Preaperd By -: Vithusan

Seaborn Tutorial

What is Seaborn?

- Seaborn is a Python data visualization library built on Matplotlib.
- Designed for creating aesthetically pleasing statistical graphics.

Key Features:

1. **High-Level Interface:**
    - Simplifies complex visualizations with minimal code.
2. **Themes and Color Palettes:**
    - Built-in themes and color palettes enhance visual appeal.
    - Easy application to change overall plot appearance.
3. **Statistical Estimation:**
    - Performs statistical estimation while plotting.
    - Automatically adds features like regression lines.
4. **Integration with Pandas:**
    - Seamless integration with Pandas DataFrames.
    - Facilitates easy visualization of data patterns.
5. **Categorical Plotting:**
    - Specialized functions for categorical data visualization.
    - Includes bar plots, count plots, and box plots.

Basic Usage Example:

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

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Create a scatter plot with Seaborn
sns.scatterplot(x=x, y=y)

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

Use Cases:

- Ideal for exploratory data analysis.

- Effective in visualizing relationships, distributions, and patterns in data.

In [50]:
```
import seaborn as sns
```

In [51]:
```
df = sns.load_dataset("tips")
```

In [52]:
```
df.head()
```

Out[52]:

	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

Correlation with Heatmap

What is it?

- A heatmap is a graphical representation of data where values are depicted as colors.
- Correlation measures the statistical association between two variables.

Correlation Heatmap in Seaborn:

- **Purpose:** Visualizing correlation matrices between variables in a dataset.
- **Implementation:** Use `sns.heatmap()` to create a color-coded matrix displaying correlation coefficients.
- **Color Coding:** Positive correlations in one color, negative in another, making patterns easily discernible.

Key Benefits:

- **Quick Insights:** Easily identify relationships between variables.
- **Pattern Recognition:** Visualize complex correlation structures efficiently.
- **Aesthetically Pleasing:** Seaborn's default styles enhance the visual appeal of heatmaps.

Note: Adjust the colormap ("cmap") and other parameters based on preferences and specific use cases.

Use Cases:

- Identifying relationships between features in a dataset.
- Feature selection in machine learning based on correlation strength.

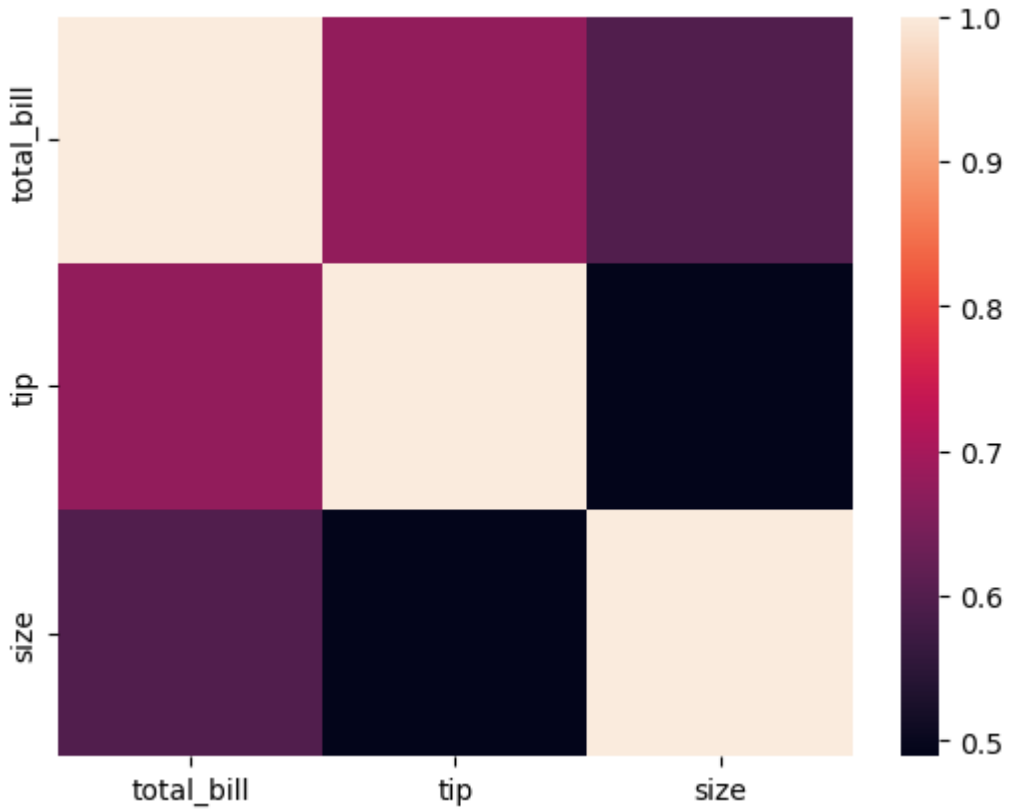
In [53]:
```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
numeric_df.corr()
```

Out[53]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [54]:
```
sns.heatmap(numeric_df.corr())
```

Out[54]:
```
<Axes: >
```



JoinPlot

- **Purpose:** Seaborn's `jointplot` is used to visualize the relationship between two numeric variables along with their univariate distributions.
- **Representation:** It combines a scatter plot of the two variables with histograms on the margins.
- **Insights:** Helps in understanding the joint distribution of variables and their individual distributions.

Key Features:

1. **Scatter Plot:** The main plot displays the joint distribution as a scatter plot.
2. **Marginal Histograms:** Histograms along the X and Y axes show the distribution of each variable individually.

3. **Kernel Density Estimation (KDE):** Optionally, kernel density estimation can be added to represent the data's probability density.

Usage:

- Ideal for exploring the correlation and distribution patterns between two variables.

