** Why Seaborn?**

- provides a layer of abstraction hence simpler to use
- better aesthetics
- more graphs included

** Seaborn Roadmap**

Types of Functions

- Figure Level
- Axis Level

Main Classification

- Relational Plot
- Distribution Plot
- Categorical Plot
- Regression Plot
- Matrix Plot
- Multiplots

https://seaborn.pydata.org/api.html

** 1. Relational Plot**

- to see the statistical relation between 2 or more variables.
- Bivariate Analysis

Plots under this section

- scatterplot
- lineplot

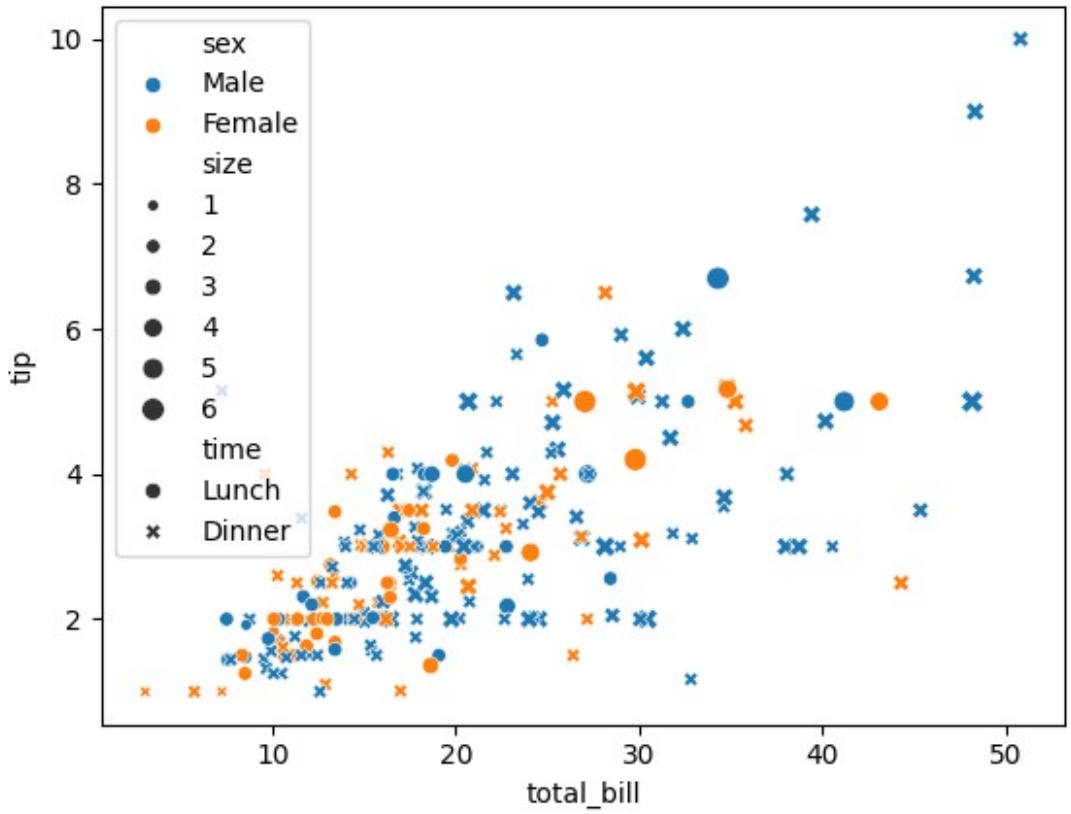
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px

tips = sns.load_dataset('tips')
tips.head()
```

```
{"summary":"{\n  \"name\": \"tips\",\n  \"rows\": 244,\n  \"fields\":
[\n    {\n      \"column\": \"total_bill\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8.902411954856856,\n
\"min\": 3.07,\n        \"max\": 50.81,\n
\"num_unique_values\": 229,\n        \"samples\": [\n          22.12,\
n          20.23,\n          14.78\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"tip\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.3836381890011826,\n
```

\"min\": 1.0,\n   \"max\": 10.0,\n   \"num_unique_values\": 123,\n   \"samples\": [\n   3.35,\n   1.5,\n   6.73\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"sex\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Male\",\n   \"Female\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"smoker\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Yes\",\n   \"No\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"day\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"Sat\",\n   \"Fri\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"time\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Lunch\",\n   \"Dinner\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"size\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 6,\n   \"num_unique_values\": 6,\n   \"samples\": [\n   2,\n   3\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"tips"}

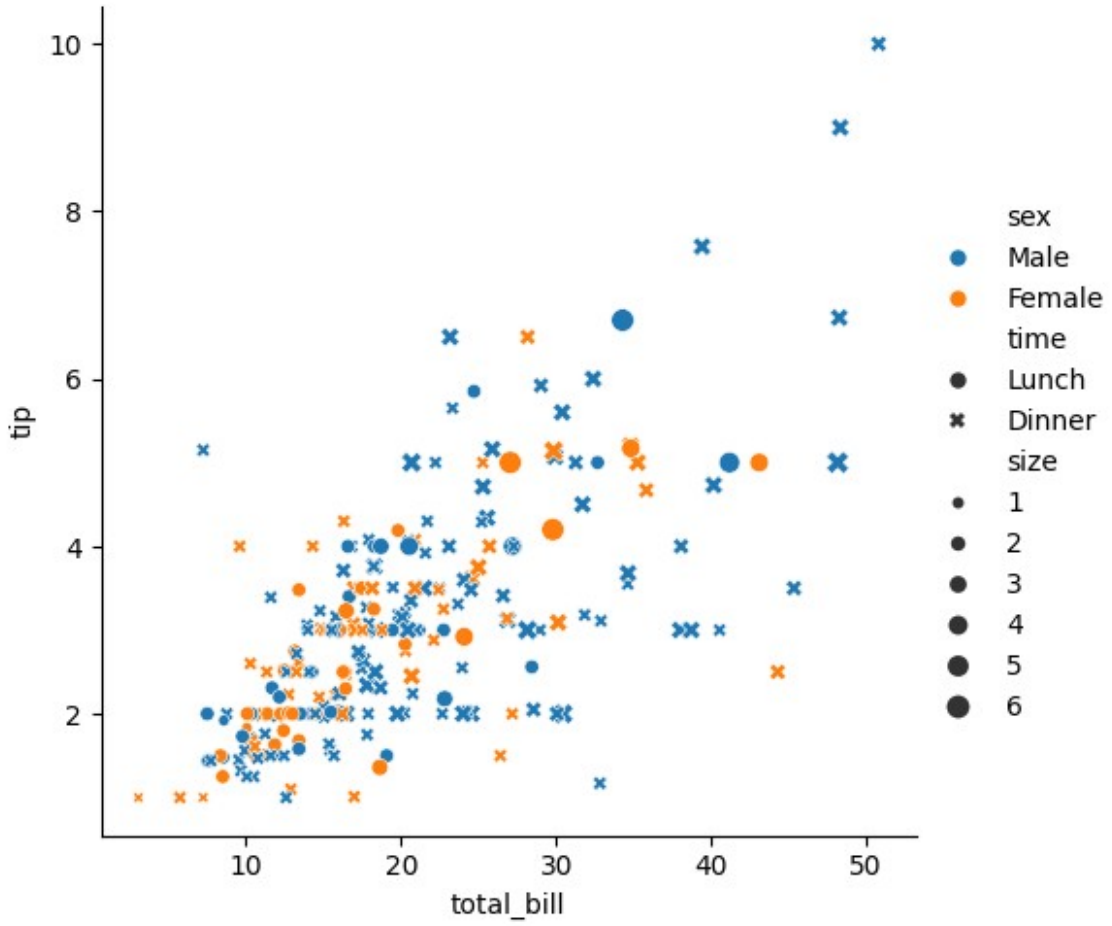
```
# scatter plot -> axes level function
sns.scatterplot(data=tips,x='total_bill',y='tip',hue='sex',style='time',size='size')
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```




```
# relplot -> figure level -> square shape
sns.relplot(data=tips,x='total_bill',y='tip',kind='scatter',hue='sex',
style='time',size='size')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66a4159c50>
```



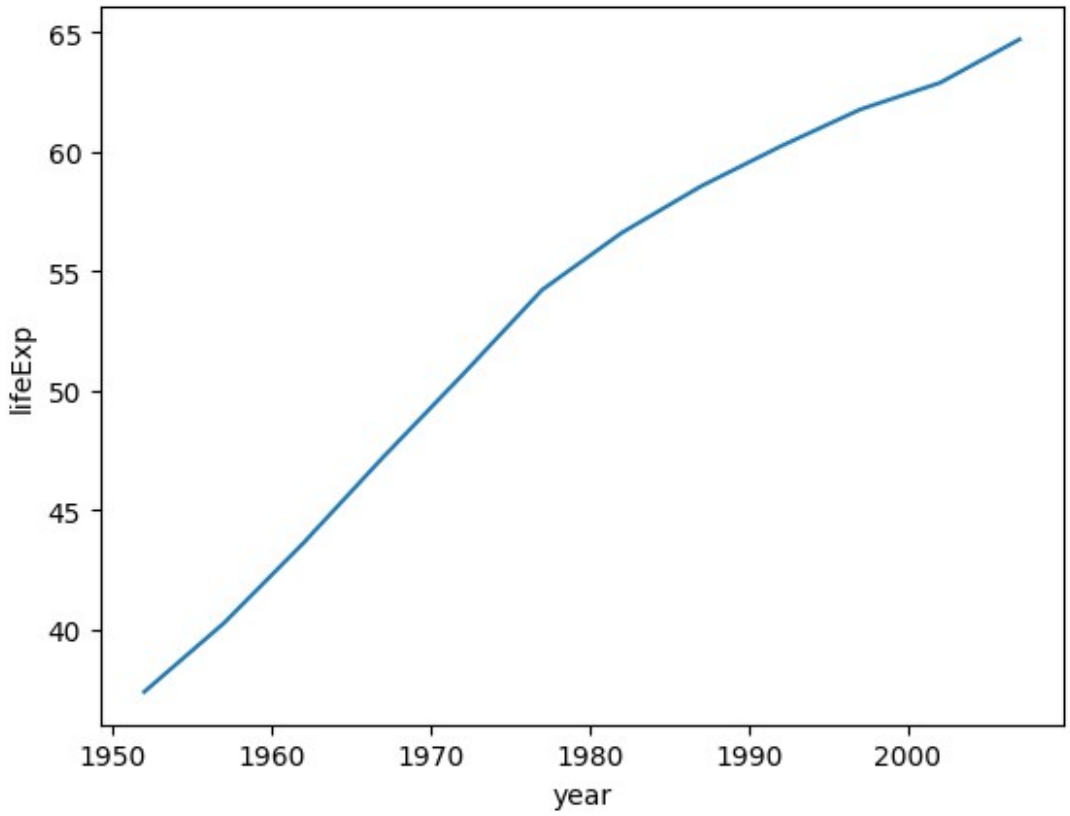

```
# line plot
gap = px.data.gapminder()
temp_df = gap[gap['country'] == 'India']
temp_df
```


```
{"summary":"{\n  \"name\": \"temp_df\",\n  \"rows\": 12,\n
\"fields\": [\n    {\n      \"column\": \"country\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n
\"India\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"continent\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"Asia\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 18,\n        \"min\": 1952,\n        \"max\": 2007,\n
\"num_unique_values\": 12,\n        \"samples\": [\n          2002\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"lifeExp\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

```
9.257067515818614,\n        \"min\": 37.37300000000001,\n
\"max\": 64.69800000000001,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          62.879\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"pop\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 251724253,\n        \"min\":
372000000,\n        \"max\": 1110396331,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
1034172547\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"gdpPercap\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 570.248220505152,\n        \"min\":
546.5657493,\n        \"max\": 2452.210407,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
1746.769454\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"iso_alpha\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"IND\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"iso_num\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0,\n        \"min\": 356,\n        \"max\": 356,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          356\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"temp_df"}
```

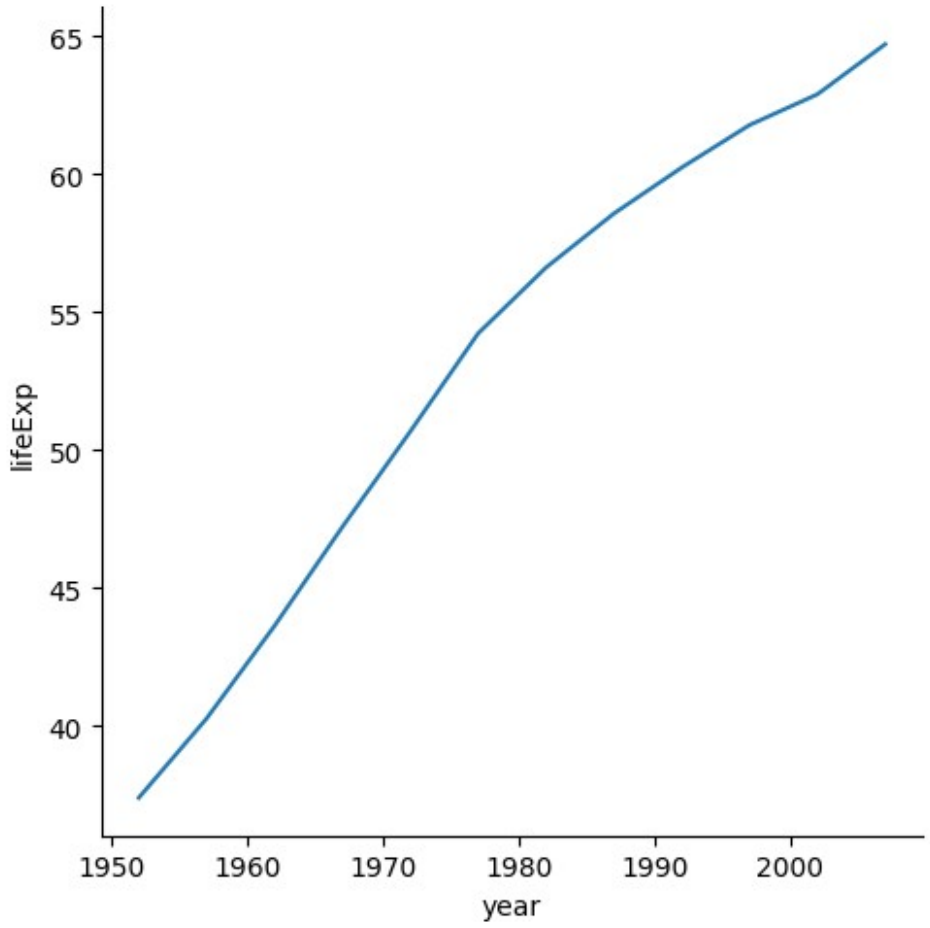

```
#axes level function
sns.lineplot(data=temp_df,x='year',y='lifeExp')
```

```
<Axes: xlabel='year', ylabel='lifeExp'>
```




```
#using relpolt
sns.relplot(data=temp_df,x='year',y='lifeExp',kind='line')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66a4752150>
```



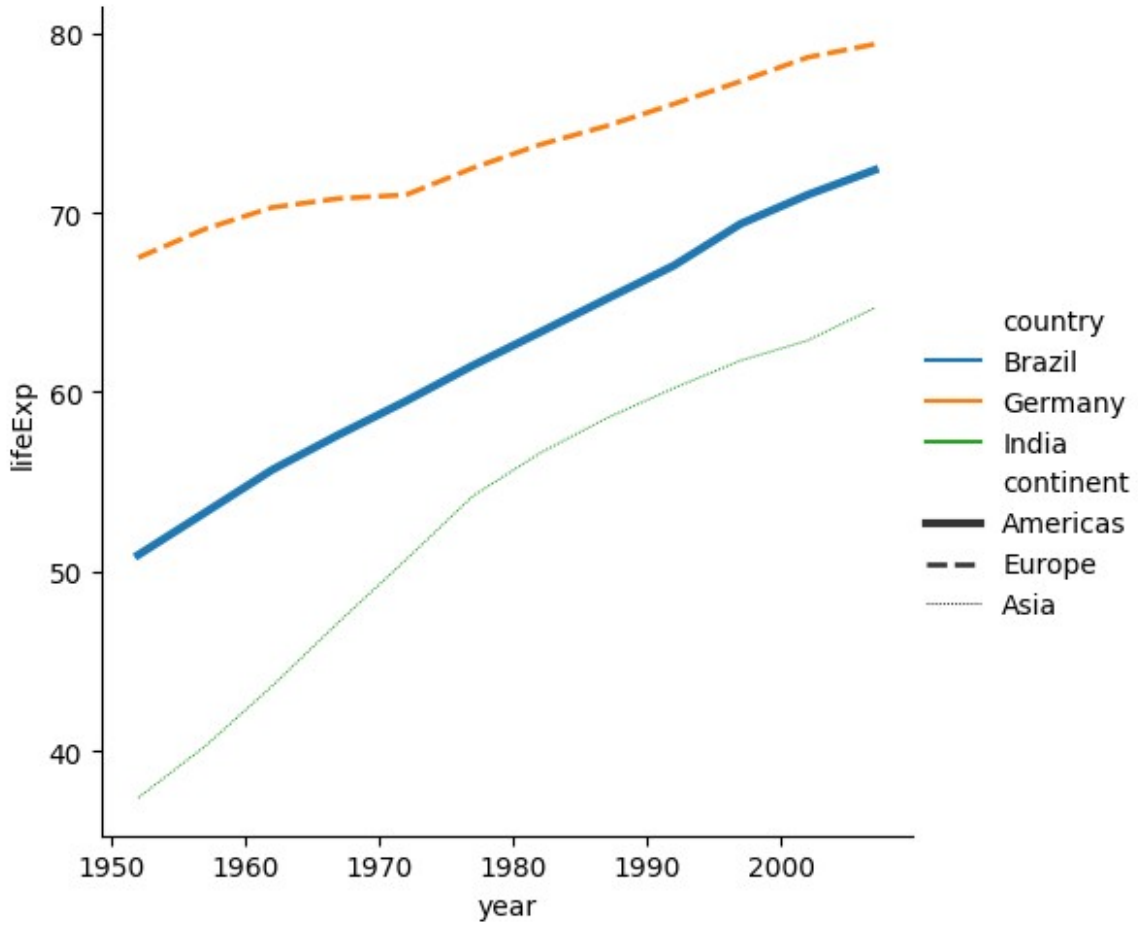

```
# hue and style
temp_df = gap[gap['country'].isin(['India','Brazil','Germany'])]
temp_df
```


```
{"summary":"{\n  \"name\": \"temp_df\",\n  \"rows\": 36,\n
\"fields\": [\n    {\n      \"column\": \"country\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Brazil\",\n          \"Germany\",\n          \"India\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"continent\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Americas\",\n          \"Europe\",\n          \"Asia\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"year\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 17,\n        \"min\": 1952,\n
\"max\": 2007,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          2002,\n          1997,\n          1952\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"lifeExp\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
10.861487254120798,\n        \"min\": 37.37300000000001,\n
\"max\": 79.406,\n        \"num_unique_values\": 36,\n
\"samples\": [\n          64.69800000000001,\n          69.1,\n
43.605\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"pop\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 321772361,\n        \"min\": 56602560,\n        \"max\":
1110396331,\n        \"num_unique_values\": 36,\n        \"samples\":
[\n          1110396331,\n          71019069,\n          454000000\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"gdpPercap\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9659.839054167098,\n        \"min\": 546.5657493,\n        \"max\":
32170.37442,\n        \"num_unique_values\": 36,\n        \"samples\":
[\n          2452.210407,\n          10187.82665,\n
658.3471509\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"iso_alpha\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"BRA\",\n          \"DEU\",\n          \"IND\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"iso_num\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 119,\n        \"min\": 76,\n        \"max\": 356,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          76,\n
276,\n          356\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"temp_df"}
```

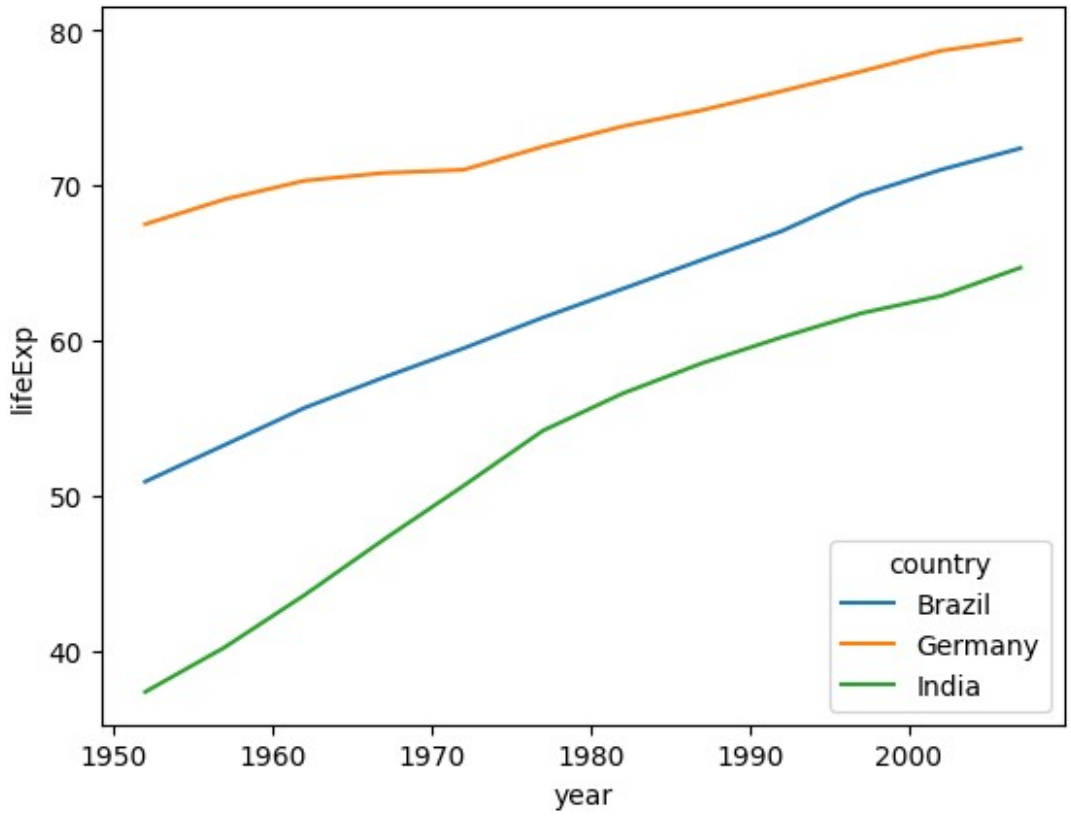

