Seaborn: Statistical Data Visualization

Seaborn is a Python library that simplifies creating statistical graphs. It works directly with DataFrames and arrays, automating statistical aggregation for informative visualizations.

Feature	Seaborn	Matplotlib	Plotly
Focus	Emphasizes statistical data visualization	General purpose plotting library	Geared toward interactive, web-based graphs
Ease of Use	Simplifies the creation of complex visualizations	Very customizable but technically demanding	User-friendly with a focus on interactivity and web integration
Design	Provides visually appealing plots	Offers basic styles, requiring customization for enhanced visuals	Delivers advanced, interactive visuals that are out-of-the-box

Install seaborn along with its required dependencies.

```
pip install seaborn
```

Begin your Python script by importing seaborn.

```
import seaborn as sns
```

Creating a plot with seaborn is straightforward.

```
sns.plotter_function(x="x_column", y="y_column", data=dataframe_name)
```

Seaborn provides an intuitive interface for creating statistical visuals with pandas DataFrames. It offers plot types like relational, categorical, and distribution for varied data analysis. Mastering seaborn involves choosing plot types, customizing visuals, and integrating statistical models.

Understanding Data for Visualization

Built-in Datasets

Seaborn provides several built-in datasets for practicing and demonstration, including popular ones like:

```
sns.load_dataset('dataset_name')
```

Dataset	Details
tips	Data of a restaurant’s total bill, tip amount, and other factors
iris	Iris flower dataset, including sepal and petal sizes, etc.
titanic	Passenger data from the Titanic like fare, age, etc.
flights	Historical flight passenger data by month and year
diamonds	Prices and attributes of approximately 54,000 diamonds
mpg	Fuel economy data for cars like horsepower, cylinder, etc.
penguins	Data of penguins from Palmer Station, Antarctica

Integration with NumPy and pandas

NumPy Arrays: Ensure your data is organized into 2D arrays for matrix-based plots and 1D array for vector-based plots.

```
import seaborn as sns
import numpy as np

data = np.random.rand(10, 12)
sns.plotting_function(data)
```

Pandas DataFrames: Utilize DataFrame's structure to label your data with meaningful column names.

```
import seaborn as sns
import pandas as pd

df = pd.DataFrame({
    'Category': ['A', 'B', 'A', 'B'],
    'Value': [10, 20, 30, 40]
})

sns.plotting_function(x='Category', y='Value', data=df)
```

Figure Aesthetics and Layout

Customizing Seaborn Styles

Seaborn offers a variety of predefined styles designed to be visually appealing and improve the readability of plots.

```
sns.set_style("style_name")
```

Style Name	Details
darkgrid	A dark background and white gridlines
whitegrid	A light background with white gridlines
dark	A dark background style without gridlines
white	A light background style without gridlines
ticks	A variation of the white style that adds ticks to the axes

Adjusting Plot Context

Seaborn's context settings are designed to optimize the readability of your plots depending on where they are being presented. The available presets include:

```
sns.set_context("preset_name")
```

Preset value	Details
paper	Ideal for visuals intended for printed media
notebook	The default context, optimized for Jupyter notebooks
talk	Scales up elements for presentations
poster	Maximizes readability by further scaling up plot elements

Working with Color Palettes

Seaborn has a set of predefined color palettes that can easily enhance the visual appeal of plots.

```
sns.set_palette("palette_name")    # Apply a built-in palette

# Create and apply a custom palette
custom_palette = sns.color_palette(["#9b59b6", "#3498db", "#95a5a6"])
sns.set_palette(custom_palette)
```

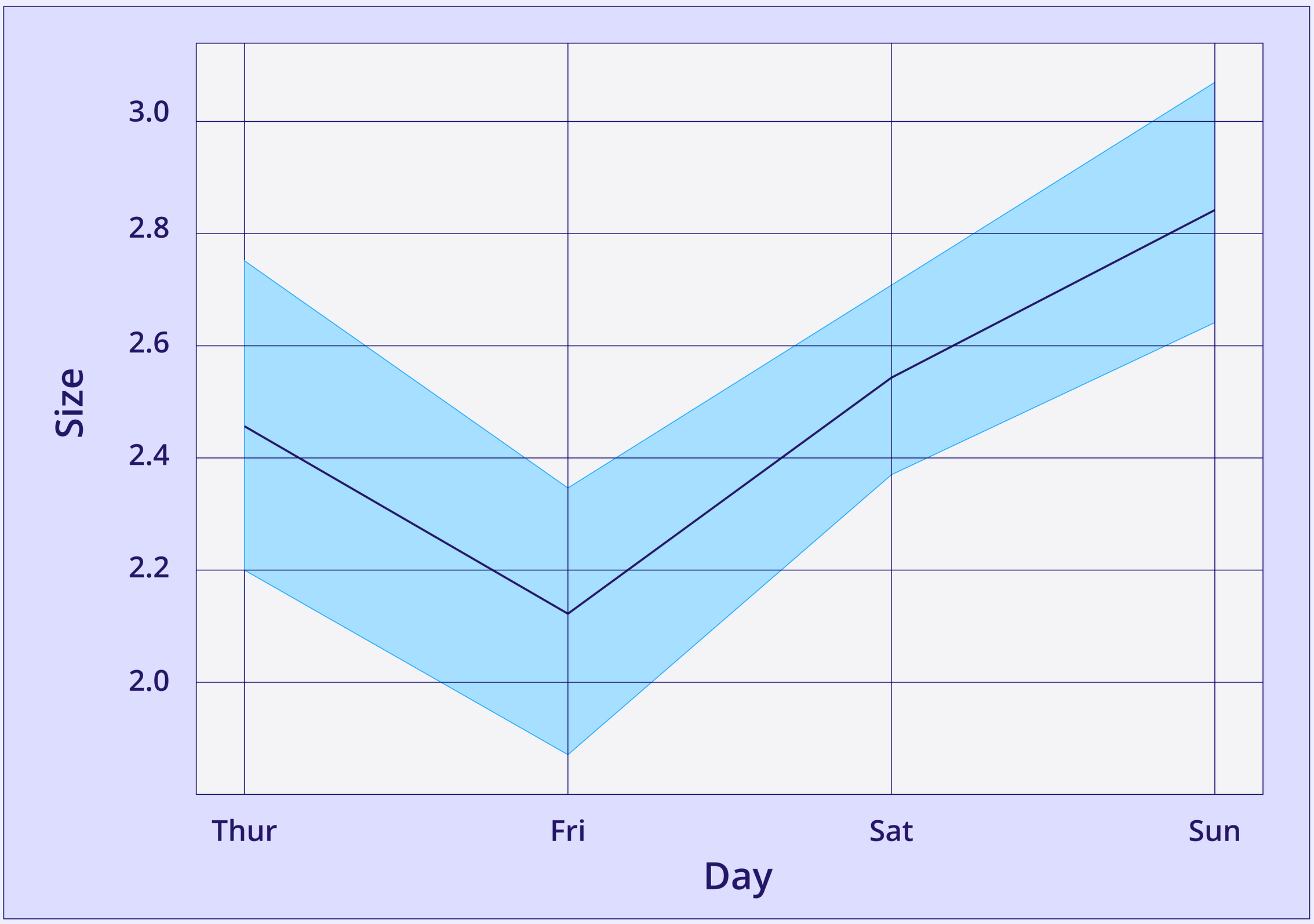
Theme	Details
Deep	Default theme: A mix of relatively strong colors
Muted	Similar to Deep, but less intense
Pastel	Lighter and softer variations of colors
Bright	Bright and vibrant versions of colors
Dark	A darker version is useful for plots with white backgrounds
Colorblind	Designed for those with colorblindness

Visualizing Numerical Data

Line Plot

It visualizes the change in a trend over time or another continuous variable.

```
sns.lineplot(x="day", y="size", data=tips)
```



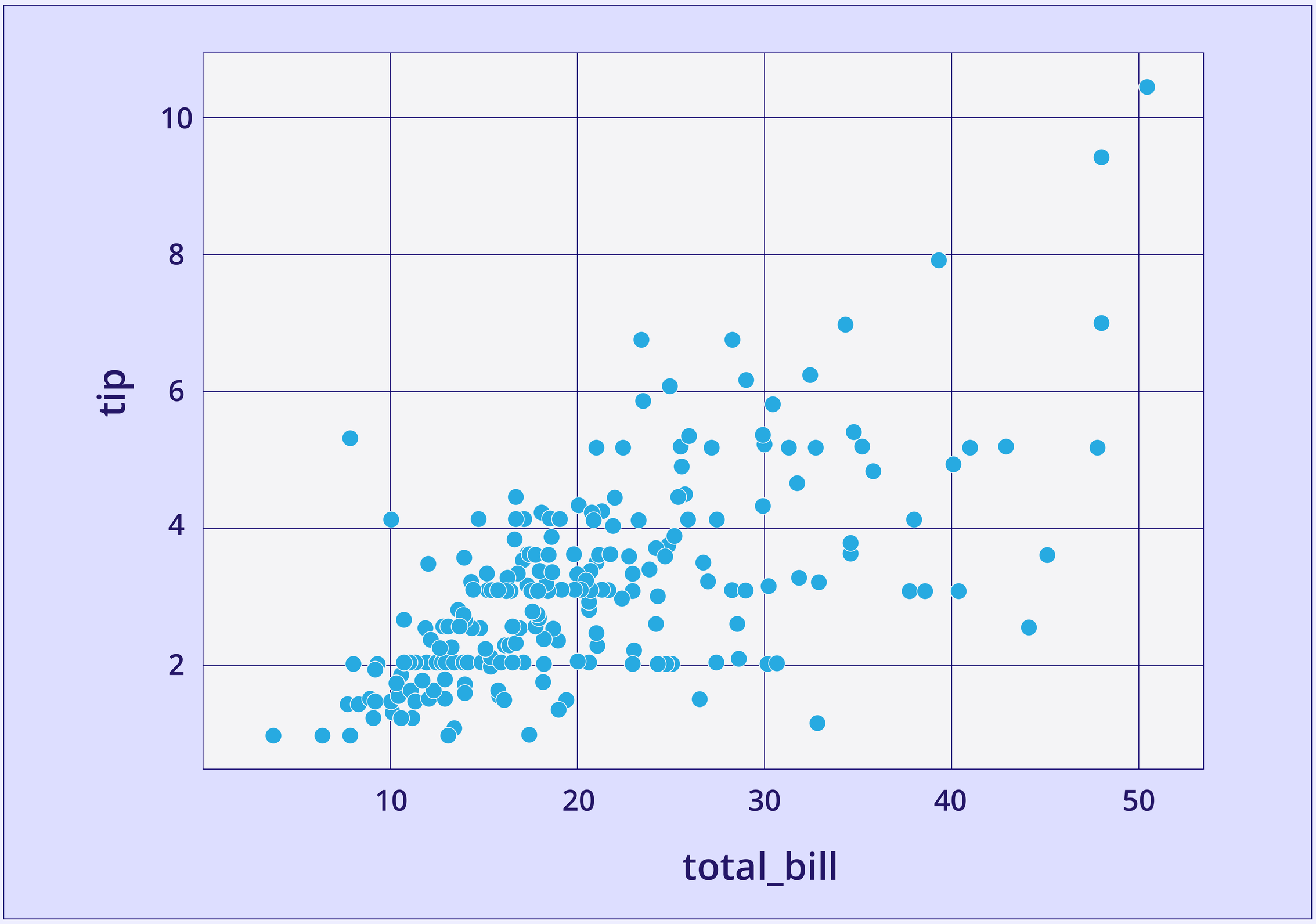
Seaborn: Statistical Data Visualization

educative

Scatter Plot

It illustrates relationships between two numerical variables.

```
sns.scatterplot(x="total_bill", y="tip", data=tips)
```