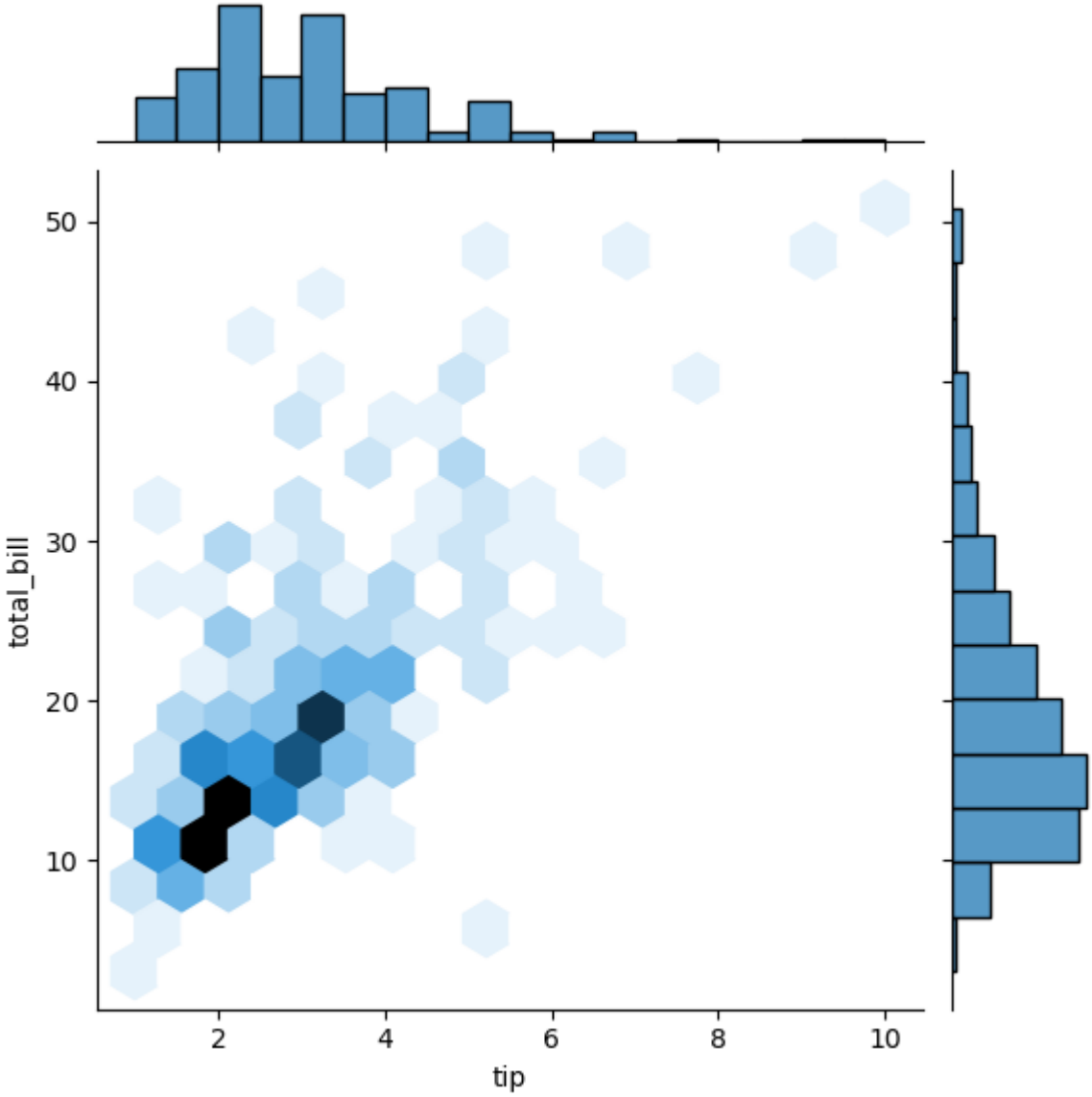
Example Code:

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

# Creating a jointplot
sns.jointplot(x='variable1', y='variable2', data=df, kind='scatter')
plt.show()
```

Univariate Analysis

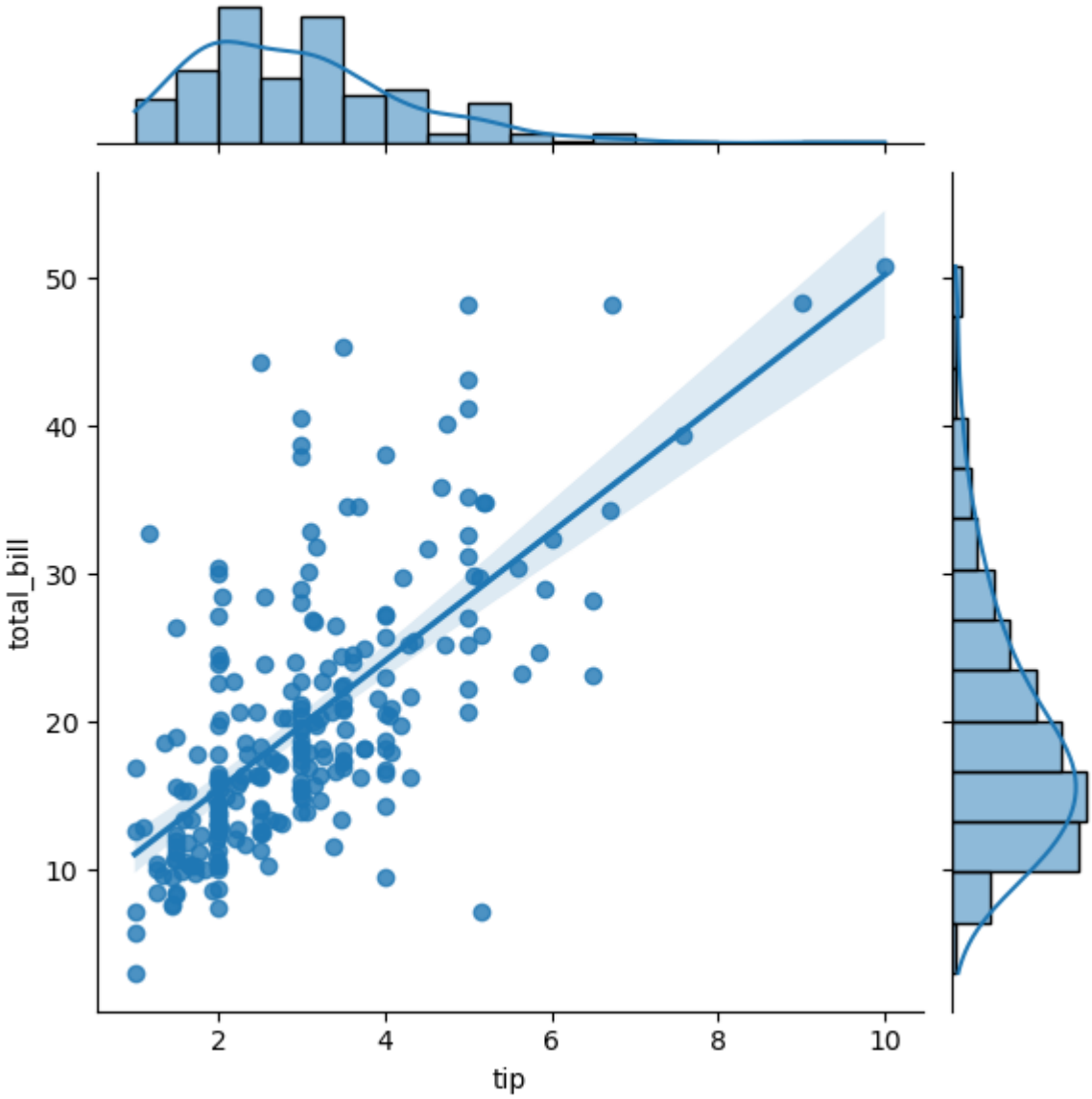
In [55]:
```
sns.jointplot(x = "tip", y = "total_bill", data = df, kind = "hex")
```

Out[55]:
```
<seaborn.axisgrid.JointGrid at 0x19a44ebb050>
```



In [56]:
```
sns.jointplot(x = "tip", y = "total_bill", data = df, kind = "reg")
```

Out[56]:
```
<seaborn.axisgrid.JointGrid at 0x19a46342990>
```



Pair plot

Explanation:

- **Purpose:** Seaborn's `pairplot` is designed for exploring pairwise relationships between numerical variables in a dataset.
- **Representation:** It creates a matrix of scatter plots for each pair of numerical features.
- **Insights:** Enables quick identification of patterns, trends, and correlations within the dataset.

Key Features:

1. **Scatter Plots:** Main diagonal contains histograms, while scatter plots fill the lower and upper triangles.

2. **Kernel Density Estimation (KDE):** Optional inclusion for a more detailed representation of data distribution.
3. **Color-Coding:** Categorical variables can be used to color points for additional insights.

Usage:

- Ideal for understanding relationships between multiple variables in a dataset.
- Quick visual exploration of data structure.

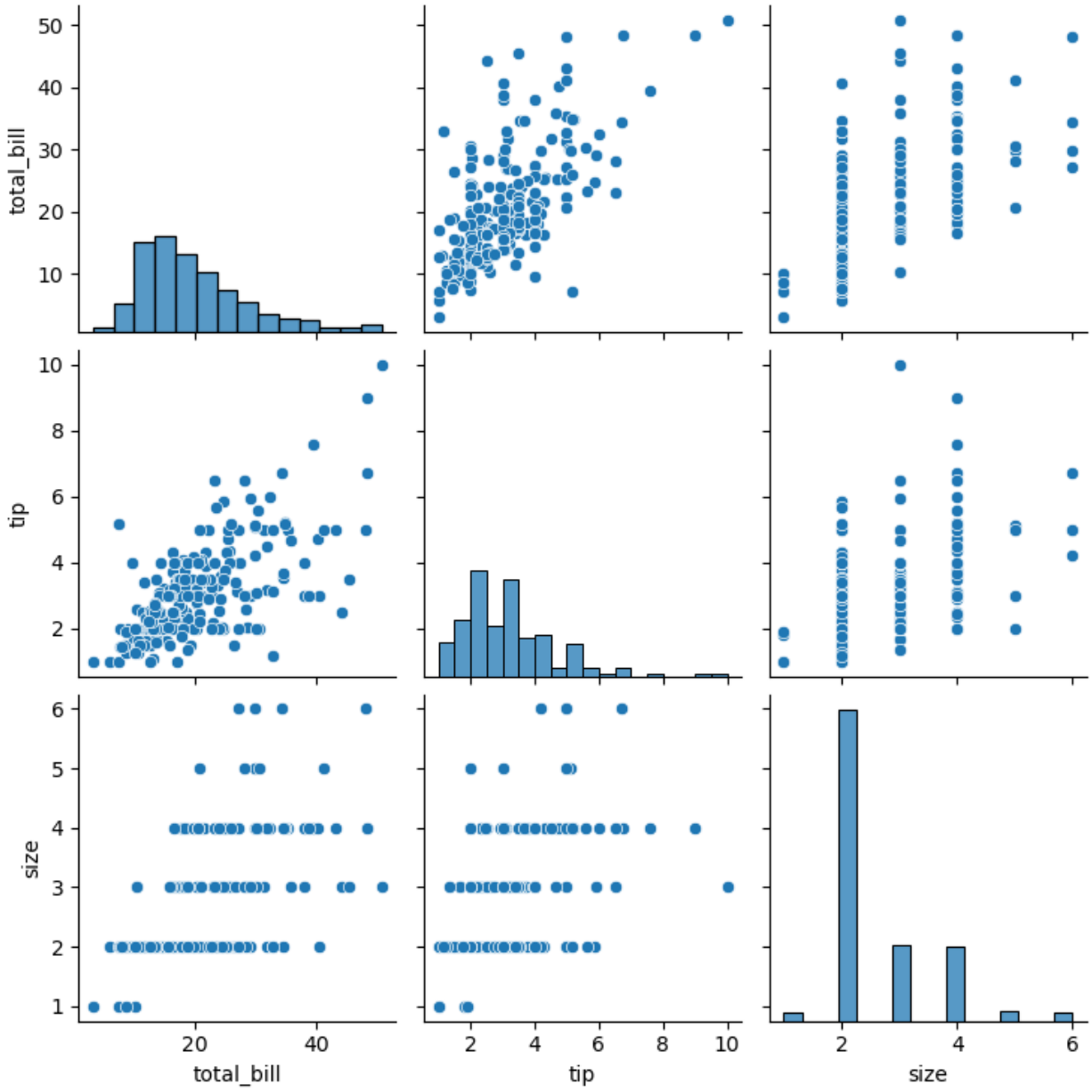
Example Code:

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

# Creating a pair plot
sns.pairplot(df, hue='categorical_variable', kind='scatter')
```

