SEABORN

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on top matplotlib library and is also closely integrated with the data structures from pandas.

Different categories of plot in Seaborn

Relational plots: This plot is used to understand the relation between two variables.

Categorical plots: This plot deals with categorical variables and how they can be visualized.

Distribution plots: This plot is used for examining univariate and bivariate distributions

Univariate = 1 variable → what is happening?

Bivariate = 2 variables → how are they related?

Regression plots: The regression plots in Seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.

Matrix plots: A matrix plot is an 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.

In [5]:
```
pip install seaborn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: seaborn in c:\users\mohap\appdata\roaming\python\python312\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\mohap\appdata\roaming\python\python312\site-packages
(from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from sea
born) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\mohap\appdata\roaming\python\python312\site-packag
es (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mohap\appdata\roaming\python\python312\site-packages (fro
m matplotlib!=3.6.1,>=3.4->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mohap\appdata\roaming\python\python312\site-packages (fr
om matplotlib!=3.6.1,>=3.4->seaborn) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mohap\appdata\roaming\python\python312\site-packages (fr
om matplotlib!=3.6.1,>=3.4->seaborn) (1.4.7)
Requirement already satisfied: packaging>=20.0 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from matpl
otlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mohap\appdata\roaming\python\python312\site-packages (fro
m matplotlib!=3.6.1,>=3.4->seaborn) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\mohap\appdata\roaming\python\python312\site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from pa
ndas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from
pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\mohap\appdata\roaming\python\python312\site-packages (from python
-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Histplot: Seaborn Histplot is used to visualize the univariate set of distributions(single variable). It plots a histogram, with some other variations like kdeplot and rugplot. The Histplot function takes several arguments but the important ones are

data: This is the array, series, or dataframe that you want to visualize. It is a required parameter.

x: This specifies the column in the data to use for the histogram. If your data is a dataframe, you can specify the column by name.

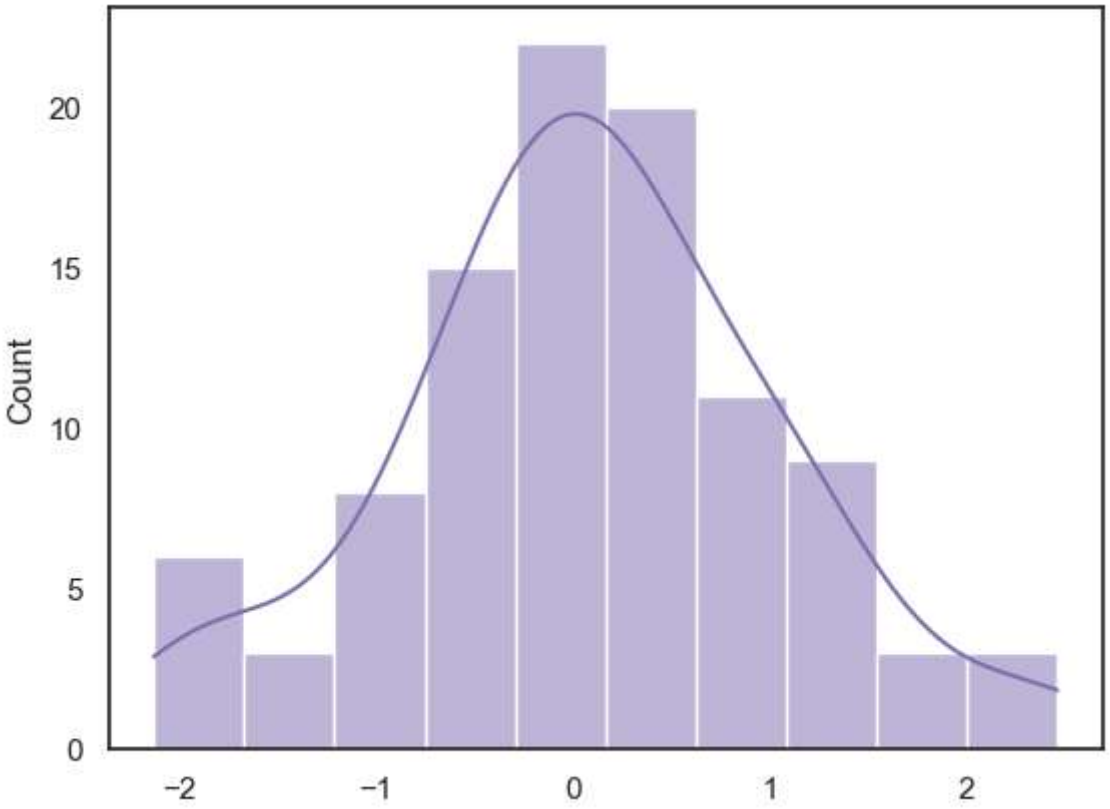
y: This specifies the column in the data to use for the histogram when you want to create a bivariate histogram. By default, it is set to None, meaning that a univariate histogram will be plotted.

bins: This specifies the number of bins to use when dividing the data into intervals for plotting. By default, it is set to "auto", which uses an algorithm to determine the optimal number of bins.

kde: This parameter controls whether to display a kernel density estimate (KDE) of the data in addition to the histogram. By default, it is set to False, meaning that a KDE will not be plotted.

In [8]:
```
import numpy as np
import seaborn as sns

sns.set(style="white")

# Generate a random univariate dataset
rs = np.random.RandomState(10)  # 10 -seed
d = rs.normal(size=100)

# Plot a simple histogram and kde
sns.histplot(d, kde=True, color="m")

# From a normal (Gaussian) distribution

#Mean = 0, Standard deviation = 1 (default)

#📌 This is a univariate dataset (single variable)
```

Out[8]:
```
<Axes: ylabel='Count'>
```



Distplot: Seaborn distplot is used to visualize the univariate set of distributions(Single features) and plot the histogram with some other variations like kdeplot and rugplot.

The function takes several parameters, but the most important ones are:

a: This is the array, series, or list of data that you want to visualize. It is a required parameter.

bins: This specifies the number of bins to use when dividing the data into intervals for plotting. By default, it is set to "auto", which uses an algorithm to determine the optimal number of bins.

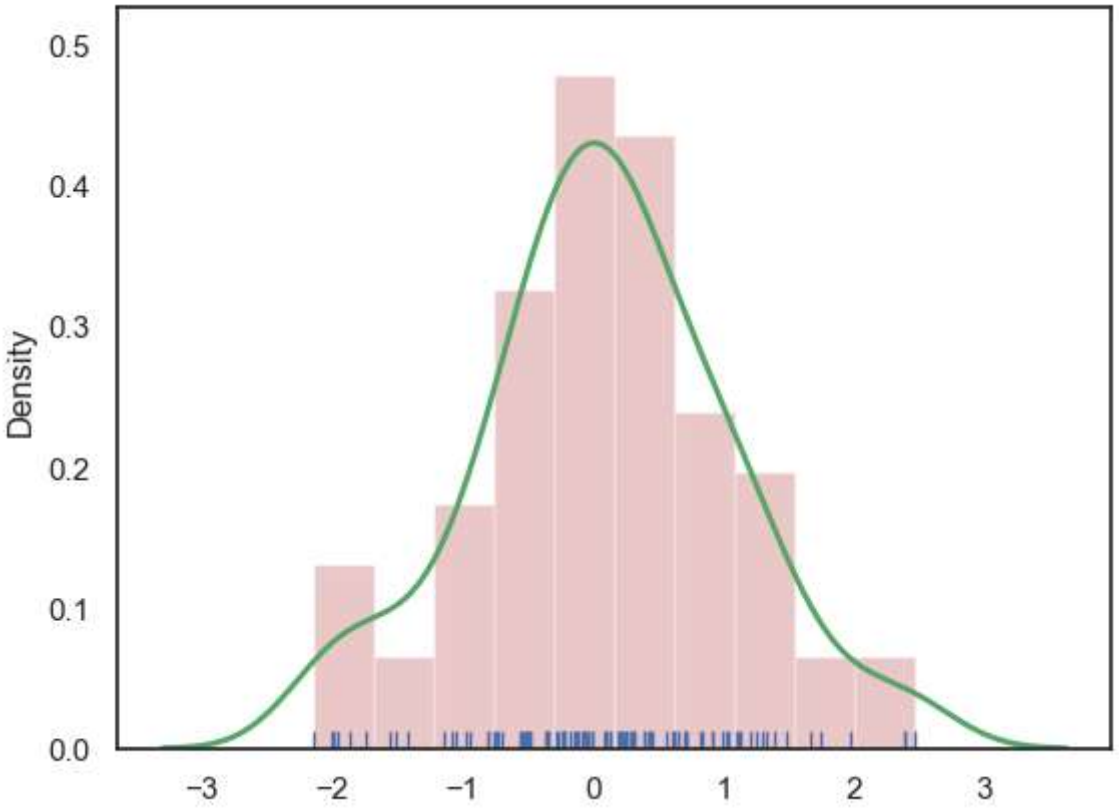
kde: This parameter controls whether to display a kernel density estimate (KDE) of the data in addition to the histogram. By default, it is set to True, meaning that a KDE will be plotted.

hist: This parameter controls whether to display the histogram of the data. By default, it is set to True, meaning that a histogram will be plotted.

In [11]:
```
import numpy as np
import seaborn as sns

sns.set(style="white")  # background color

# Generate a random univariate dataset
rs = np.random.RandomState(10)
d = rs.normal(size=100)

# Define the colors to use
colors = ["r", "g", "b"]  #red,green,blue

# Plot a histogram with multiple colors
sns.distplot(d, kde=True, hist=True, bins=10,
             rug=True,hist_kws={"alpha": 0.3,   #vertical single line -rug
                                "color": colors[0]},  # red
             kde_kws={"color": colors[1], "lw": 2},  #green ,lw-line width
             rug_kws={"color": colors[2]})  # blue
```