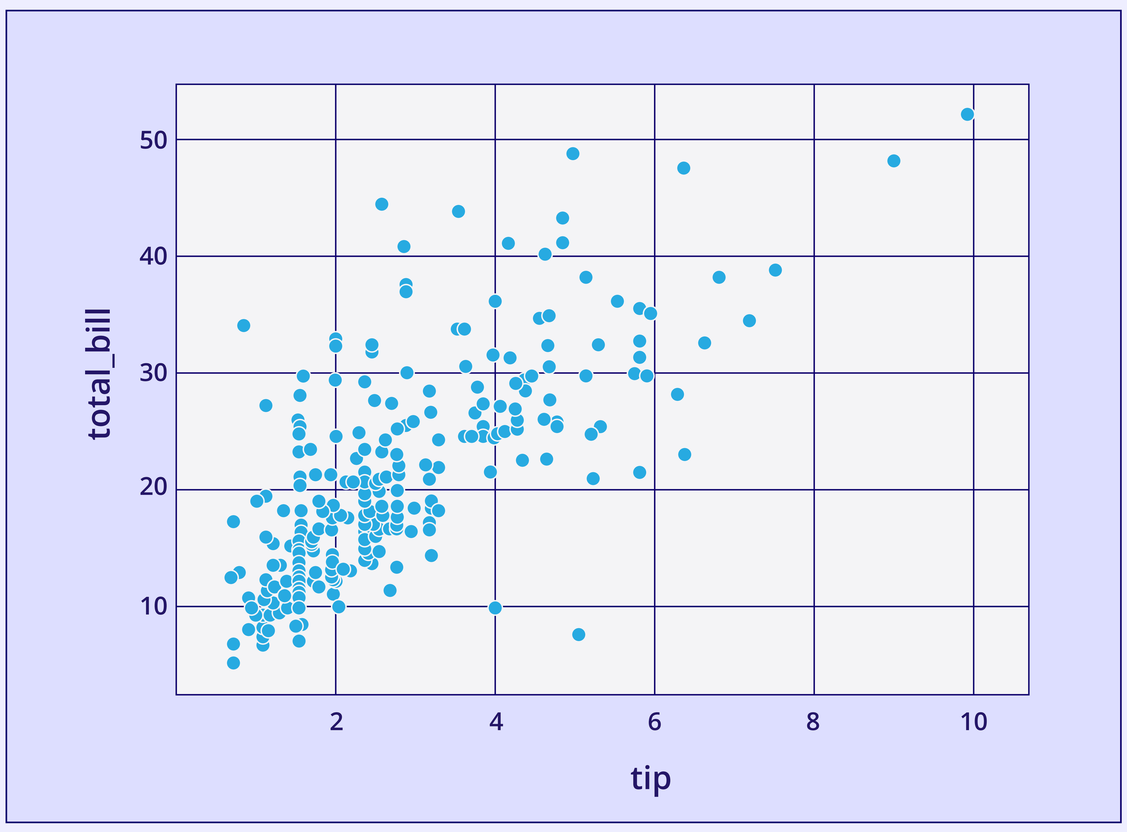


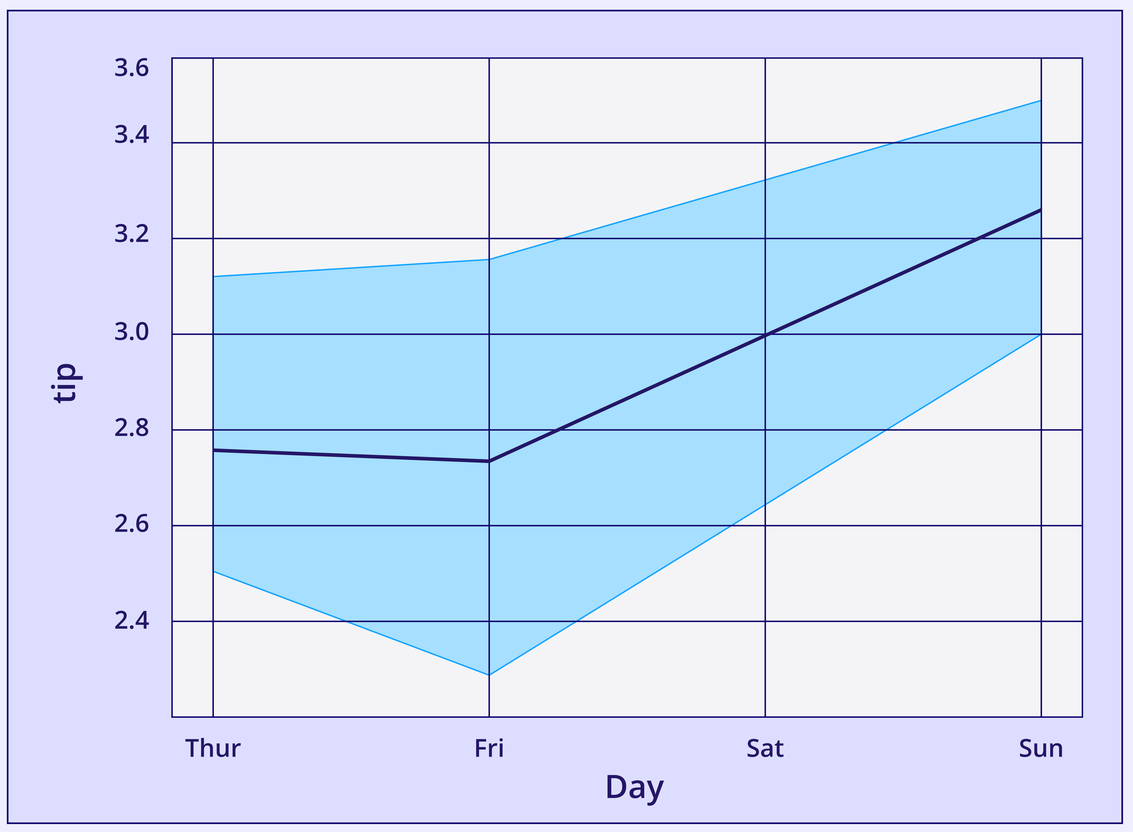
RelPlot

It visualizes relationships between two quantitative variables and can create scatter and line plots.

```
sns.relplot(x="day", y="tip", data=tips, kind="line")
```

The value of “kind” can either be “scatter” or “line”.

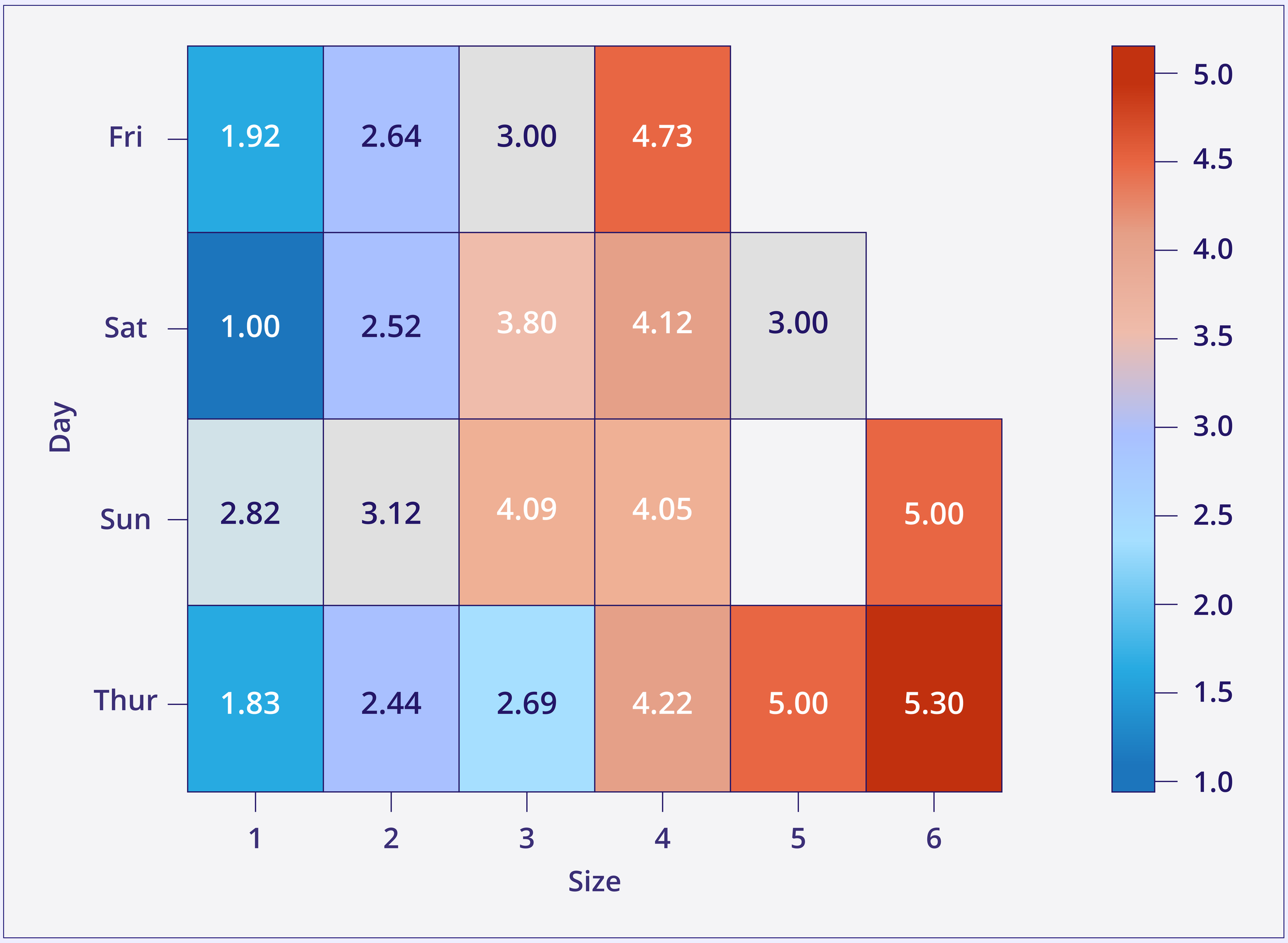




Heatmap

This is a two-dimensional representation of data where the intensity of colors represents the magnitude of values.

```
# Create a pivot table with average tips by day and time
pivot_table = tips.pivot_table(index='Day', columns='Size', values='tip',
aggfunc='mean')

# Create a heatmap showing the average tips by day and time
ax = sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="coolwarm",
linewidths=.5)
```

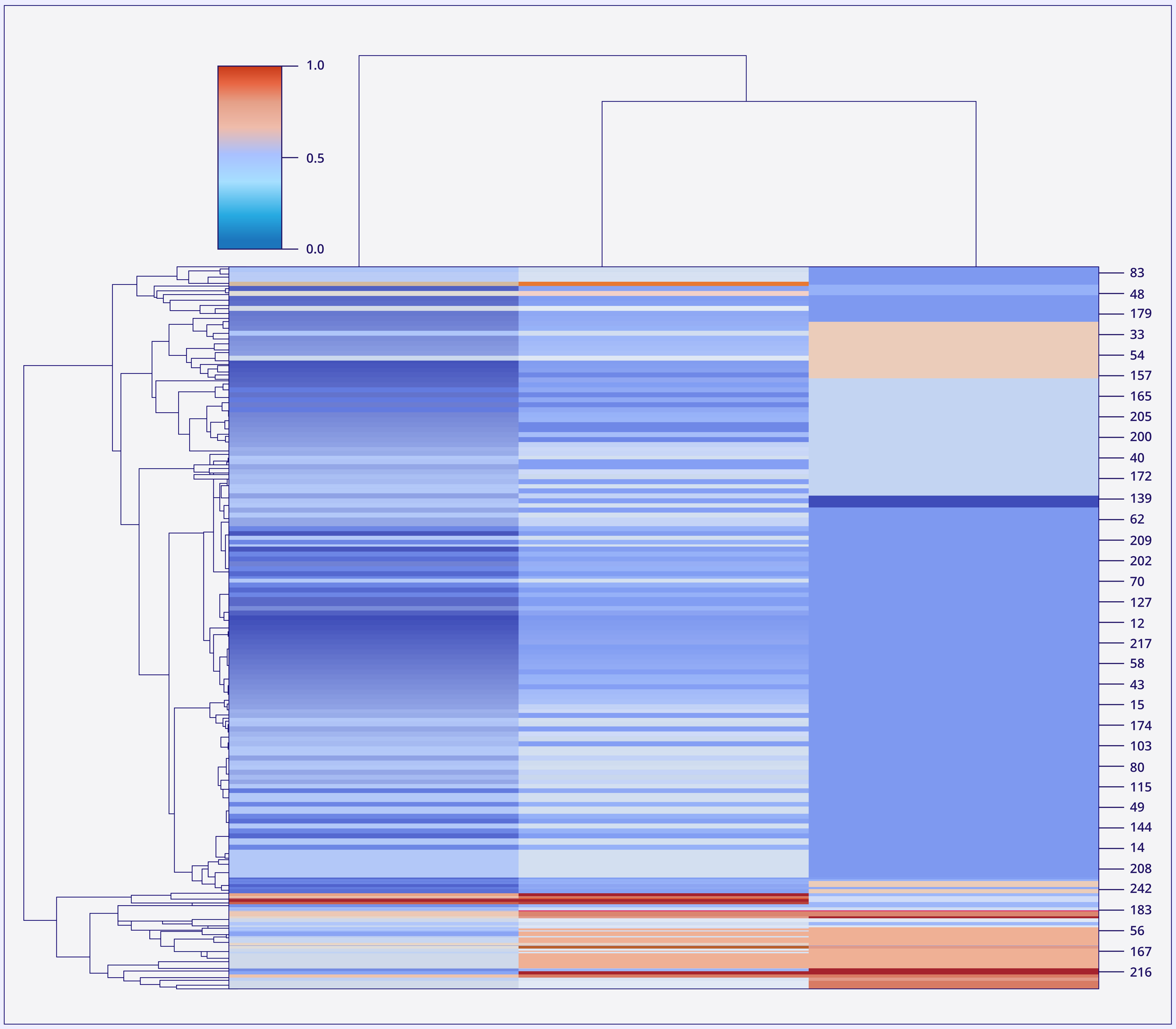


Clustermap

This is a heatmap that uses hierarchical clustering to organize the data, providing insights into the structure of the dataset.

```
# Select only numerical columns
numerical_tips = tips[['total_bill', 'tip', 'size']]

# Generate the clustermap
sns.clustermap(numerical_tips, cmap='coolwarm', standard_scale=1)
```