In [57]:
```
sns.pairplot(df)
```

```
D:\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure l
ayout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

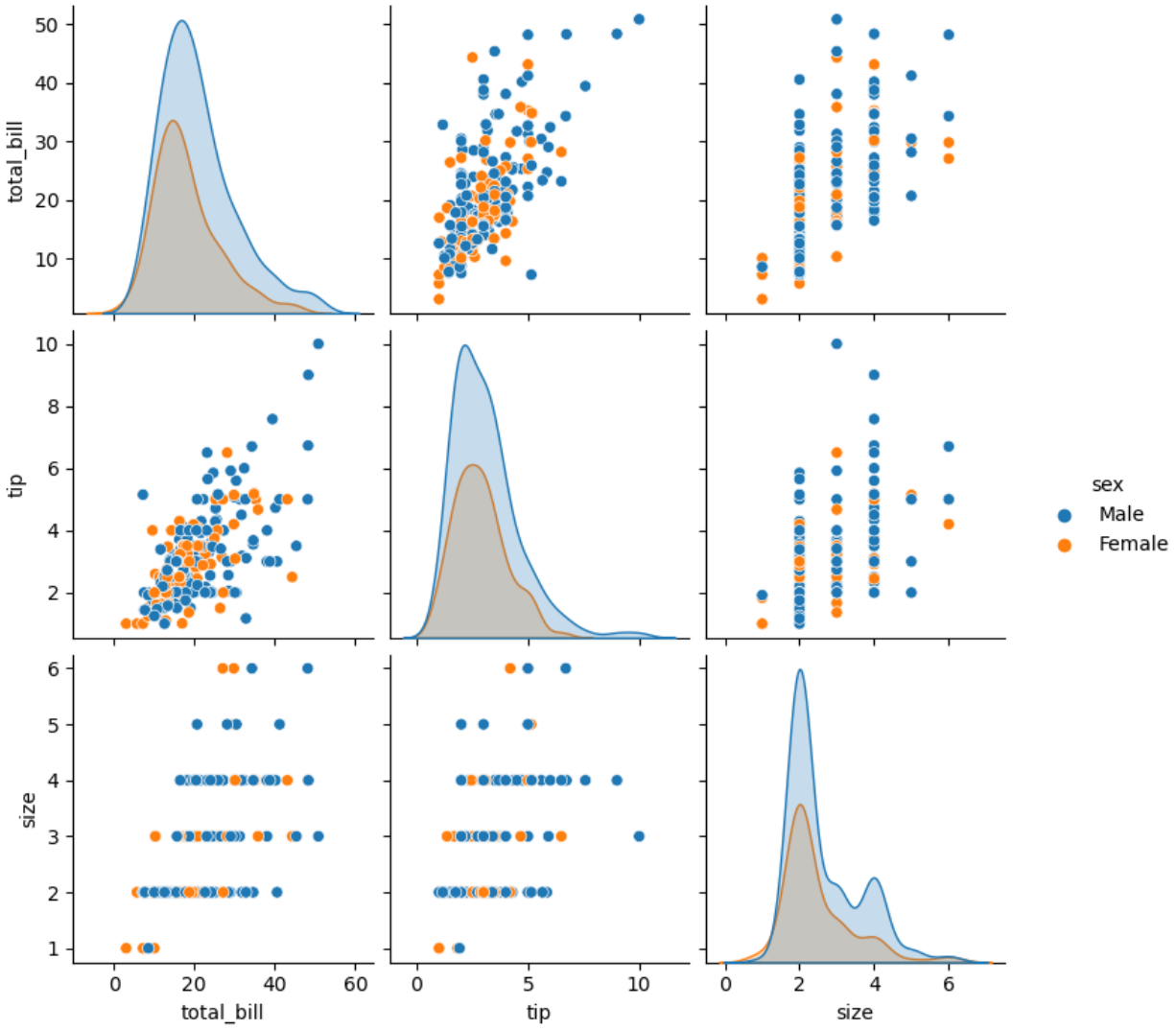
Out[57]:
```
<seaborn.axisgrid.PairGrid at 0x19a46350990>
```



In [58]:
```
sns.pairplot(df, hue = "sex")
```

```
D:\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure l
ayout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[58]:
```
<seaborn.axisgrid.PairGrid at 0x19a46f66dd0>
```



In [59]:
```
df["smoker"].value_counts()
```

Out[59]:
```
smoker
No     151
Yes     93
Name: count, dtype: int64
```

Dist plot

Explanation:

- **Purpose:** Seaborn's `distplot` is used to visualize the distribution of a univariate set of observations.