```
sns.relplot(data=temp_df,x='year',y='lifeExp',kind='line',hue='country
',style='continent',size='continent')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66a0bda050>
```



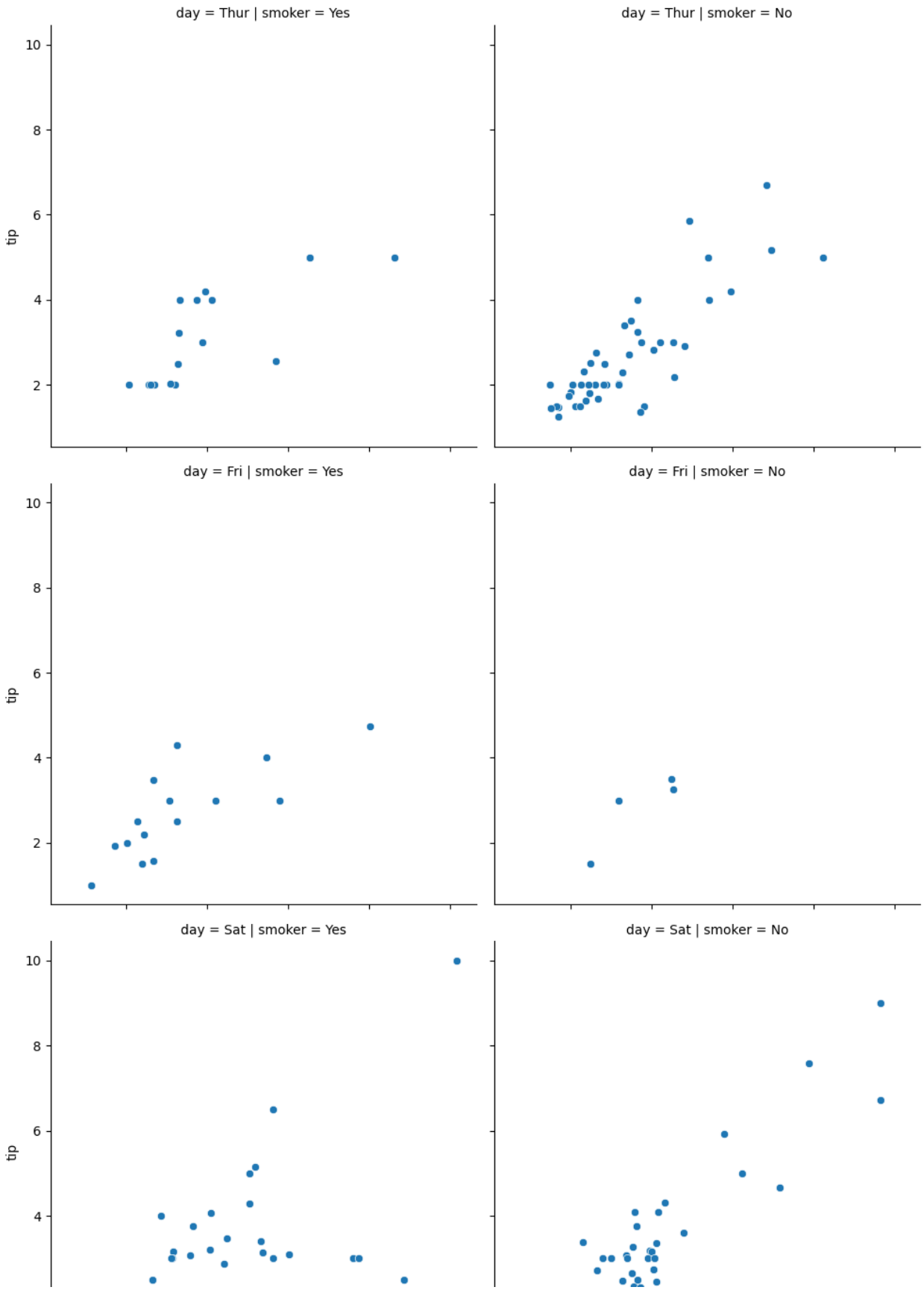

```
sns.lineplot(data=temp_df, x='year', y='lifeExp', hue='country')
```

```
<Axes: xlabel='year', ylabel='lifeExp'>
```



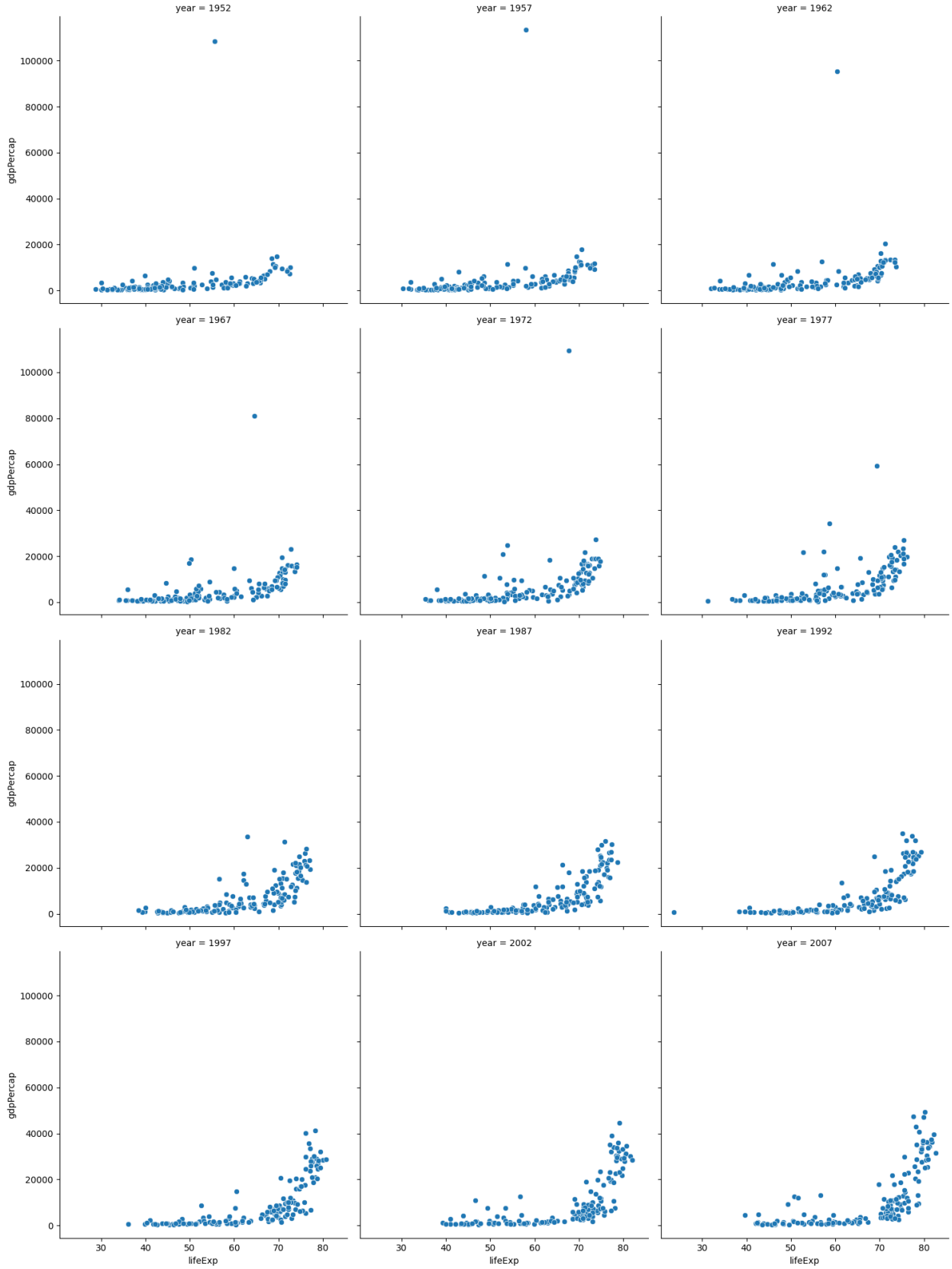

```
# facet plot-> figure level function -> work with relplot
# it will not work with scatterplot and lineplot
sns.relplot(data=tips,x='total_bill',y='tip',col='smoker',kind='line',
row='day')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d669feed550>
```


day = Thur | smoker = Yes
day = Thur | smoker = No
day = Fri | smoker = Yes
day = Fri | smoker = No
day = Sat | smoker = Yes
day = Sat | smoker = No
tip
10
8
6
4
2

```
# colwrap
sns.relplot(data=gap,x='lifeExp',y='gdpPercap',kind='scatter',col='yea
r',col_wrap=3)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d669ccdbc10>
```



2. Distribution Plots

used for univariate analysis

used to find out the distribution

Range of the observation

Central Tendency

is the data bimodal?

Are there outliers?

Plots under distribution plot

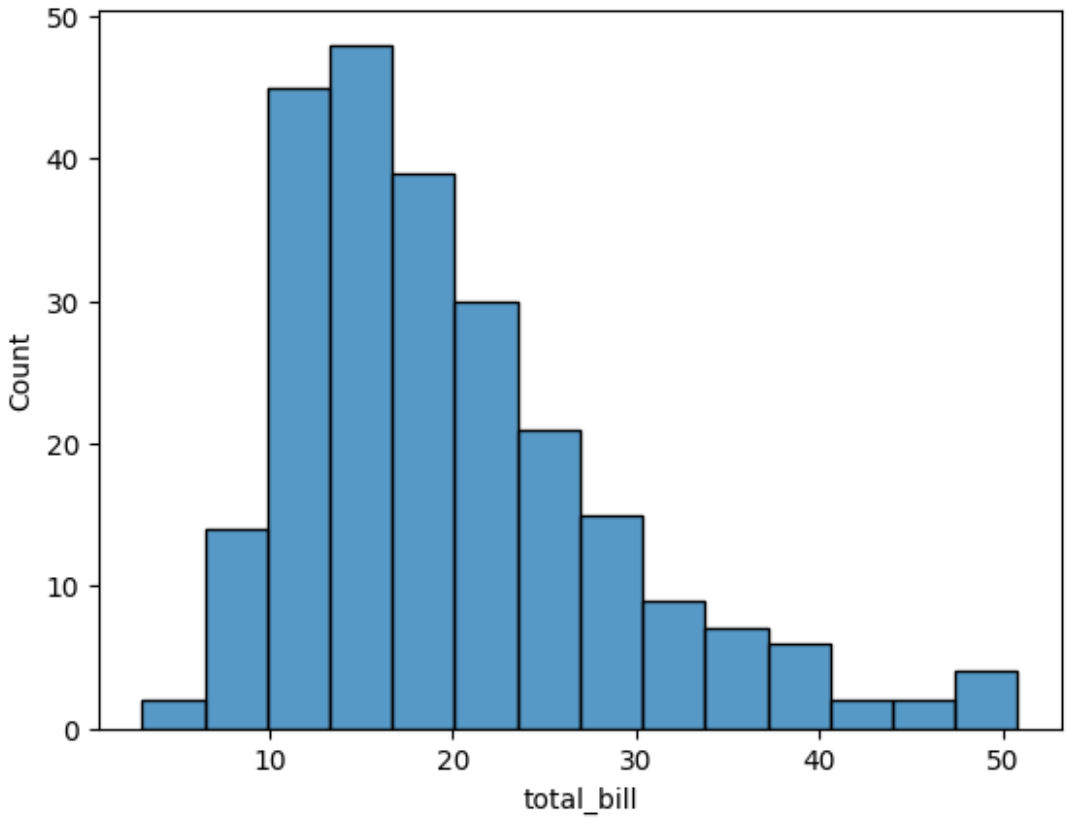
histplot

kdeplot

rugplot

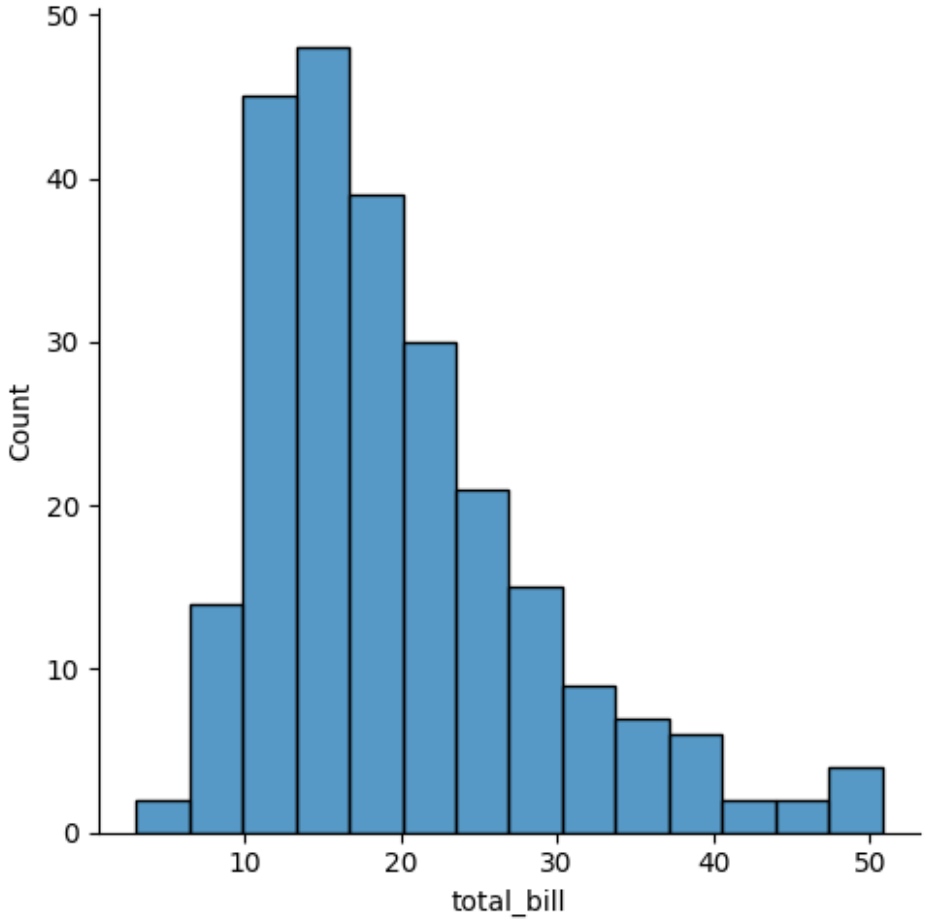
```
# figure level -> displot
# axes level -> histplot -> kdeplot -> rugplot

# plotting univariate histogram
sns.histplot(data=tips, x='total_bill')
```

```
<Axes: xlabel='total_bill', ylabel='Count'>
```



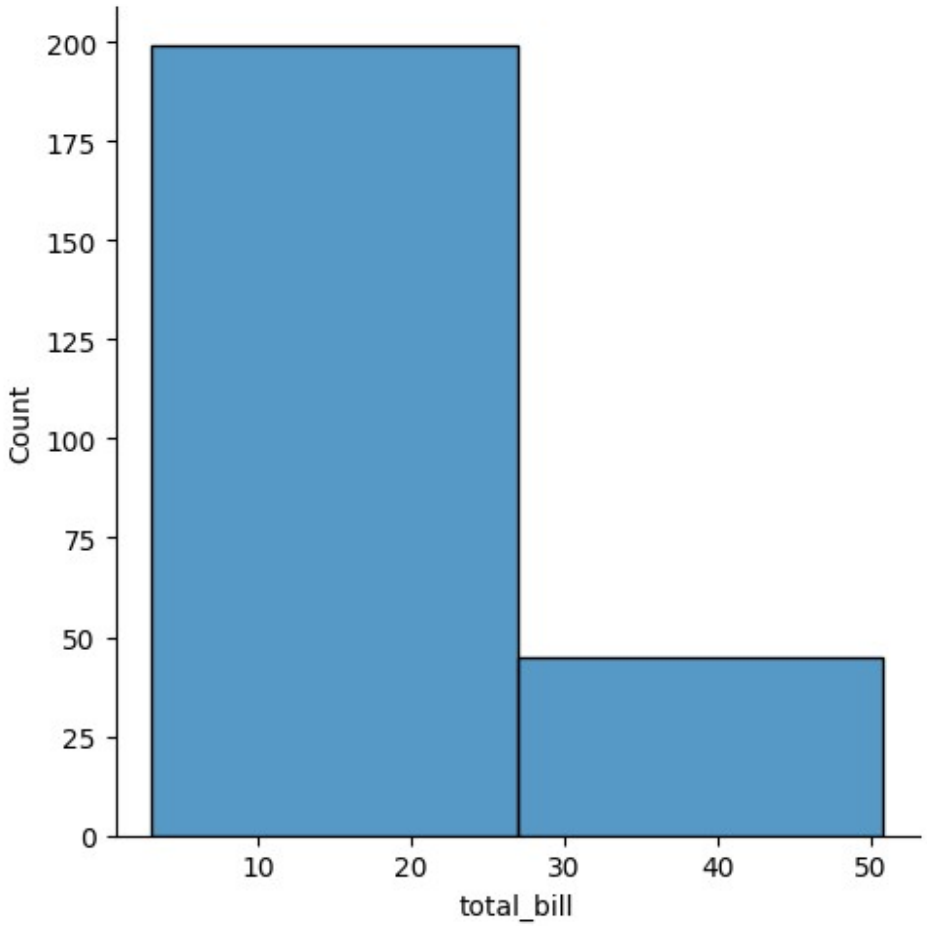
```
sns.displot(data=tips, x='total_bill',kind='hist')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d669d9d51d0>
```



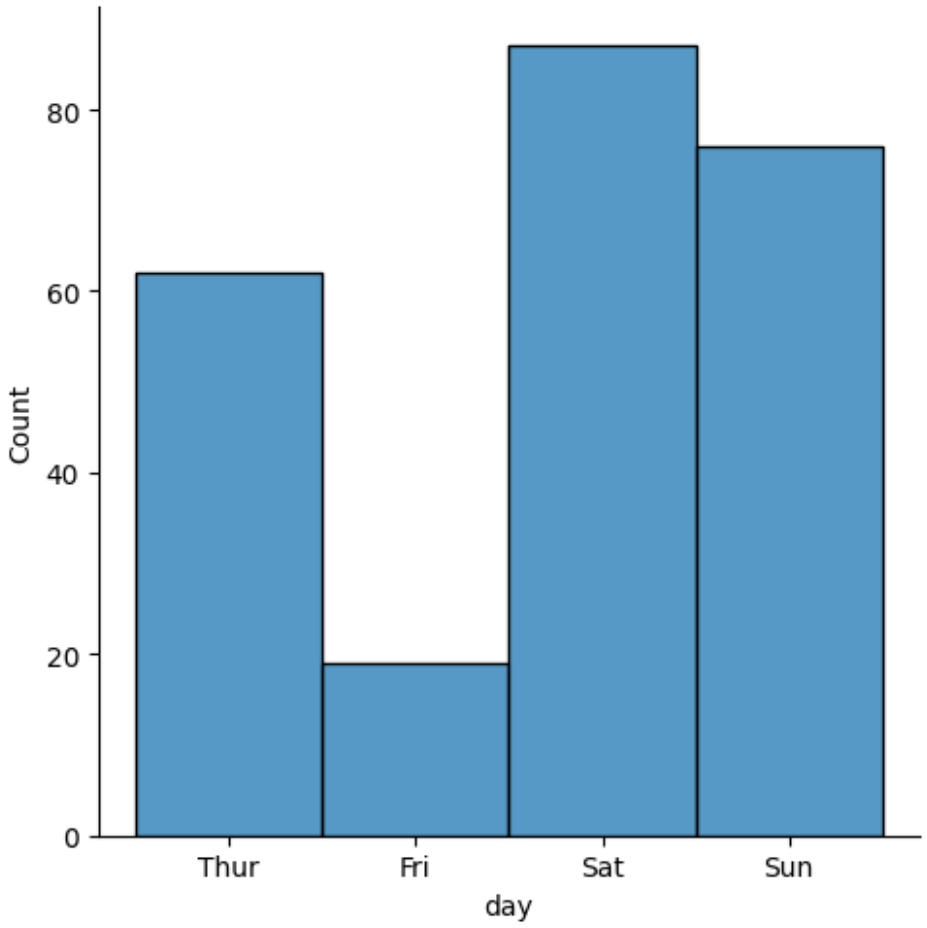

```
# bins
sns.displot(data=tips, x='total_bill',kind='hist',bins=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6697782050>
```



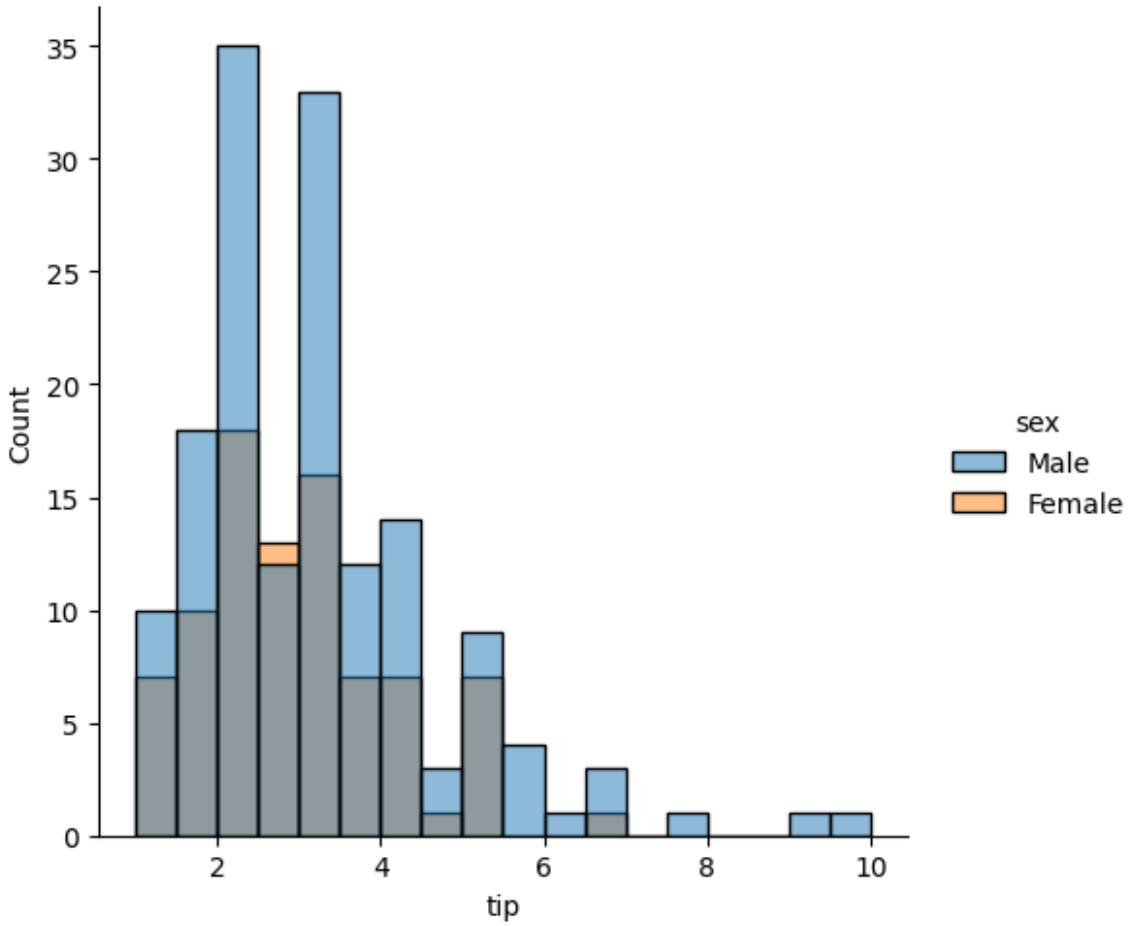