```
C:\Users\mohap\AppData\Local\Temp\ipykernel_2404\2207689057.py:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(d, kde=True, hist=True, bins=10,
```

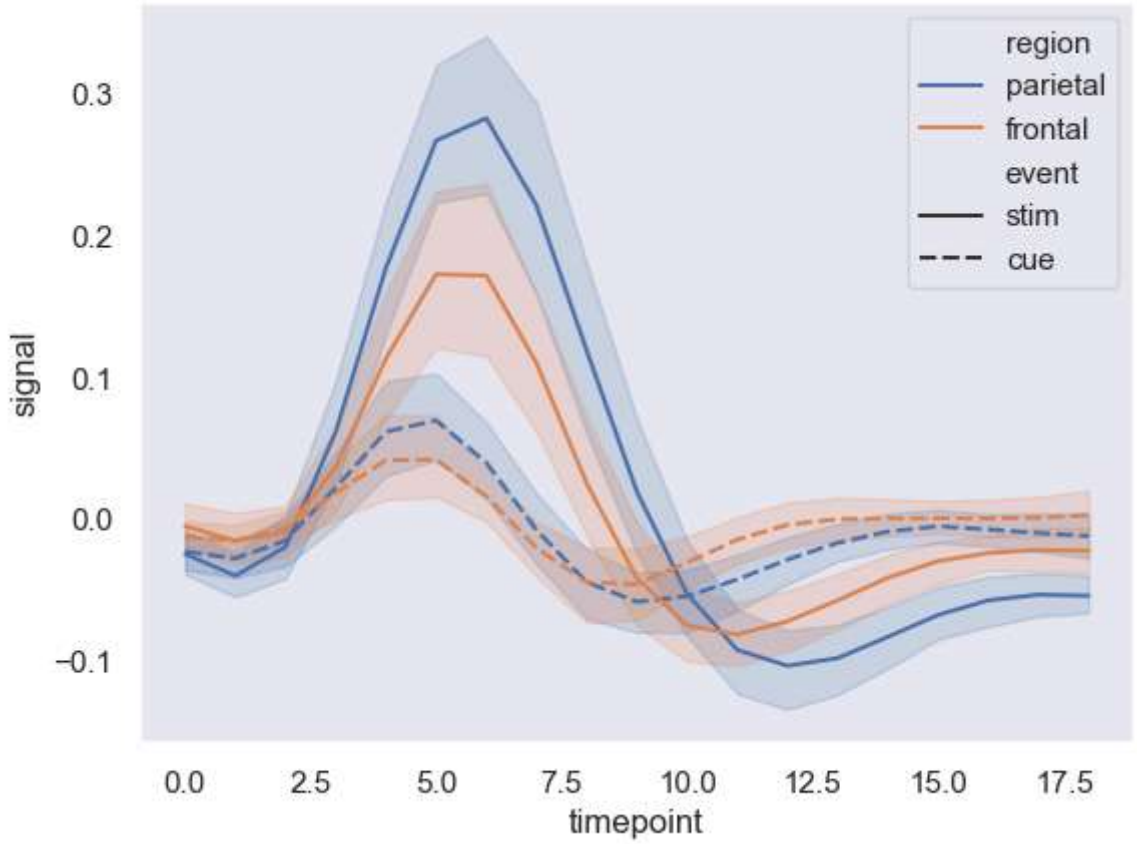
Out[11]: <Axes: ylabel='Density'>



Lineplot: The line plot is one of the most basic plots in the seaborn library. This plot is mainly used to visualize the data in the form of some time series, i.e. in a continuous manner.

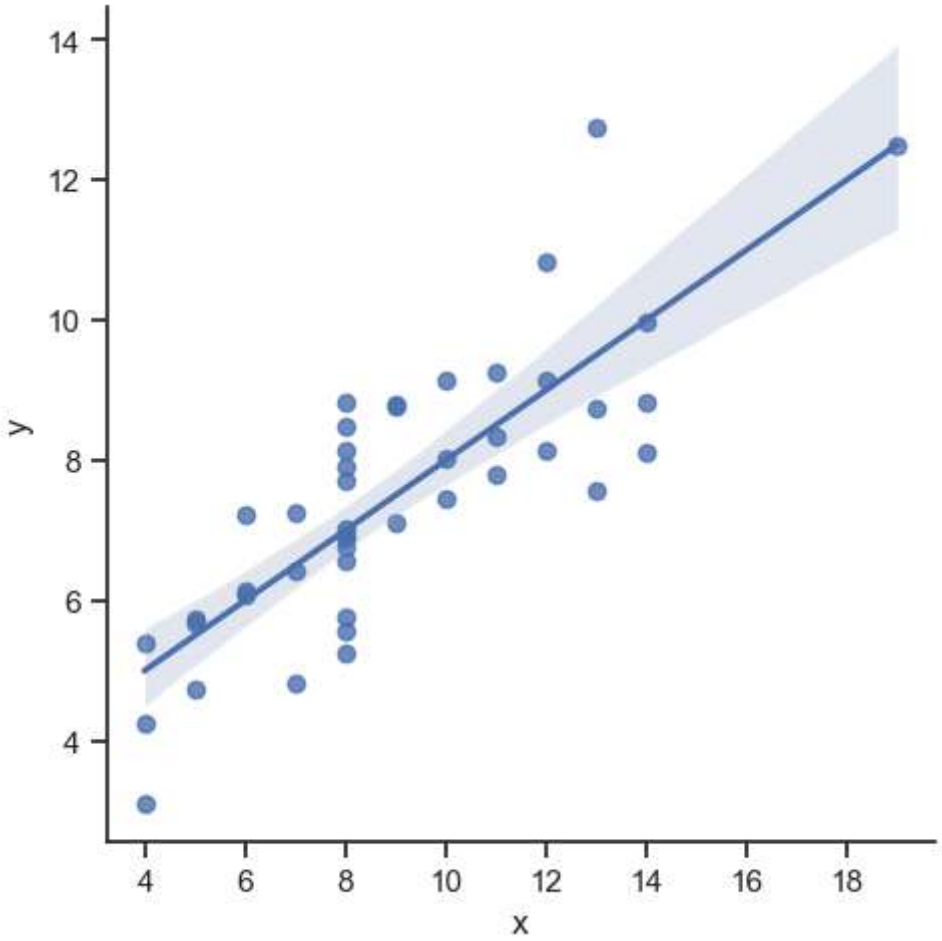
In [16]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="dark")
fmri = sns.load_dataset("fmri")

# Plot the responses for different\
# events and regions
sns.lineplot(x="timepoint",
```

```
             y="signal",
             hue="region",
             style="event",
             data=fmri)
plt.show()
```