- **Representation:** Combines a histogram with a kernel density estimate (KDE) to represent the probability density of the data.
- **Insights:** Offers insights into the central tendency and spread of the data.

Key Features:

1. **Histogram:** Displays the distribution of data as bars.
2. **Kernel Density Estimation (KDE):** Smooth curve overlaid on the histogram for a continuous representation.
3. **Customization:** Adjustable parameters for controlling plot appearance.

Usage:

- Ideal for understanding the distribution, skewness, and central tendency of a single variable.

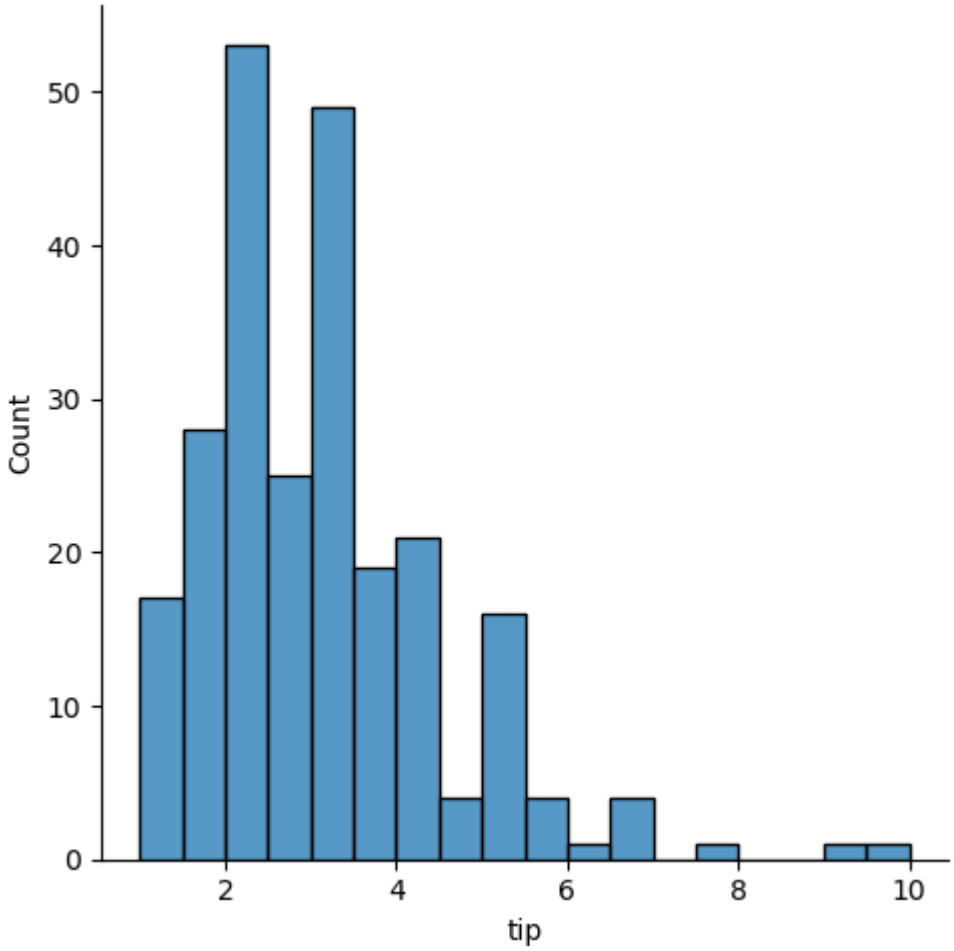
Example Code:

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

# Creating a distribution plot
sns.distplot(df['numeric_variable'], bins=30, kde=True, color='skyblue')
plt.show()
```

In [60]:
```
sns.displot(df["tip"])
```

```
D:\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure l
ayout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

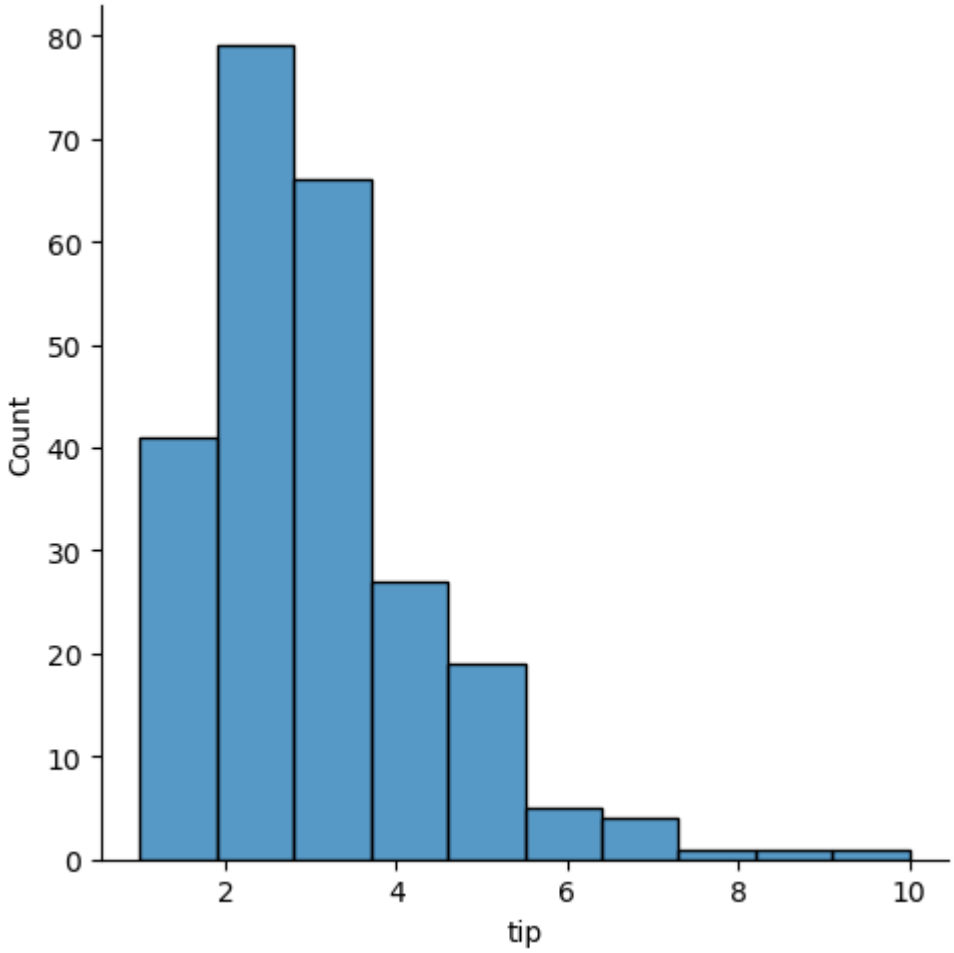
Out[60]:
```
<seaborn.axisgrid.FacetGrid at 0x19a47c58990>
```