```
# It’s also possible to visualize the distribution of a categorical
variable using the logic of a histogram.
# Discrete bins are automatically set for categorical variables

# countplot
sns.displot(data=tips, x='day',kind='hist')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6696e42610>
```



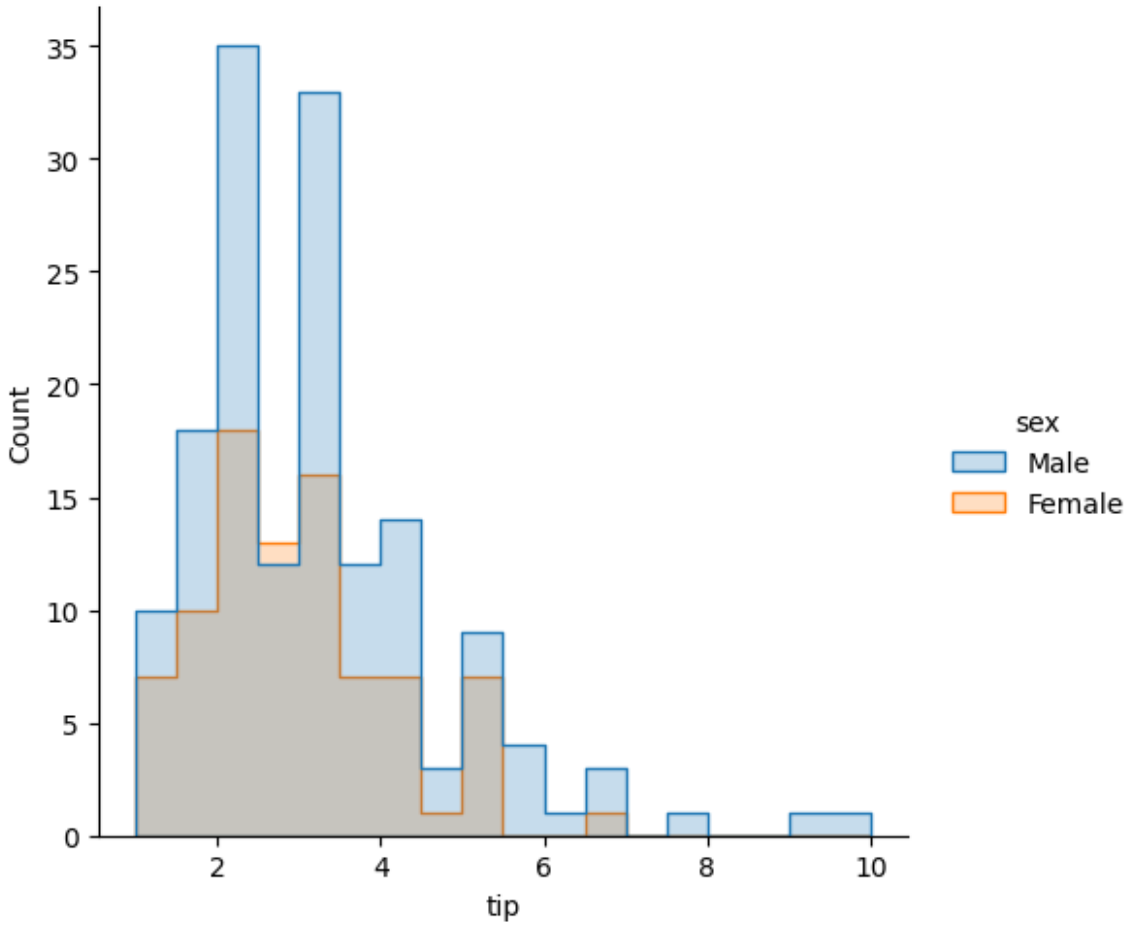

```
#hue parameter
sns.displot(data=tips, x='tip',hue='sex')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6696876750>
```




```
#element -> step
sns.displot(data=tips, x='tip',hue='sex',element='step')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6696662a90>
```




```
titanic = sns.load_dataset('titanic')
titanic
```

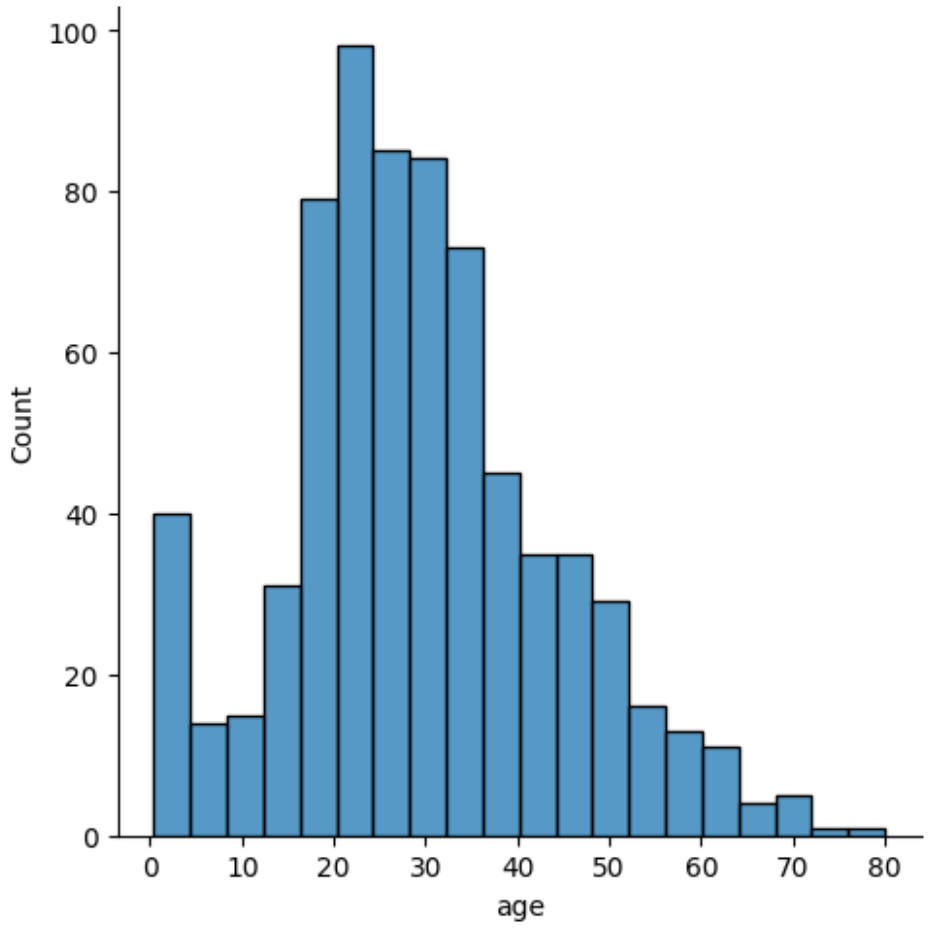

```
{"summary":"{\n  \"name\": \"titanic\",\n  \"rows\": 891,\n
\"fields\": [\n    {\n      \"column\": \"survived\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"pclass\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"female\",\n          \"male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14.526497332334044,\n
```

```
\"min\": 0.42,\n        \"max\": 80.0,\n        \"num_unique_values\":
88,\n        \"samples\": [\n          0.75,\n          22.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sibsp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 8,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"parch\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 6,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"fare\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 49.693428597180905,\n        \"min\": 0.0,\n        \"max\":
512.3292,\n        \"num_unique_values\": 248,\n        \"samples\":
[\n          11.2417,\n          51.8625\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"embarked\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"S\",\n          \"C\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"class\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"Third\",\n          \"First\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"who\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"man\",\
n          \"woman\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"adult_male\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"deck\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"C\",\n          \"E\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"embark_town\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Southampton\",\n          \"Cherbourg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"alive\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"yes\",\n          \"no\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"alone\",\n      \"properties\": {\
n        \"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"titanic"}
```

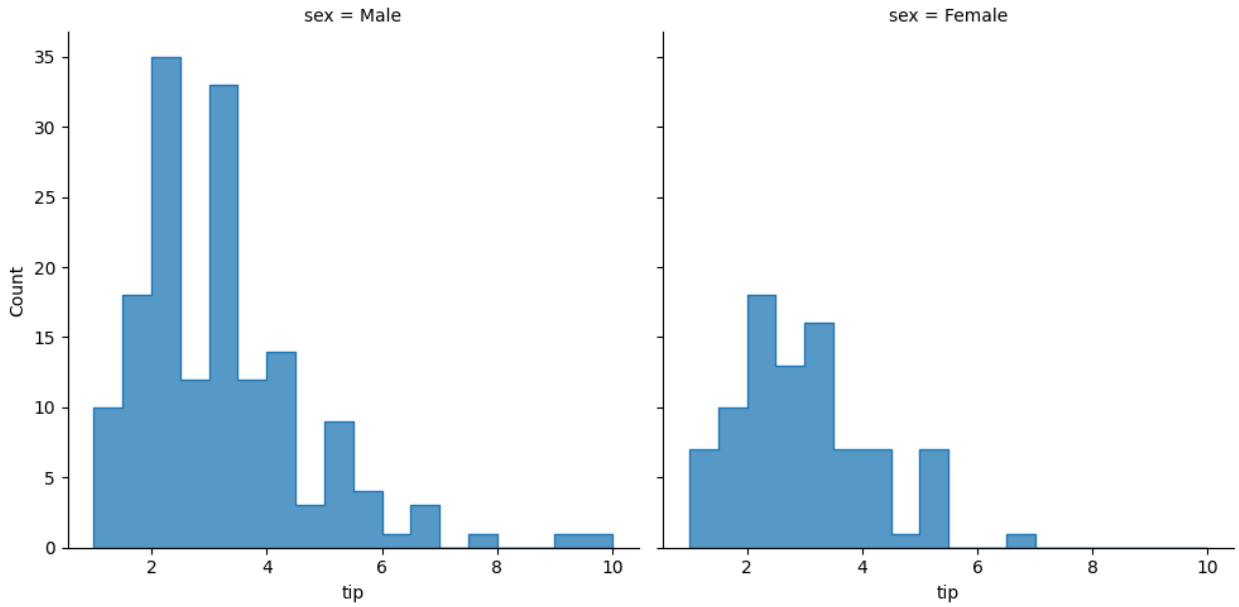
```
sns.displot(data=titanic,x='age',kind='hist')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66967886d0>
```



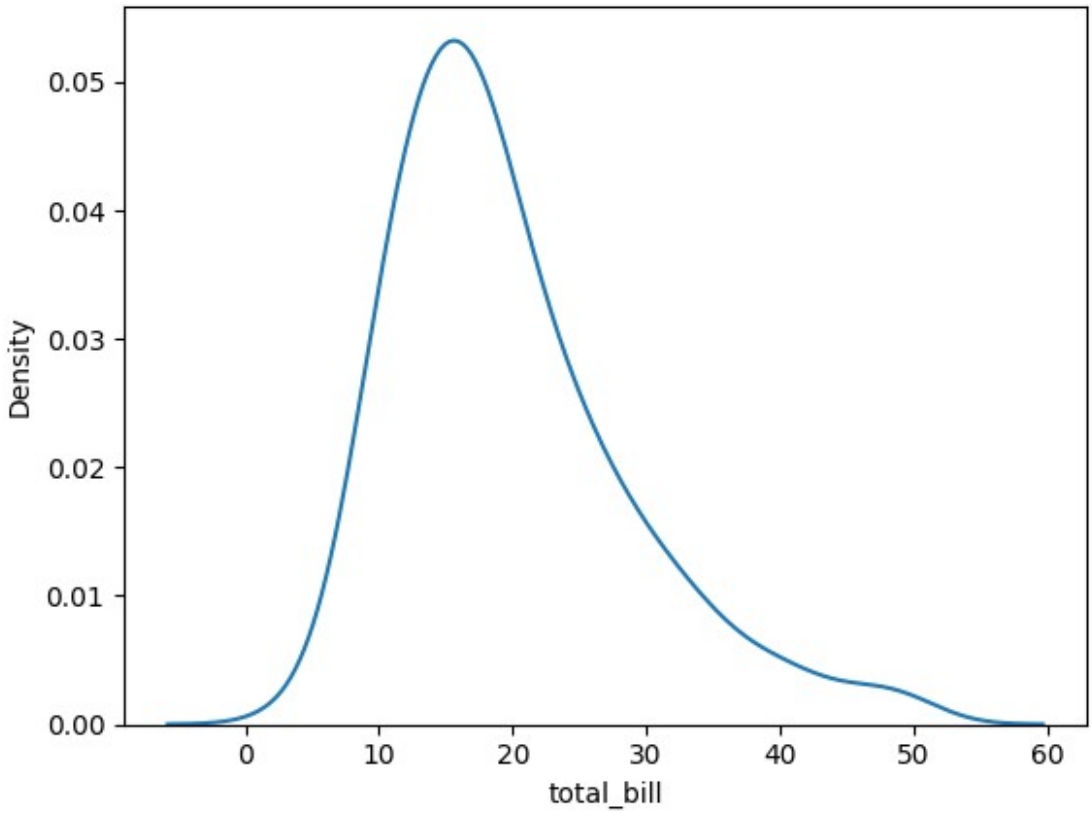

```
#faceting using col and row
sns.displot(data=tips,x='tip',kind='hist',col='sex',element='step')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6696ea24d0>
```



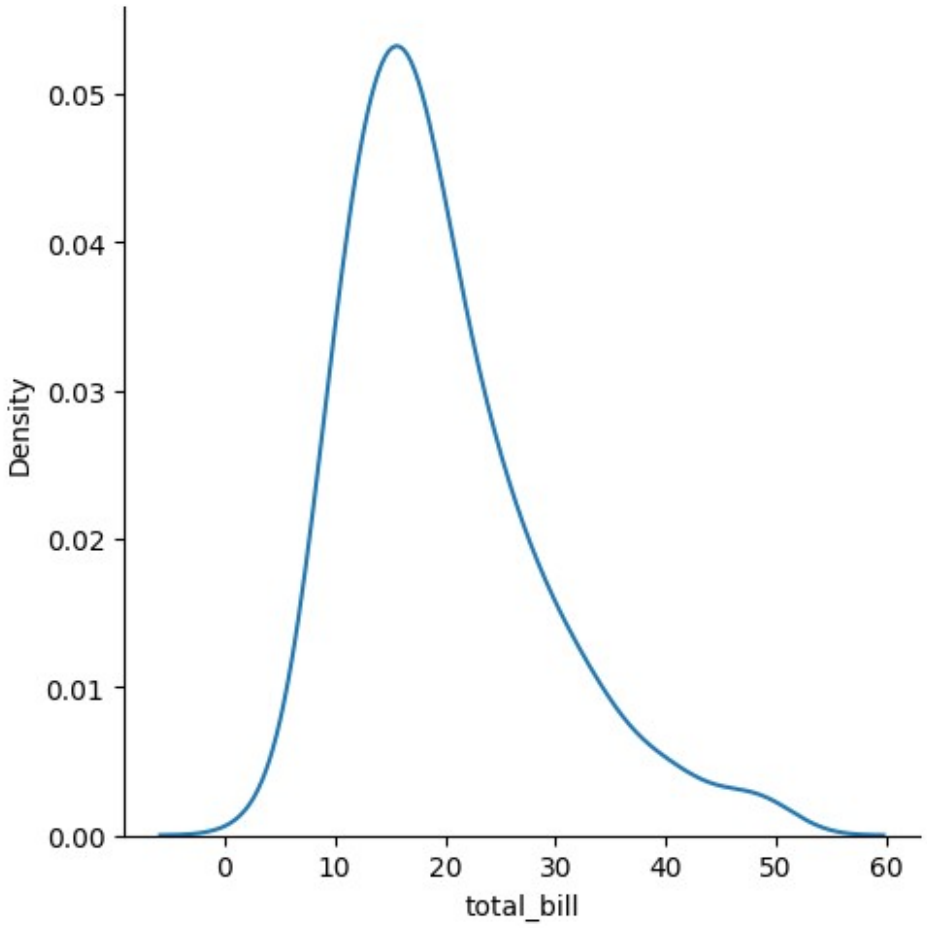

```
# kdeplot
# Rather than using discrete bins, a KDE plot smooths the observations
with a Gaussian kernel, producing a continuous density estimate
sns.kdeplot(data=tips,x='total_bill')
```

```
<Axes: xlabel='total_bill', ylabel='Density'>
```



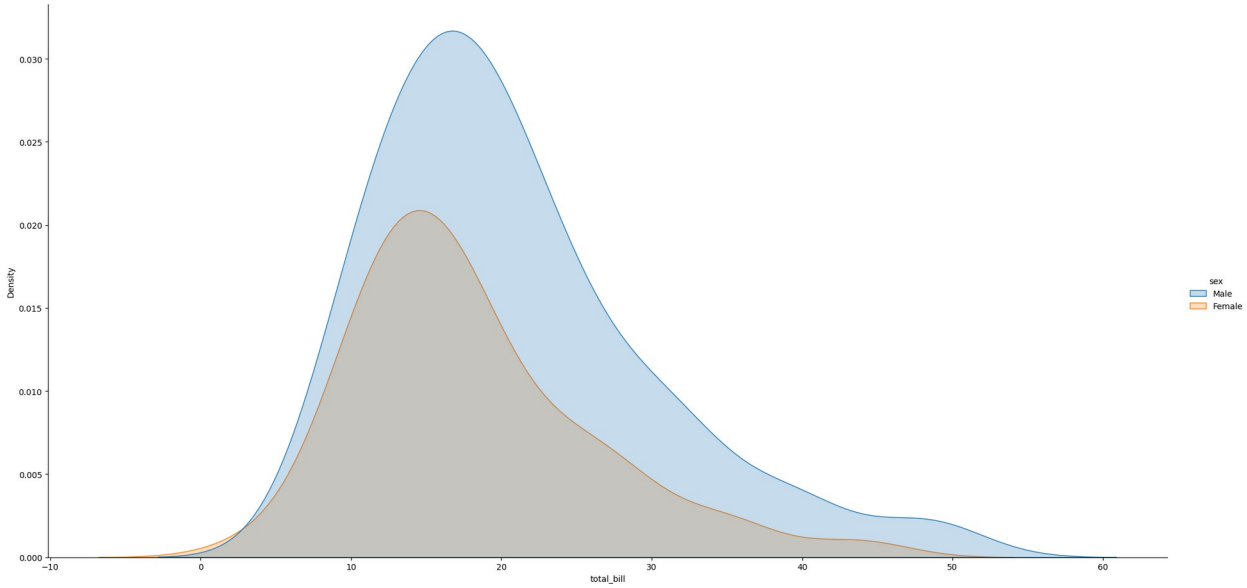

```
sns.displot(data=tips,x='total_bill',kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6695987610>
```



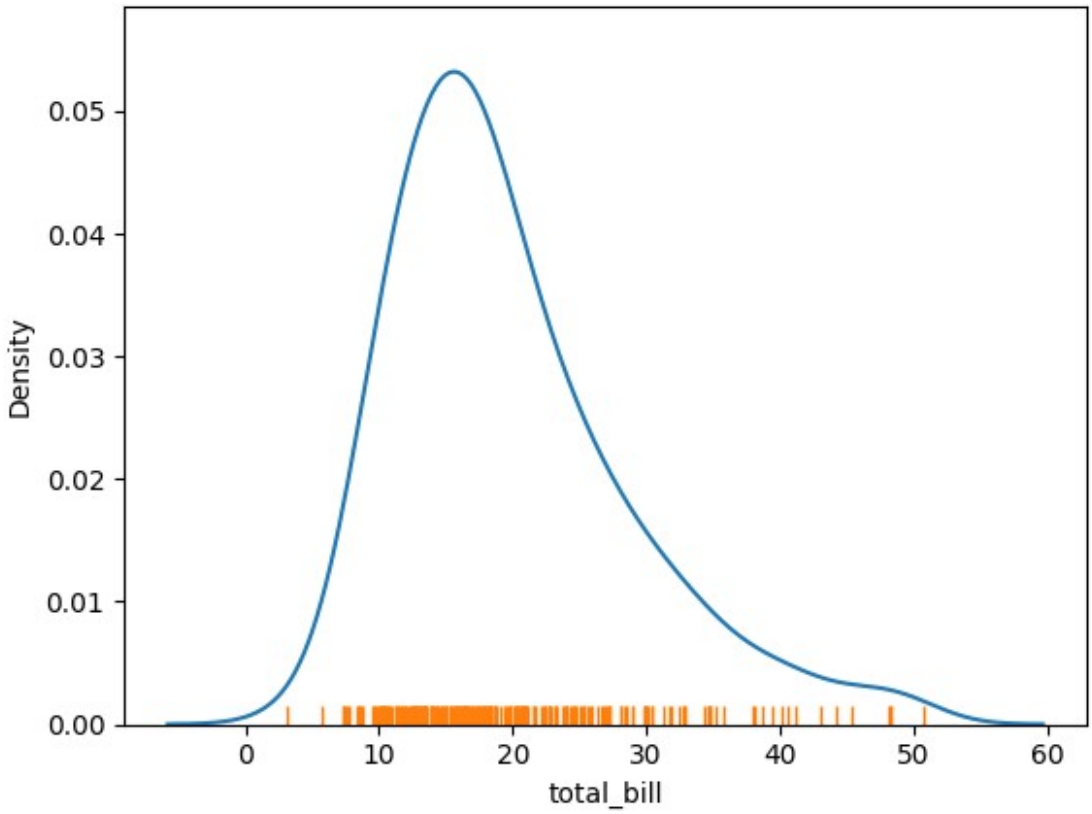

```
#hue -> fill
sns.displot(data=tips,x='total_bill',kind='kde',hue='sex',fill=True,he
ight=10,aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6696cc6e50>
```



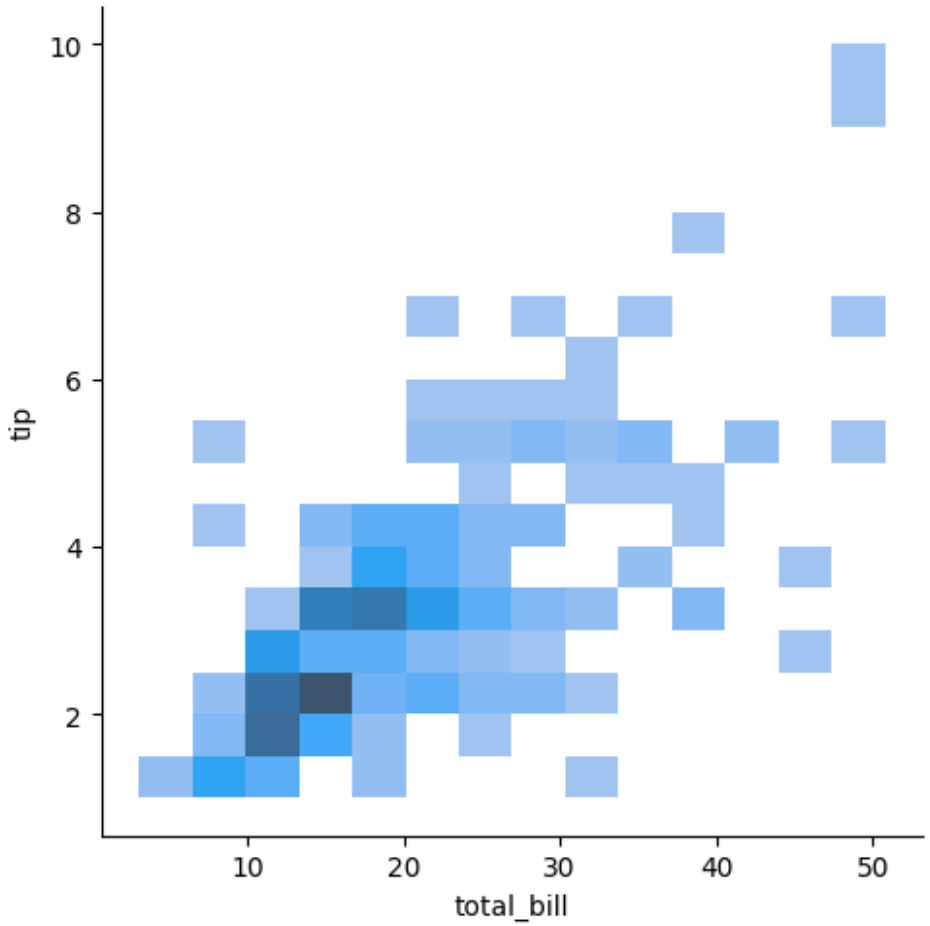