In [18]:
```
import seaborn as sns

sns.set(style="ticks")

# Loading the dataset
df = sns.load_dataset("anscombe")

# Show the results of a linear regression
sns.lmplot(x="x", y="y", data=df)
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x295cc8deea0>



Key Features

Comes with built-in datasets like iris, tips, etc.

Provides statistical plots such as boxplots, violin plots, swarm plots, etc.

Handles categorical data visualization better than Matplotlib.

Supports aesthetic customization (themes, color palettes, styles).

Simplifies working with DataFrames by auto-labeling axes.

1. Strip Plot

A strip plot is a categorical scatter plot where data points are plotted along one categorical axis. It is useful for visualizing the distribution of values but may suffer from overlapping points.

Applications

Used when we want to visualize raw distribution of numerical data across categories.

Helpful for detecting clusters or general spread of values.

Advantages

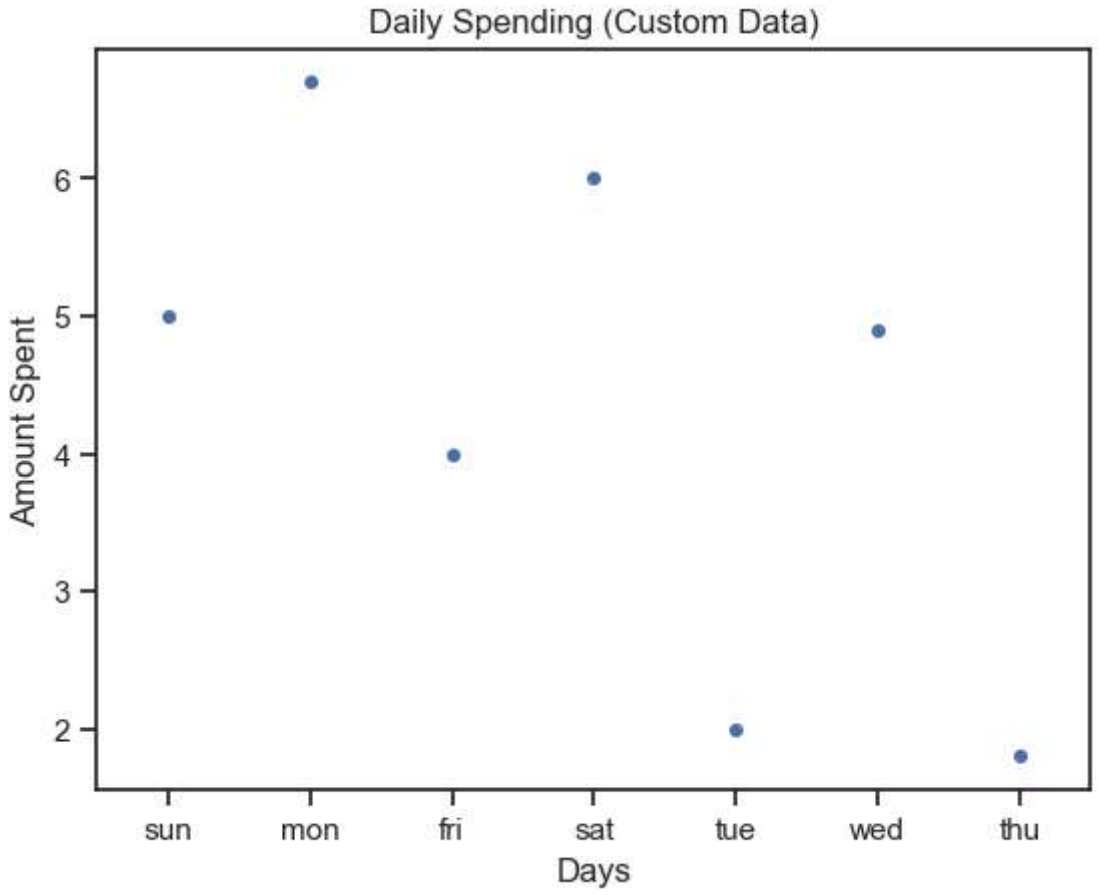
Simple and easy to interpret.

Shows individual data points clearly.

Limitations

Overlapping points may cause loss of clarity in dense datasets.

In [24]:
```
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=x, y=y)
ax.set(xlabel='Days', ylabel='Amount Spent')
plt.title('Daily Spending (Custom Data)')
plt.show()
```



2. Swarm Plot

A swarm plot is similar to a strip plot, but points are arranged to avoid overlap. This ensures all data points are visible, making it more informative.

Applications

Useful when dataset is small/medium and we want to show all observations.

Comparing sub-groups clearly without stacking.

Advantages

Prevents overlap of data points.

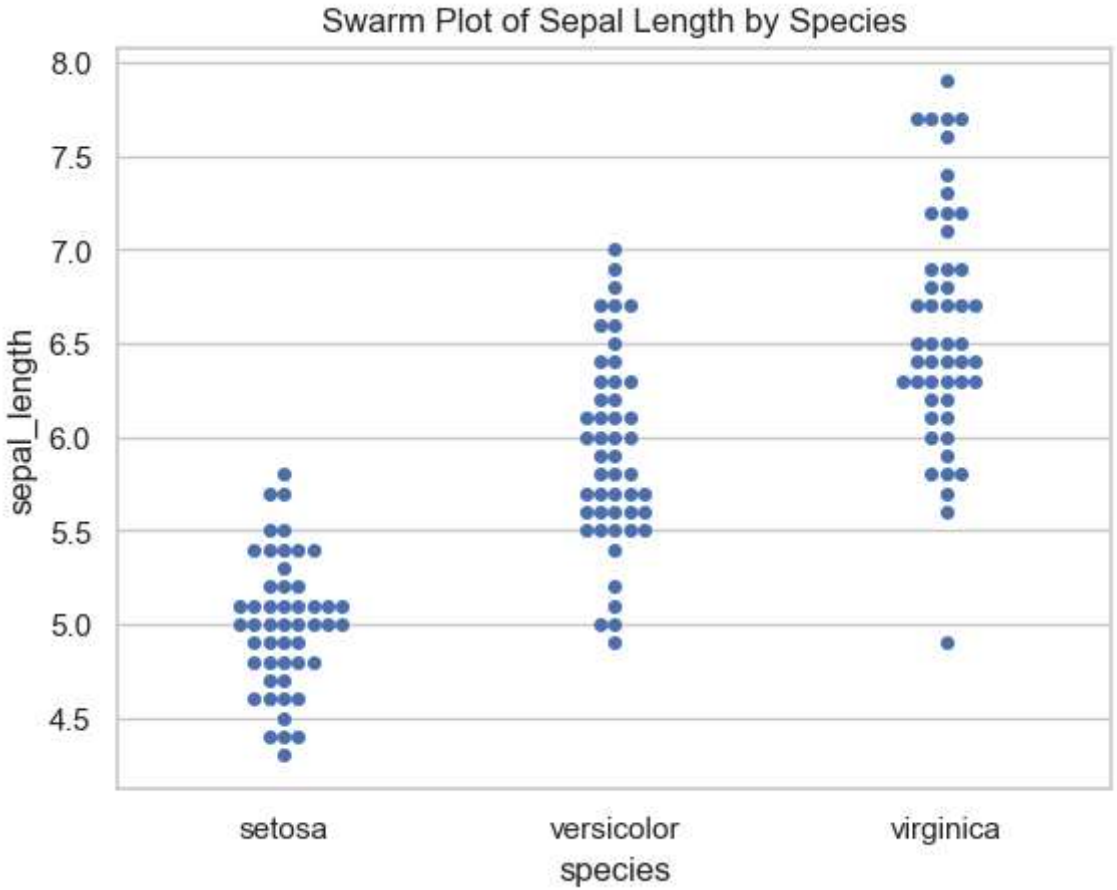
Provides clearer visual insight than strip plot.

Limitations

Can be slow for large datasets.

May look cluttered when categories have thousands of points.

In [27]:
```
sns.set(style="whitegrid")
iris = sns.load_dataset("iris")
sns.swarmplot(x="species", y="sepal_length", data=iris)
plt.title("Swarm Plot of Sepal Length by Species")
plt.show()
```



3. Bar Plot

A bar plot shows the average (by default mean) of a numerical variable across categories. It can use different estimators (mean, median, std, etc.) for aggregation.

Applications

Comparing average values across categories.

Displaying results of group-by operations visually.

ADVANTAGES:

Flexible can use different statistical functions.

Limitations:

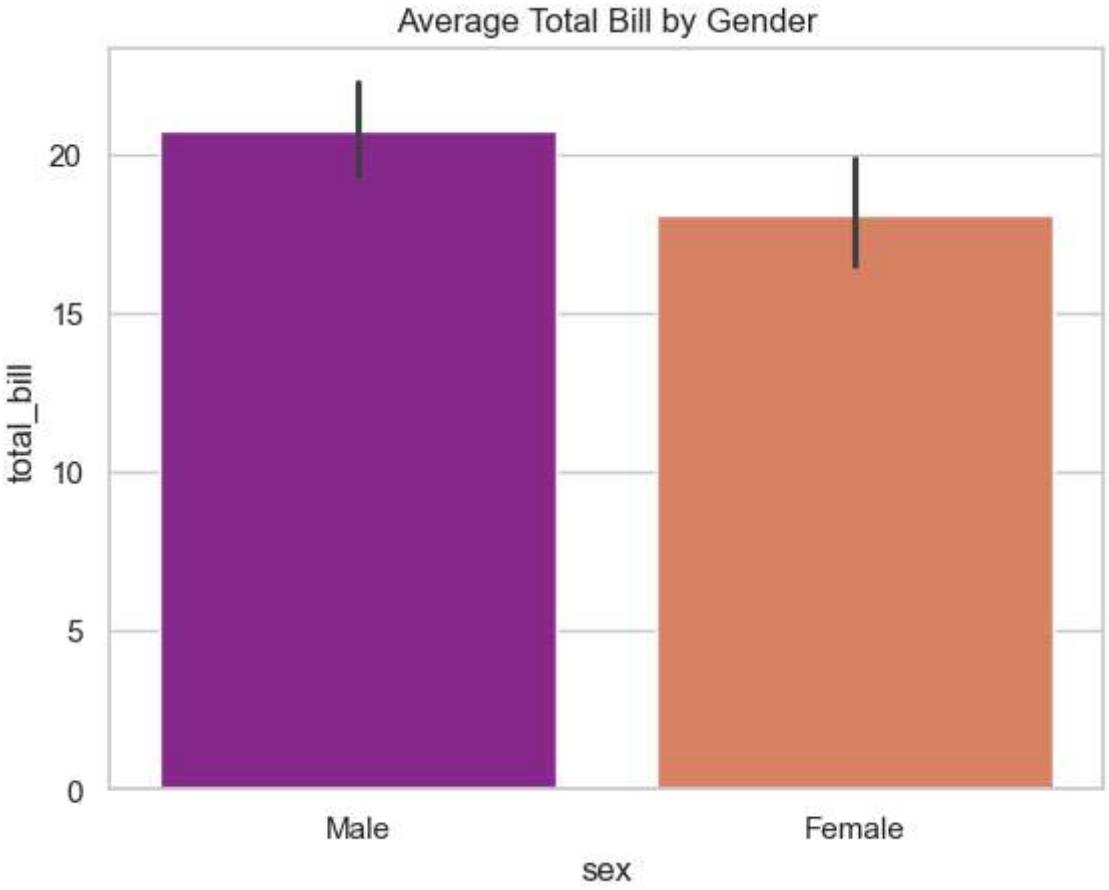
Does not show individual data distribution.

Can hide variability when using only mean.

==========

In [32]:
```
tips = sns.load_dataset("tips")
sns.barplot(x="sex", y="total_bill", data=tips, palette="plasma")
plt.title("Average Total Bill by Gender")
plt.show()
```

```
C:\Users\mohap\AppData\Local\Temp\ipykernel_2404\1241397627.py:2: FutureWarning:

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

  sns.barplot(x="sex", y="total_bill", data=tips, palette="plasma")
```



4. Count Plot

A count plot simply counts the occurrences of each category. It is like a histogram for categorical variables.

Applications

Checking frequency distribution of categorical values.

Advantages

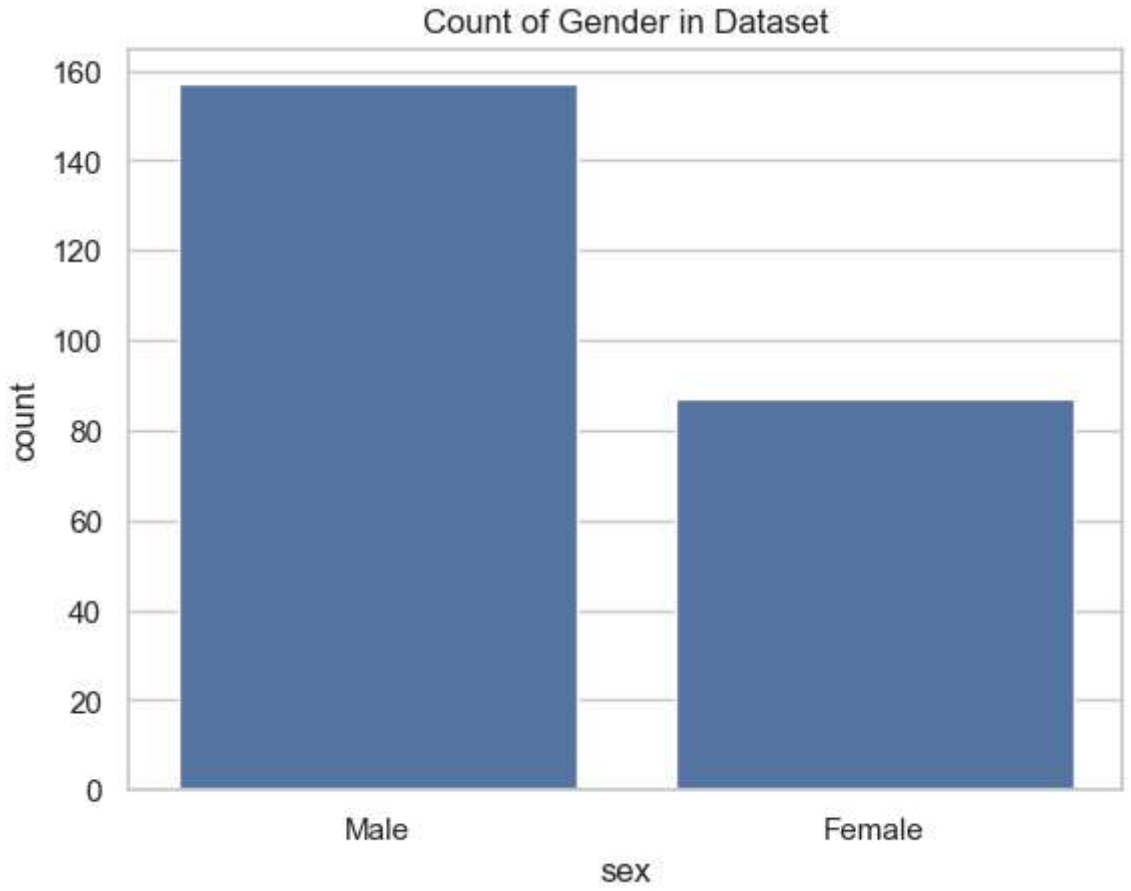
Very simple and quick to interpret.

No need for numerical data, only categorical required.

Limitations

Cannot display numerical spread inside categories.

In [35]:
```
tips = sns.load_dataset("tips")
sns.countplot(x="sex", data=tips)
plt.title("Count of Gender in Dataset")
plt.show()
```



5. Box Plot

A box plot (or whisker plot) summarizes numerical data using quartiles, median and outliers. It helps in detecting variability and spread.

Applications

Detecting outliers.

Comparing spread of distributions across categories.

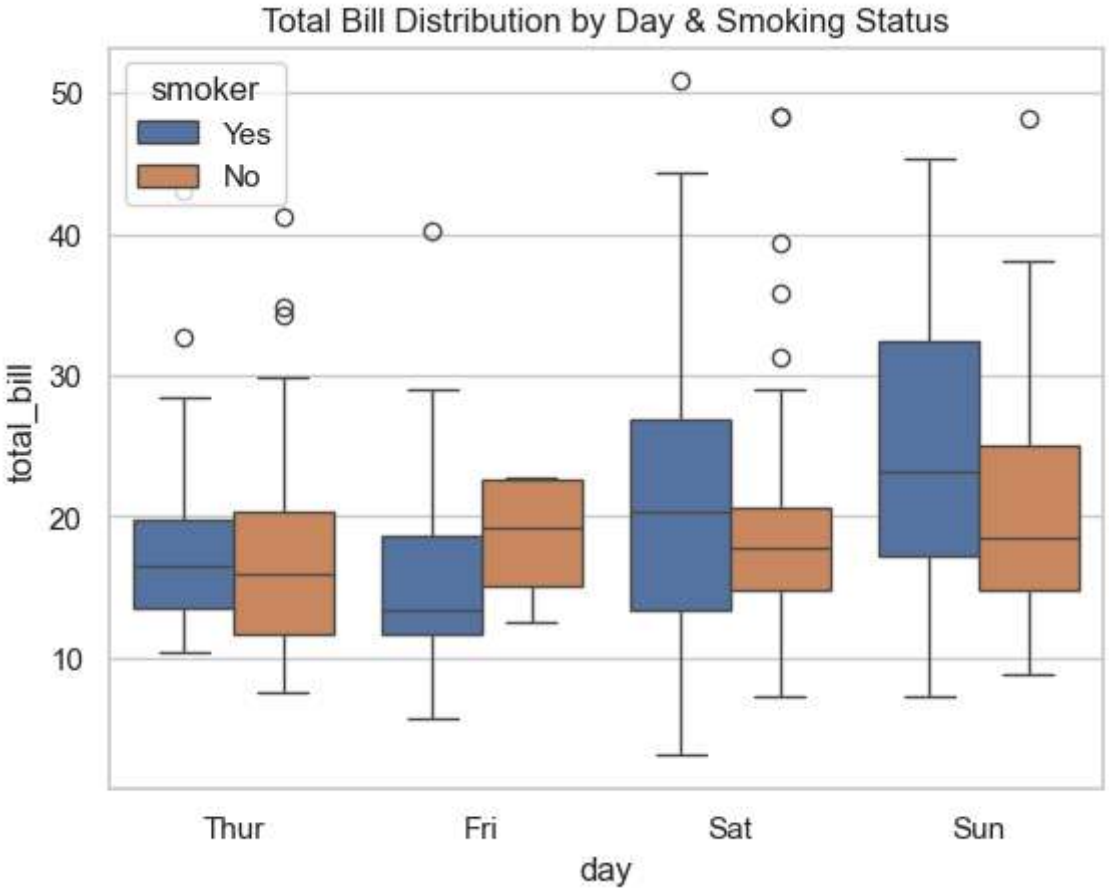
Advantages

Useful for large datasets.

Limitations

Does not show exact data distribution shape.

In [38]:
```
tips = sns.load_dataset("tips")
sns.boxplot(x="day", y="total_bill", data=tips, hue="smoker")
plt.title("Total Bill Distribution by Day & Smoking Status")
plt.show()
```



6. Violin Plot

A violin plot combines a box plot with a density plot, showing both summary stats and distribution shape.

Applications

Comparing distributions more deeply than boxplot.

Helpful for detecting multimodal distributions.

Advantages

Shows both summary statistics and data distribution.

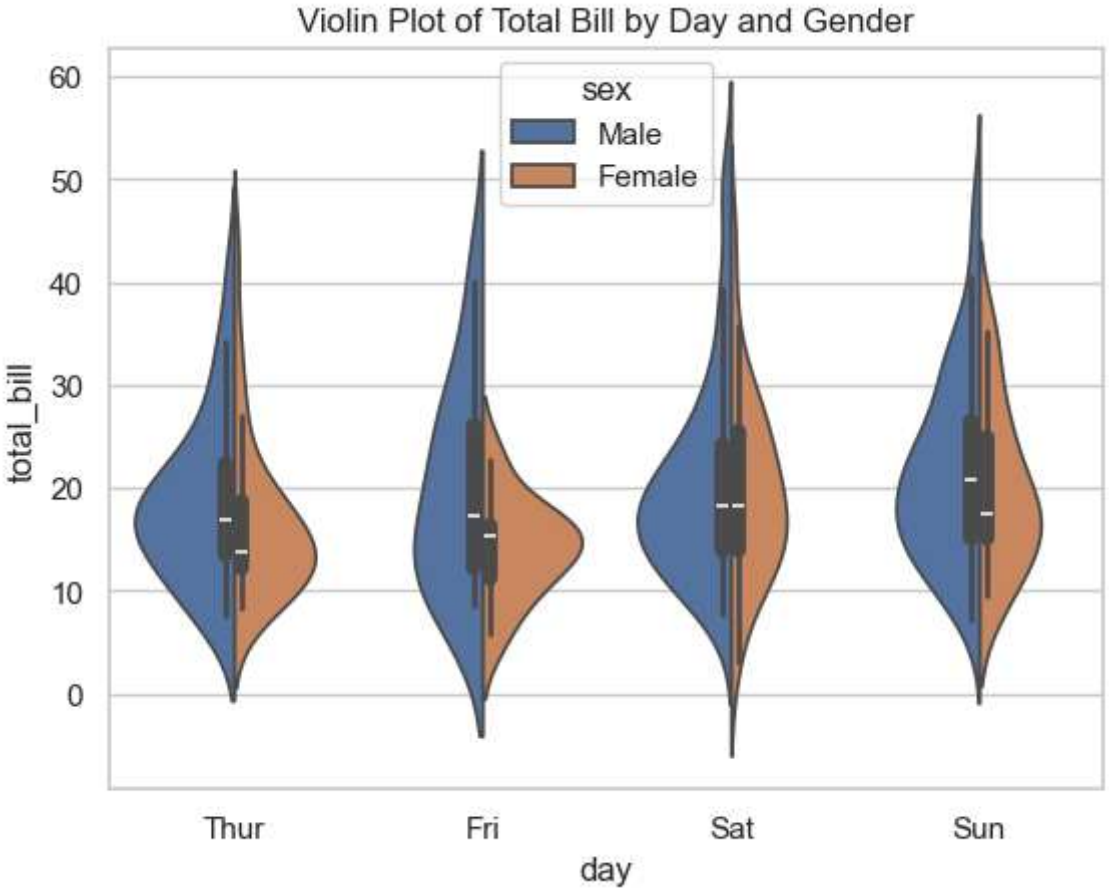
Easier to see differences in distribution shapes.

Limitations

Can be harder to interpret for beginners.

May be misleading if sample size is small.

In [41]:
```
tips = sns.load_dataset("tips")
sns.violinplot(x="day", y="total_bill", data=tips, hue="sex", split=True)
plt.title("Violin Plot of Total Bill by Day and Gender")
plt.show()
```



7. Strip Plot with Hue

This is an enhanced strip plot where categories are further divided using hue. It allows comparing multiple sub-groups within a category.

hue is used to split data into groups and color them differently in a plot

Applications

Comparing subgroups inside categories.

Visualizing interaction between two categorical variables.

Advantages

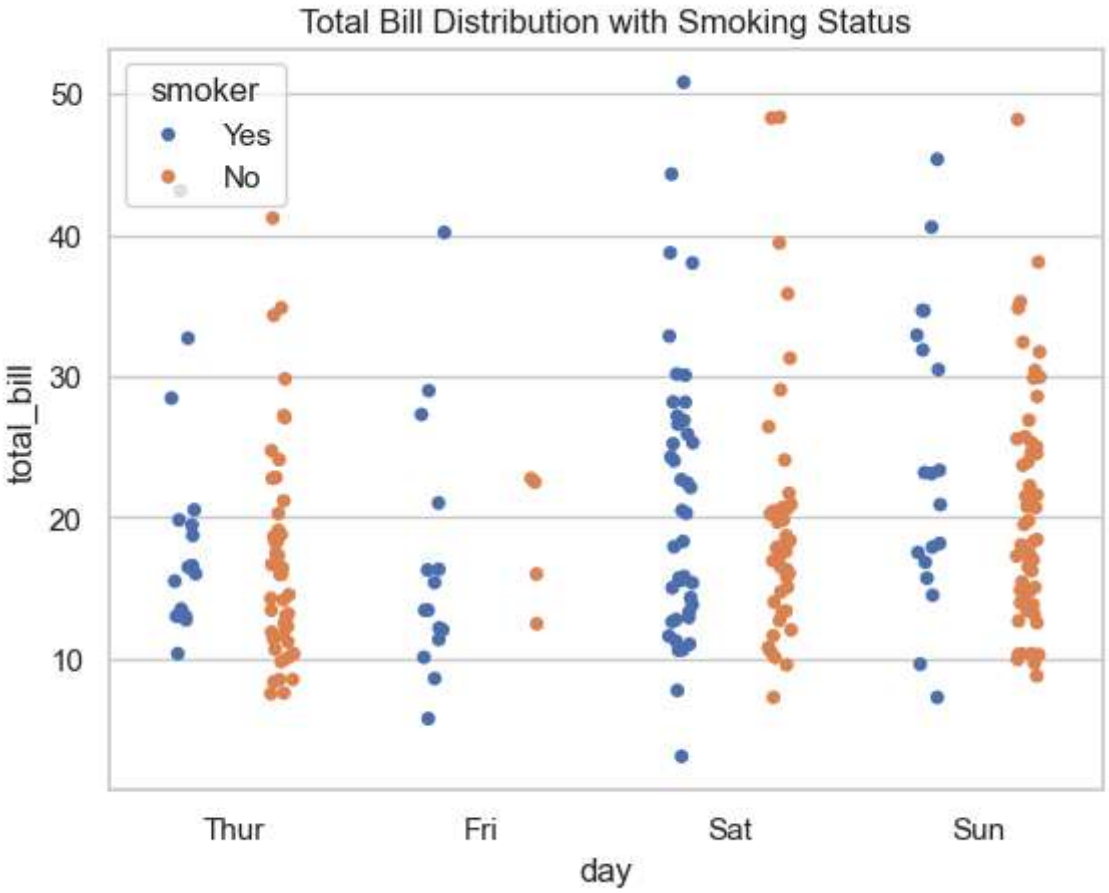
Adds extra dimension to strip plot.

Useful for multivariate visualization.

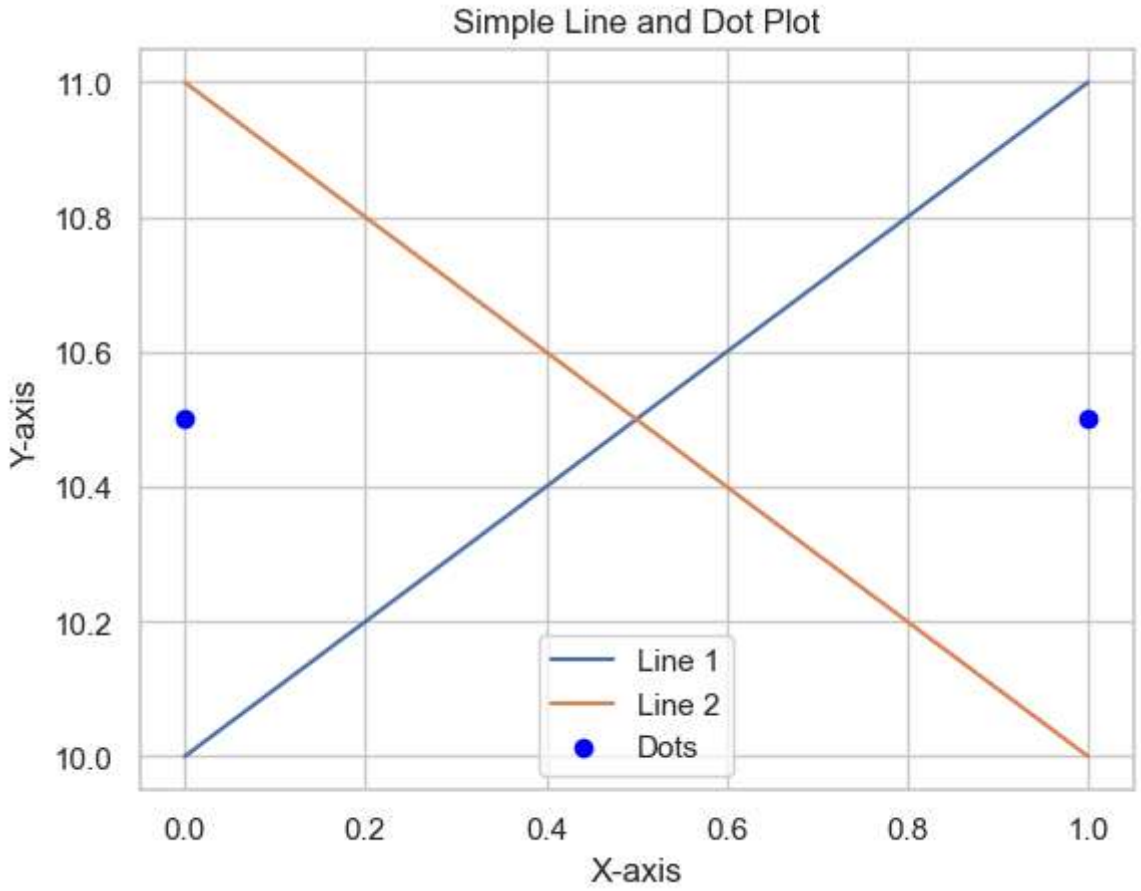
Limitations

Overlap issue exists.

In [54]:
```
tips = sns.load_dataset("tips")
sns.stripplot(x="day", y="total_bill", data=tips,
              jitter=True, hue="smoker", dodge=True)
plt.title("Total Bill Distribution with Smoking Status")
plt.show()
#jitter=True
#Adds small random horizontal movement #Prevents points from overlapping
```

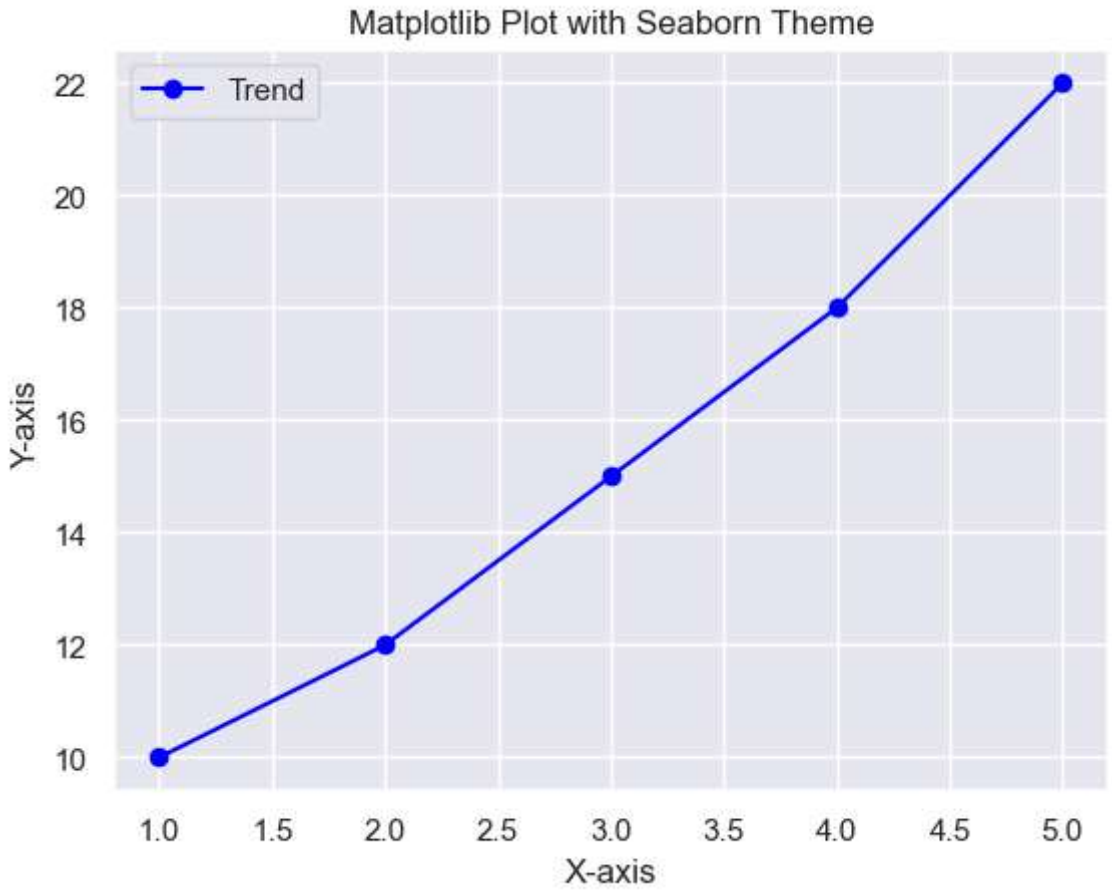


In [60]:
```
import matplotlib.pyplot as plt