In [61]:
```
sns.displot(df["tip"], kde = False, bins = 10)
```

```
D:\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure l
ayout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[61]:
```
<seaborn.axisgrid.FacetGrid at 0x19a47c53b50>
```



Categorical Plots

Explanation:

- **Purpose:** Seaborn provides various categorical plot functions to visualize relationships in categorical data.
- **Representation:** These plots reveal statistical relationships and distributions across different categories.
- **Insights:** Effective for understanding patterns and variations in categorical variables.

Key Categorical Plots:

1. **Bar Plot:**
    - **Purpose:** Displays the mean or other estimator with confidence intervals.
    - **Insights:** Ideal for comparing values of different categories.

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

# Creating a bar plot
sns.barplot(x='category', y='values', data=df)
plt.show()
```

2. **Count Plot:**

- **Purpose:** Shows the counts of observations in each category.
- **Insights:** Useful for visualizing the distribution of categorical data.

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

# Creating a count plot
sns.countplot(x='category', data=df)
plt.show()
```

3. **Box Plot:**

- **Purpose:** Displays the distribution of quantitative data across different categories.
- **Insights:** Reveals the spread and presence of outliers.

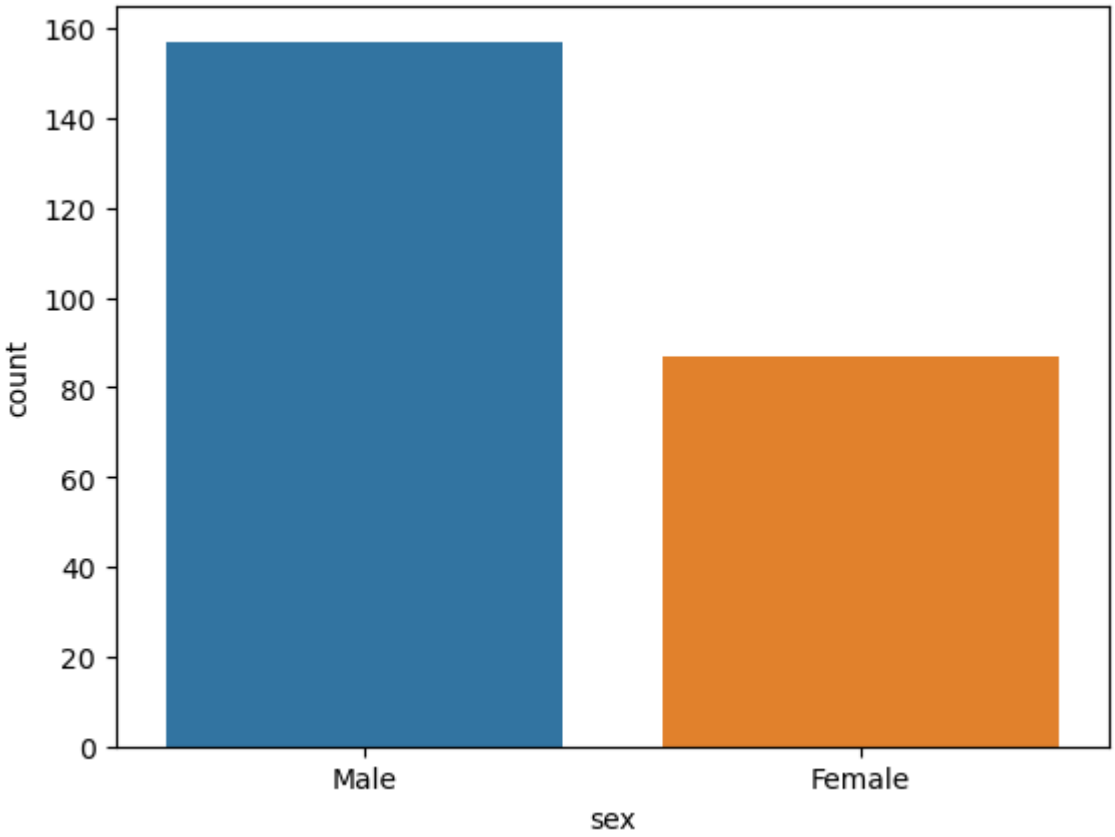
```
import seaborn as sns
import matplotlib.pyplot as plt