```
# Rugplot

# Plot marginal distributions by drawing ticks along the x and y axes.

# This function is intended to complement other plots by showing the
location of individual observations in an unobtrusive way.
sns.kdeplot(data=tips,x='total_bill')
sns.rugplot(data=tips,x='total_bill')
```

```
<Axes: xlabel='total_bill', ylabel='Density'>
```



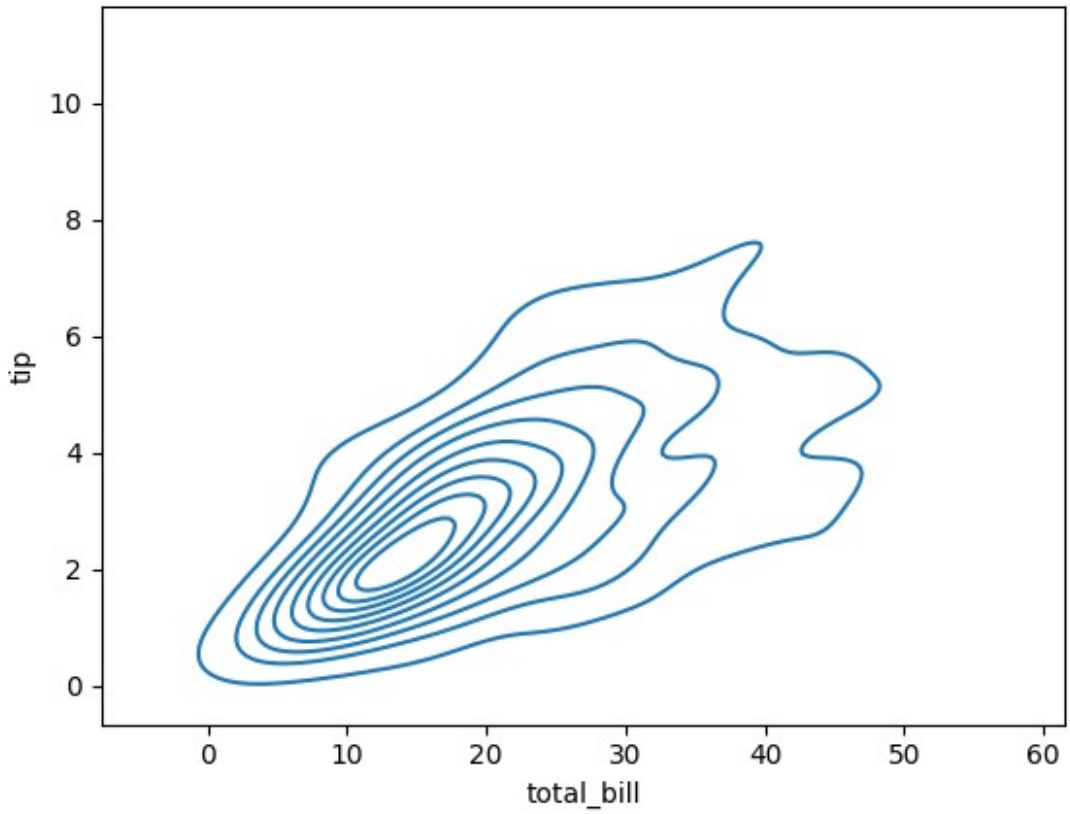

```
# Bivariate histogram
# A bivariate histogram bins the data within rectangles that tile the
plot
# and then shows the count of observations within each rectangle with
the fill color

# sns.histplot(data=tips, x='total_bill', y='tip')
sns.displot(data=tips,x='total_bill',y='tip',kind='hist')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d669691f2d0>
```



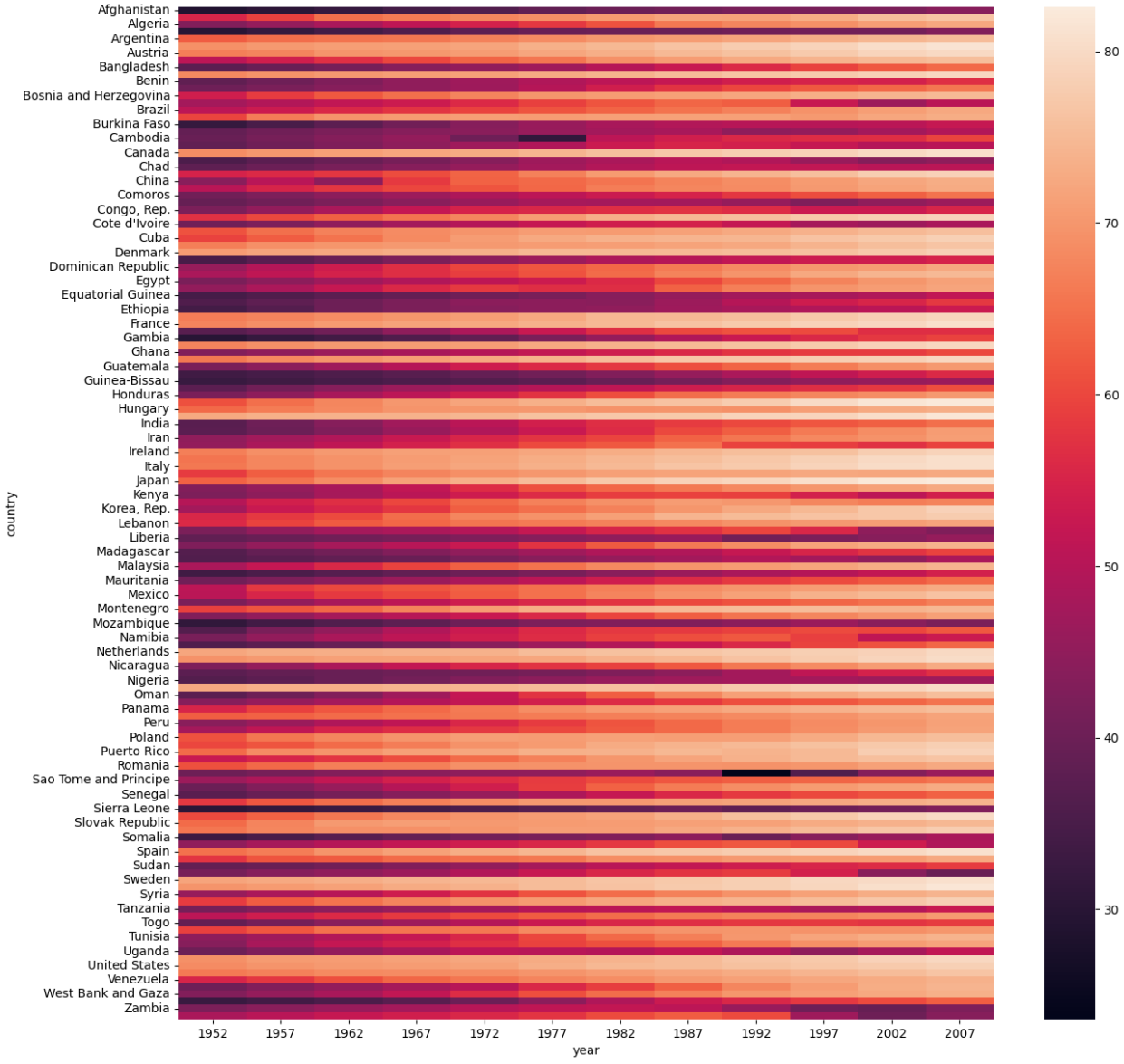

```
# Bivariate Kdeplot
# a bivariate KDE plot smoothes the (x, y) observations with a 2D
Gaussian
sns.kdeplot(data=tips, x='total_bill', y='tip')
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```




```
# Heatmap

# Plot rectangular data as a color-encoded matrix
temp_df = gap.pivot(index='country',columns='year',values='lifeExp')

# axes level function
plt.figure(figsize=(15,15))
sns.heatmap(temp_df)
```

```
<Axes: xlabel='year', ylabel='country'>
```




```
# annot
temp_df = gap[gap['continent'] ==
'Europe'].pivot(index='country',columns='year',values='lifeExp')

plt.figure(figsize=(15,15))
sns.heatmap(temp_df,annot=True,linewidth=0.5, cmap='summer')
```

```
<Axes: xlabel='year', ylabel='country'>
```

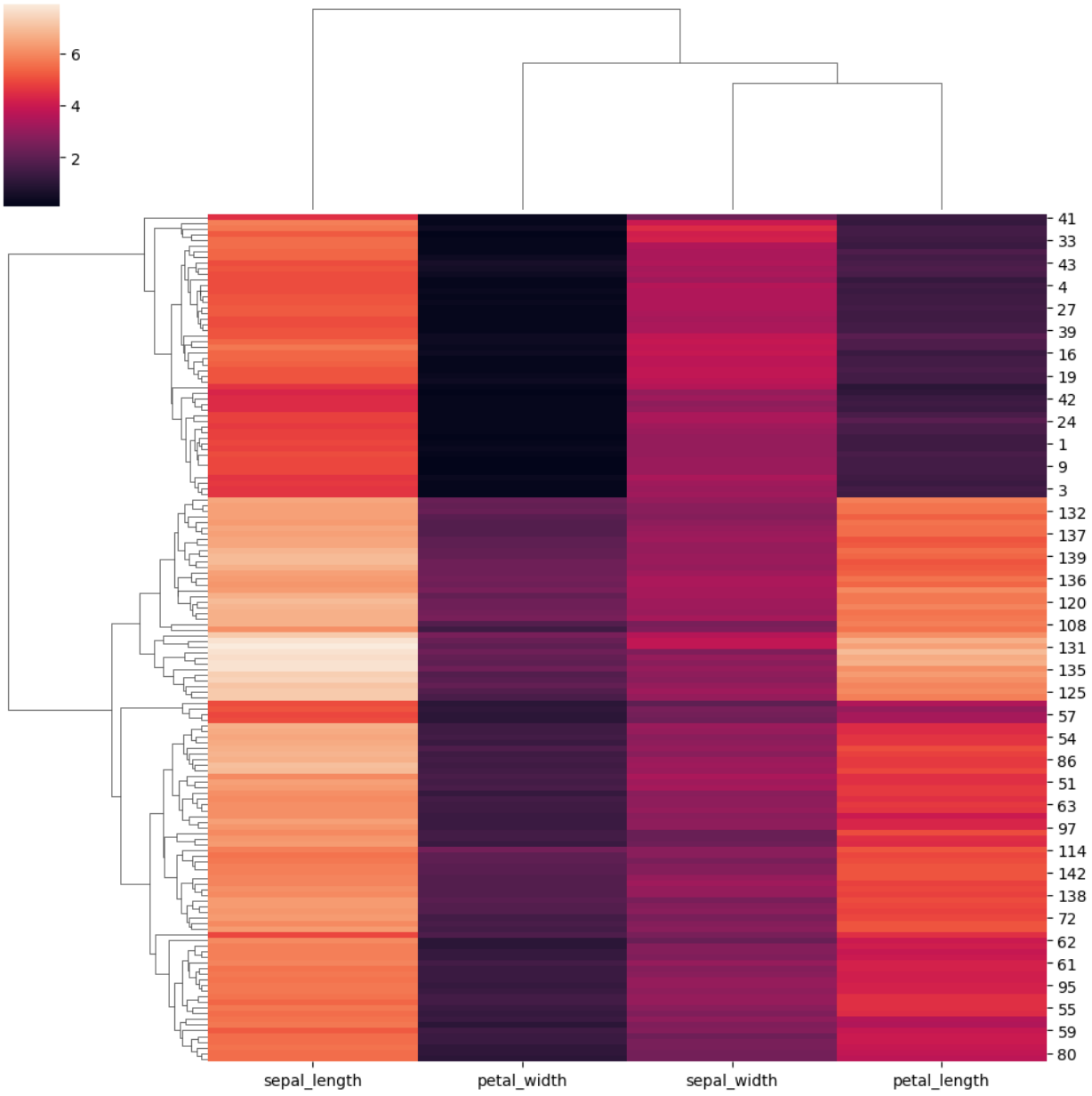

```
# Clustermap

# Plot a matrix dataset as a hierarchically-clustered heatmap.

# This function requires scipy to be available.

iris = px.data.iris()
iris
```


```
{"summary":"{\n  \"name\": \"iris\",\n  \"rows\": 150,\n  \"fields\":
[\n    {\n      \"column\": \"sepal_length\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.8280661279778629,\n        \"min\": 4.3,\n        \"max\": 7.9,\n
\"num_unique_values\": 35,\n        \"samples\": [\n          6.2,\n
4.5,\n          5.6\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"sepal_width\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.4335943113621737,\n   \"min\":
2.0,\n   \"max\": 4.4,\n   \"num_unique_values\": 23,\n
\"samples\": [\n   2.3,\n   4.0,\n   3.5\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"petal_length\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1.7644204199522617,\n   \"min\": 1.0,\n   \"max\": 6.9,\n
\"num_unique_values\": 43,\n   \"samples\": [\n   6.7,\n
3.8,\n   3.7\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"petal_width\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.7631607417008414,\n   \"min\":
0.1,\n   \"max\": 2.5,\n   \"num_unique_values\": 22,\n
\"samples\": [\n   0.2,\n   1.2,\n   1.3\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"species\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n
\"setosa\",\n   \"versicolor\",\n   \"virginica\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"species_id\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 1,\n   \"max\": 3,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   1,\n
2,\n   3\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"iris"}

```
sns.clustermap(iris.iloc[:,[0,1,2,3]])
```

```
<seaborn.matrix.ClusterGrid at 0x7d66955cbb90>
```



Categorical Plots

###Categorical Scatter Plot

- Stripplot
- Swarmplot

Categorical Distribution Plots

- Boxplot
- Violinplot

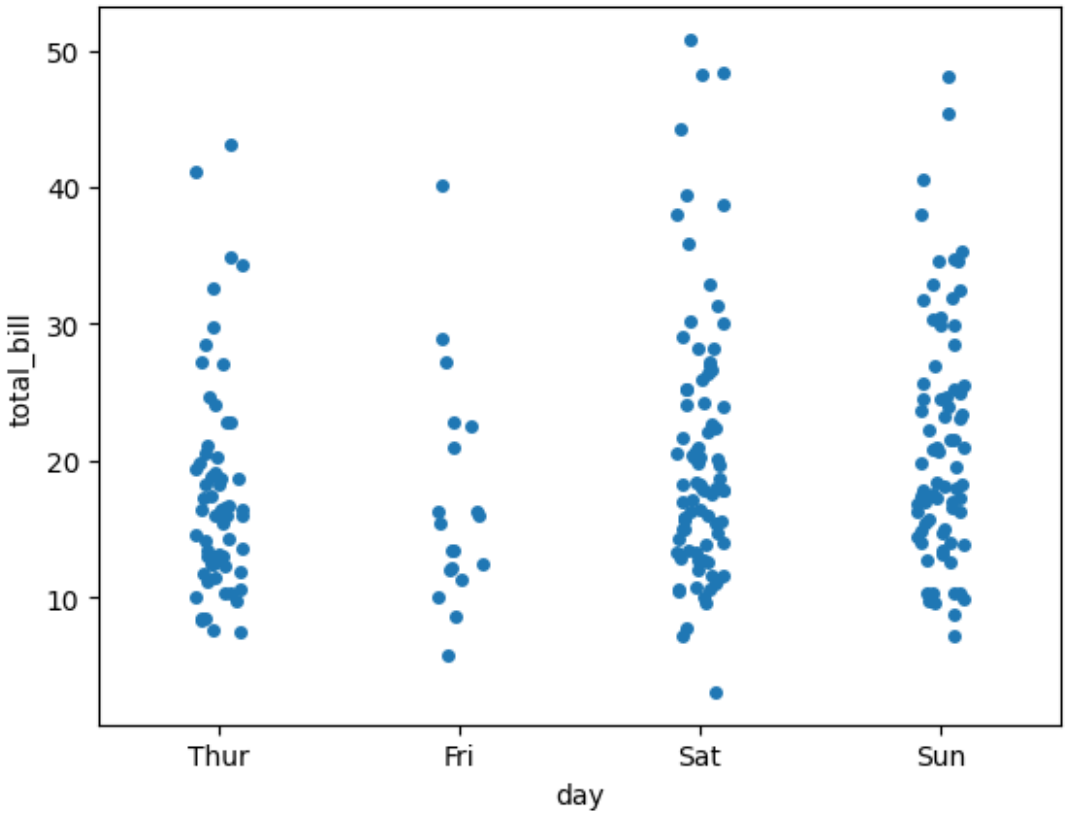
Categorical Estimate Plot -> for central tendency

- Barplot
- Pointplot
- Countplot

Figure level function -> `catplot`

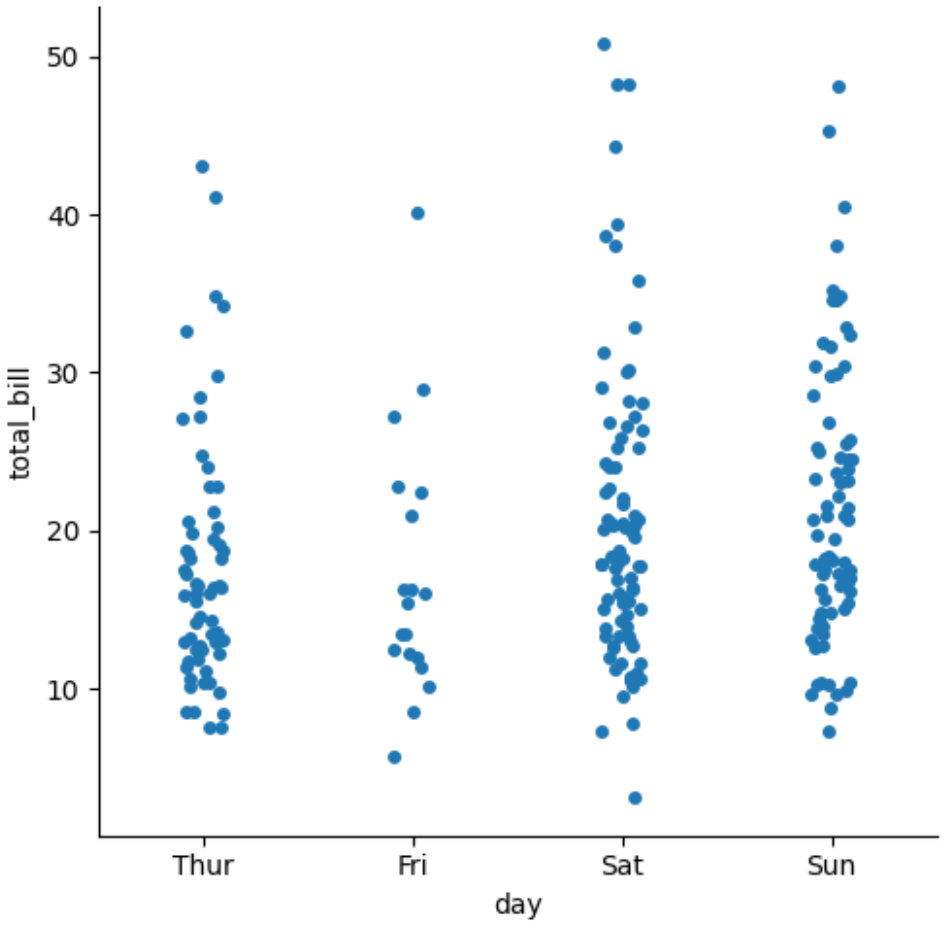
```
# strip plot
# axes level function
sns.stripplot(data=tips,x='day',y='total_bill')
```

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



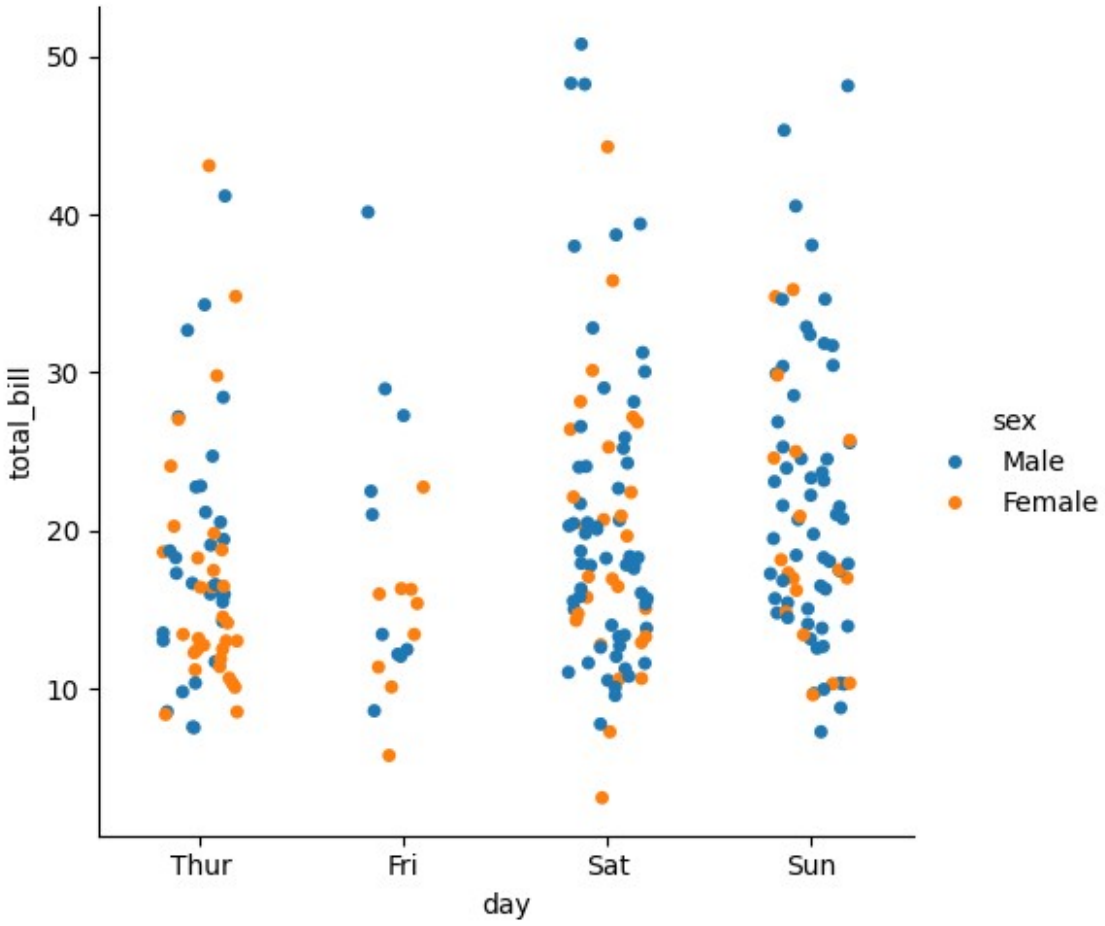