plt.plot([0, 1], [10, 11], label='Line 1')
plt.plot([0, 1], [11, 10], label='Line 2')
plt.scatter([0, 1], [10.5, 10.5], color='blue', marker='o', label='Dots')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Simple Line and Dot Plot')
plt.legend()
plt.show()
```



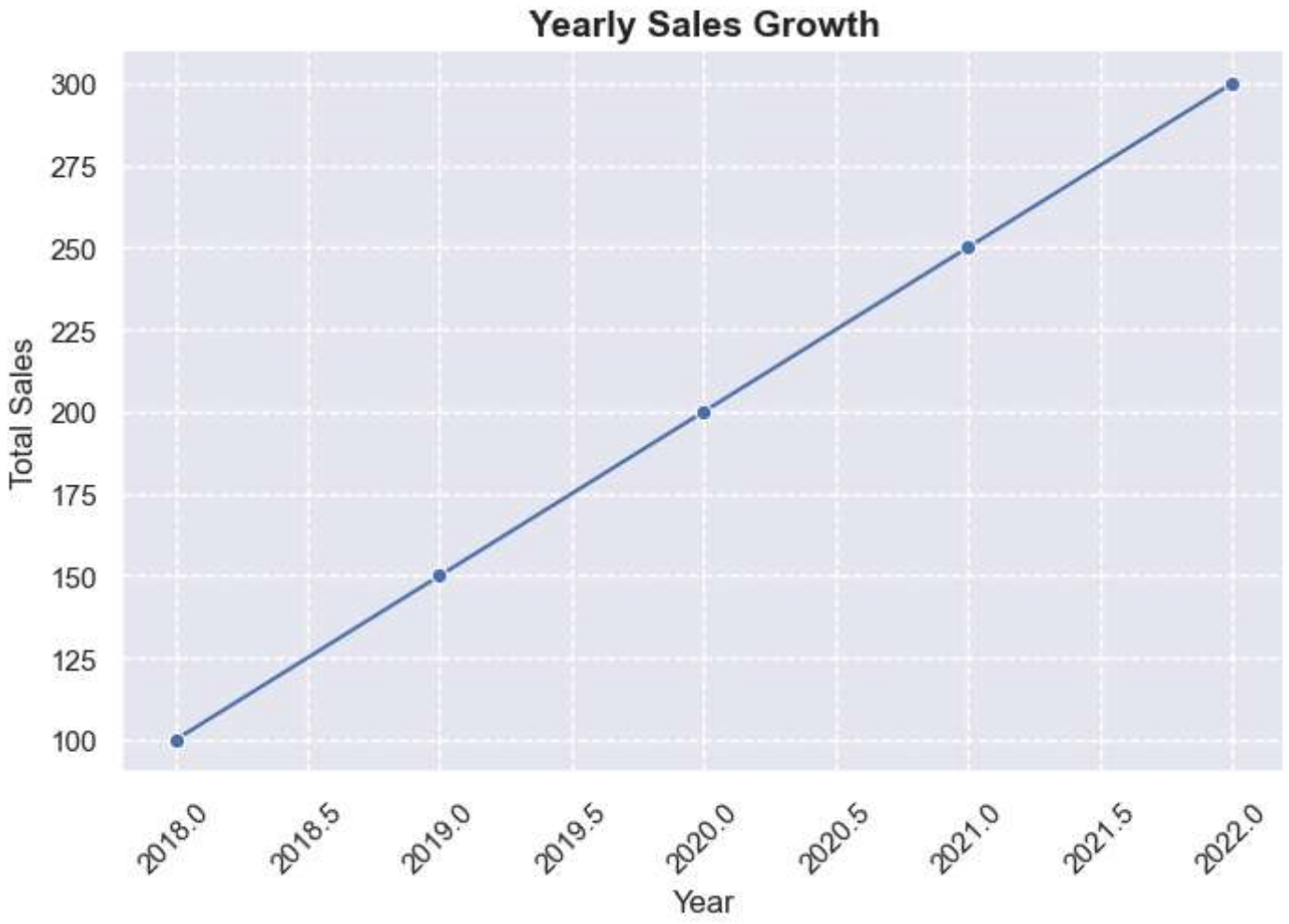
In [62]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Apply Seaborn theme
sns.set_theme(style="darkgrid")

# Creating a simple Matplotlib plot
x = [1, 2, 3, 4, 5]
y = [10, 12, 15, 18, 22]

plt.plot(x, y, marker='o', linestyle='-', color='blue', label="Trend")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Matplotlib Plot with Seaborn Theme")
```

```
plt.legend()
plt.show()
```