# Creating a box plot
sns.boxplot(x='category', y='values', data=df)
plt.show()
```

Usage:

- Effective for comparing and visualizing distributions within different categories.
- Useful for understanding patterns and variations in categorical variables.

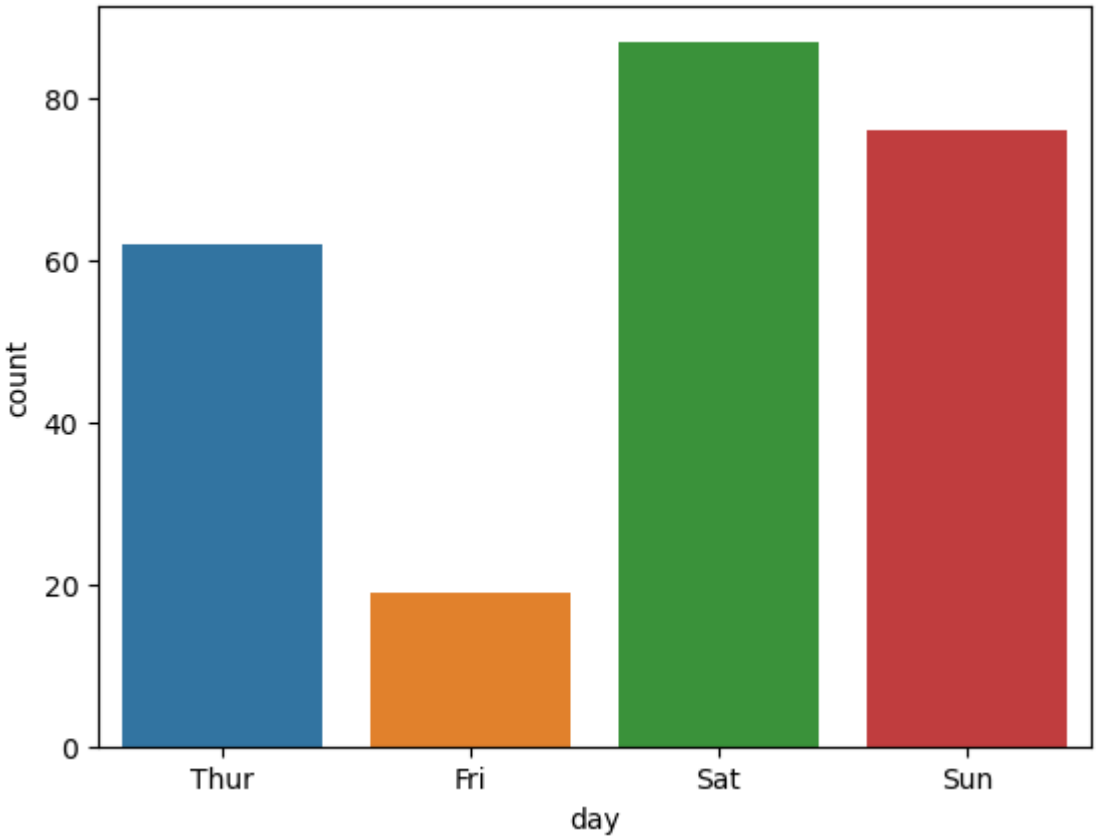
In [63]:
```
#count plot
sns.countplot(x = "sex", data = df)
```

Out[63]: <Axes: xlabel='sex', ylabel='count'>



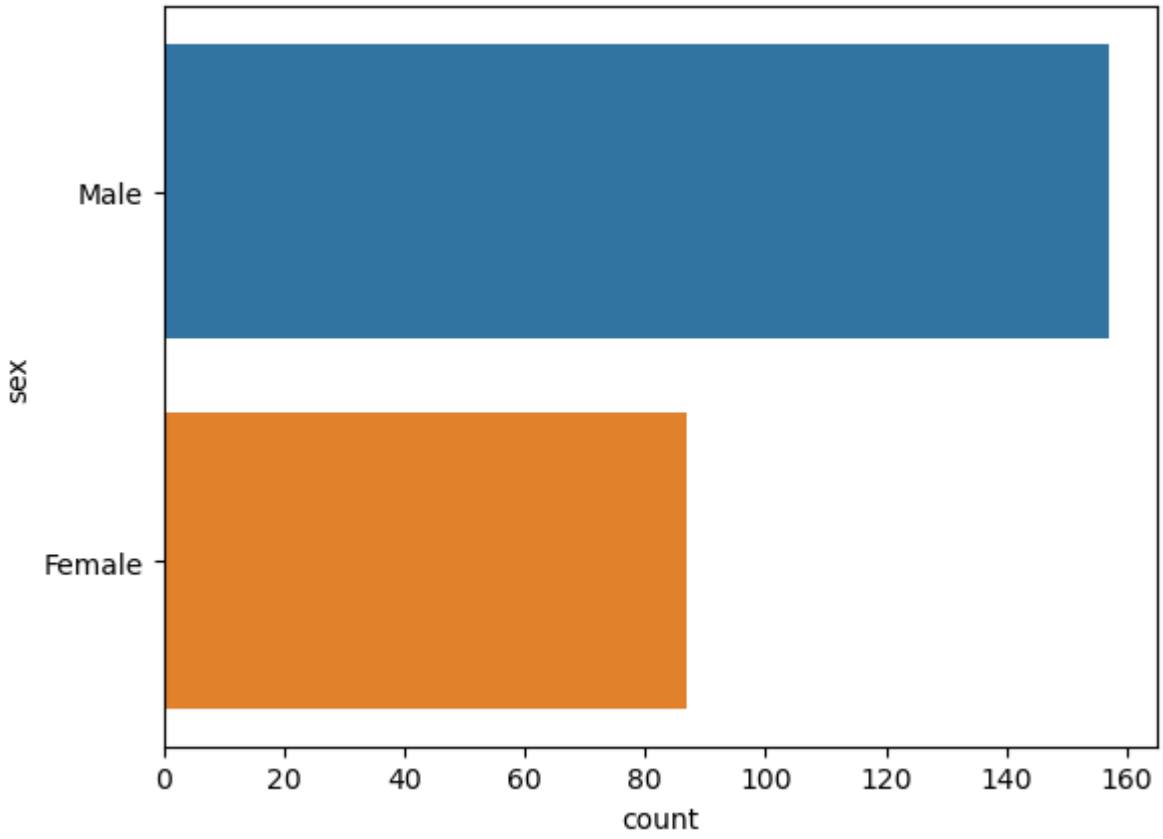
In [64]:
```
sns.countplot(x = "day", data = df)
```

Out[64]: <Axes: xlabel='day', ylabel='count'>



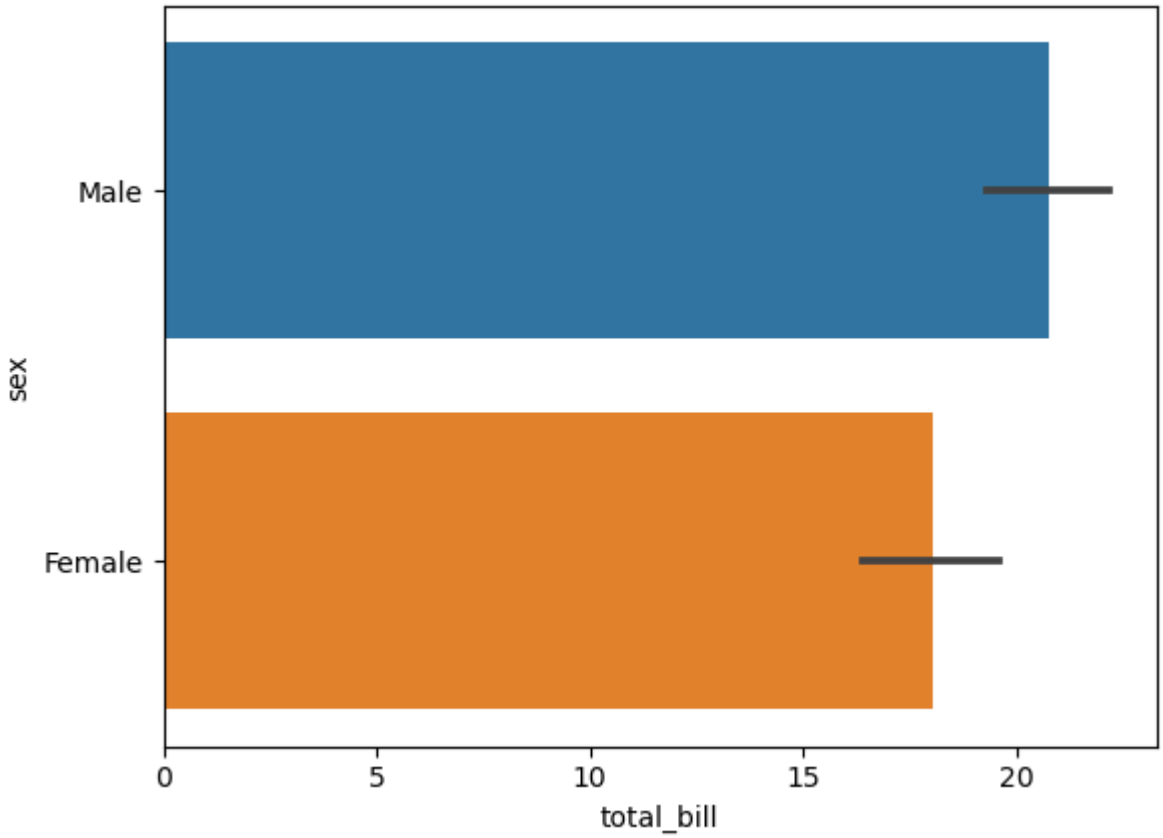
In [66]:
```
sns.countplot(y = "sex", data = df)
```

Out[66]: <Axes: xlabel='count', ylabel='sex'>



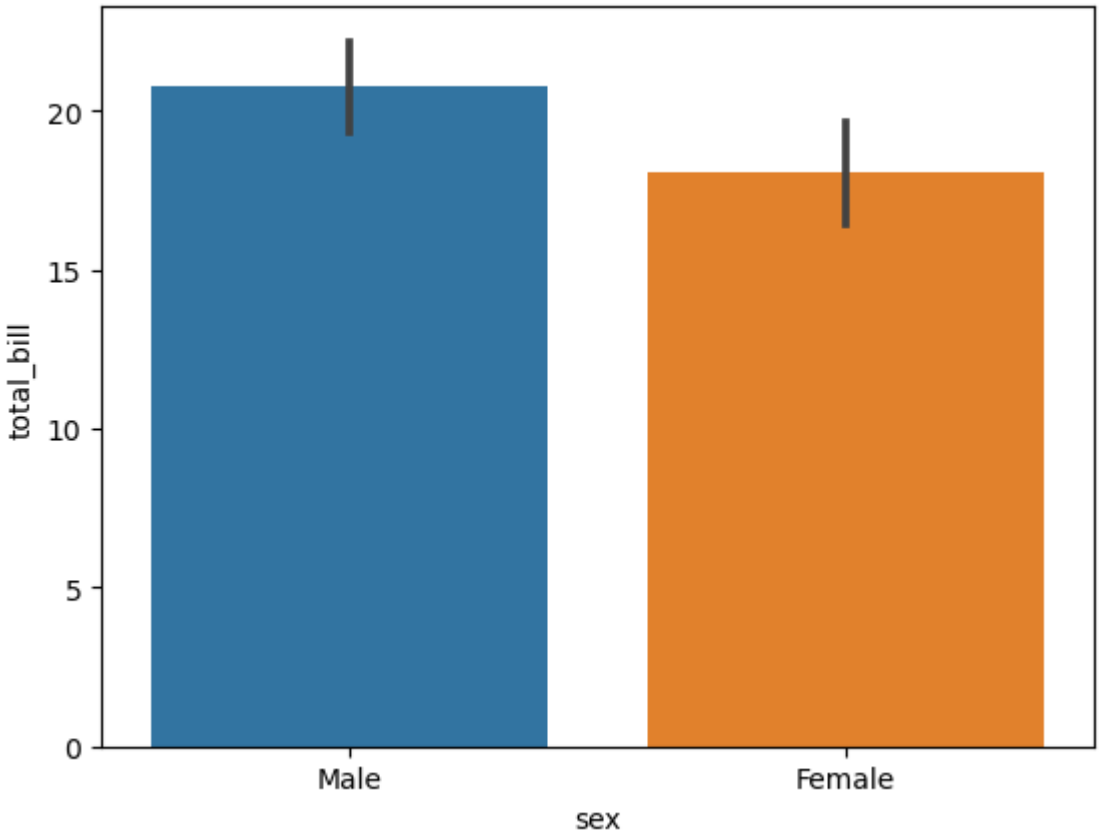
In [67]:
```
#bar plot
sns.barplot(x = "total_bill", y = "sex", data = df)
```

Out[67]:
```
<Axes: xlabel='total_bill', ylabel='sex'>
```



In [68]:
```
sns.barplot(y = "total_bill", x = "sex", data = df)
```

Out[68]:
```
<Axes: xlabel='sex', ylabel='total_bill'>
```



Box plot

Explanation:

- **Purpose:** Seaborn's `boxplot` visualizes the distribution of quantitative data, highlighting key statistics.
- **Representation:** It displays the quartiles, median, and potential outliers in a compact form.
- **Insights:** Useful for comparing the spread and central tendency of data across different categories.

Example Code:

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

# Creating a box plot
sns.boxplot(x="sex", y="age", data=df)
plt.show()
```