```
# using catplot
# figure level function
sns.catplot(data=tips,x='day',y='total_bill',kind='strip')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6695595f90>
```



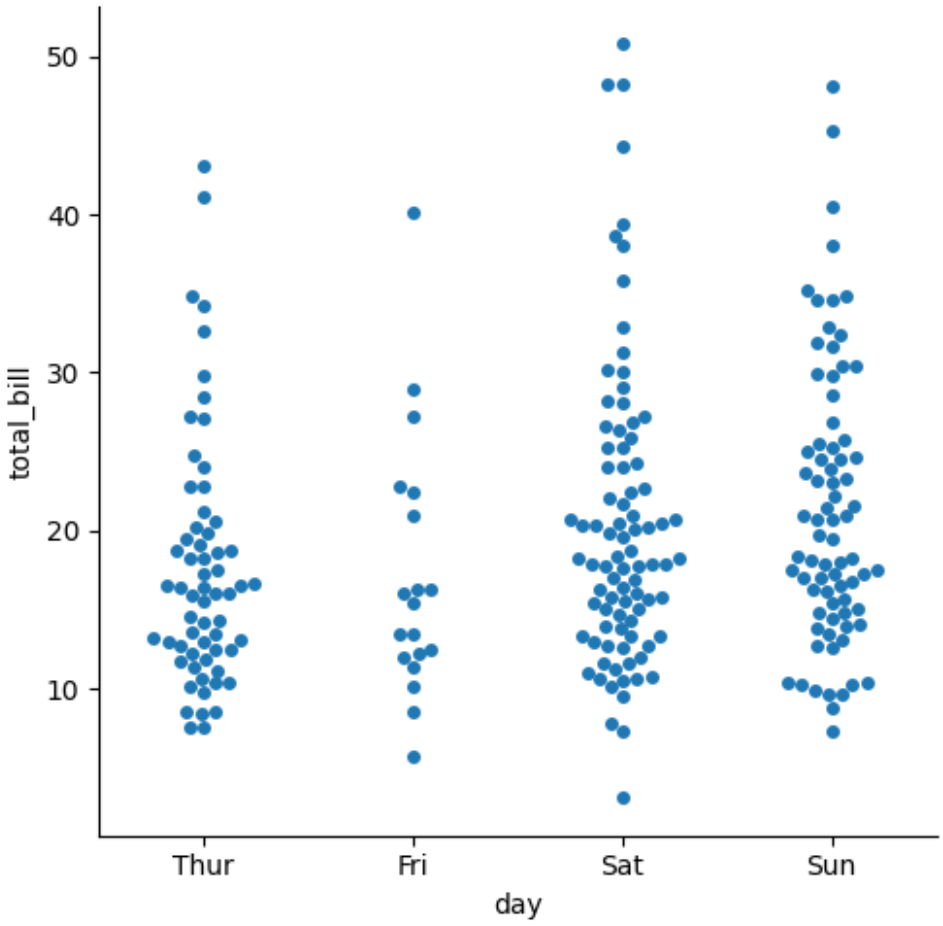

```
# jitter
sns.catplot(data=tips,x='day',y='total_bill',kind='strip',jitter=0.2,h
ue='sex')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6694c26750>
```



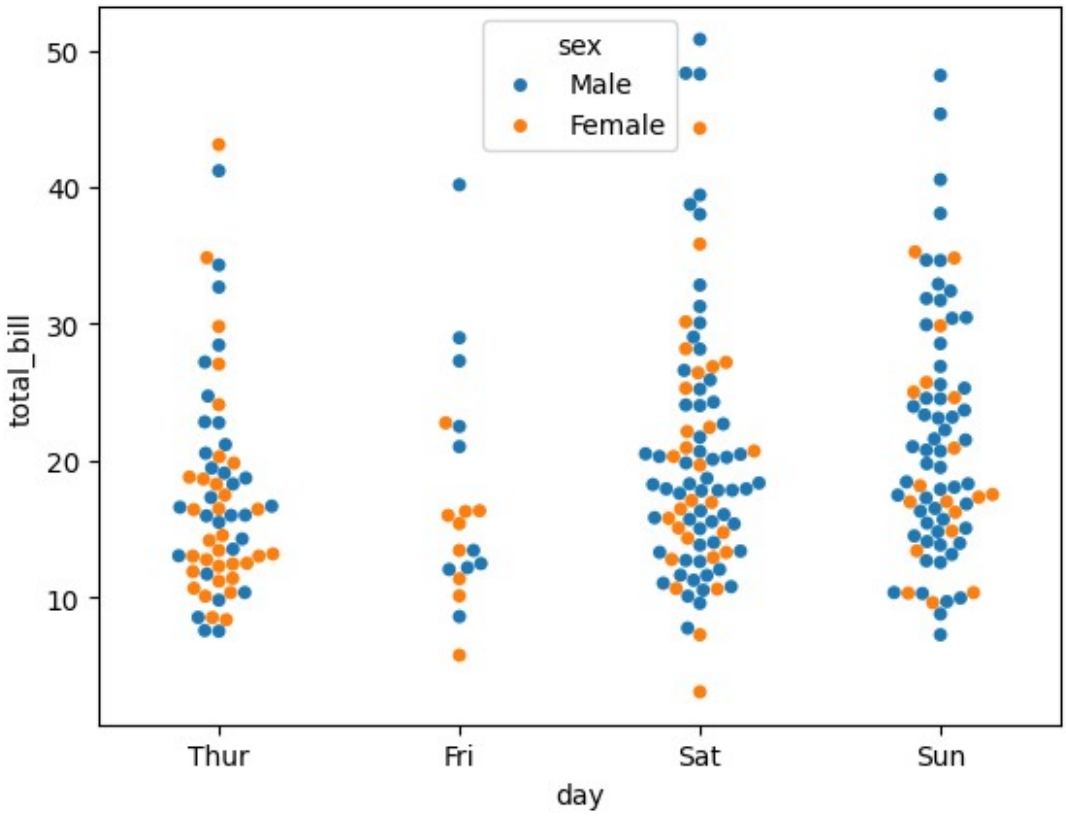

```
#swarmplot
sns.catplot(data=tips,x='day',y='total_bill',kind='swarm')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66945fc250>
```




```
#hue
sns.swarmplot(data=tips,x='day',y='total_bill',hue='sex')
```

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

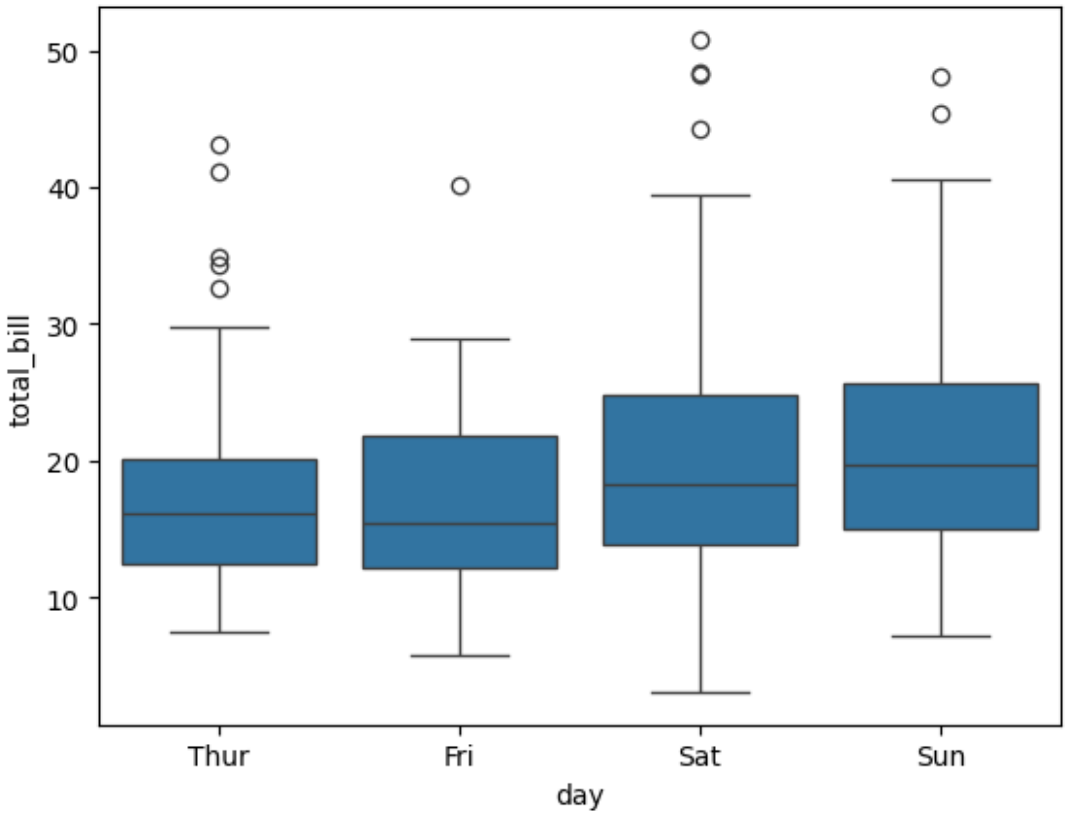


Boxplot

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile [Q1], median, third quartile [Q3] and “maximum”). It can tell you about your outliers and what their values are. Boxplots can also tell you if your data is symmetrical, how tightly your data is grouped and if and how your data is skewed.

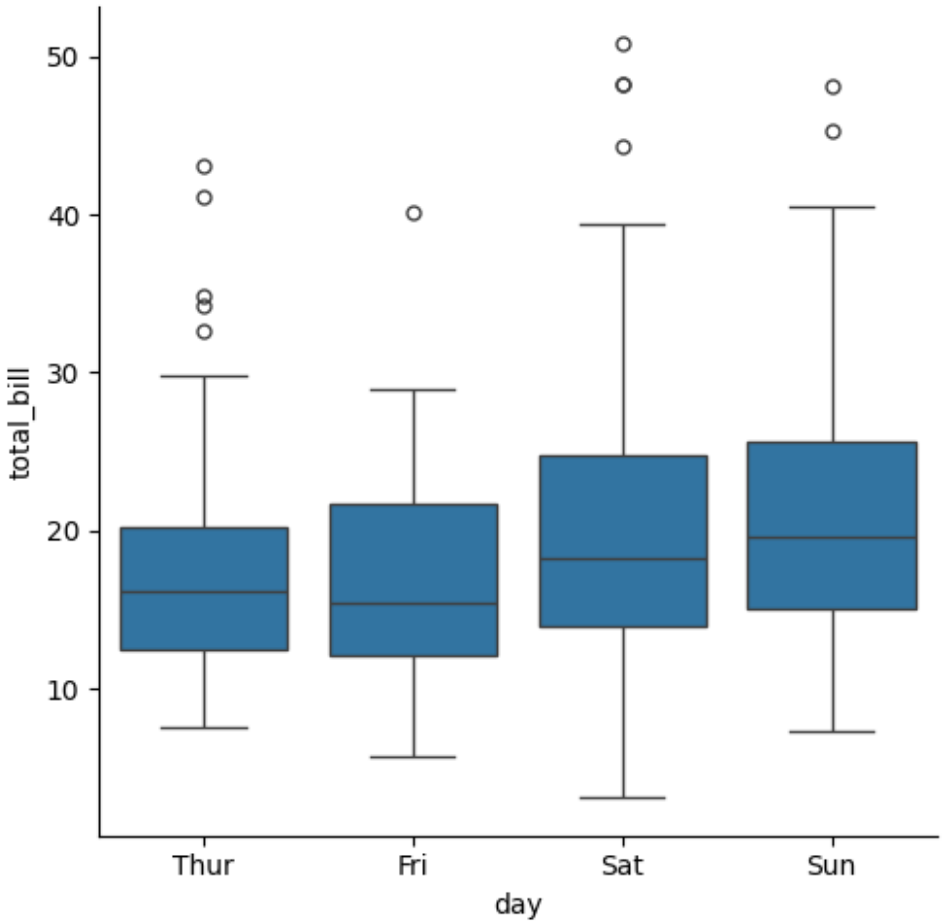
```
#Box plot
sns.boxplot(data=tips,x='day',y='total_bill')
```

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



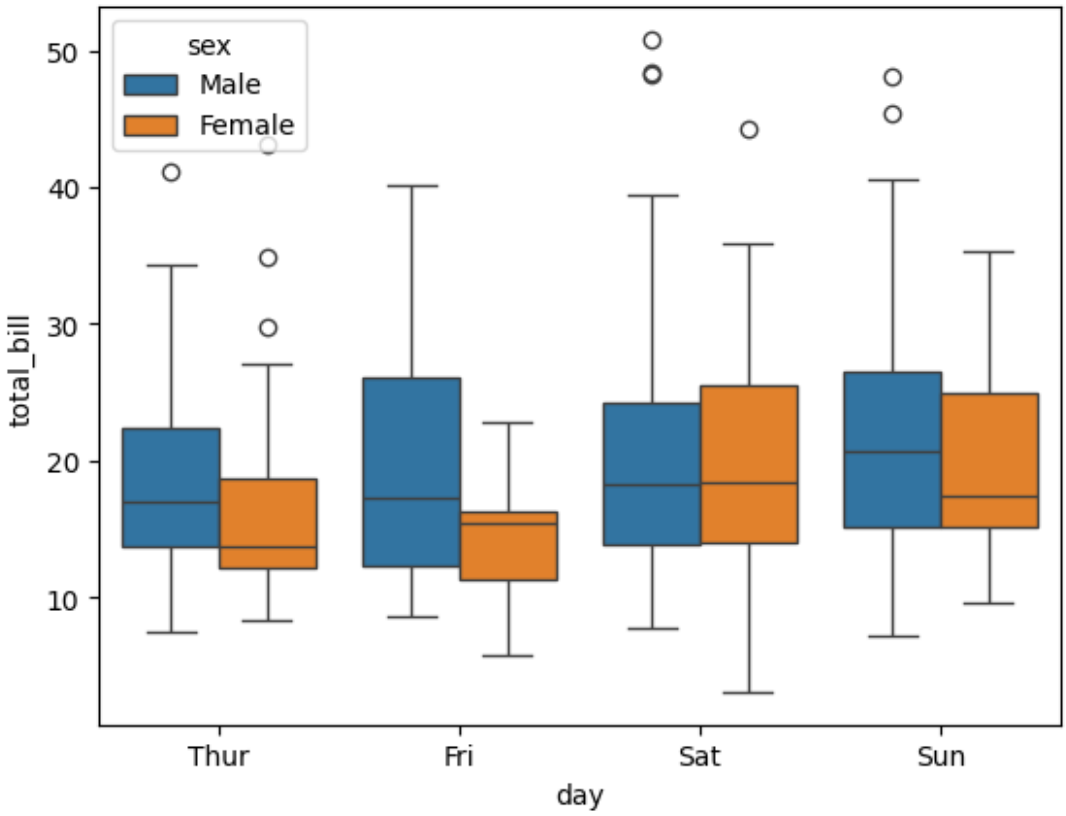

```
# Using catplot
sns.catplot(data=tips,x='day',y='total_bill',kind='box')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d6694587f90>
```



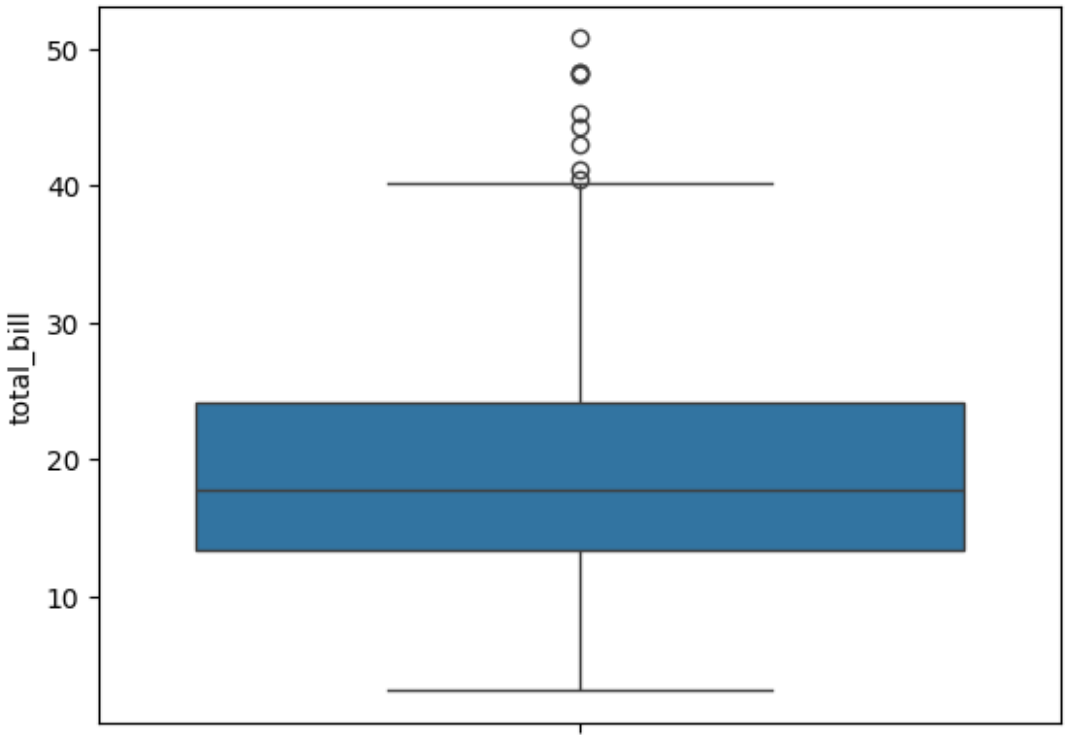

```
# hue
sns.boxplot(data=tips,x='day',y='total_bill',hue='sex')
```

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