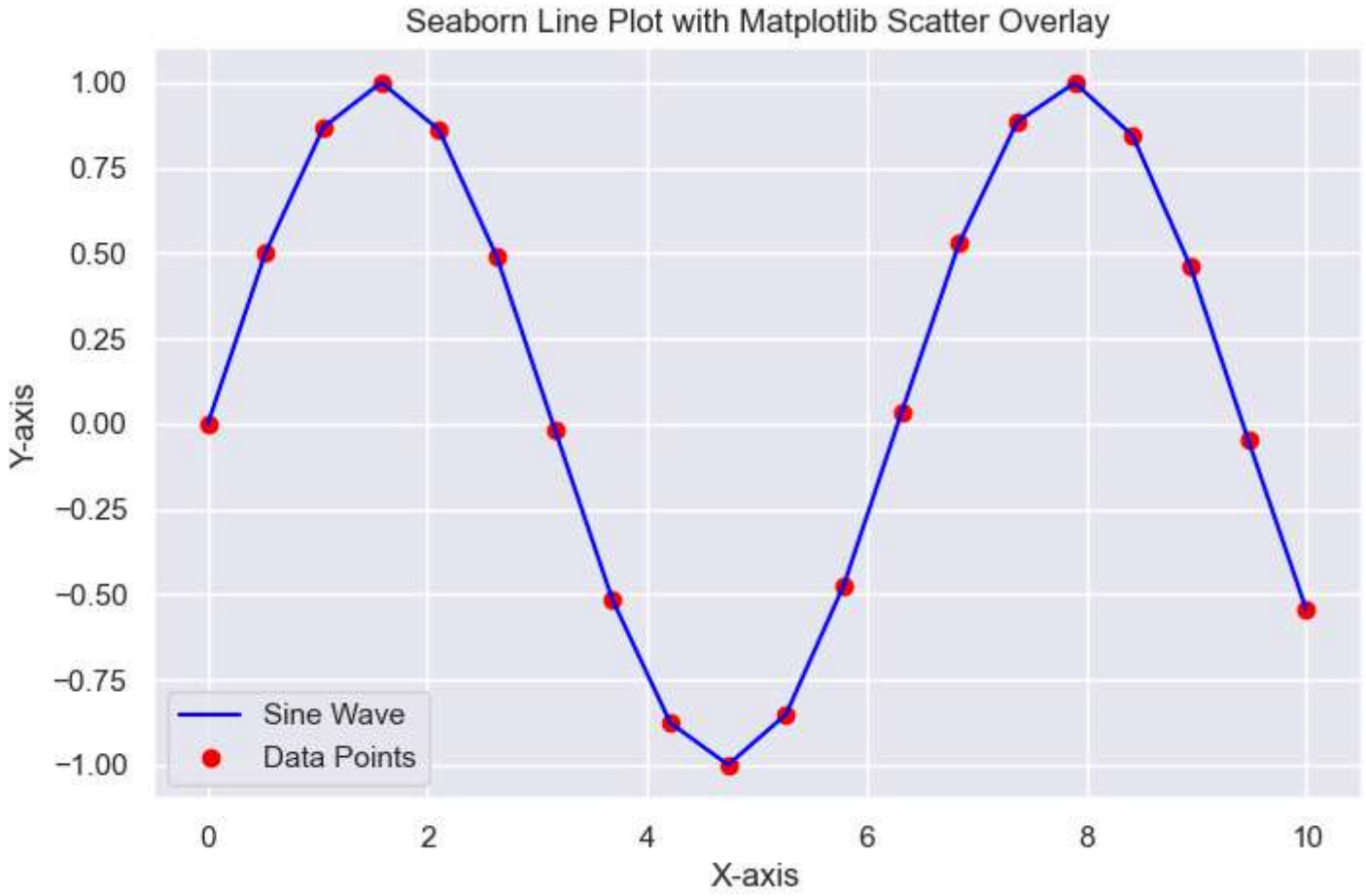
In [68]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

data = pd.DataFrame({
    'Year': [2018, 2019, 2020, 2021, 2022],
    'Sales': [100, 150, 200, 250, 300]
})

plt.figure(figsize=(8, 5))
sns.lineplot(x='Year', y='Sales', data=data, marker='o')
```

```
# Customizing using Matplotlib
plt.title("Yearly Sales Growth", fontsize=14, fontweight='bold')
plt.xlabel("Year", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--')
plt.show()
```



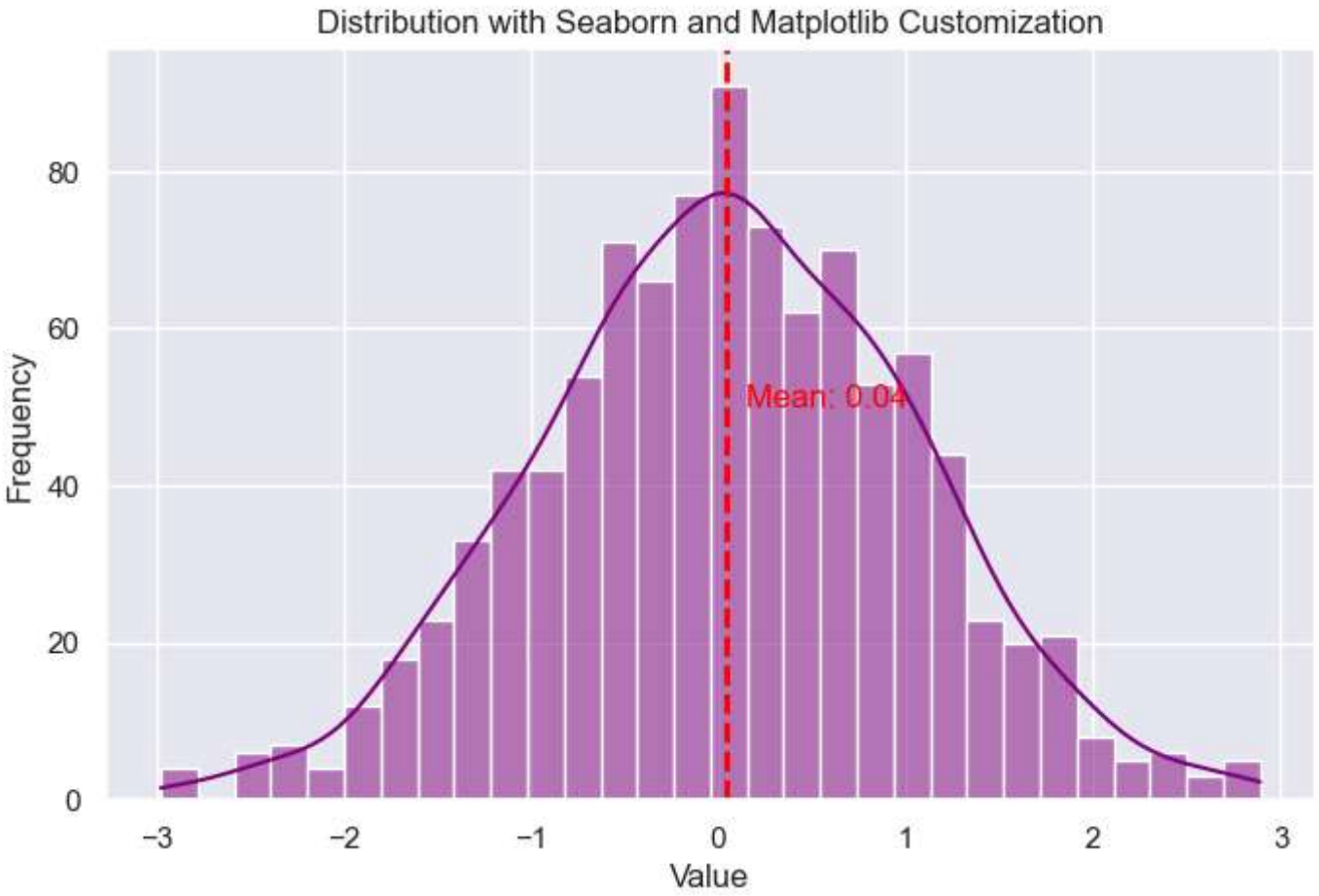
In [70]:
```
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns

x = np.linspace(0, 10, 20)
y = np.sin(x)

plt.figure(figsize=(8, 5))

# Seaborn Line Plot
sns.lineplot(x=x, y=y, color='blue', label='Sine Wave')

# Matplotlib Scatter Plot
plt.scatter(x, y, color='red', marker='o', label="Data Points")

plt.title("Seaborn Line Plot with Matplotlib Scatter Overlay")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.show()
```



In [72]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = np.random.randn(1000)

plt.figure(figsize=(8, 5))
sns.histplot(data, kde=True, bins=30, color='purple')

# Adding Mean Line using Matplotlib
mean_value = np.mean(data)
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2)
plt.text(mean_value + 0.1, 50, f'Mean: {mean_value:.2f}', color='red')
```