Seaborn: Statistical Data Visualization

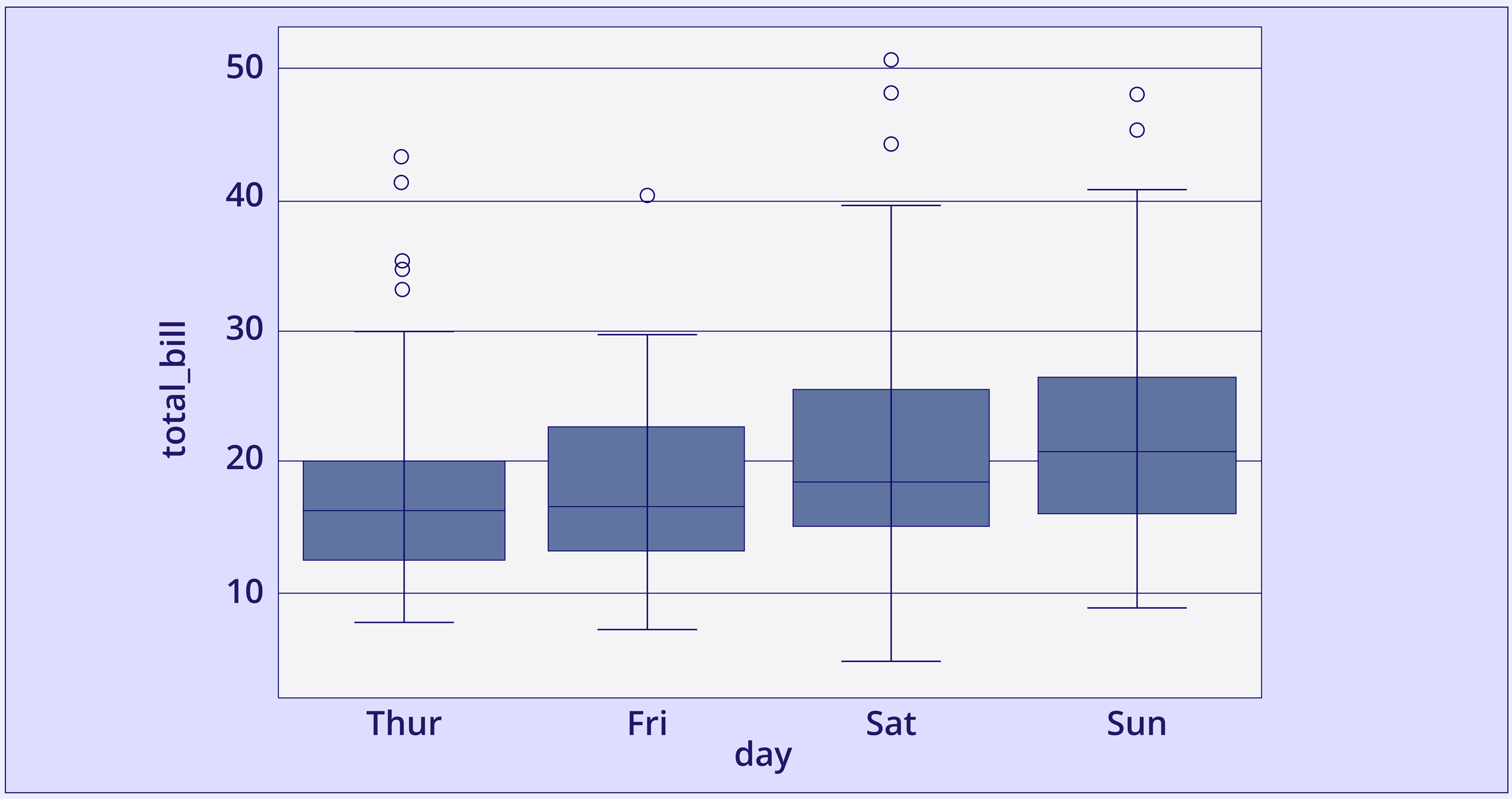


Visualizing Categorical Data

Box Plot

It displays the distribution of data within each category through quartiles and outliers.

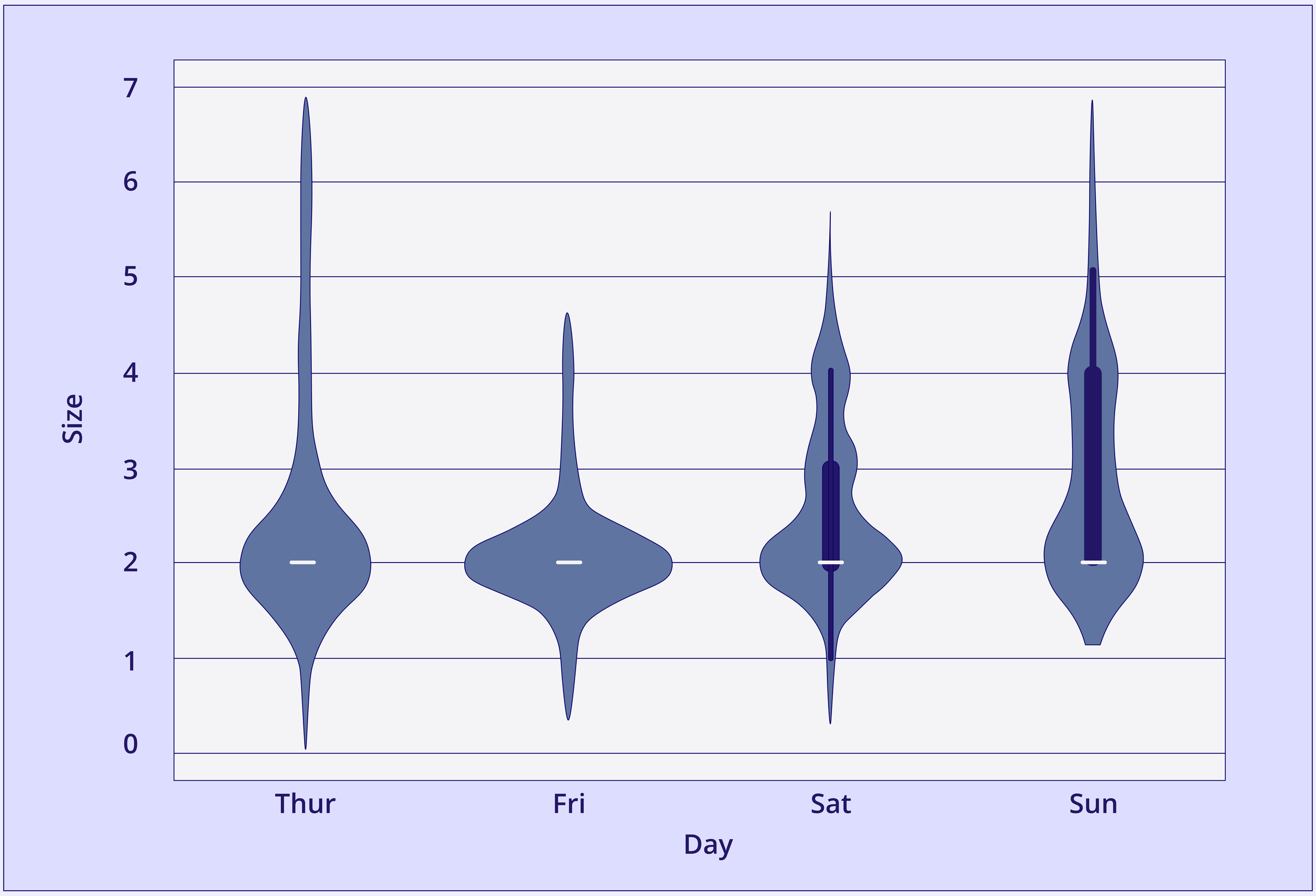
```
plot = sns.boxplot(x="day", y="total_bill", data=tips)
```



Violin Plot

It combines box plots and kernel density plots, showing both the summary statistics and the density of the data distribution.

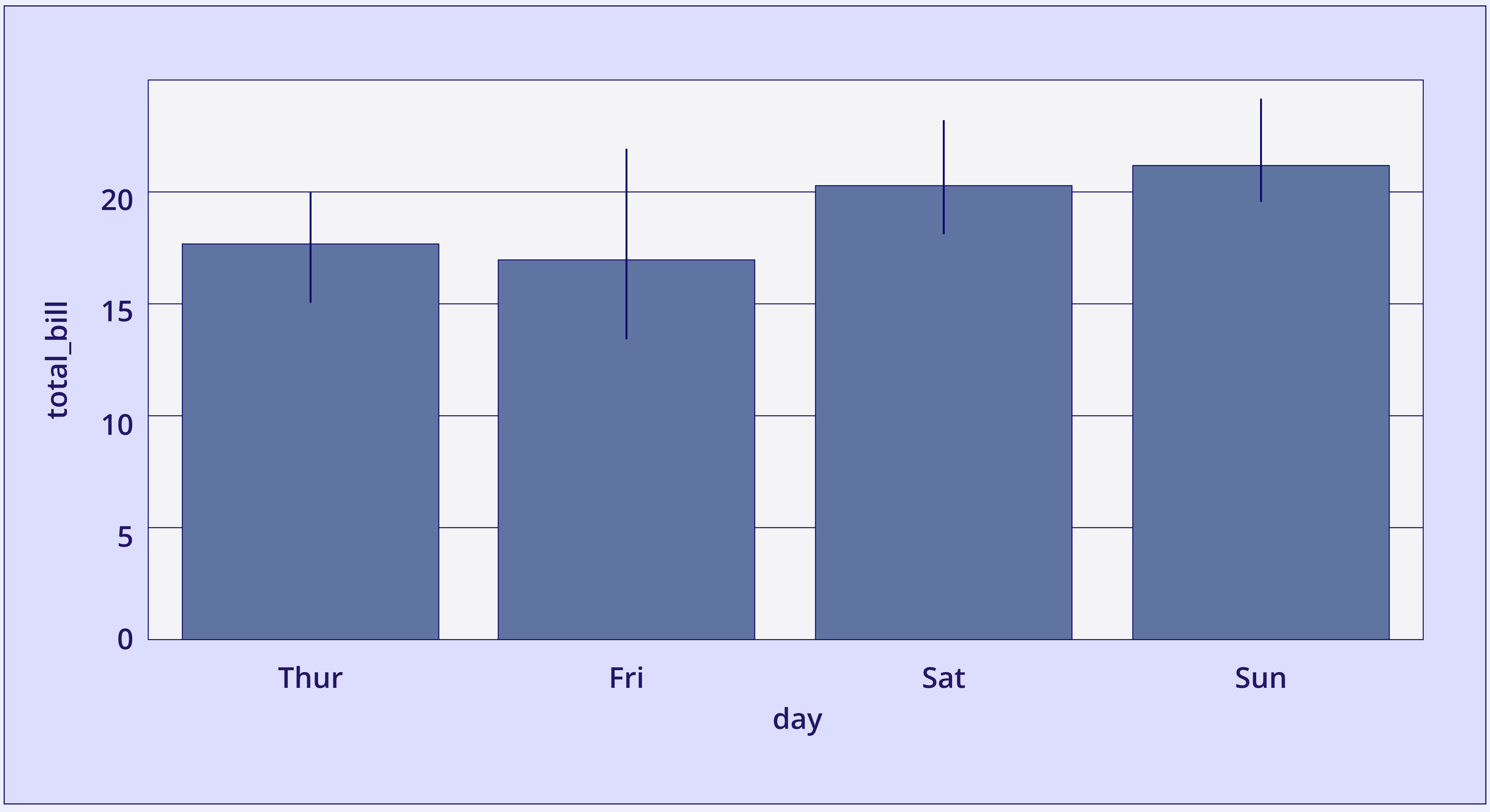
```
plot = sns.violinplot(x="day", y="size", data=tips)
```



Bar Plot

It displays data with rectangular bars with heights proportional to the values they represent.

```
sns.barplot(x="day", y="total_bill", data=tips)
```