```
sns.boxplot(data=tips,y='total_bill')
```

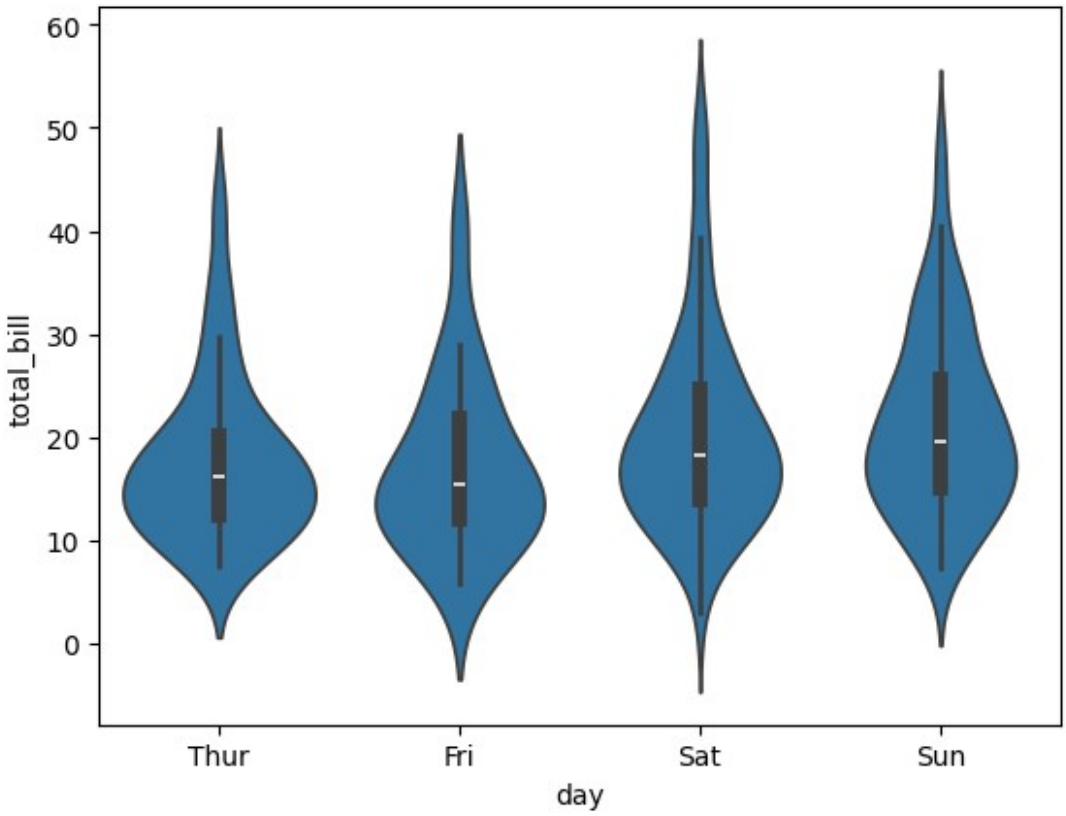
```
<Axes: ylabel='total_bill'>
```



Violinplot = (Boxplot + KDEplot)

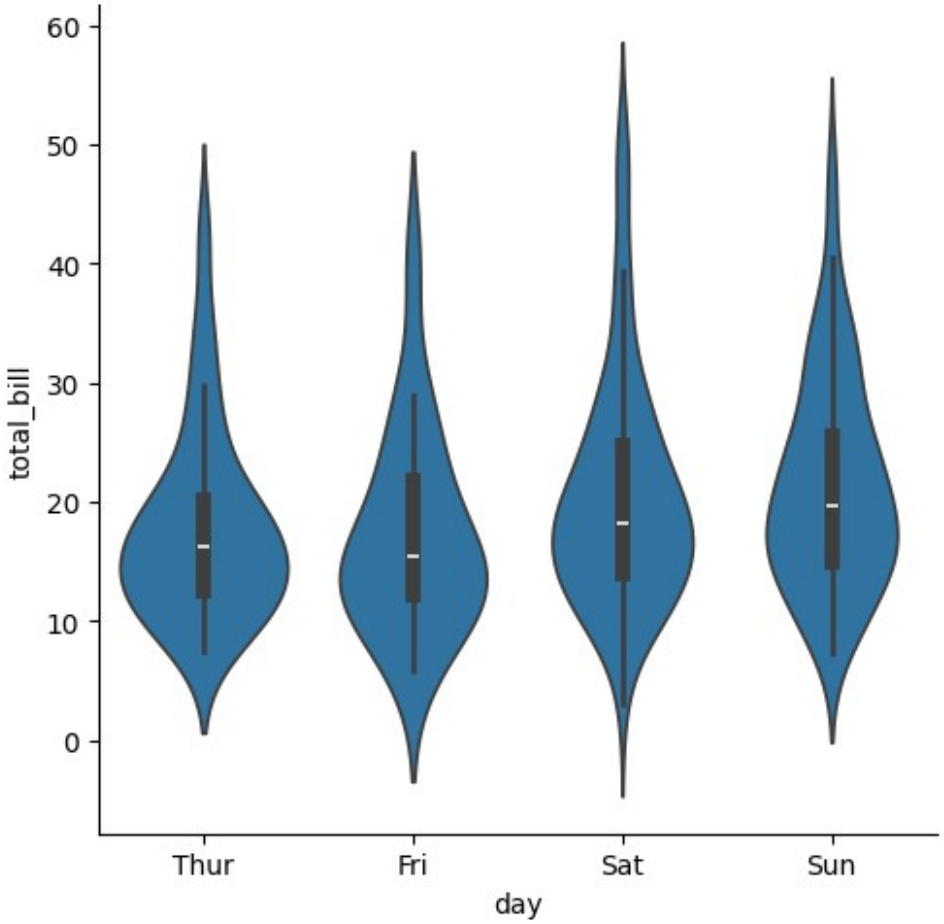
```
# violinplot
sns.violinplot(data=tips,x='day',y='total_bill')
```

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



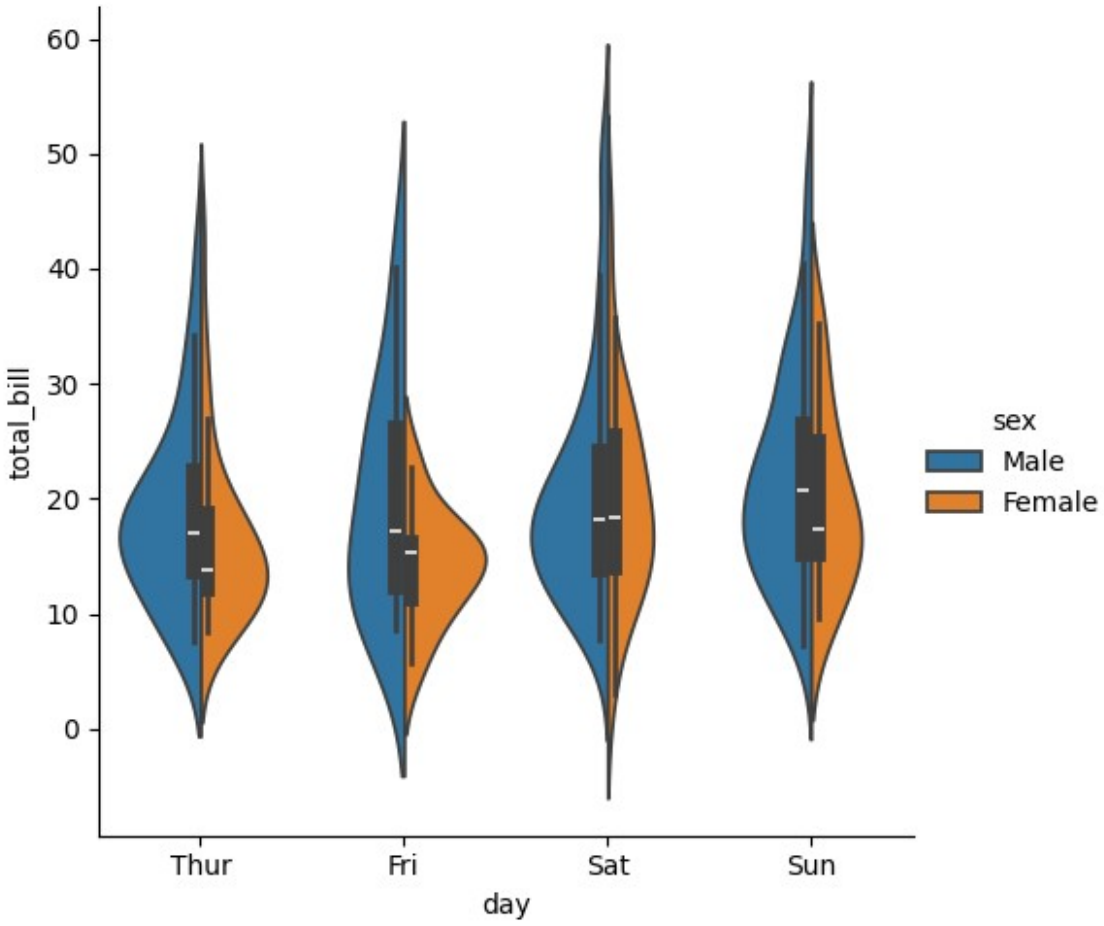

```
sns.catplot(data=tips,x='day',y='total_bill',kind='violin')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66944cafd0>
```



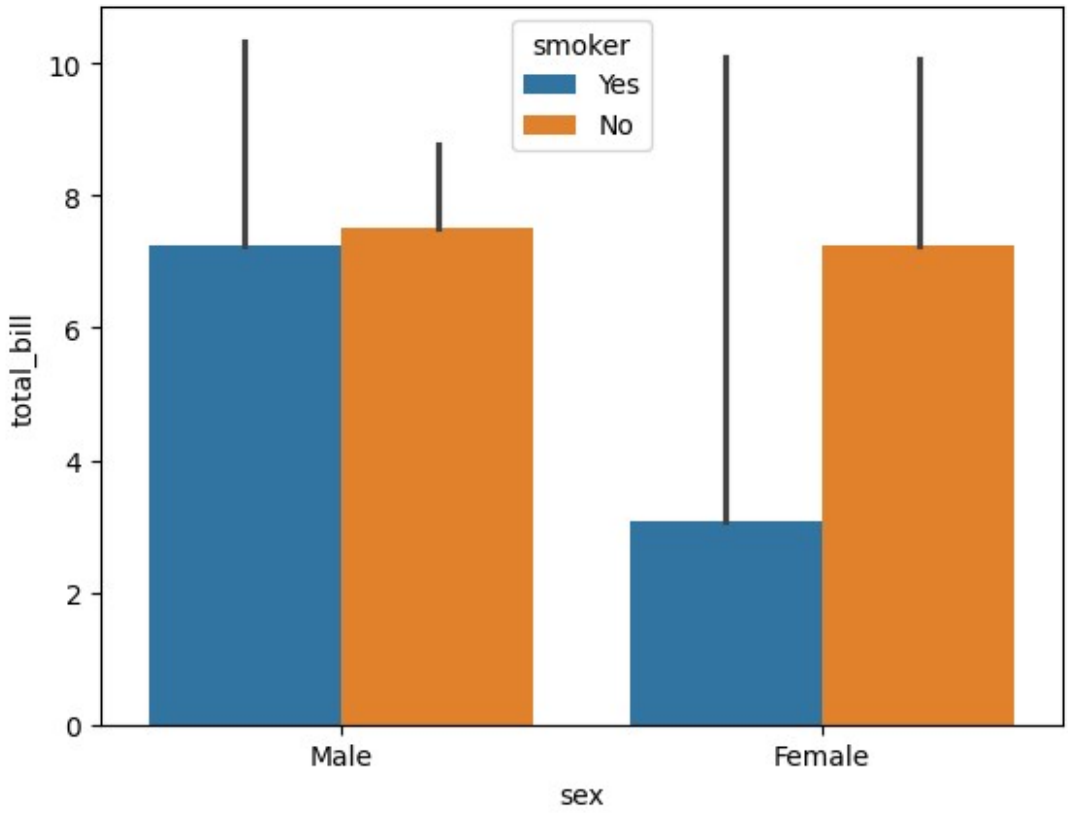

```
# hue

sns.catplot(data=tips,x='day',y='total_bill',kind='violin',hue='sex',s
plit=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66941609d0>
```



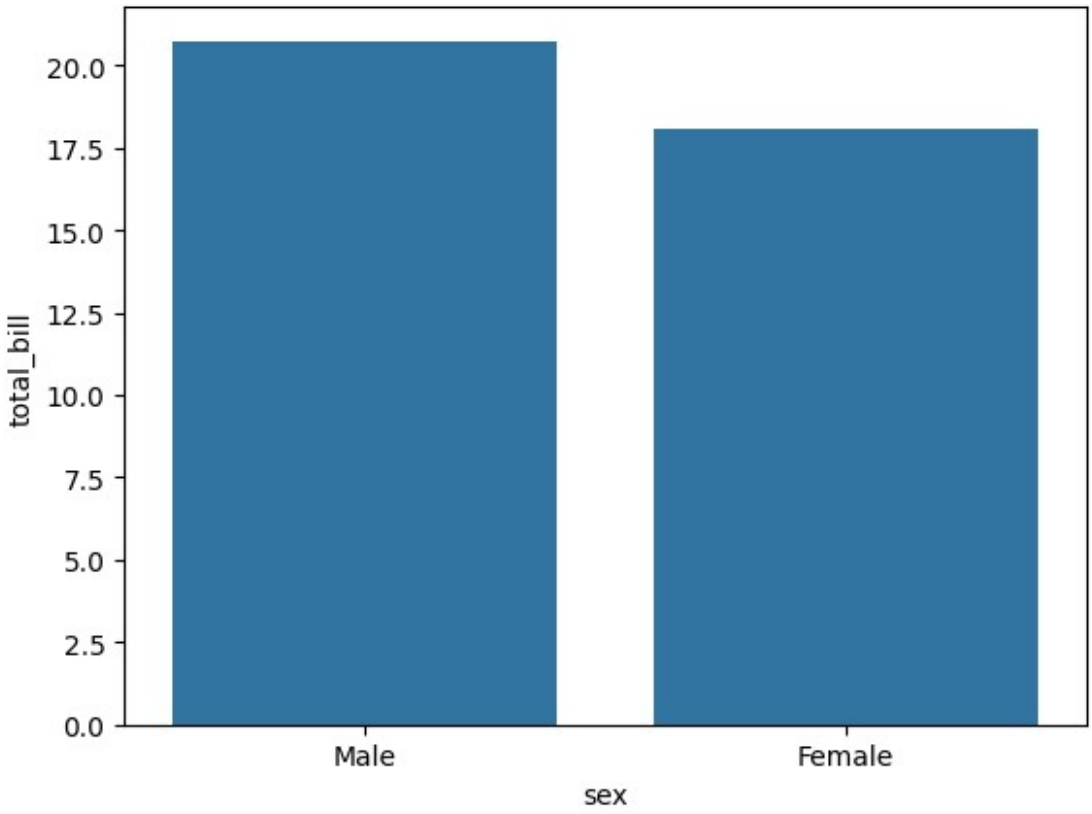

```
# barplot
# some issue with errorbar

sns.barplot(data=tips, x='sex',
y='total_bill',hue='smoker',estimator=np.min)
```

```
<Axes: xlabel='sex', ylabel='total_bill'>
```



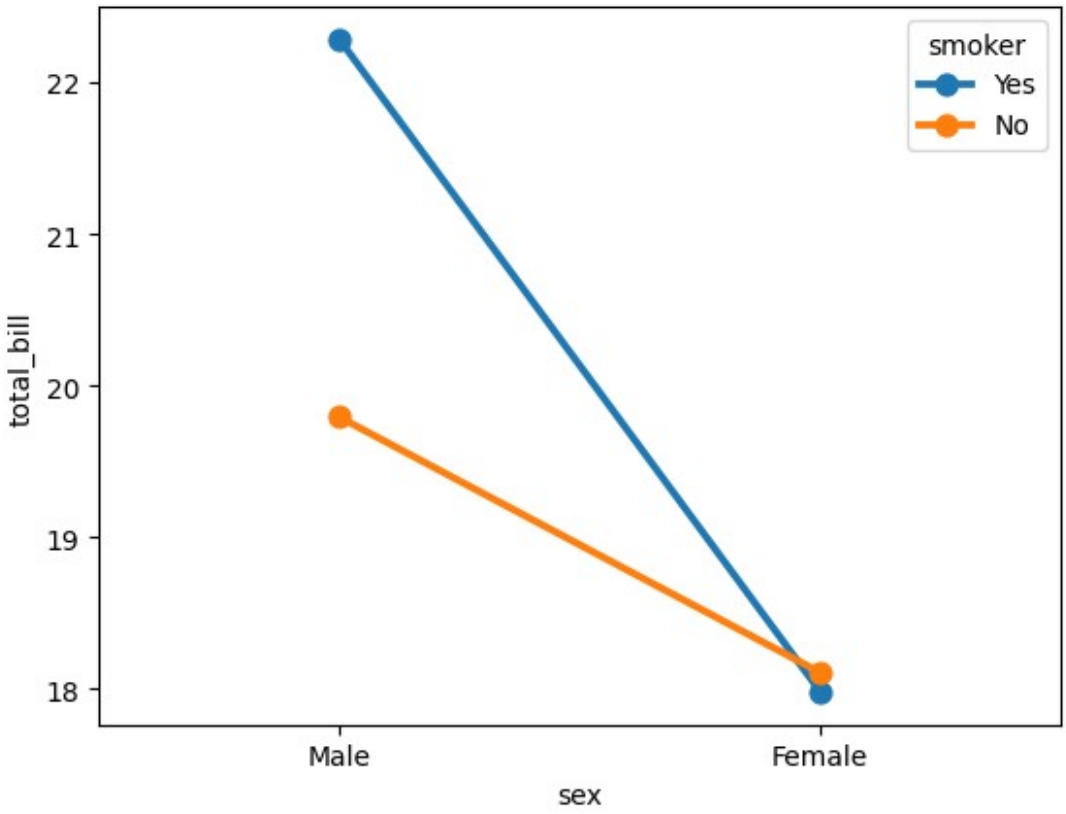

```
sns.barplot(data=tips, x='sex', y='total_bill',ci=None)
```

```
<ipython-input-87-535865fa5ddf>:1: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.


<Axes: xlabel='sex', ylabel='total_bill'>
```



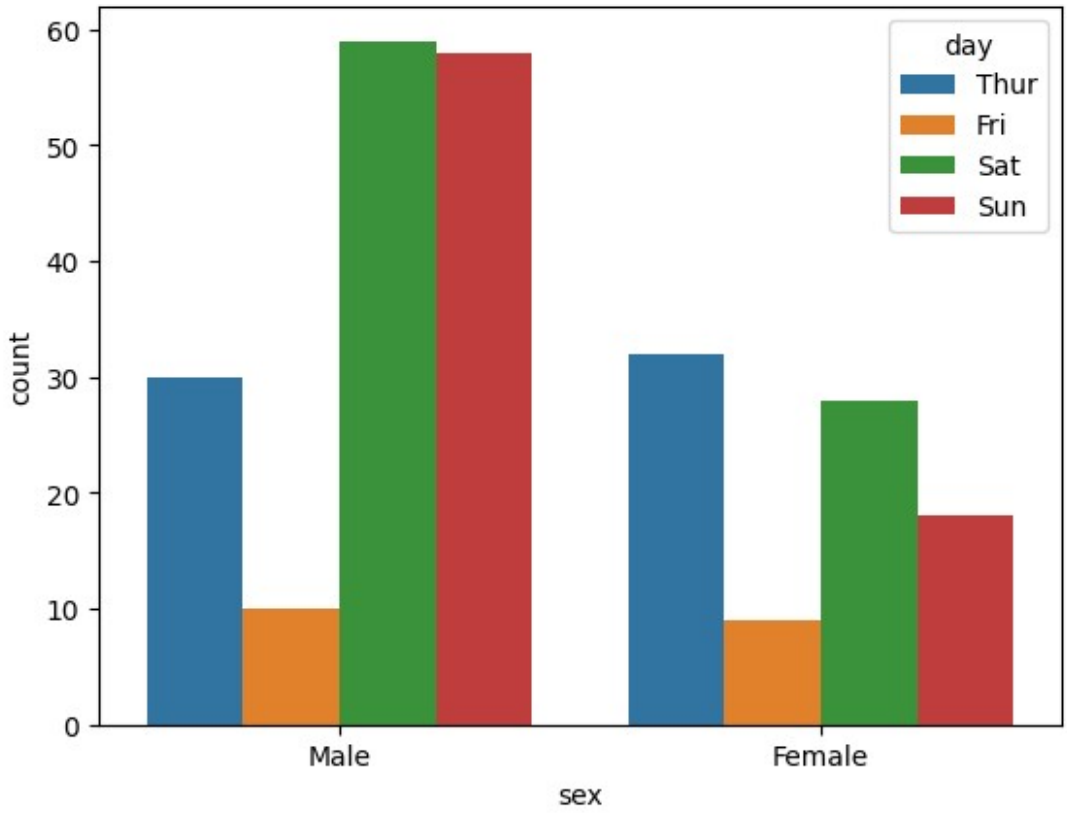

```
# point plot
sns.pointplot(data=tips, x='sex', y='total_bill',hue='smoker',ci=None)
```

```
<ipython-input-88-52b49bbff557>:2: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.


<Axes: xlabel='sex', ylabel='total_bill'>
```



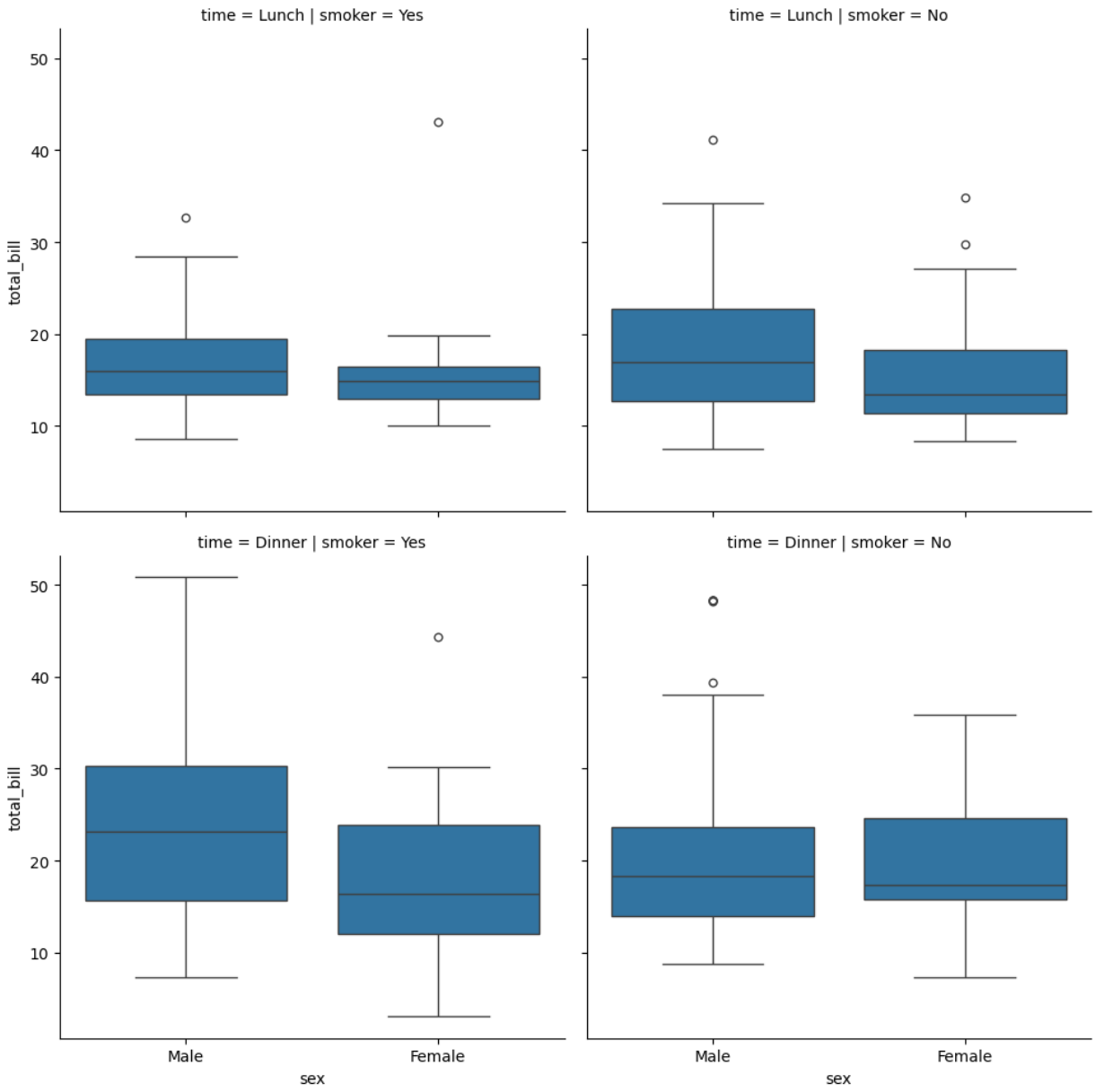

```
# countplot
sns.countplot(data=tips,x='sex',hue='day')
```

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