Key Features:

1. **Quartiles:** Box represents the interquartile range (IQR) containing 50% of the data.
2. **Median:** Line inside the box indicates the median value.

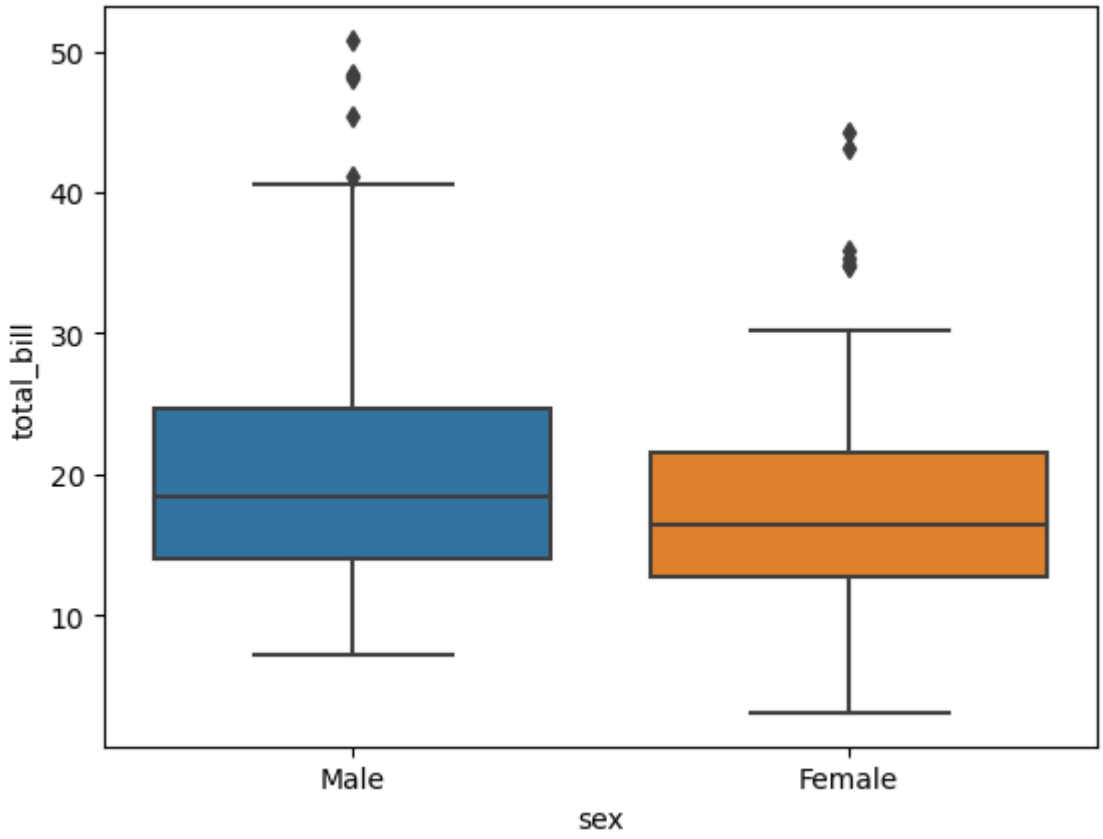
3. **Whiskers:** Extend to the minimum and maximum values within a calculated range.
4. **Outliers:** Dots outside the whiskers denote potential outliers.

Usage:

- Compares the distribution and spread of numerical data across different categories.
- Identifies potential outliers and provides insights into data variability.