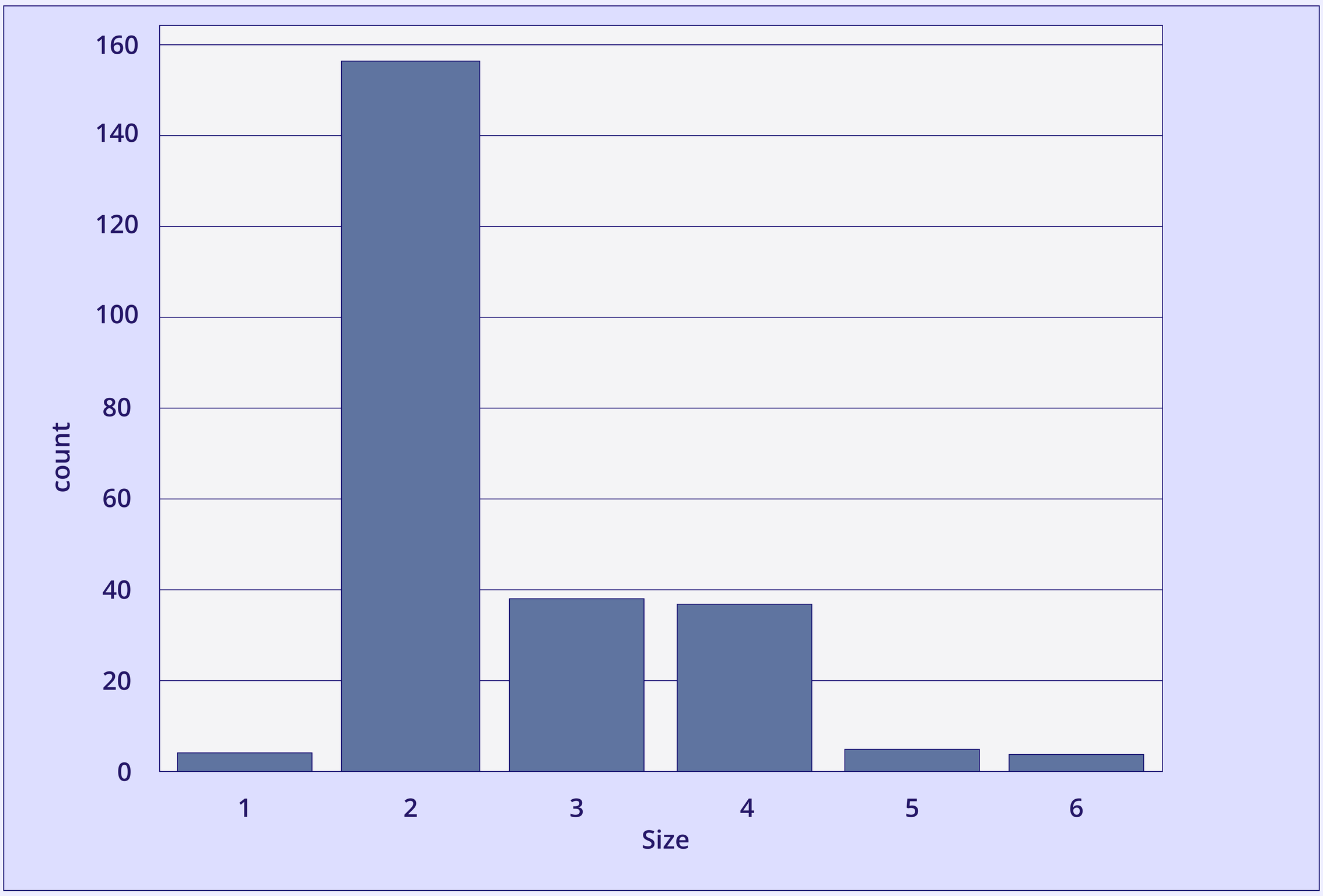
Seaborn: Statistical Data Visualization

educative

Count Plot

It is a type of bar plot that shows the count of cases in each categorical bin using bars.

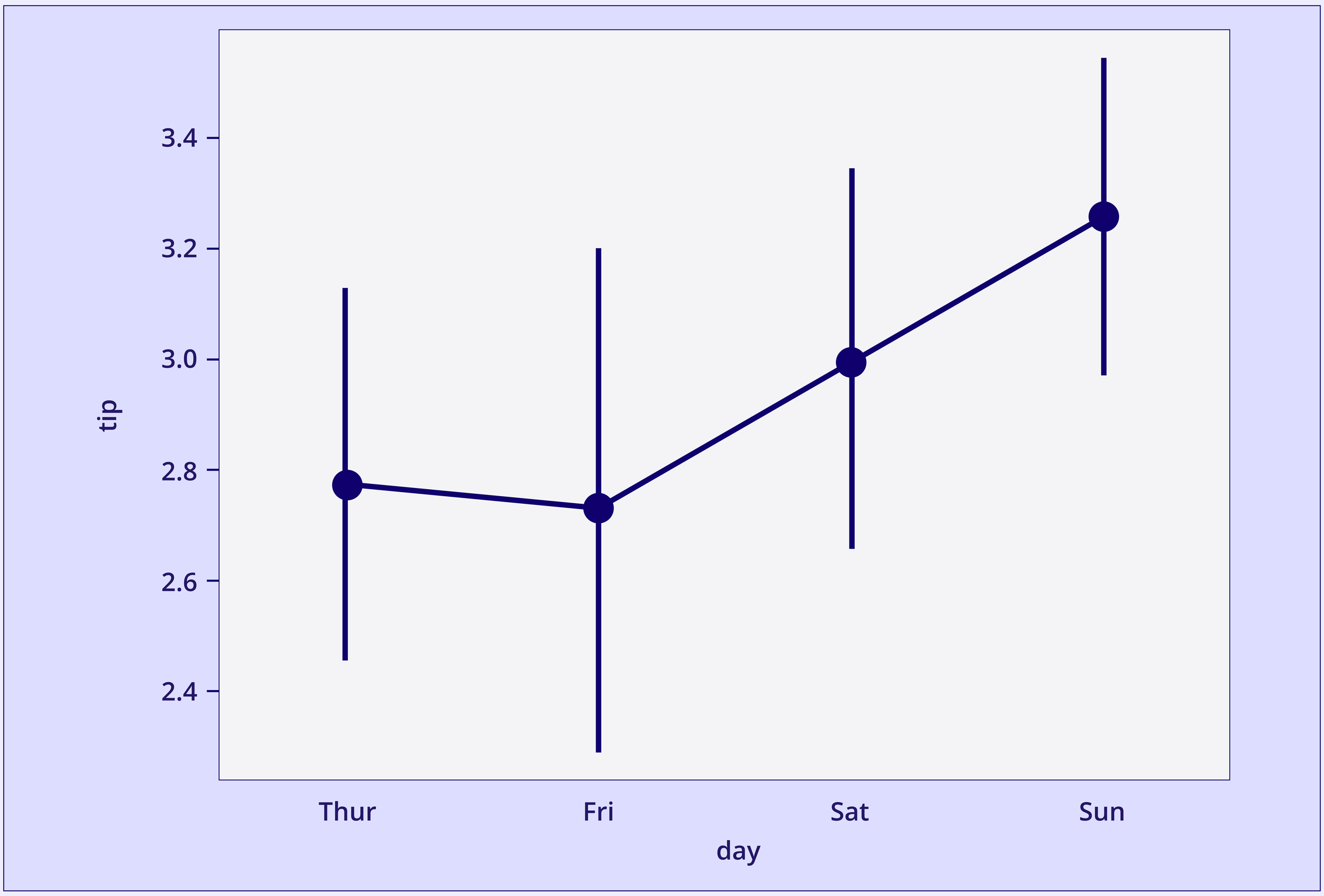
```
plot = sns.countplot(x="size", data=tips)
```



Point Plot

Points are plotted at the estimated value of a quantitative variable.

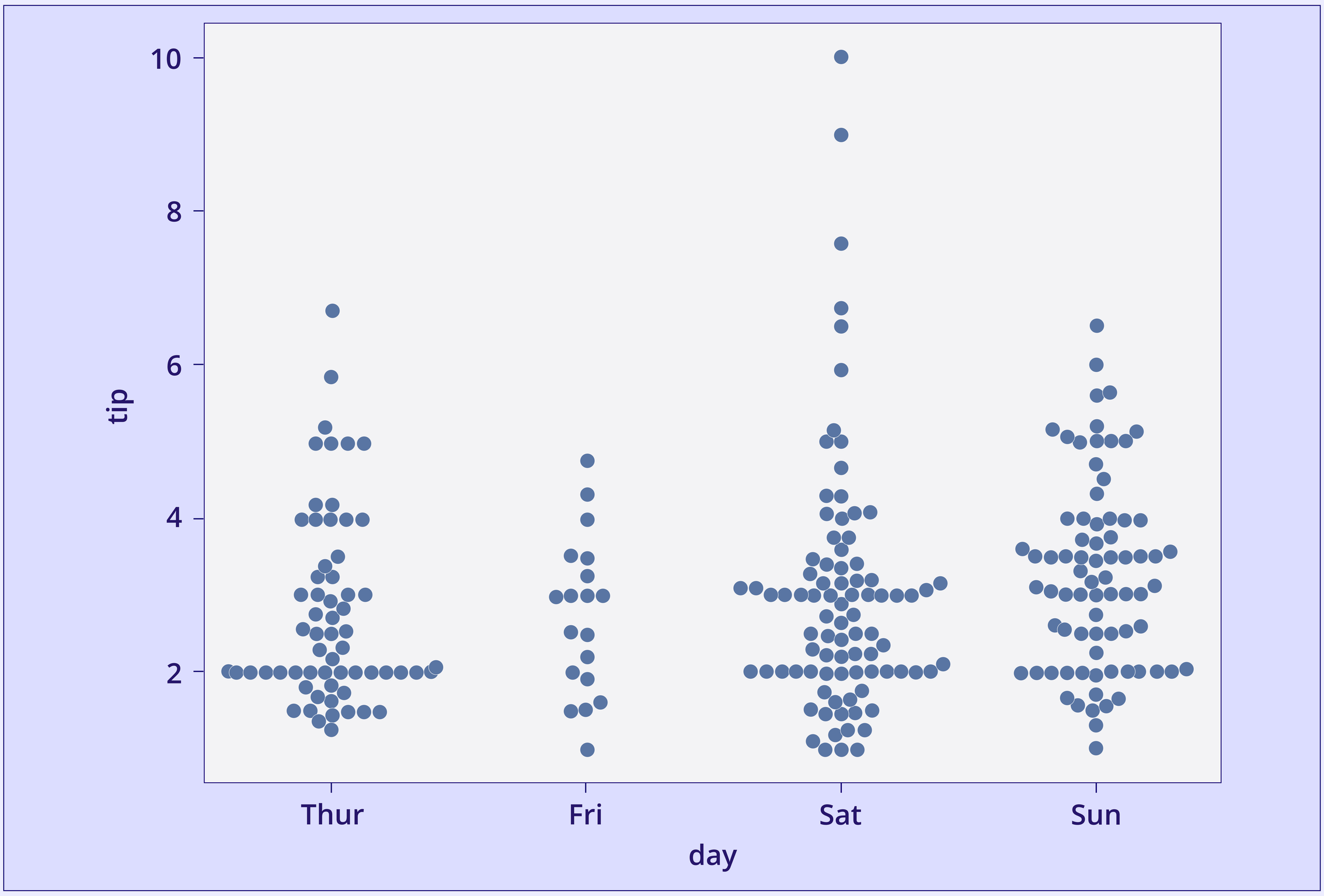
```
sns.pointplot(data=tips, x='day', y='tip')
```



Swarm Plot

It displays all data points for categorical data, adjusting their position to prevent overlap and clearly show distributions.