```
# pointplot

# faceting using catplot
sns.catplot(data=tips,
x='sex',y='total_bill',col='smoker',kind='box',row='time')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d669450d510>
```

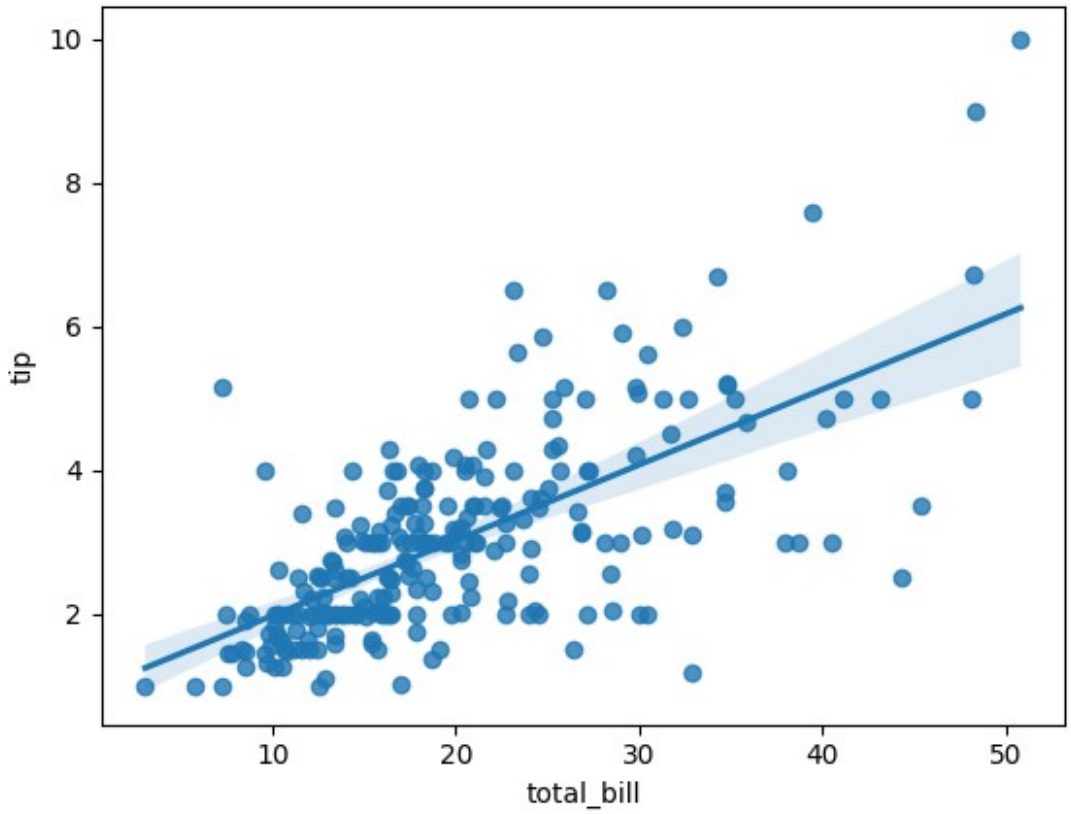


Regression Plots

regplot lmplot In the simplest invocation, both functions draw a scatterplot of two variables, x and y, and then fit the regression model y ~ x and plot the resulting regression line and a 95% confidence interval for that regression.

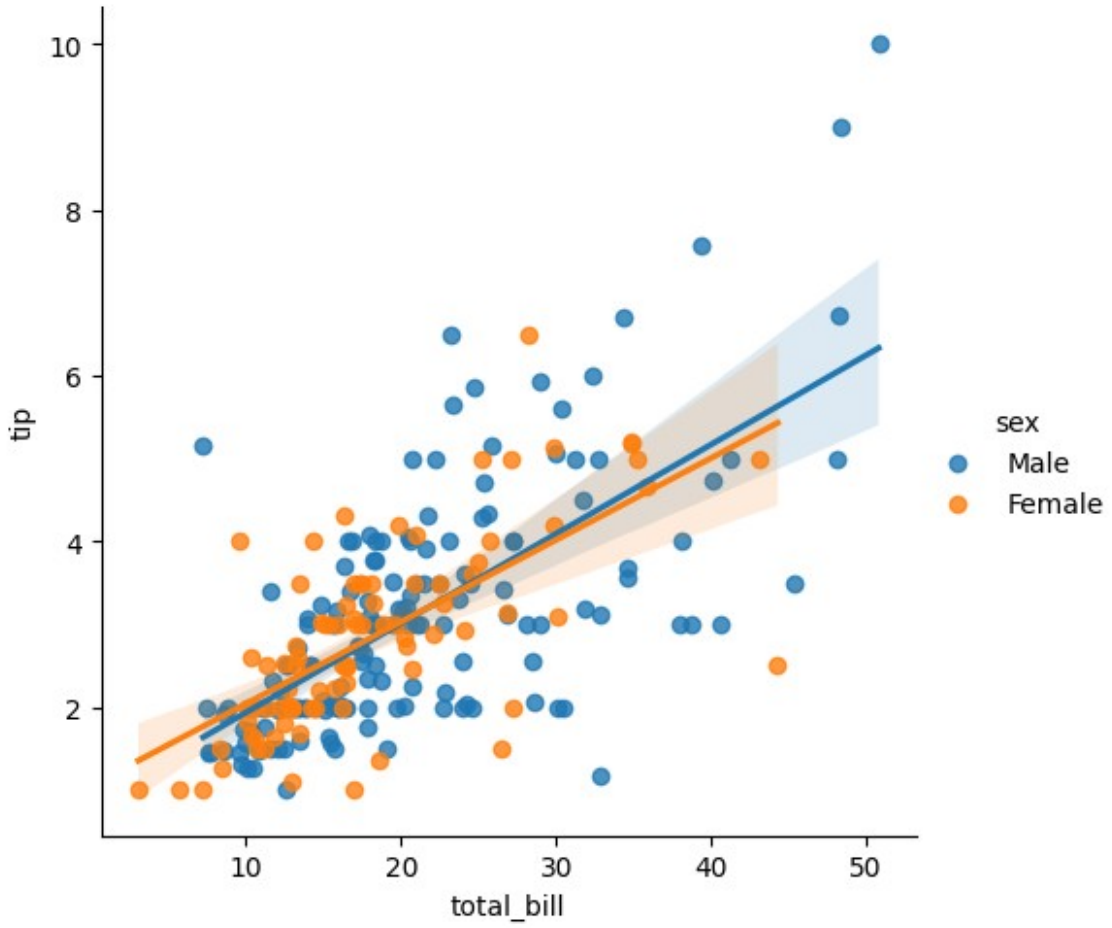
```
# axes level
# hue parameter is not available
sns.regplot(data=tips,x='total_bill',y='tip')
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



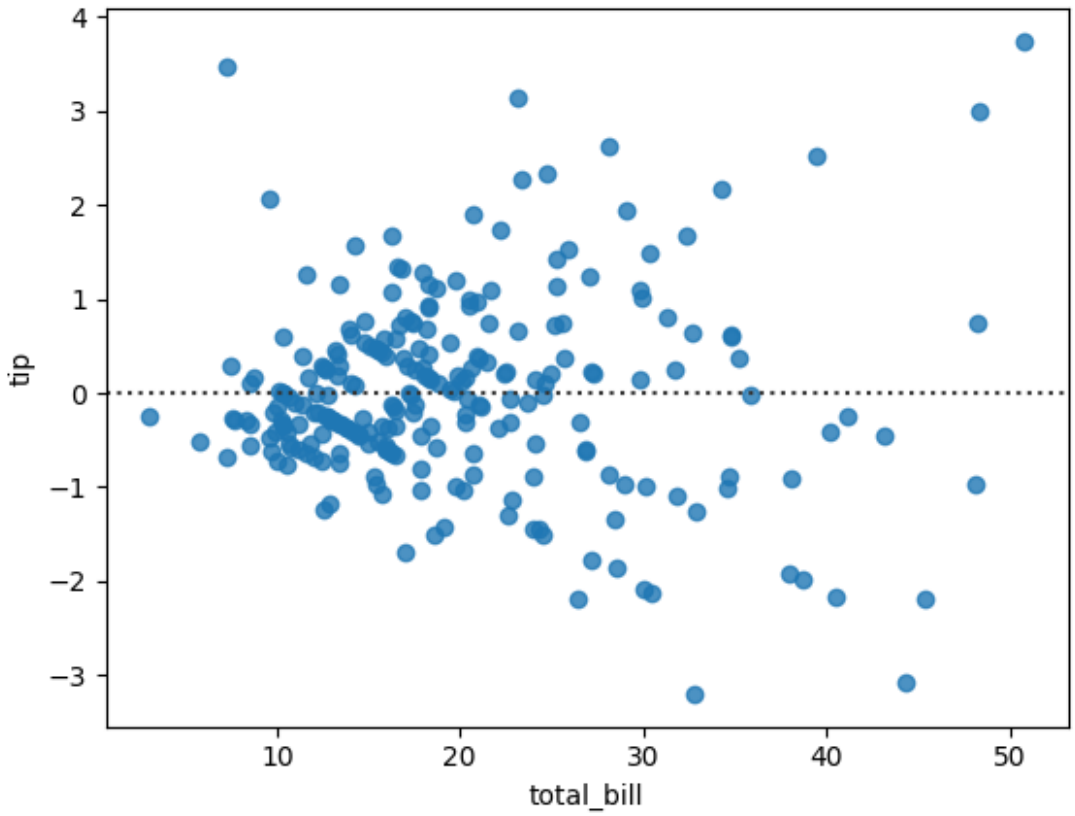

```
sns.lmplot(data=tips,x='total_bill',y='tip',hue='sex')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d66942757d0>
```




```
# residplot
sns.residplot(data=tips,x='total_bill',y='tip')
```

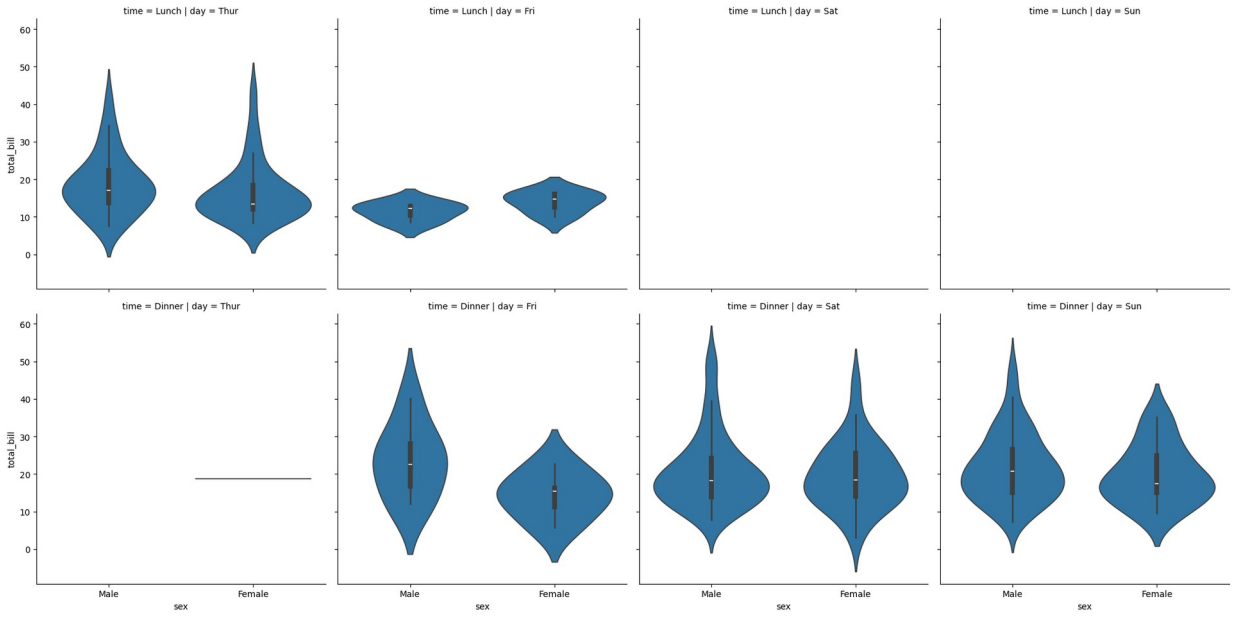
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



A second way to plot Facet plots -> FacetGrid

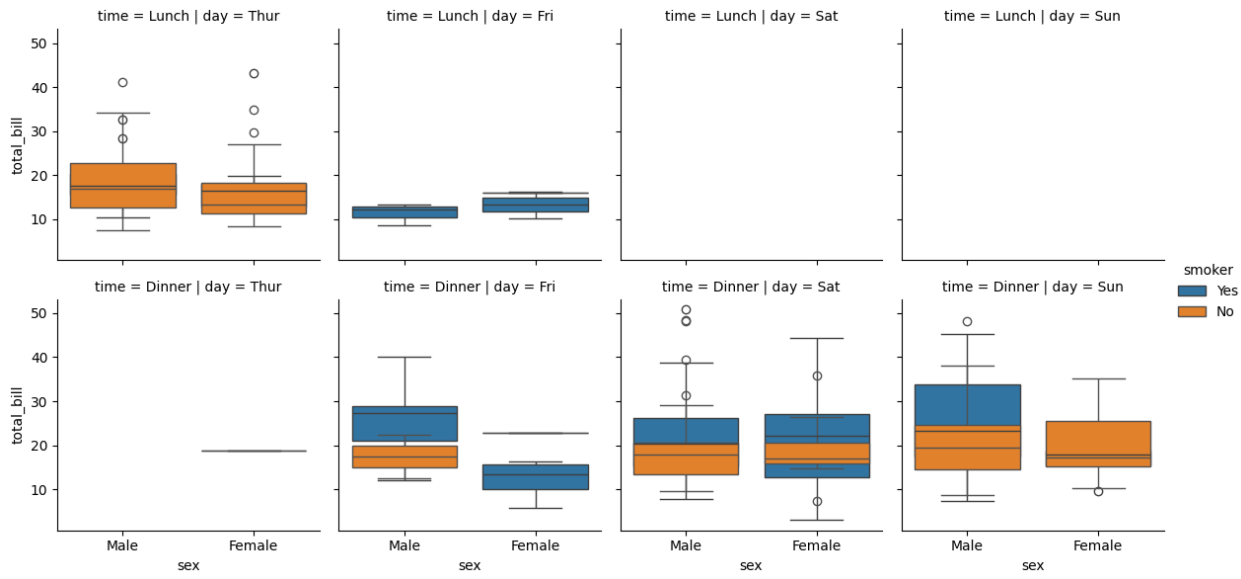
```
# figure level -> relplot -> displot -> catplot -> lmplot
sns.catplot(data=tips,x='sex',y='total_bill',kind='violin',col='day',r
ow='time')
```

```
<seaborn.axisgrid.FacetGrid at 0x7d668d33c790>
```




```
g = sns.FacetGrid(data=tips,col='day',row='time',hue='smoker')
g.map(sns.boxplot,'sex','total_bill')
g.add_legend()
```

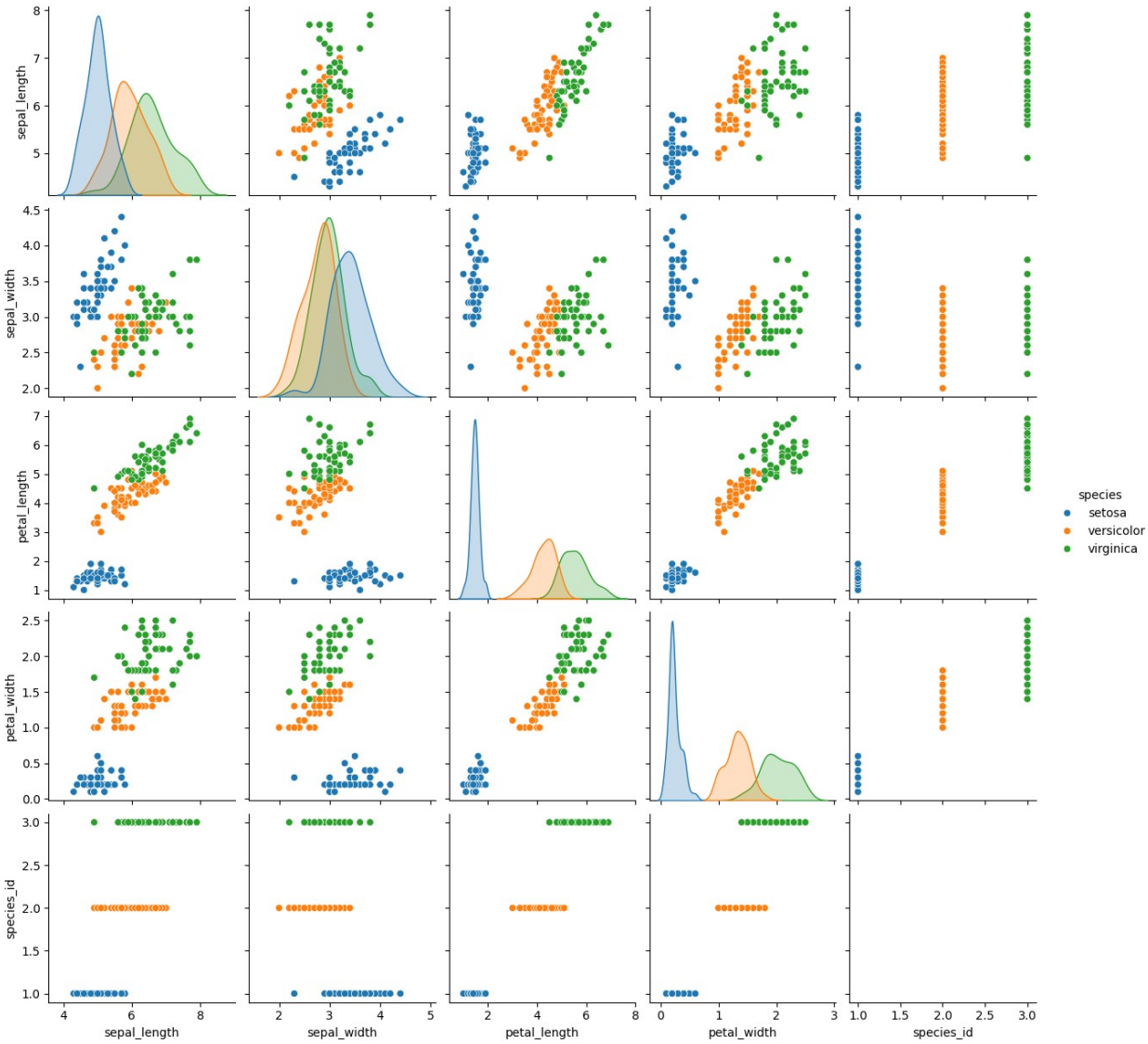
```
/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:718:
UserWarning:

Using the boxplot function without specifying `order` is likely to
produce an incorrect plot.


<seaborn.axisgrid.FacetGrid at 0x7d668cf63590>
```



lotting Pairwise Relationship (PairGrid Vs Pairplot)

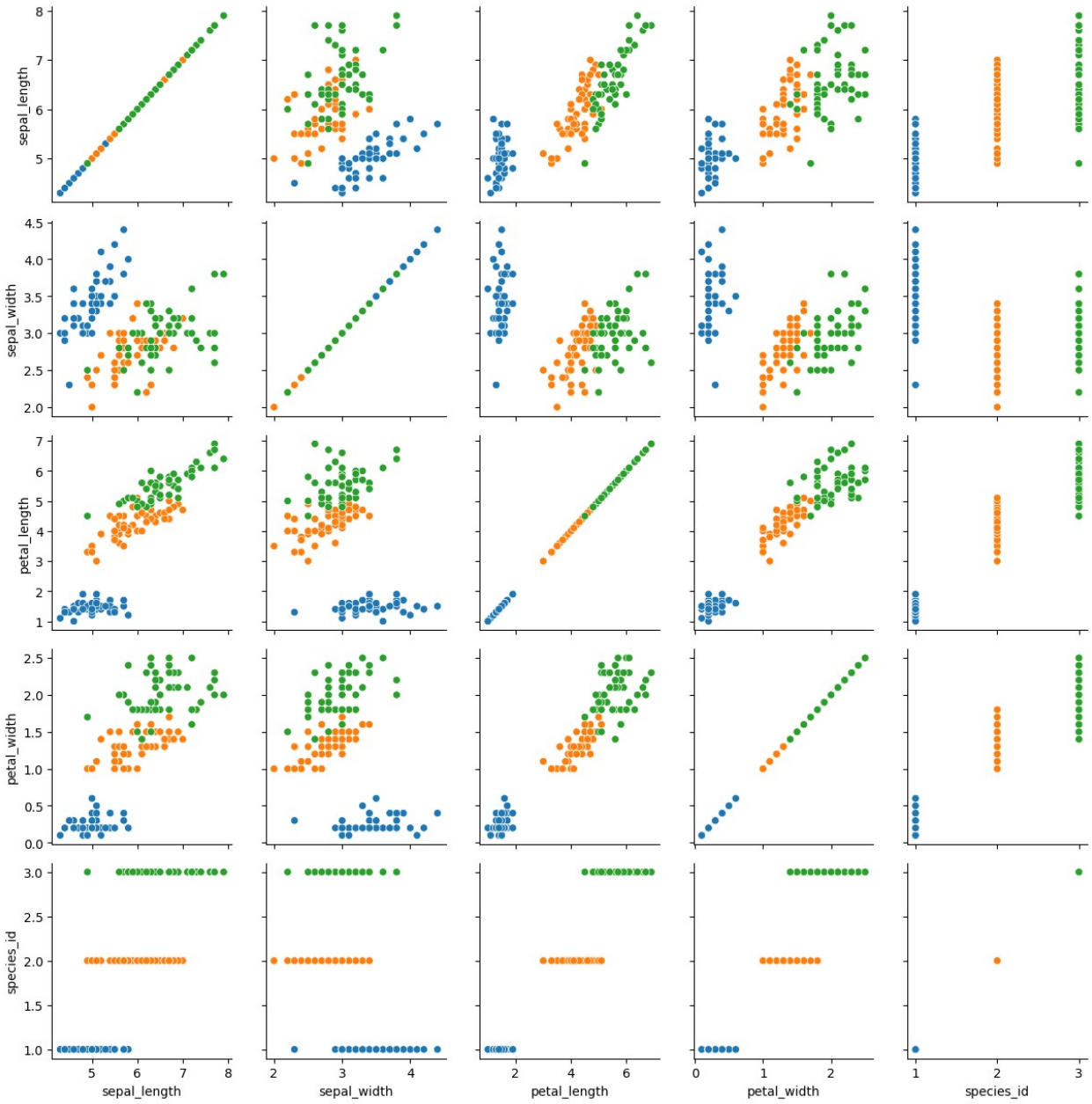
```
sns.pairplot(iris,hue='species')
```

```
<seaborn.axisgrid.PairGrid at 0x7d668d034990>
```




```
# pair grid
g = sns.PairGrid(data=iris,hue='species')
# g.map
g.map(sns.scatterplot)
```

```
<seaborn.axisgrid.PairGrid at 0x7d668cee17d0>
```



