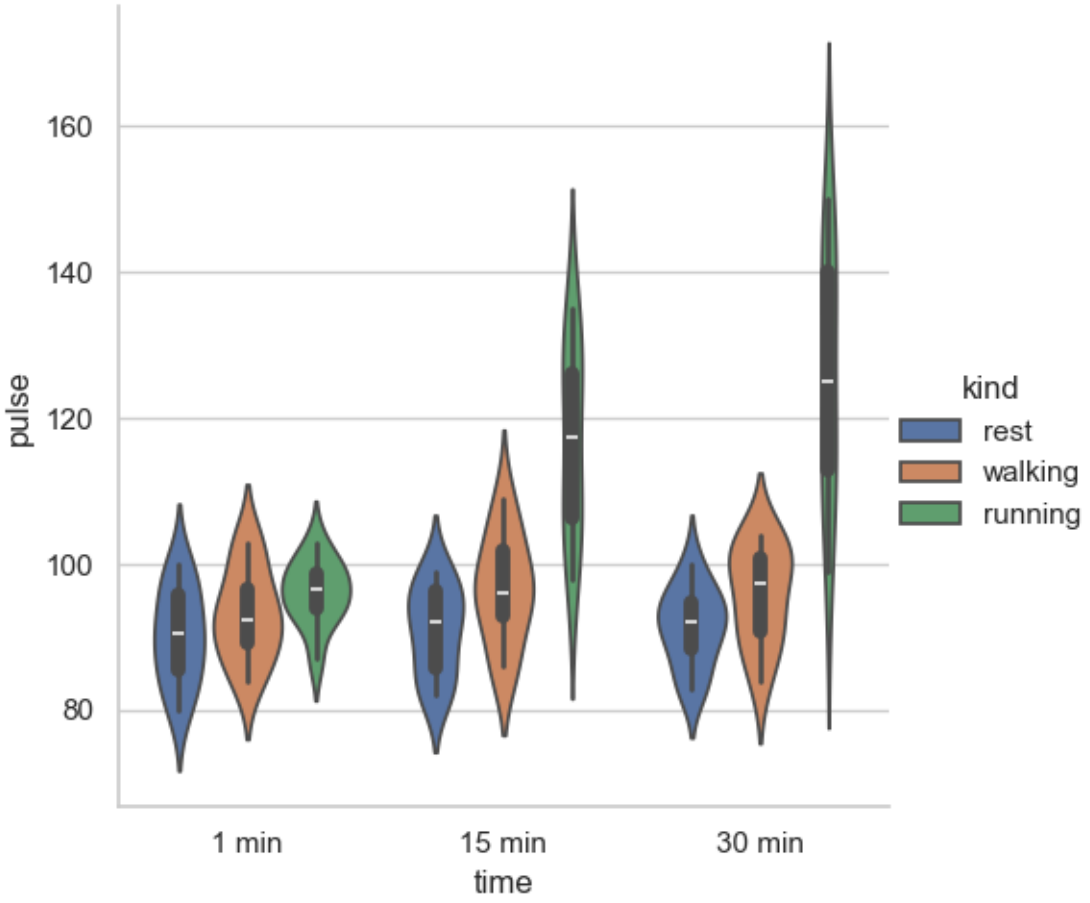
[660]:
```
sns.catplot(x='time', kind='count',data=exercise)
plt.show()
```



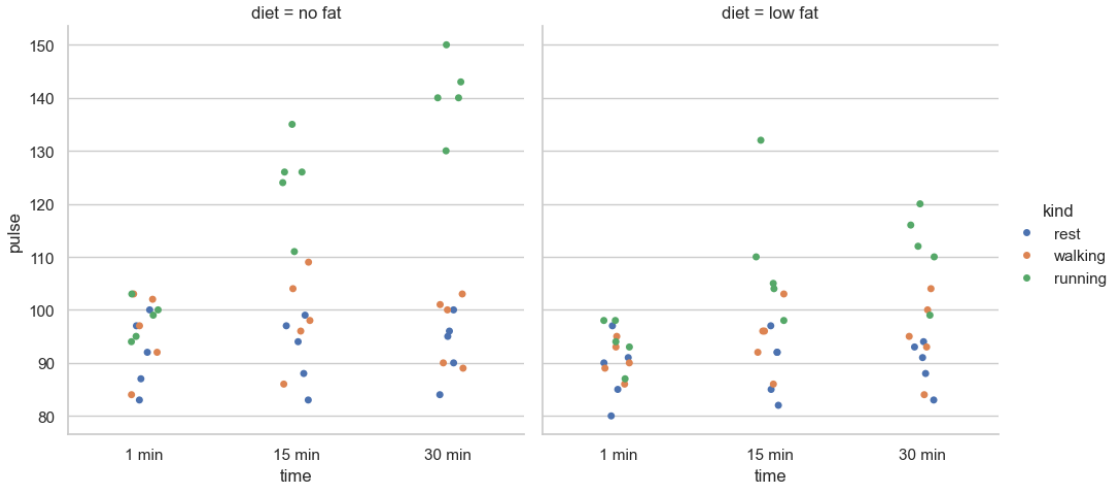
[662]:
```
sns.catplot(x='time', y='pulse',kind='bar',data=exercise)
plt.show()
```



[666]:
```
sns.catplot(x='time',y='pulse', kind='violin',data=exercise, hue='kind')
plt.show()
```