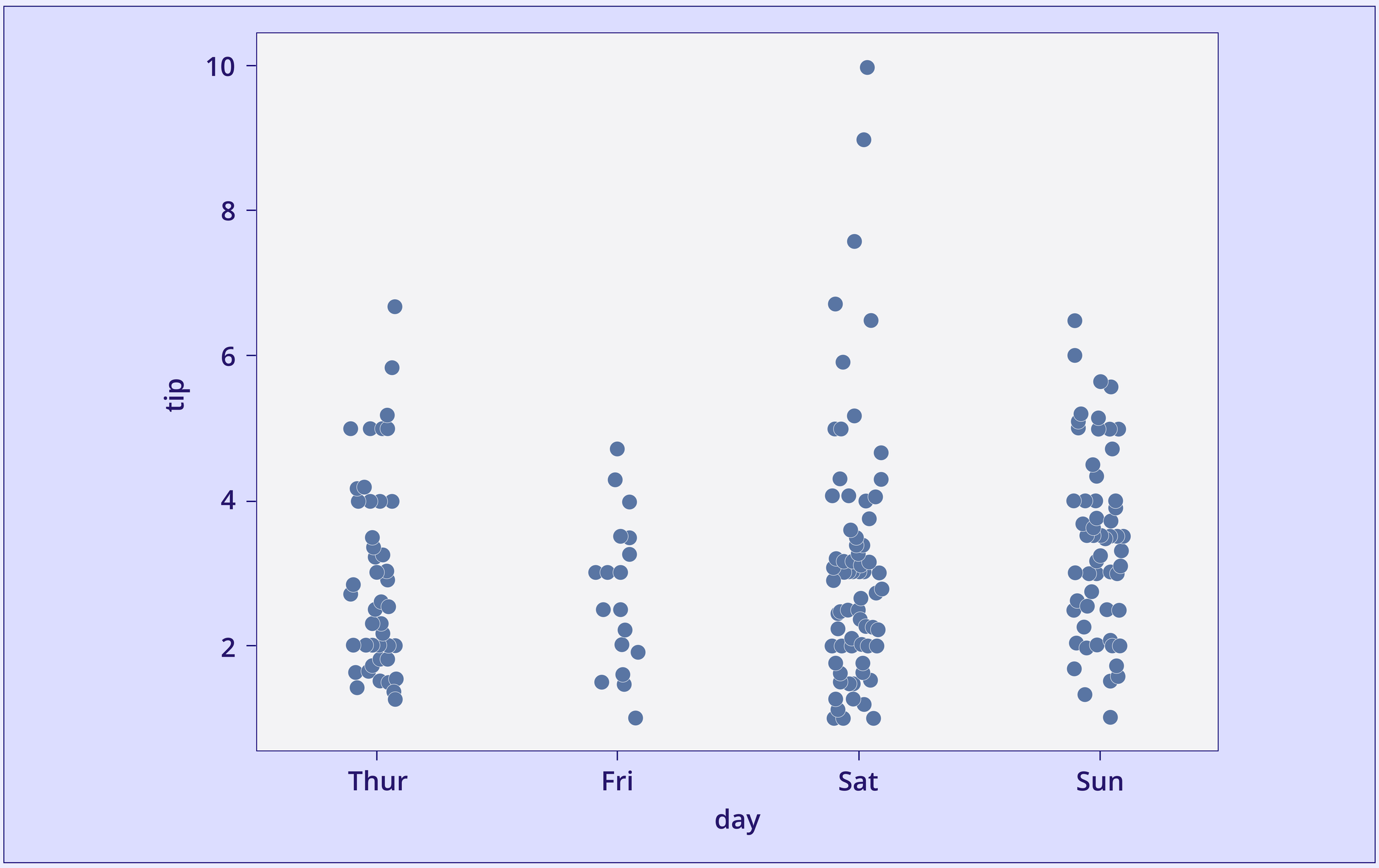
```
sns.swarmplot(data=tips, x='day', y='tip')
```



Strip Plot

It plots a scatter of the categorical data, where one axis is categorical. It’s like a scatter plot but for categorical data.

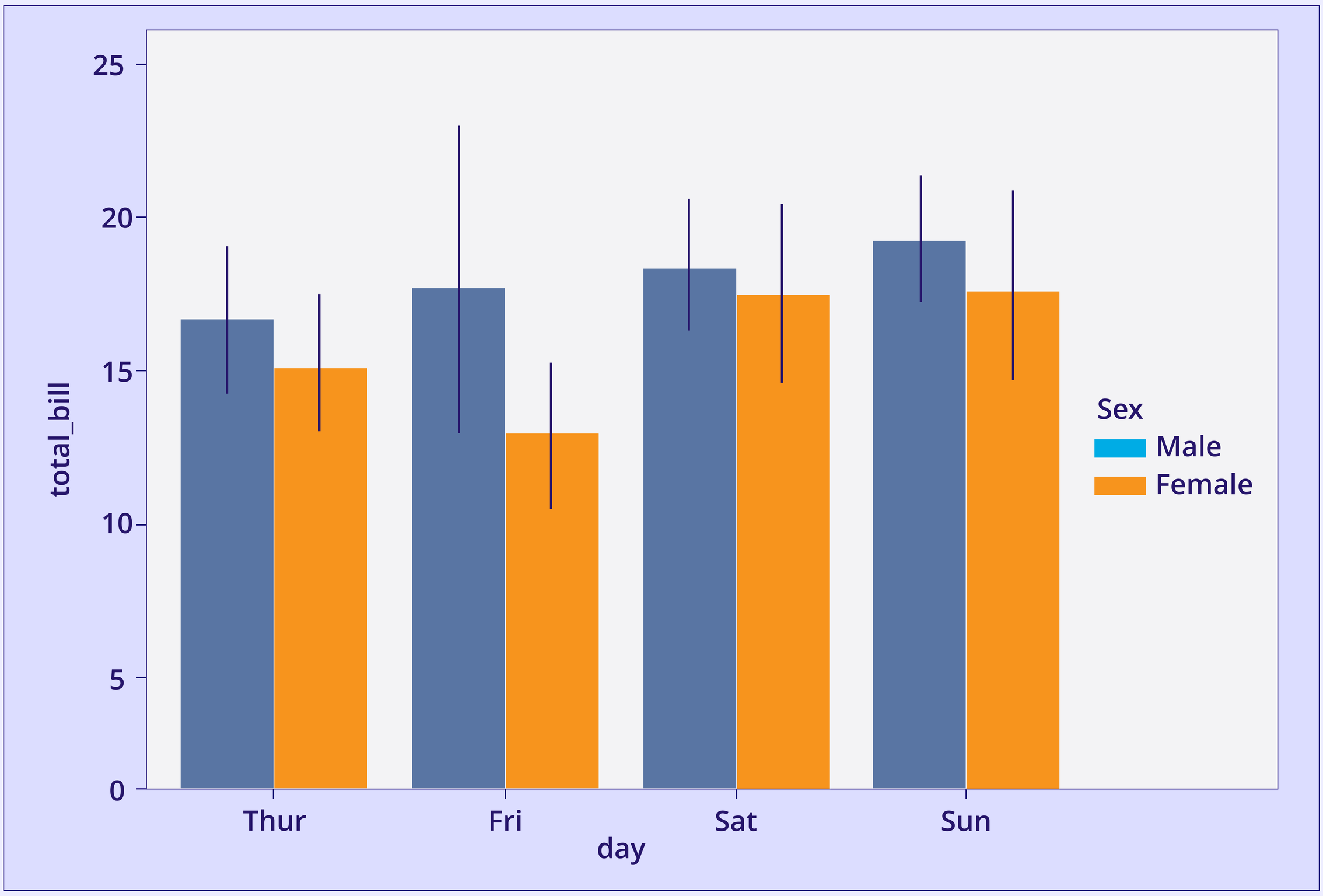
```
sns.stripplot(data=tips, x='day', y='tip')
```



Cat Plot

This is a higher-level interface for drawing categorical plots onto a FacetGrid. It allows the combining of different plot types with facets.

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

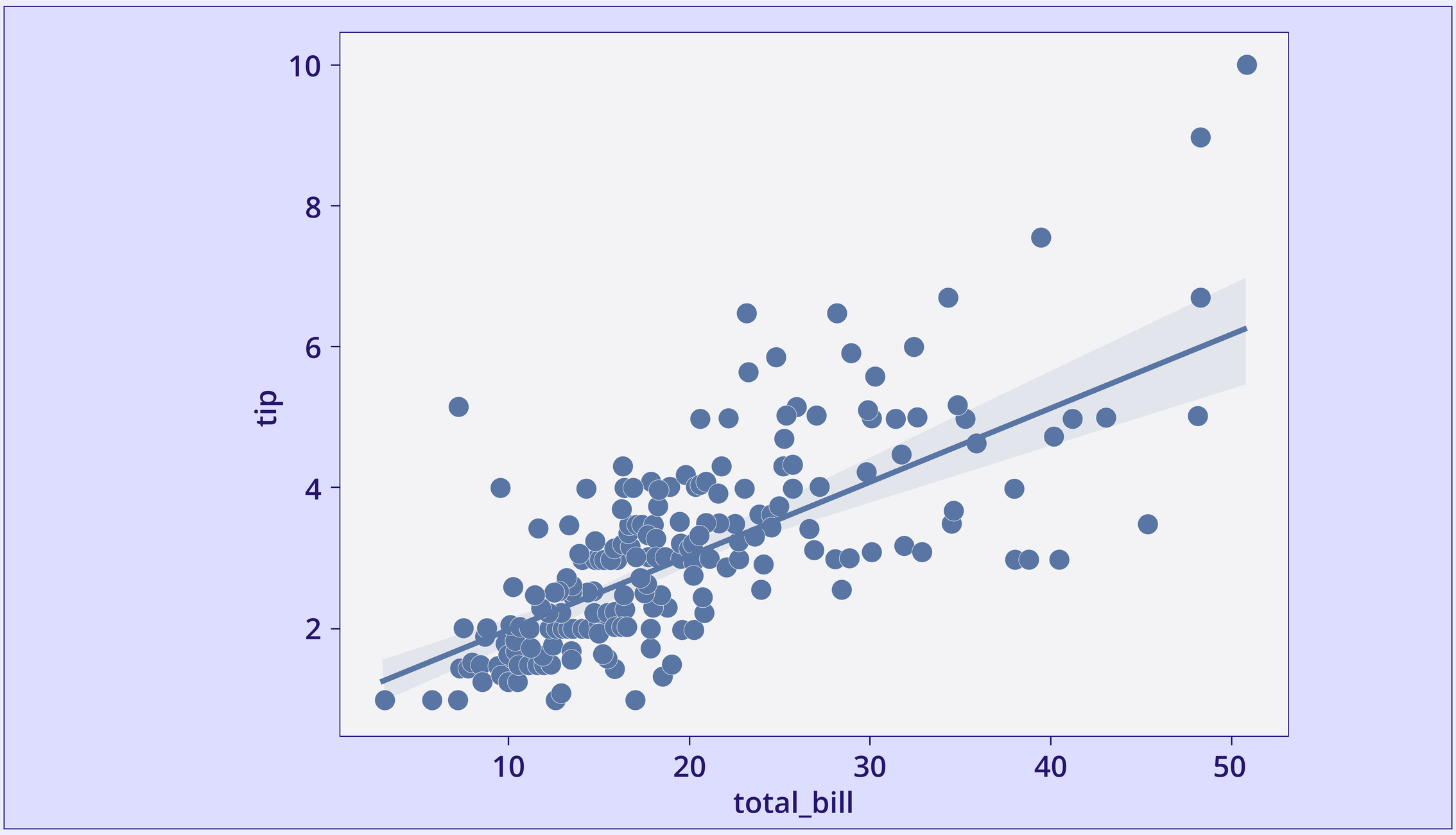


Exploring Relationships with Regression Plots

Reg Plot

It creates a scatter plot with a linear regression fit line, making it easy to see the relationship between two variables and the trend in the data.

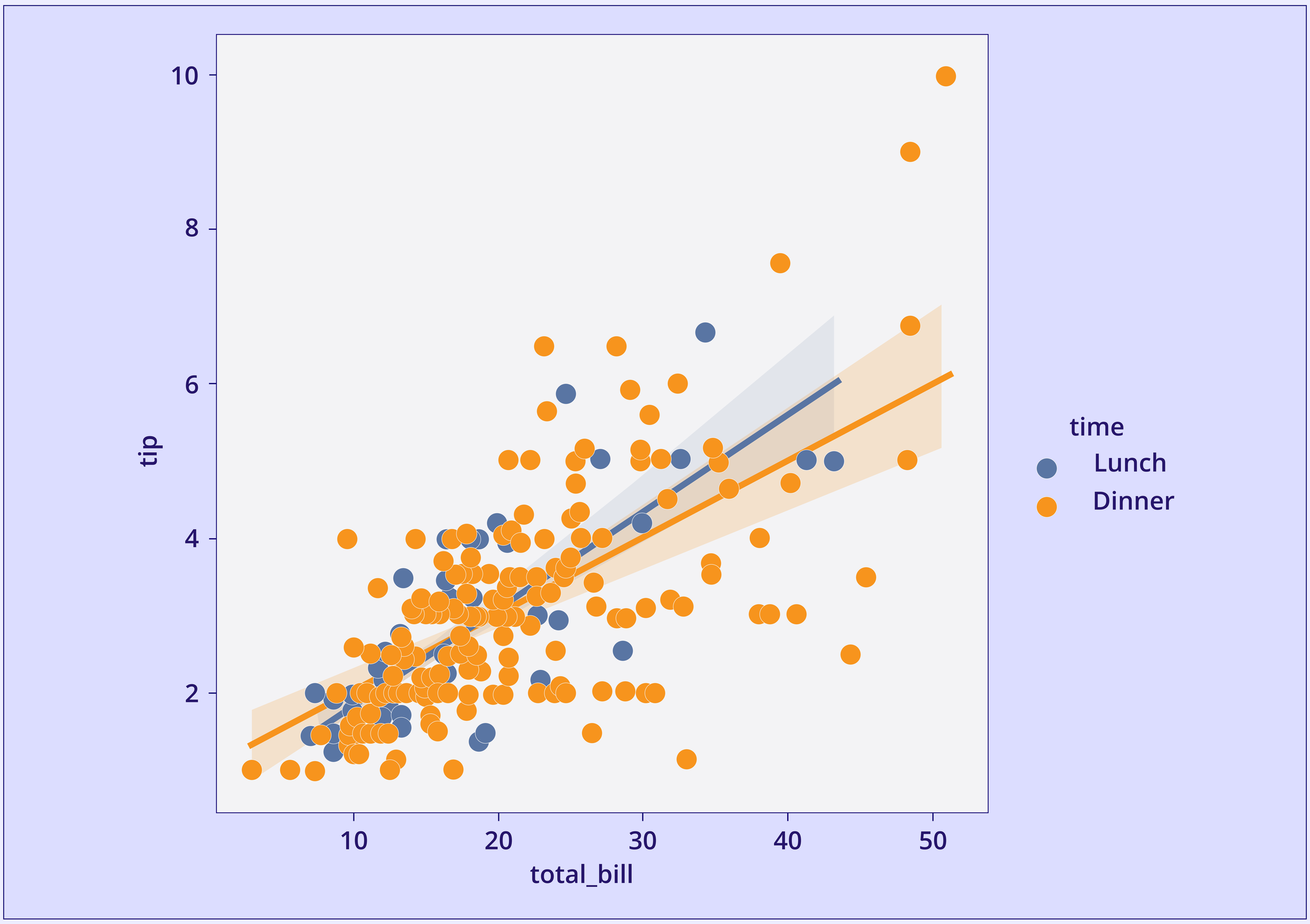
```
sns.regplot(x='total_bill', y='tip', data=tips)
```