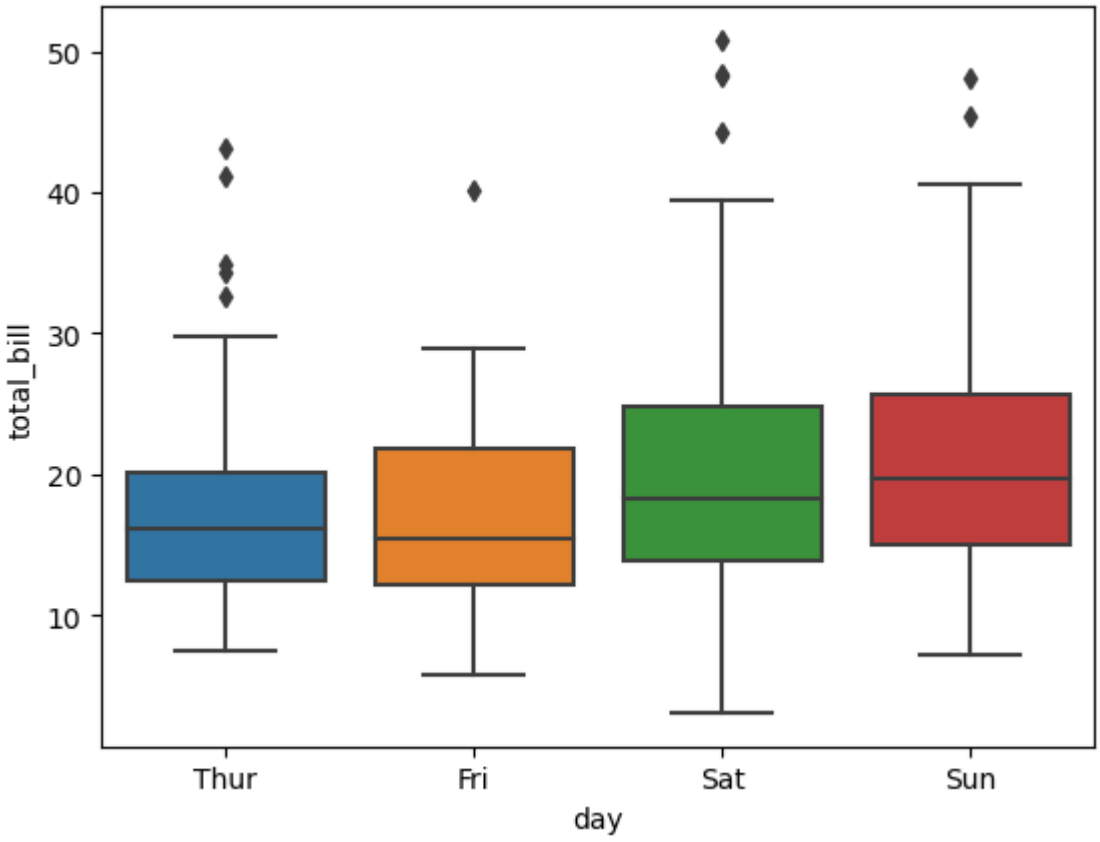
In [72]:
```
sns.boxplot(x = "sex", y = "total_bill", data = df)
```

Out[72]:
```
<Axes: xlabel='sex', ylabel='total_bill'>
```



In [73]:
```
sns.boxplot(x = "day", y = "total_bill", data = df)
```

Out[73]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



Violin Plot

Explanation:

- **Purpose:** Seaborn's `violinplot` combines aspects of a box plot and a kernel density estimate to visualize the distribution of quantitative data across different categories.
- **Representation:** It provides insights into both the central tendency and the probability density of the data.
- **Insights:** Useful for understanding the distribution shape and variations in data.

Example Code:

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

# Assuming 'sex' and 'age' are columns in your DataFrame 'df'
sns.violinplot(x="sex", y="age", data=df)
plt.show()
```

Key Features:

1. **Width of the Plot:** Represents the estimated density of the data.
2. **Interquartile Range (IQR):** Visualized by the thicker part of the plot.

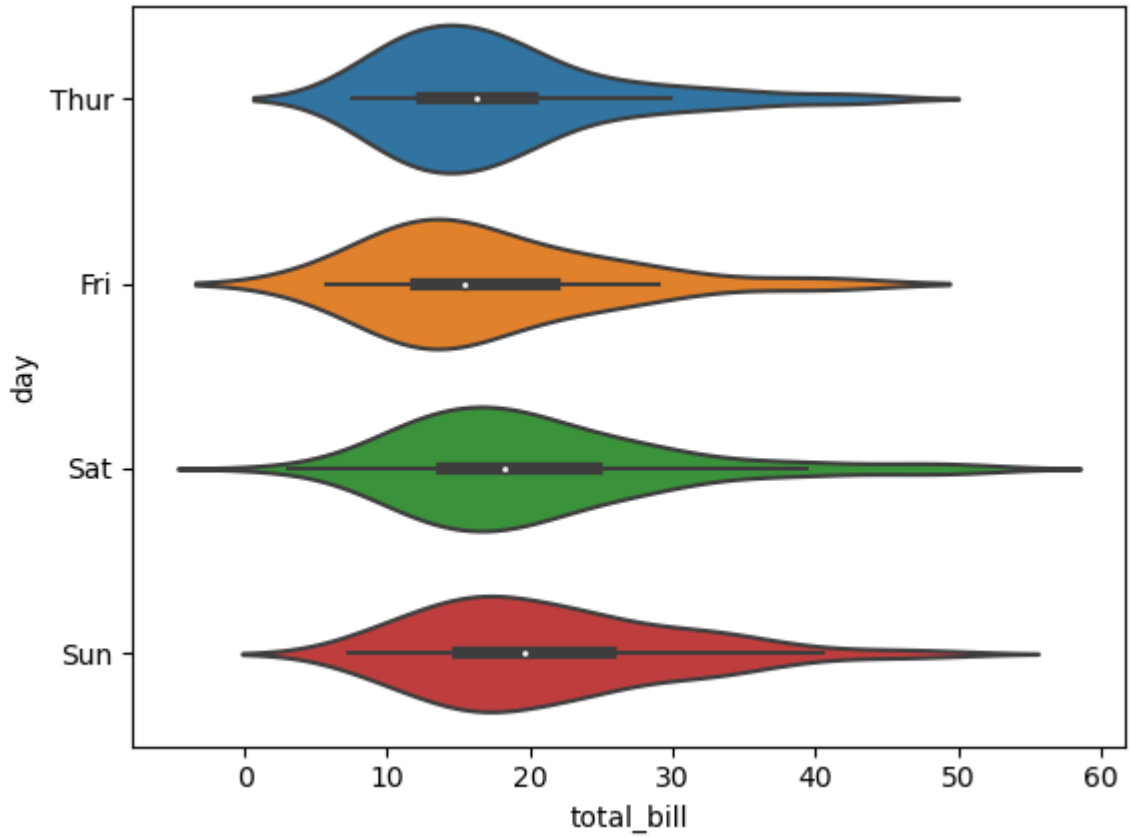
3. **Kernel Density Estimation (KDE):** Provides a smooth representation of data distribution.
4. **Potential Outliers:** Shown as points beyond the plot's whiskers.

Usage:

- Offers a detailed view of data distribution across different categories.
- Suitable for comparing multiple distributions simultaneously.

In [76]:
```
sns.violinplot(x = "total_bill", y = "day", data = df)
```

Out[76]:
```
<Axes: xlabel='total_bill', ylabel='day'>
```



In [80]:
```
#sns.load_dataset("iris")
```

2024.01.17