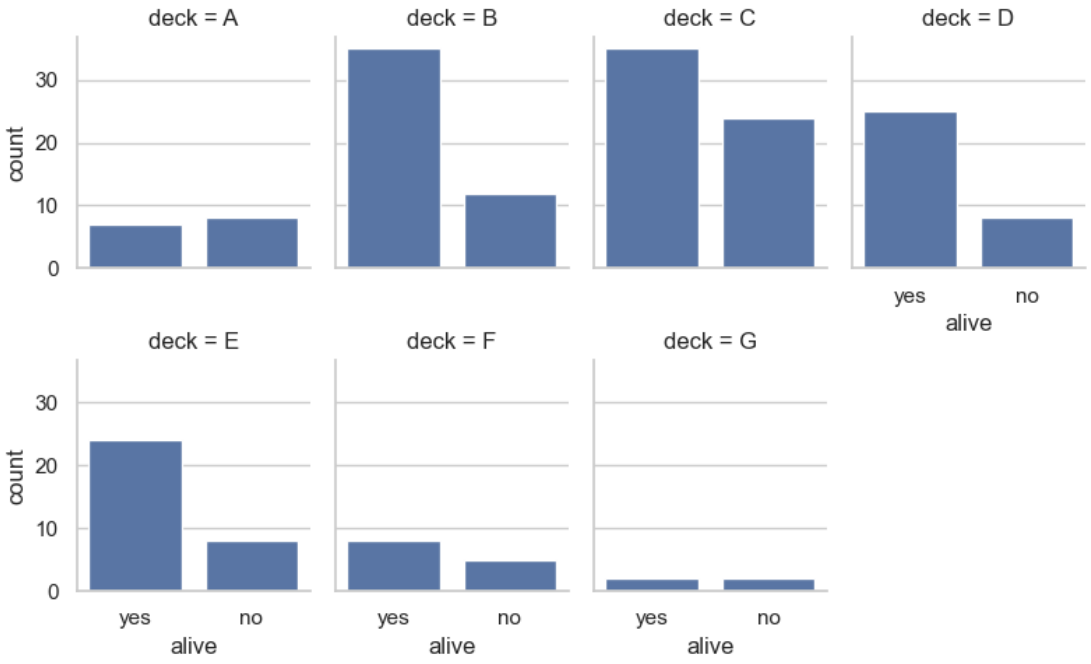


[668]:
```
sns.catplot(x='time',y='pulse', col='diet',data=exercise, hue='kind')
plt.show()
```

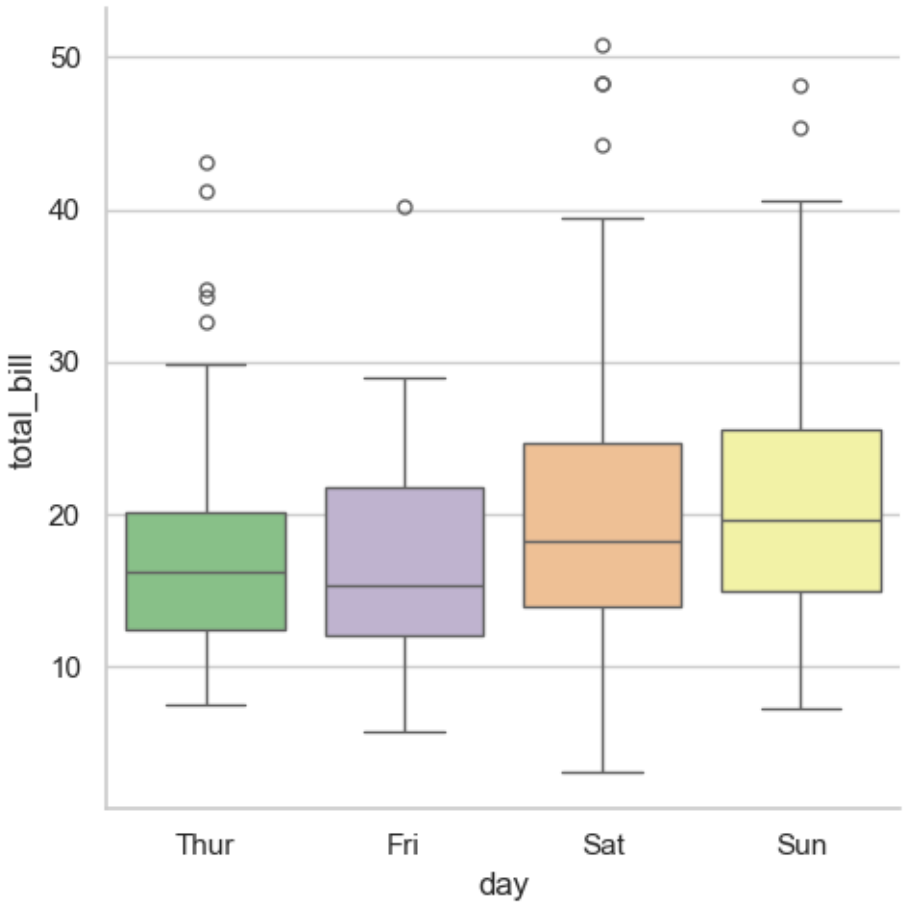


[682]:
```
titanic = sns.load_dataset("titanic")
sns.catplot(x="alive", col="deck", col_wrap=4,
                data=titanic[titanic.deck.notnull()],
                kind="count", height=2.5, aspect=.8)
```

[682]: <seaborn.axisgrid.FacetGrid at 0x1d3b92a1bb0>



[684]:
```
sns.catplot(x='day', y='total_bill', data=tip, kind='box', palette='Accent')
plt.show()
```


50
40
30
20
10
total_bill
Thur
Fri
Sat
Sun
day

[ ]:
```
#
```