Lm Plot (Linear Model Plot)

This is an extension of the reg plot that includes a regression line with a 95% confidence interval, allows for a more detailed analysis of the data.

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

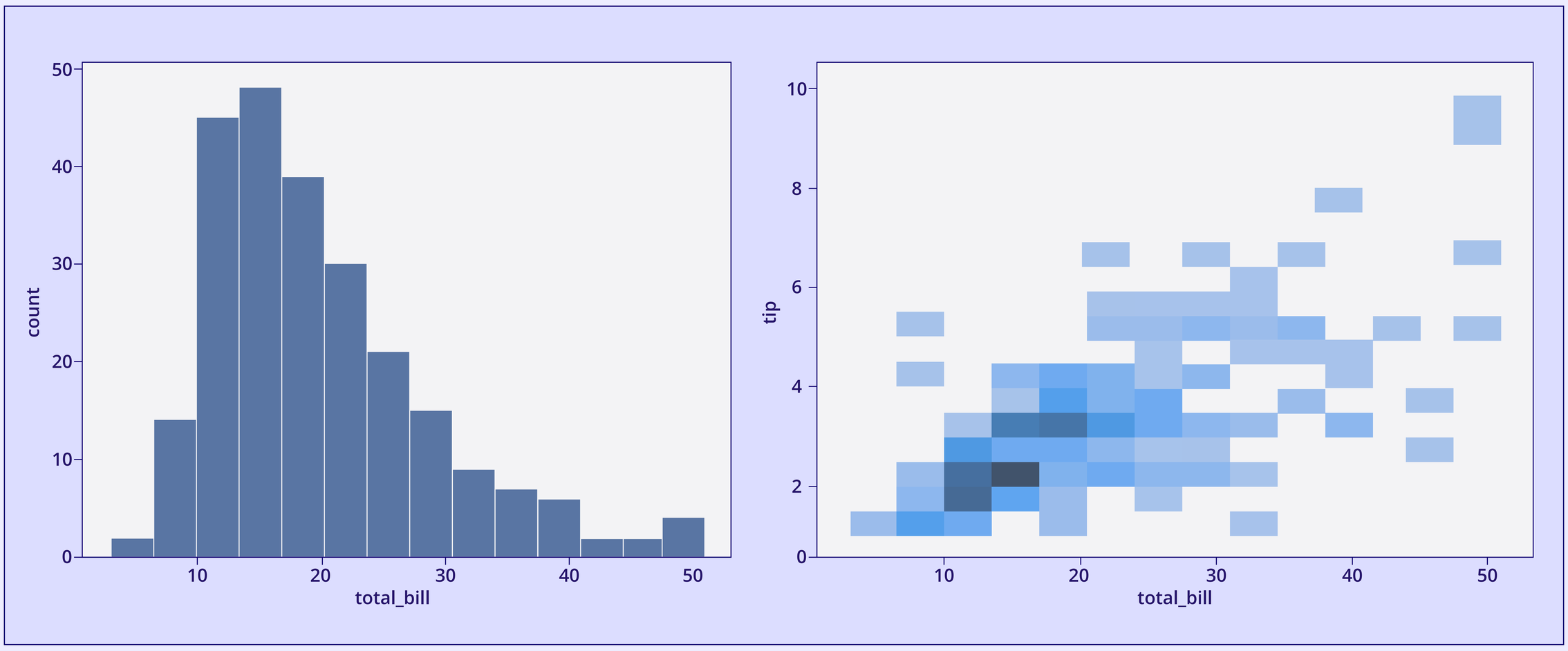


Distribution Plots for Statistical Analysis

Displot

It offers multiple methods for visualizing data distribution, either one variable at a time (univariate) or two variables at a time (bivariate).

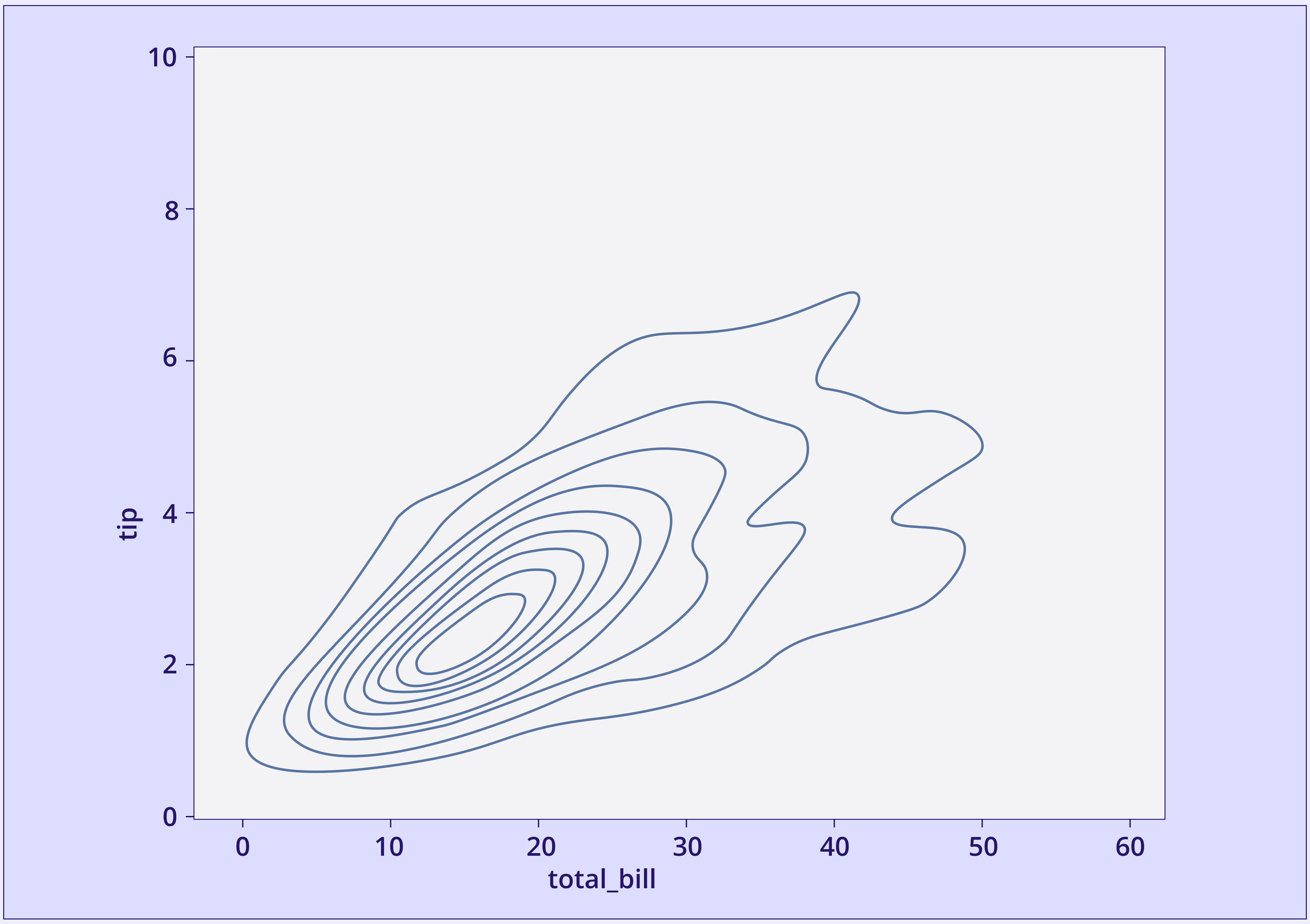
```
# One variable plot
sns.displot(x='total_bill', data=tips)


# Multi variable plot
sns.displot(x='total_bill', y='tip', data=tips)
```



KDE Plot

It displays the data to show a continuous probability density curve, providing a clear view of data distribution.

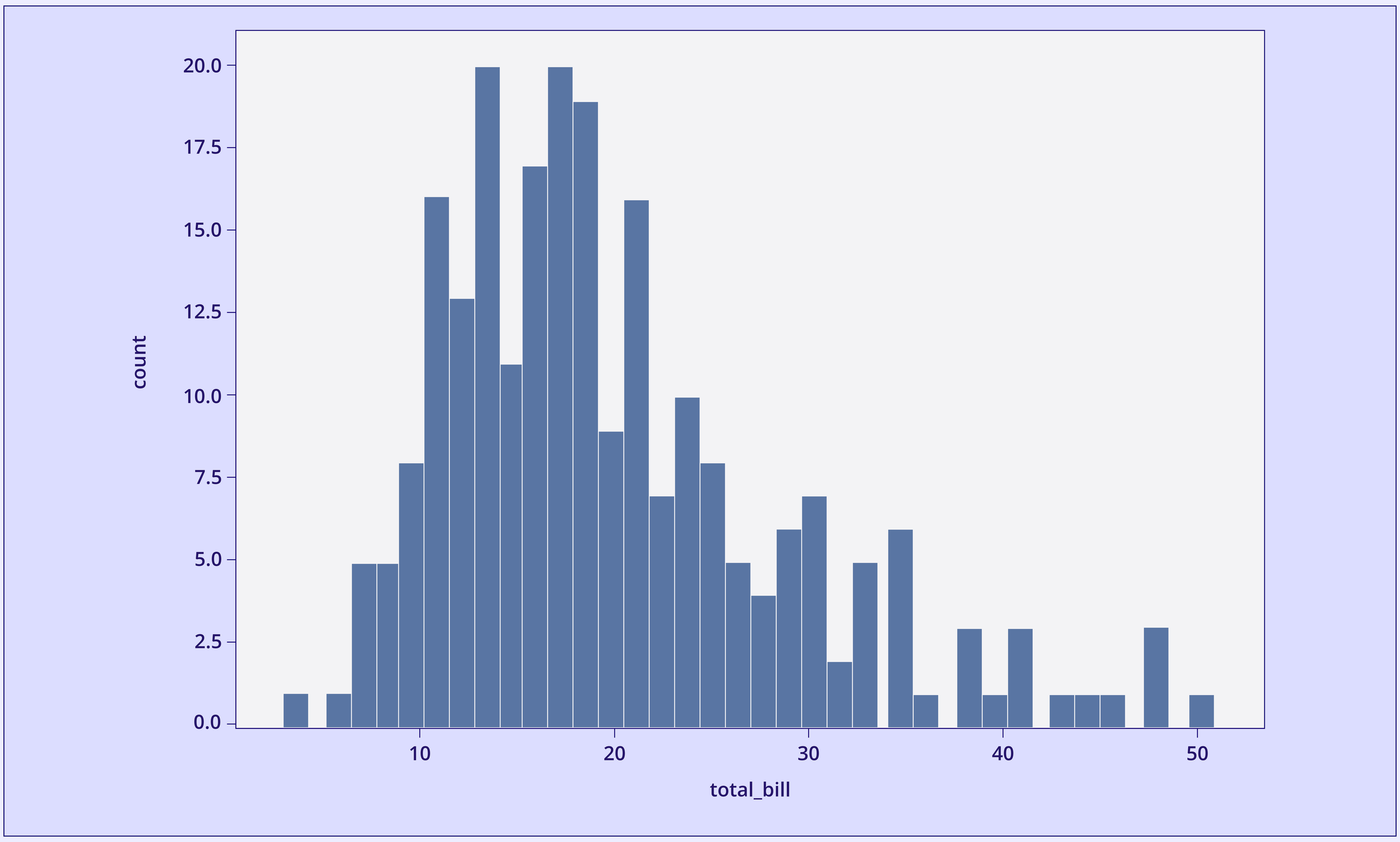
```
sns.kdeplot(x='total_bill', y='tip', data=tips)
```



Hist Plot

It displays the data distribution by counting the frequency of observations within defined intervals or bins.