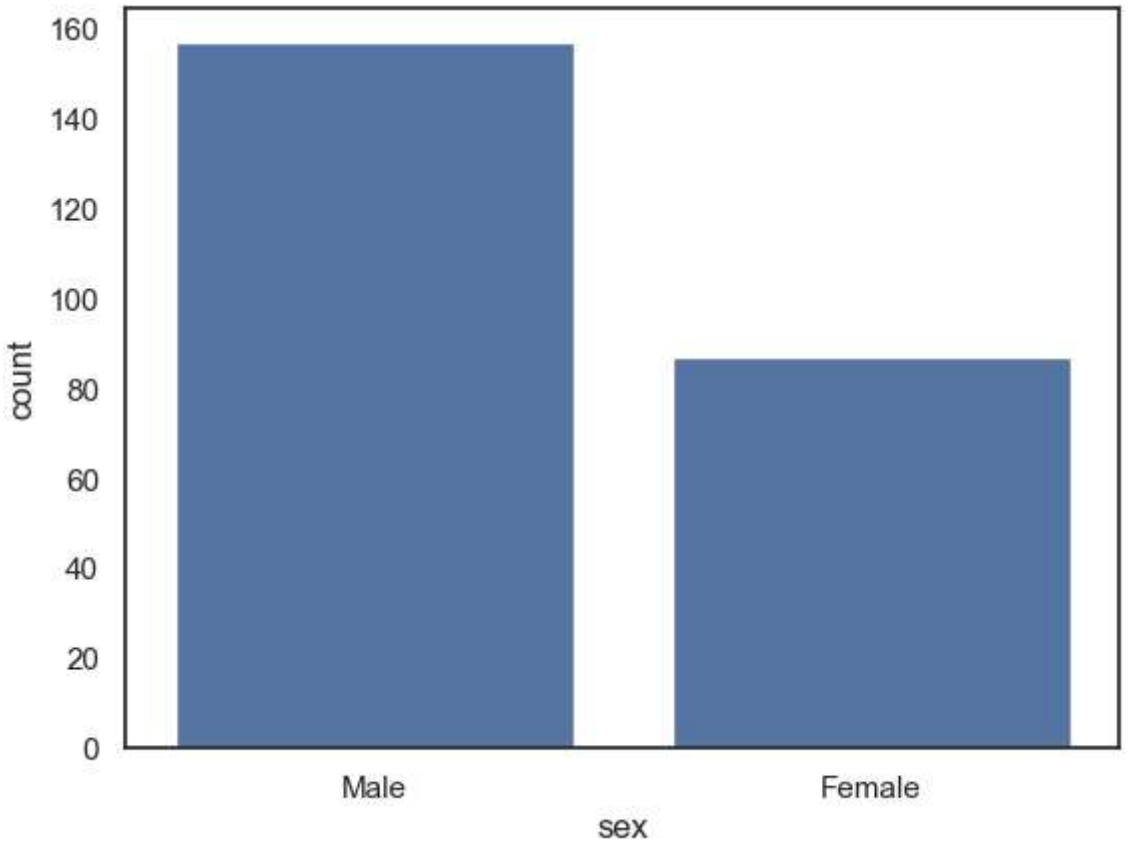
```
plt.title("Distribution with Seaborn and Matplotlib Customization")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



In [74]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# load the tips dataset present by default in seaborn
tips = sns.load_dataset('tips')
sns.set_style('white')
```

```
# make a countplot
sns.countplot(x ='sex', data = tips)
```

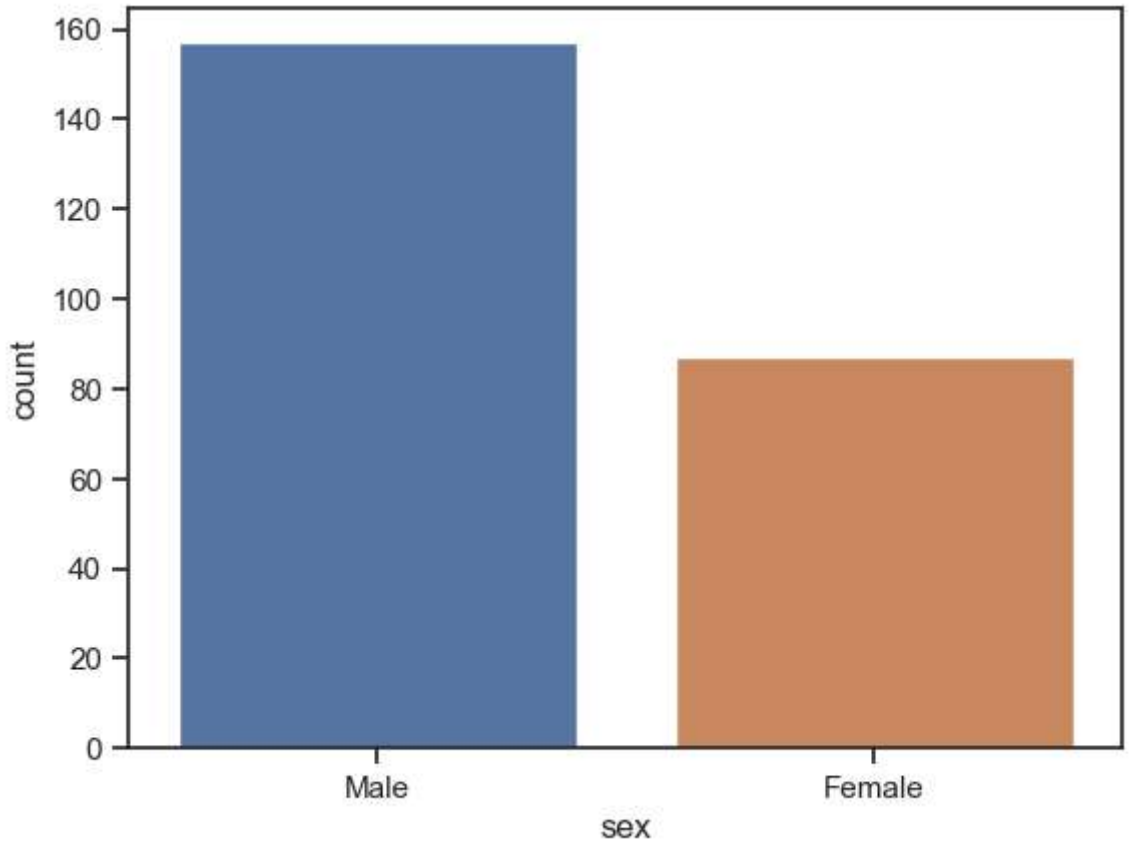
Out[74]: <Axes: xlabel='sex', ylabel='count'>



In [76]:
```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')
sns.set_style('ticks')
sns.countplot(x ='sex', data = tips, palette = 'deep')
```

```
C:\Users\mohap\AppData\Local\Temp\ipykernel_2404\2097168490.py:6: FutureWarning:

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

  sns.countplot(x ='sex', data = tips, palette = 'deep')
```

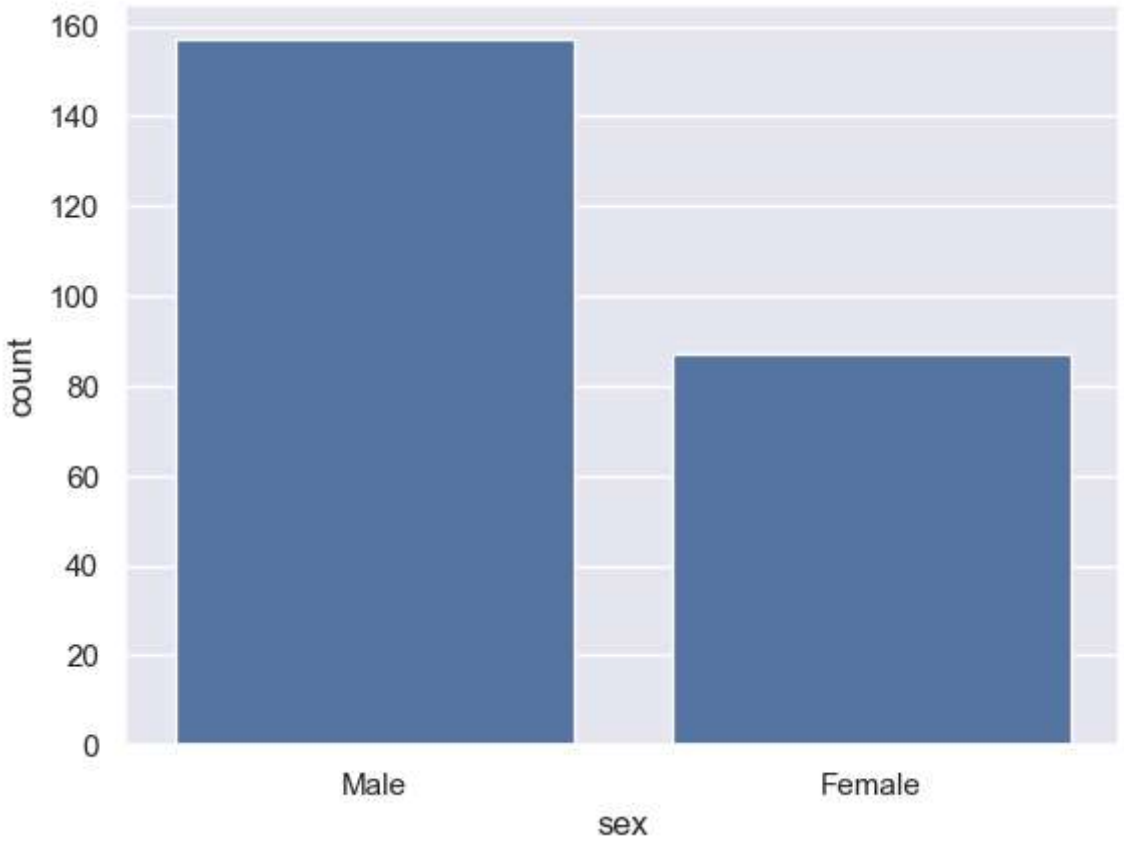
Out[76]: <Axes: xlabel='sex', ylabel='count'>