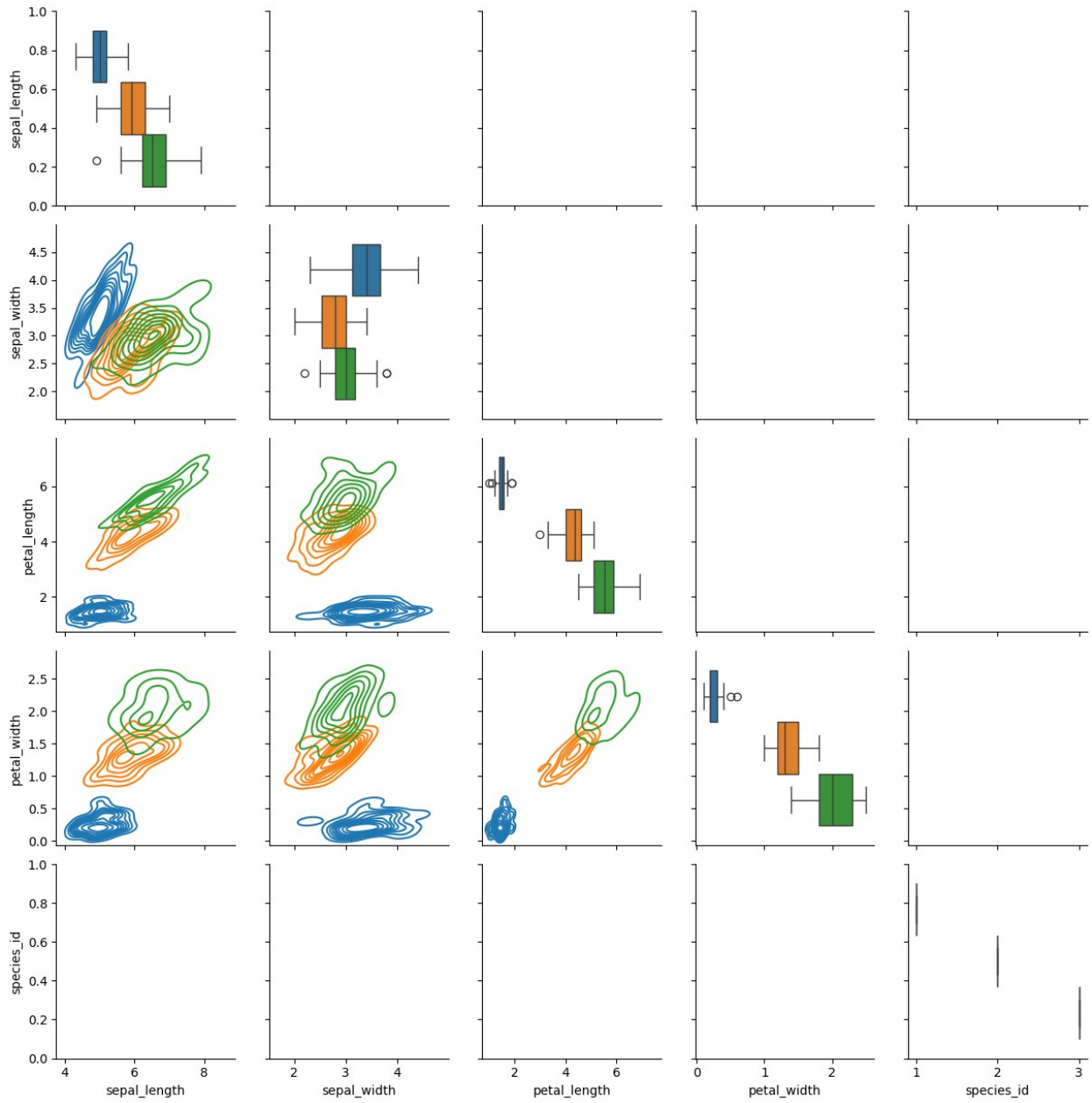

```
# map_diag -> map_offdiag
g = sns.PairGrid(data=iris,hue='species')
g.map_diag(sns.boxplot)
g.map_offdiag(sns.kdeplot)
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:1615:
```

```
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.


---------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
<ipython-input-98-6ea76c763b96> in <cell line: 0>()
      2 g = sns.PairGrid(data=iris,hue='species')
      3 g.map_diag(sns.boxplot)
----> 4 g.map_offdiag(sns.kdeplot)

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
map_offdiag(self, func, **kwargs)
   1423         """
   1424         if self.square_grid:
-> 1425             self.map_lower(func, **kwargs)
   1426             if not self._corner:
   1427                 self.map_upper(func, **kwargs)

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
map_lower(self, func, **kwargs)
   1393         """
   1394         indices = zip(*np.tril_indices_from(self.axes, -1))
-> 1395         self._map_bivariate(func, indices, **kwargs)
   1396         return self
   1397

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
_map_bivariate(self, func, indices, **kwargs)
   1572             if ax is None:  # i.e. we are in corner mode
   1573                 continue
-> 1574             self._plot_bivariate(x_var, y_var, ax, func,
**kws)
   1575         self._add_axis_labels()
   1576

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
_plot_bivariate(self, x_var, y_var, ax, func, **kwargs)
   1613                 "hue": hue, "hue_order": self._hue_order,
"palette": self._orig_palette,
```

```
   1614             })
-> 1615         func(x=x, y=y, **kwargs)
   1616
   1617         self._update_legend_data(ax)

/usr/local/lib/python3.11/dist-packages/seaborn/distributions.py in
kdeplot(data, x, y, hue, weights, palette, hue_order, hue_norm, color,
fill, multiple, common_norm, common_grid, cumulative, bw_method,
bw_adjust, warn_singular, log_scale, levels, thresh, gridsize, cut,
clip, legend, cbar, cbar_ax, cbar_kws, ax, **kwargs)
   1713     else:
   1714
-> 1715         p.plot_bivariate_density(
   1716             common_norm=common_norm,
   1717             fill=fill,

/usr/local/lib/python3.11/dist-packages/seaborn/distributions.py in
plot_bivariate_density(self, common_norm, fill, levels, thresh, color,
legend, cbar, warn_singular, cbar_ax, cbar_kws, estimate_kws,
**contour_kws)
   1111         # Transform from iso-proportions to iso-densities
   1112         if common_norm:
-> 1113             common_levels = self._quantile_to_level(
   1114                 list(densities.values()), levels,
   1115             )

/usr/local/lib/python3.11/dist-packages/seaborn/distributions.py in
_quantile_to_level(self, data, quantile)
    198         normalized_values = np.cumsum(sorted_values) /
values.sum()
    199         idx = np.searchsorted(normalized_values, 1 - isoprop)
--> 200         levels = np.take(sorted_values, idx, mode="clip")
    201         return levels
    202

/usr/local/lib/python3.11/dist-packages/numpy/_core/fromnumeric.py in
take(a, indices, axis, out, mode)
    204            [5, 7]])
    205     """
--> 206     return _wrapfunc(a, 'take', indices, axis=axis, out=out,
mode=mode)
    207
    208

/usr/local/lib/python3.11/dist-packages/numpy/_core/fromnumeric.py in
_wrapfunc(obj, method, *args, **kwds)
     55
     56     try:
---> 57         return bound(*args, **kwds)
     58     except TypeError:
```

```
     59         # A TypeError occurs if the object does have such a
method in its

IndexError: cannot do a non-empty take from an empty axes.
```

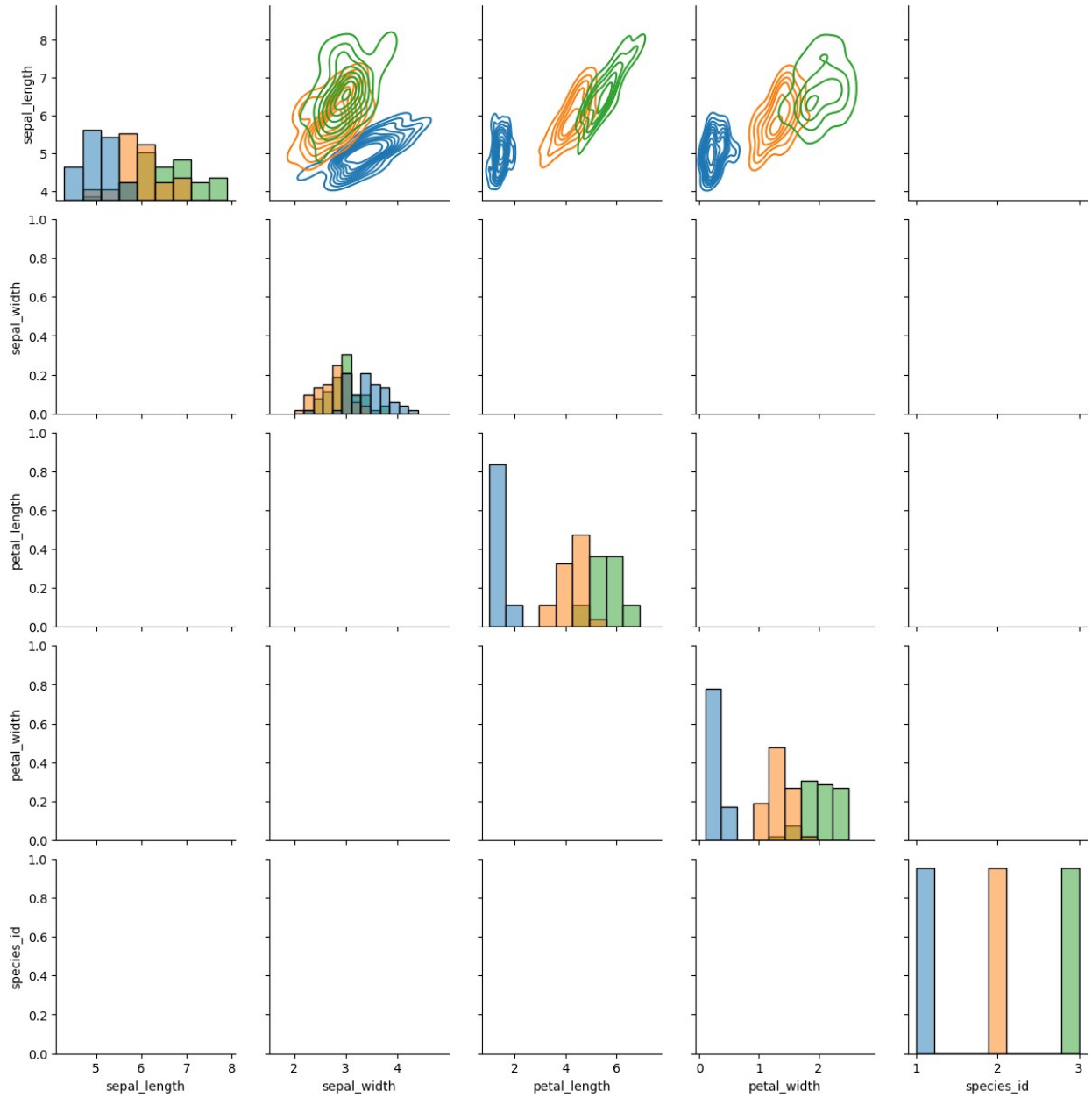



```
# map_diag -> map_upper -> map_lower
g = sns.PairGrid(data=iris,hue='species')
g.map_diag(sns.histplot)
g.map_upper(sns.kdeplot)
g.map_lower(sns.scatterplot)
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.


---------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
<ipython-input-99-5d3137e3042a> in <cell line: 0>()
      2 g = sns.PairGrid(data=iris,hue='species')
      3 g.map_diag(sns.histplot)
----> 4 g.map_upper(sns.kdeplot)
      5 g.map_lower(sns.scatterplot)

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
map_upper(self, func, **kwargs)
   1408         """
   1409         indices = zip(*np.triu_indices_from(self.axes, 1))
-> 1410         self._map_bivariate(func, indices, **kwargs)
   1411         return self
   1412

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
_map_bivariate(self, func, indices, **kwargs)
   1572             if ax is None:  # i.e. we are in corner mode
   1573                 continue
-> 1574             self._plot_bivariate(x_var, y_var, ax, func,
**kws)
   1575         self._add_axis_labels()
   1576

/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py in
_plot_bivariate(self, x_var, y_var, ax, func, **kwargs)
   1613                 "hue": hue, "hue_order": self._hue_order,
"palette": self._orig_palette,
```

```
   1614             })
-> 1615         func(x=x, y=y, **kwargs)
   1616
   1617         self._update_legend_data(ax)

/usr/local/lib/python3.11/dist-packages/seaborn/distributions.py in
kdeplot(data, x, y, hue, weights, palette, hue_order, hue_norm, color,
fill, multiple, common_norm, common_grid, cumulative, bw_method,
bw_adjust, warn_singular, log_scale, levels, thresh, gridsize, cut,
clip, legend, cbar, cbar_ax, cbar_kws, ax, **kwargs)
   1713     else:
   1714
-> 1715         p.plot_bivariate_density(
   1716             common_norm=common_norm,
   1717             fill=fill,

/usr/local/lib/python3.11/dist-packages/seaborn/distributions.py in
plot_bivariate_density(self, common_norm, fill, levels, thresh, color,
legend, cbar, warn_singular, cbar_ax, cbar_kws, estimate_kws,
**contour_kws)
   1111         # Transform from iso-proportions to iso-densities
   1112         if common_norm:
-> 1113             common_levels = self._quantile_to_level(
   1114                 list(densities.values()), levels,
   1115             )

/usr/local/lib/python3.11/dist-packages/seaborn/distributions.py in
_quantile_to_level(self, data, quantile)
    198         normalized_values = np.cumsum(sorted_values) /
values.sum()
    199         idx = np.searchsorted(normalized_values, 1 - isoprop)
--> 200         levels = np.take(sorted_values, idx, mode="clip")
    201         return levels
    202

/usr/local/lib/python3.11/dist-packages/numpy/_core/fromnumeric.py in
take(a, indices, axis, out, mode)
    204            [5, 7]])
    205     """
--> 206     return _wrapfunc(a, 'take', indices, axis=axis, out=out,
mode=mode)
    207
    208

/usr/local/lib/python3.11/dist-packages/numpy/_core/fromnumeric.py in
_wrapfunc(obj, method, *args, **kwds)
     55
     56     try:
---> 57         return bound(*args, **kwds)
     58     except TypeError:
```

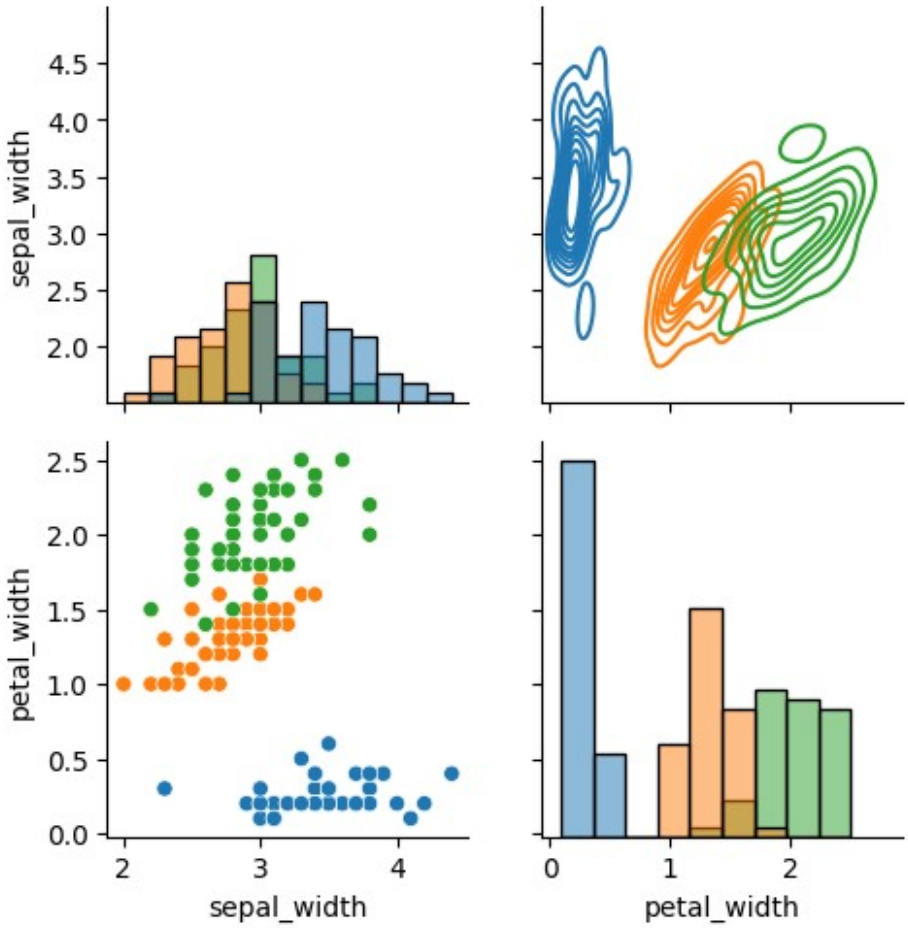
```
     59         # A TypeError occurs if the object does have such a
method in its

IndexError: cannot do a non-empty take from an empty axes.
```




```
# vars
g =
sns.PairGrid(data=iris,hue='species',vars=['sepal_width','petal_width'
])
g.map_diag(sns.histplot)
```

```
g.map_upper(sns.kdeplot)
g.map_lower(sns.scatterplot)
```

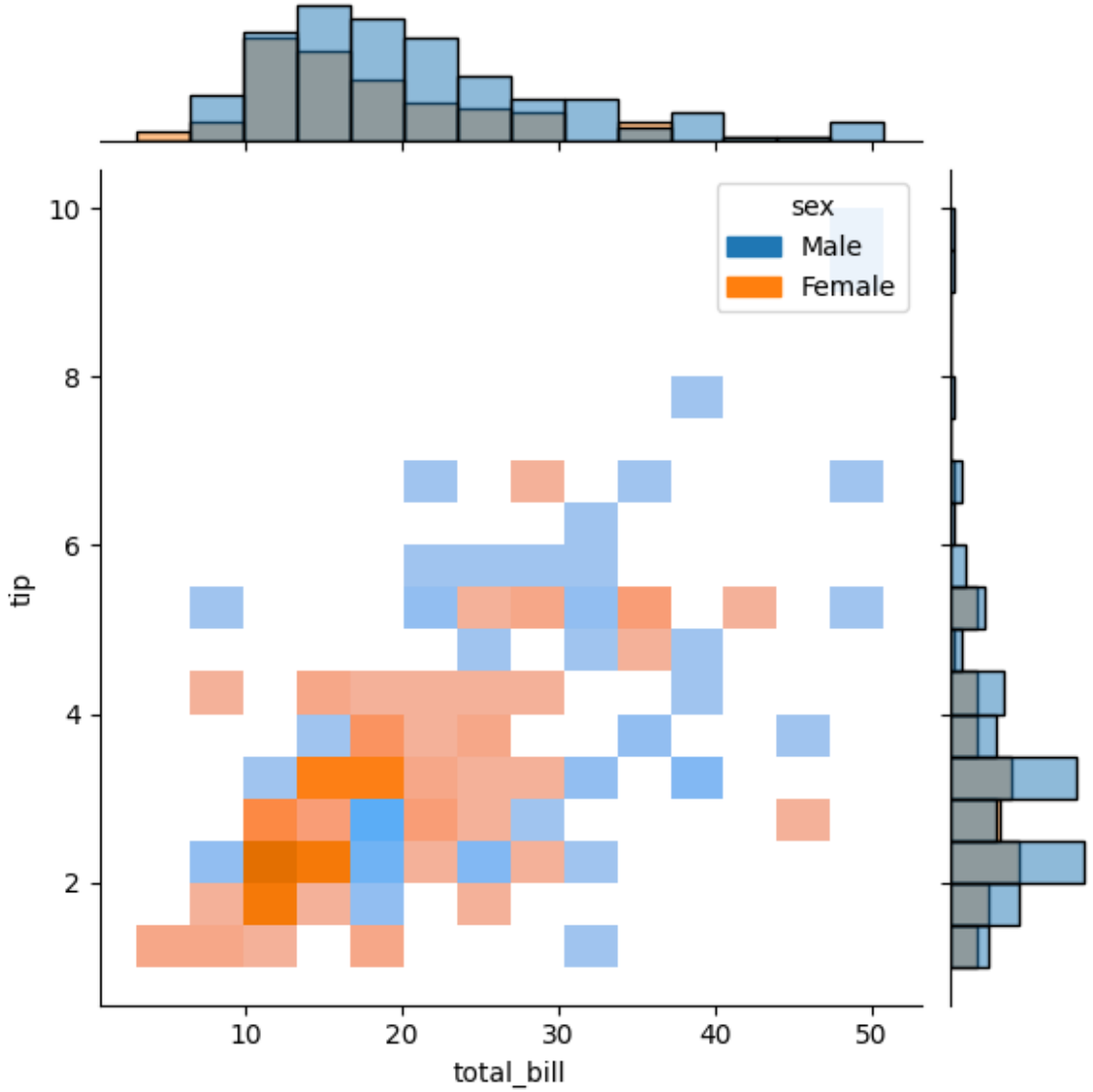
```
<seaborn.axisgrid.PairGrid at 0x7d668a8ce290>
```



JointGrid Vs Jointplot

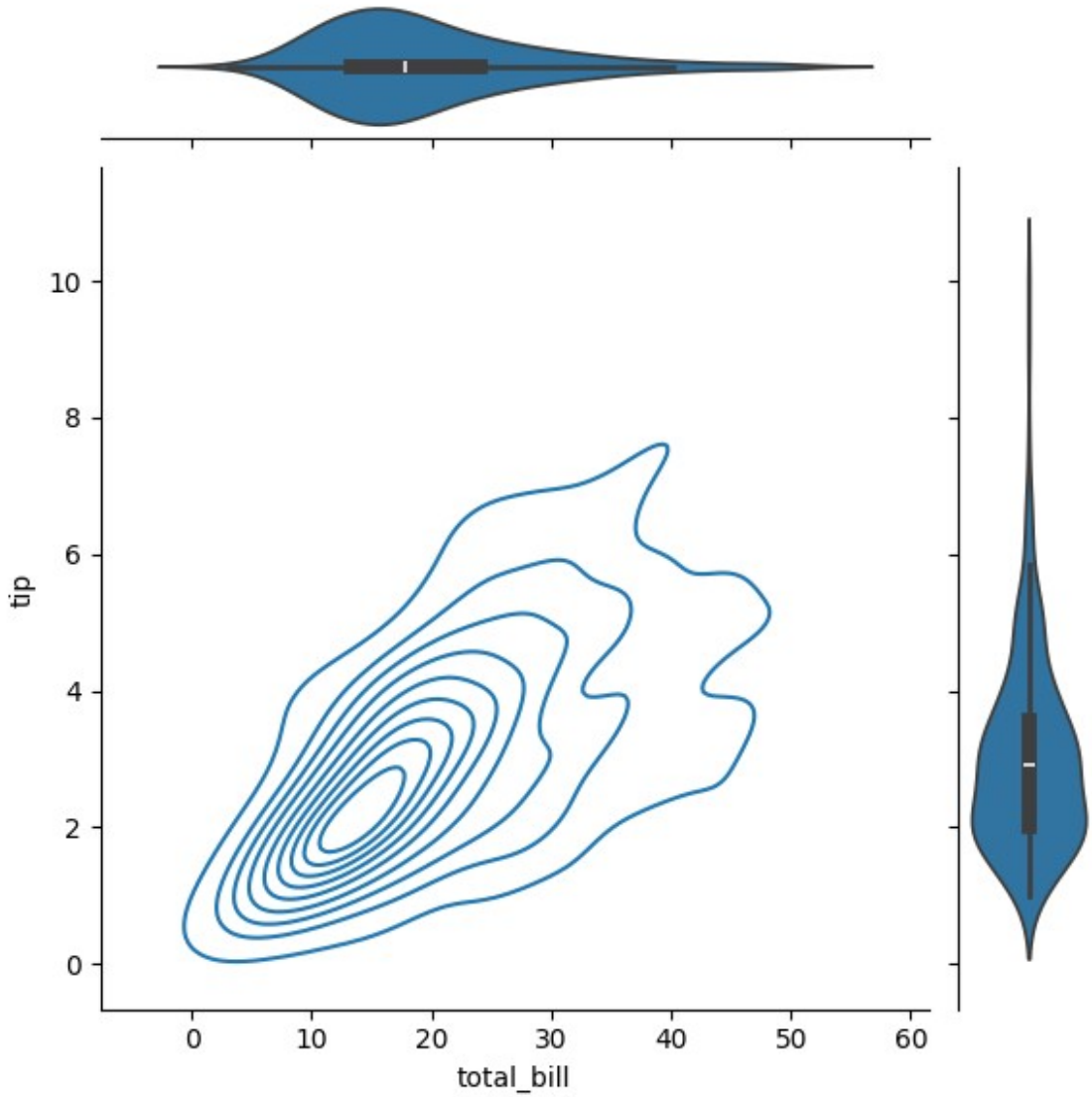
```
sns.jointplot(data=tips,x='total_bill',y='tip',kind='hist',hue='sex')
```

```
<seaborn.axisgrid.JointGrid at 0x7d668a6d4d90>
```




```
g = sns.JointGrid(data=tips,x='total_bill',y='tip')
g.plot(sns.kdeplot,sns.violinplot)
```

```
<seaborn.axisgrid.JointGrid at 0x7d668ad350d0>
```


10
8
6
4
2
0
tip
0
10
20
30
40
50
60
total_bill