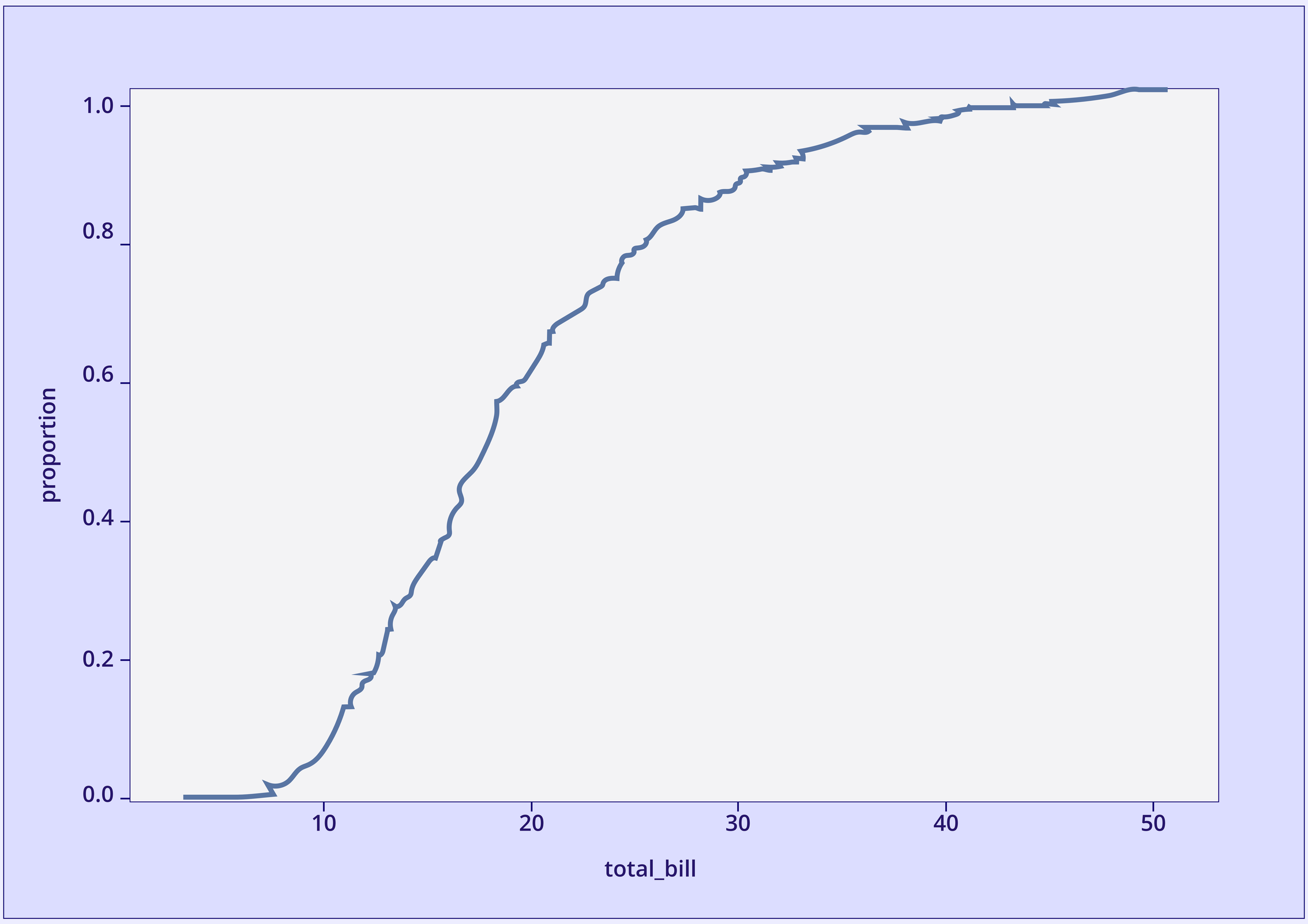
```
sns.histplot(x='total_bill', bins=40, data=tips)
```



ECDF Plot

It plots every data point to depict the cumulative distribution, facilitating direct distribution comparisons.

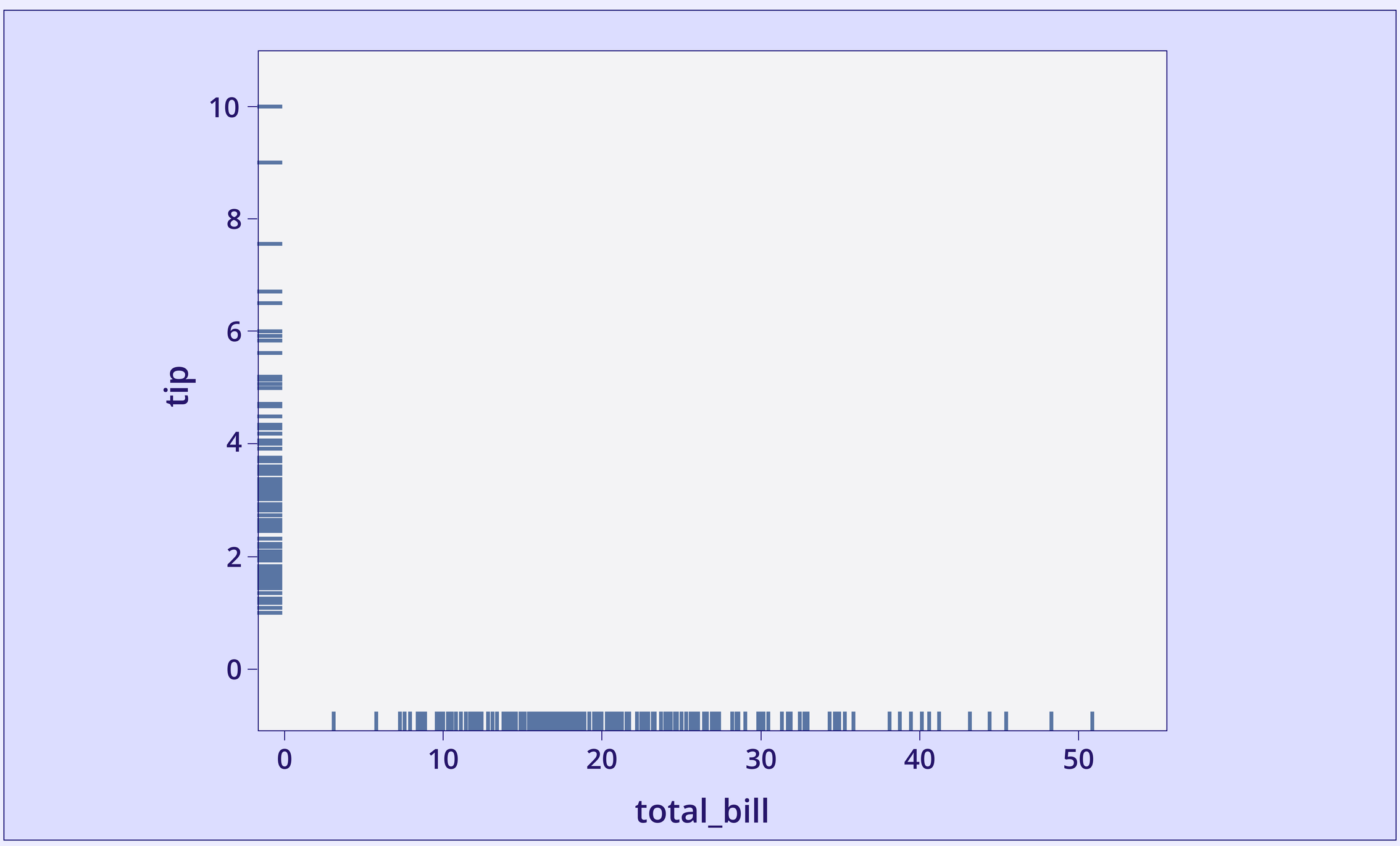
```
sns.ecdfplot(x='total_bill', data=tips)
```



Rug Plot

It places marks along an axis to represent individual data points, enhancing the depth of distribution analysis.

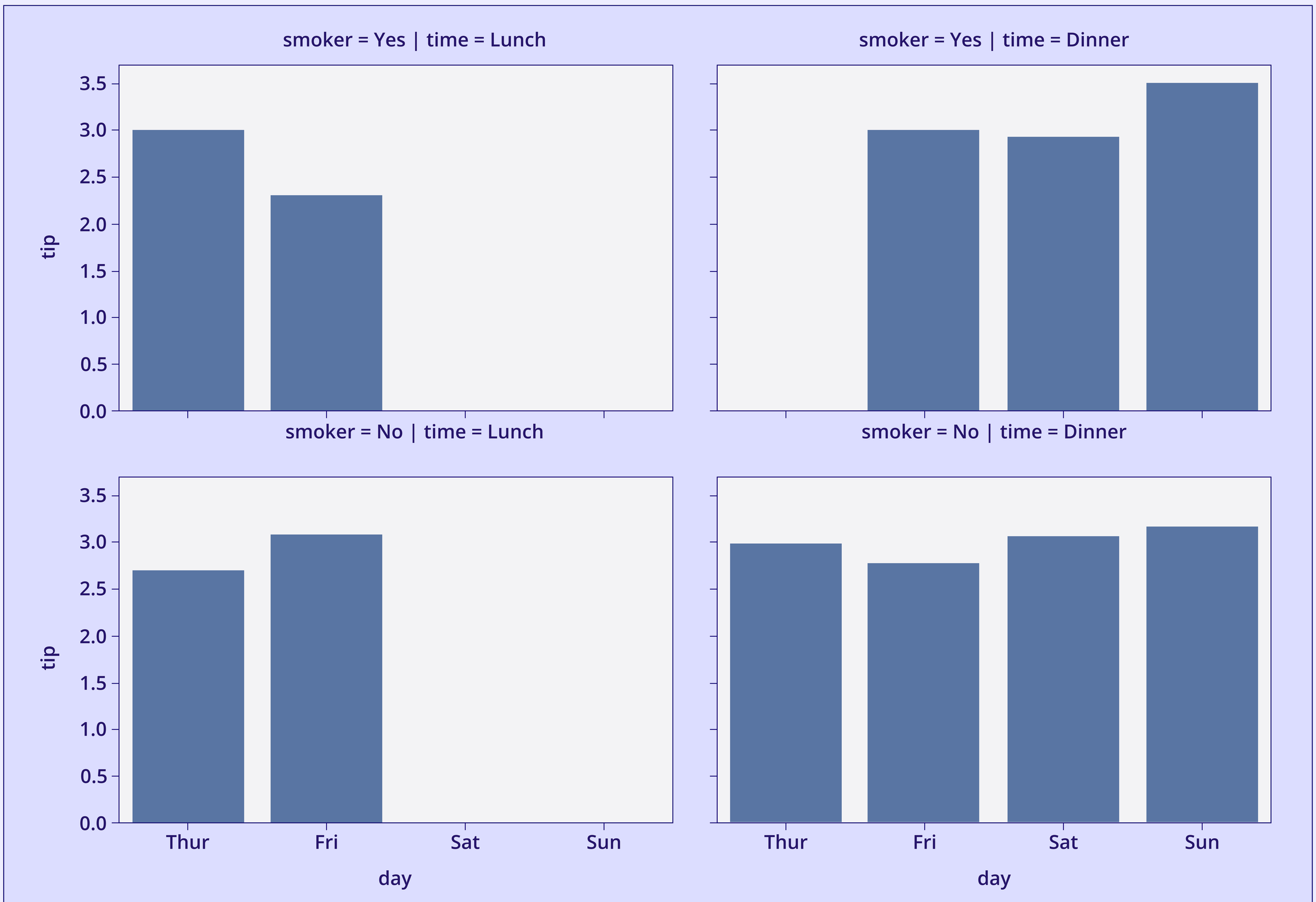
```
sns.rugplot(x='total_bill', y='tip', data=tips)
```



Multifaceted Visuals

FacetGrid

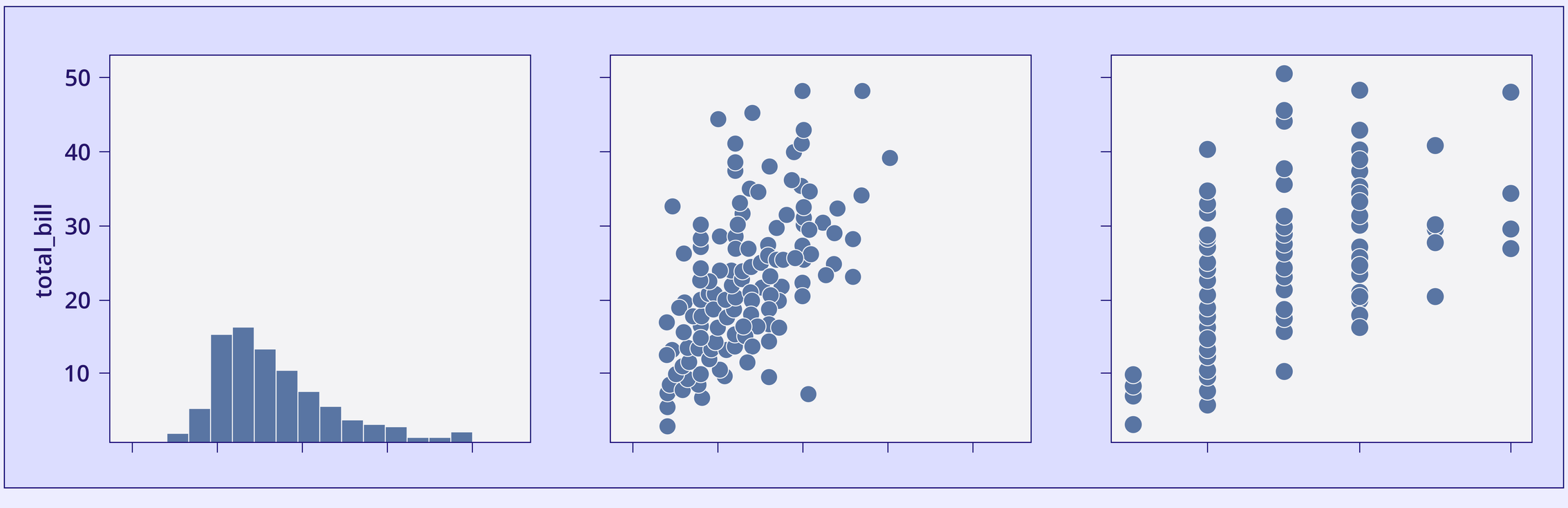
This is a multiplot grid for plotting conditional relationships.



PairGrid

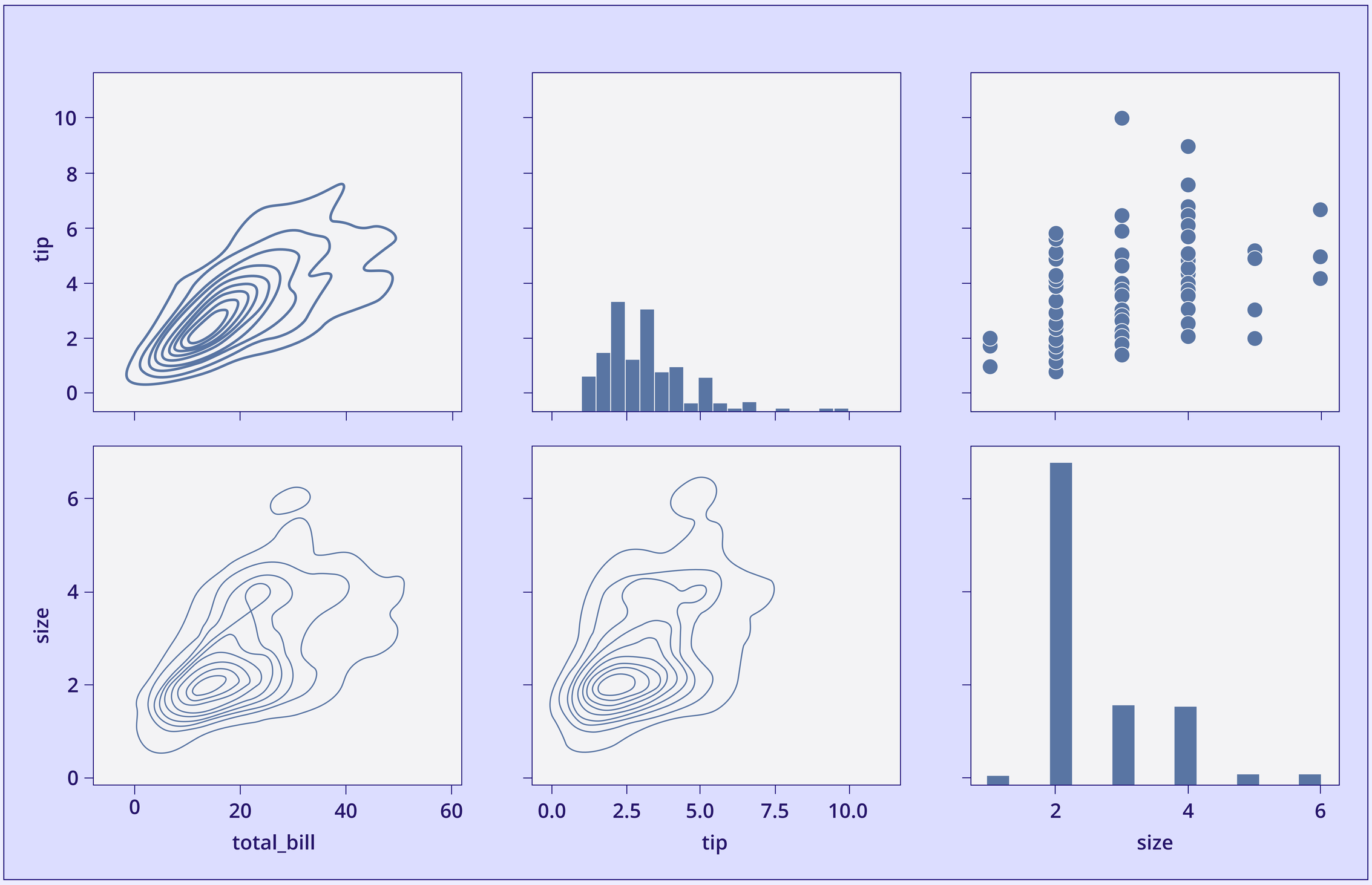
This is a subplot grid for plotting pairwise relationships in a dataset. This can be used to plot the same or different plots for each pair of variables.

```
plot = sns.PairGrid(tips)
plot.map_upper(sns.scatterplot)
plot.map_diag(sns.histplot)
plot.map_lower(sns.kdeplot)
```



Seaborn: Statistical Data Visualization

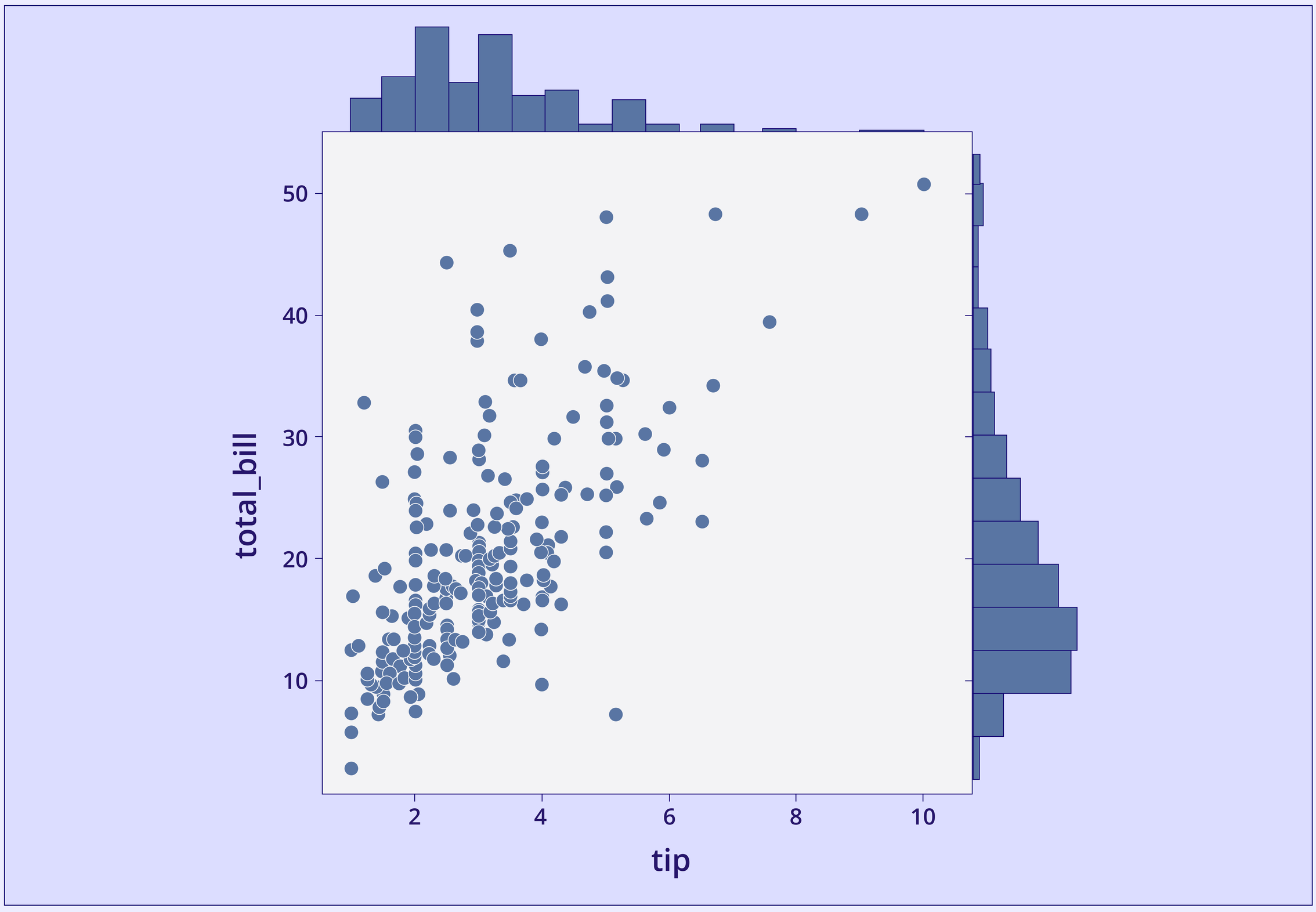
educative



JointGrid

This is a grid for plotting joint distributions by scatter plots or hex bins along with marginal distributions.

```
graph = sns.JointGrid(data=tips, x="tip", y="total_bill")
graph.plot(sns.scatterplot, sns.histplot)
```



Exporting and Displaying Plots

Preserve your visual data analysis results in various formats.

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

plot = sns.scatterplot(x="variable_x", y="variable_y", data=your_dataframe)

# Save the plot
plot.figure.savefig('plot_scatterplot.png', dpi=300)

# Display the plot
plt.show()
```

Clearing Plots

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [1, 4, 9])  # Plot some data
plt.show()  # Display the plot
plt.clear()  # Close the axes to refresh the plotting
plt.clf()  # Close the figure
plt.close()  # Close the plot window
```

Glossary

Let’s explore the utility of each type of plot.

Line Plot	Visualizes data trends over time
Scatter Plot	Examines relationships and distributions of two variables
RelPlot	Visualizes statistical relationships across different categories
Heatmap	Displays complexity in multivariate data, often in matrix formt
Clustermap	Visualizes structures in matrix data where similar clusters are grouped
Box Plot	Visualizes distributions with quartiles and outliers
Violin Plot	Visualizes the distribution and probability density of data
Bar Plot	Compares numerical values across different categories
Count Plot	Displays the counts of observations or frequency of categories
Point Plot	Displays point estimates with error bars, emphasizing differences between groups
Swarm Plot	Displays distribution of categorical data, avoiding overlapping points
Strip Plot	Represents individual data points in a categorical scatterplot
Cat Plot	All-in-one method for creating various categorical plots
Reg Plot	Visualizes a simple linear relationship between two variables
Lm Plot	Displays multiple regression plots for subsets of a dataset
Displot	Summarizes the distribution of a dataset
KDE Plot	Visualizes the distribution of a dataset
Hist Plot	Displays the frequency distribution of a numerical variable
ECDF Plo	Assesses the probability distribution of a dataset
Rug Plot	Displays individual observations along an axis
Joint Plot	Displays the relationship between two variables and their marginal distributions
FacetGrid	Creates a grid of plots based on a dataset’s features to compare distributions
PairGrid	Explores pairwise relationships in a dataset
JointGrid	Creates custom joint and marginal plots