In [78]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# load the tips dataset present by default in seaborn
tips = sns.load_dataset('tips')
sns.set_style('darkgrid')
```

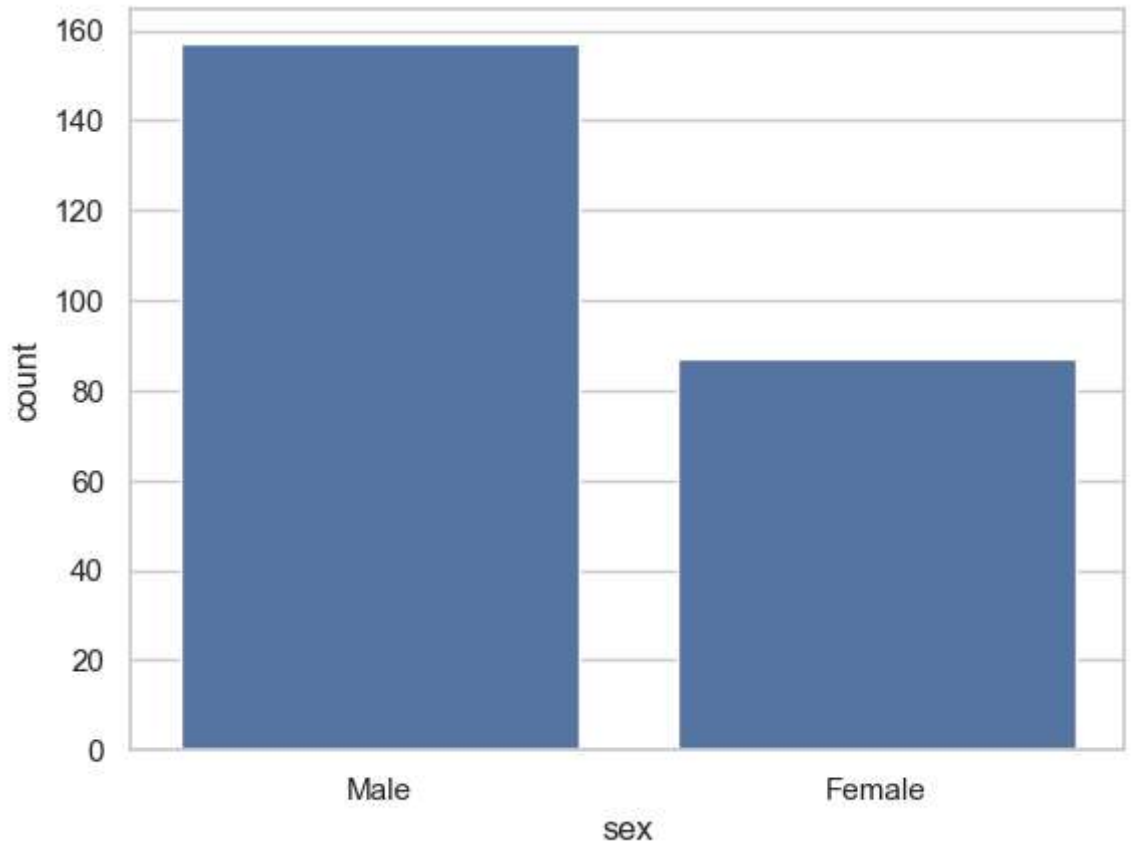
```
# make a countplot
sns.countplot(x ='sex', data = tips)
```

Out[78]: <Axes: xlabel='sex', ylabel='count'>

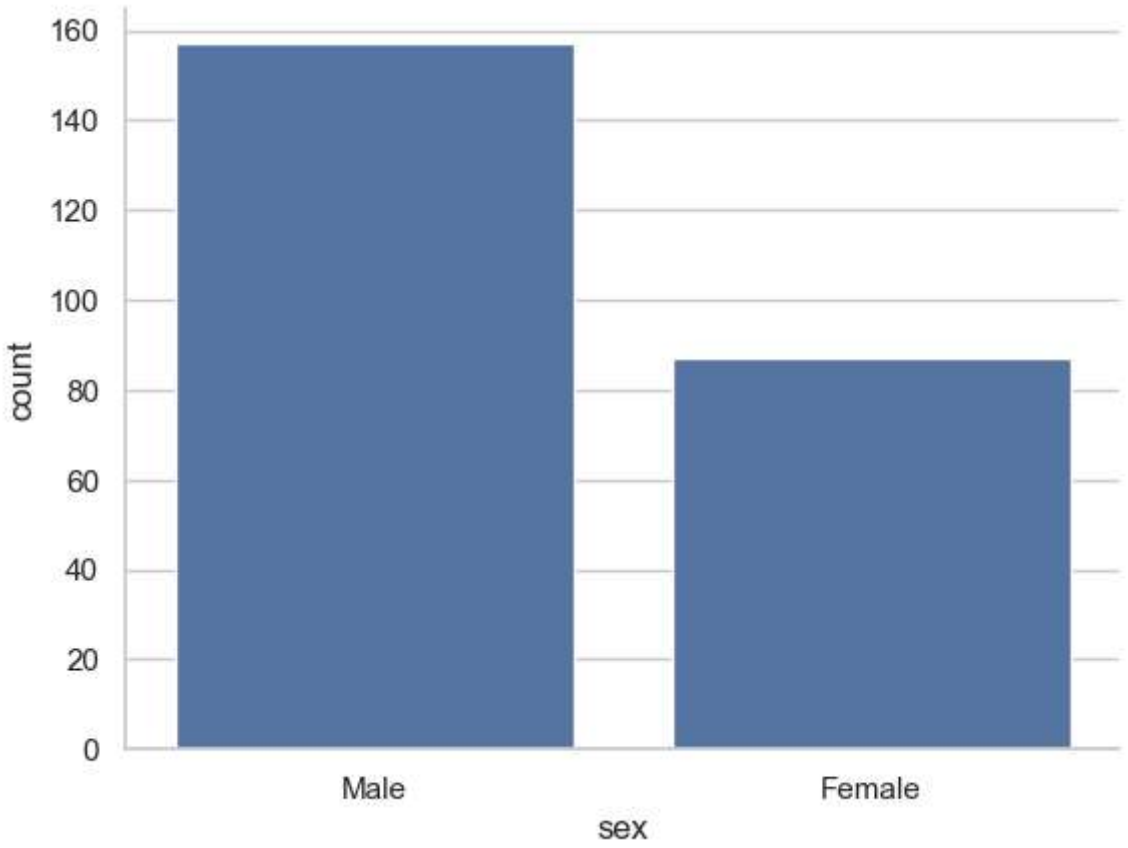


In [80]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# load the tips dataset present by default in seaborn
tips = sns.load_dataset('tips')
sns.set_style('whitegrid')

# make a countplot
sns.countplot(x ='sex', data = tips)
```

Out[80]: <Axes: xlabel='sex', ylabel='count'>

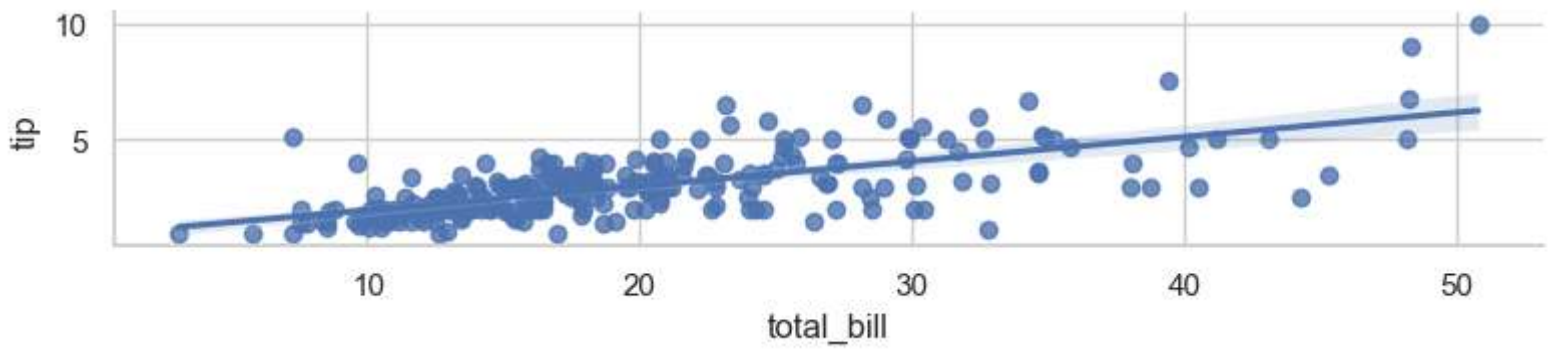


In [82]:
```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')
sns.countplot(x ='sex', data = tips)
sns.despine()
```



In [90]:

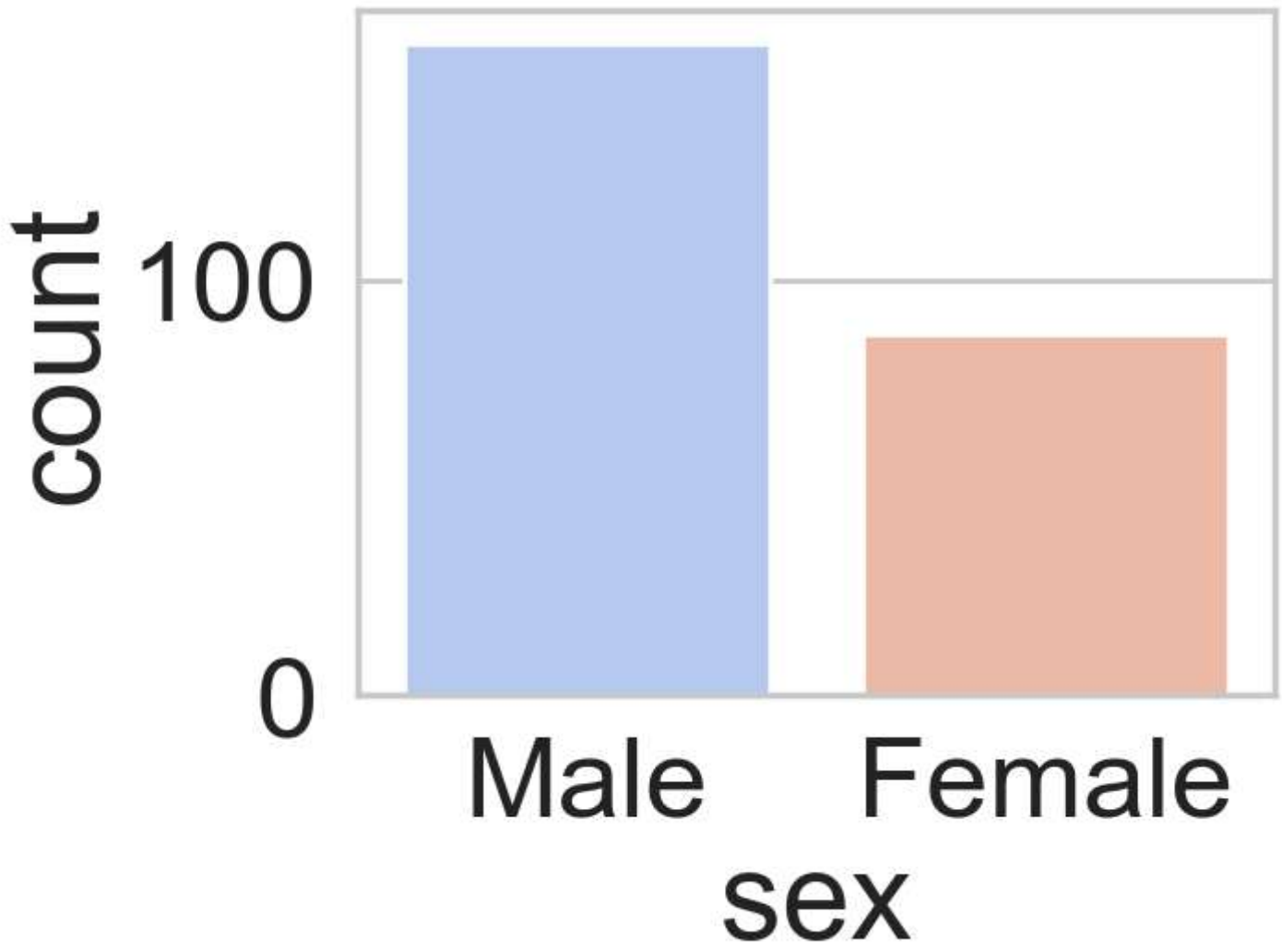
```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')

sns.lmplot(
    x='total_bill',
    y='tip',
    height=2,      # ✅ correct
    aspect=4,
    data=tips
)

plt.show()
```



In [92]:
```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')
sns.set_context('poster', font_scale = 2)
sns.countplot(x ='sex', data = tips, palette ='coolwarm')
```

```
C:\Users\mohap\AppData\Local\Temp\ipykernel_2404\1621432276.py:6: FutureWarning:

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

  sns.countplot(x ='sex', data = tips, palette ='coolwarm')
```

Out[92]: <Axes: xlabel='sex', ylabel='count'>



In [ ]: