Seaborn

- A Python library built on top of Matplotlib for creating visually appealing statistical graphics.
- Best for creating heatmaps, violin plots, pair plots, and categorical plots (like bar, box, and strip plots).
- It simplifies complex visualization tasks with high-level functions

Importing Seaborn and Dataset

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

# Load an example dataset
tips = sns.load_dataset("tips")
print(tips.head())
```

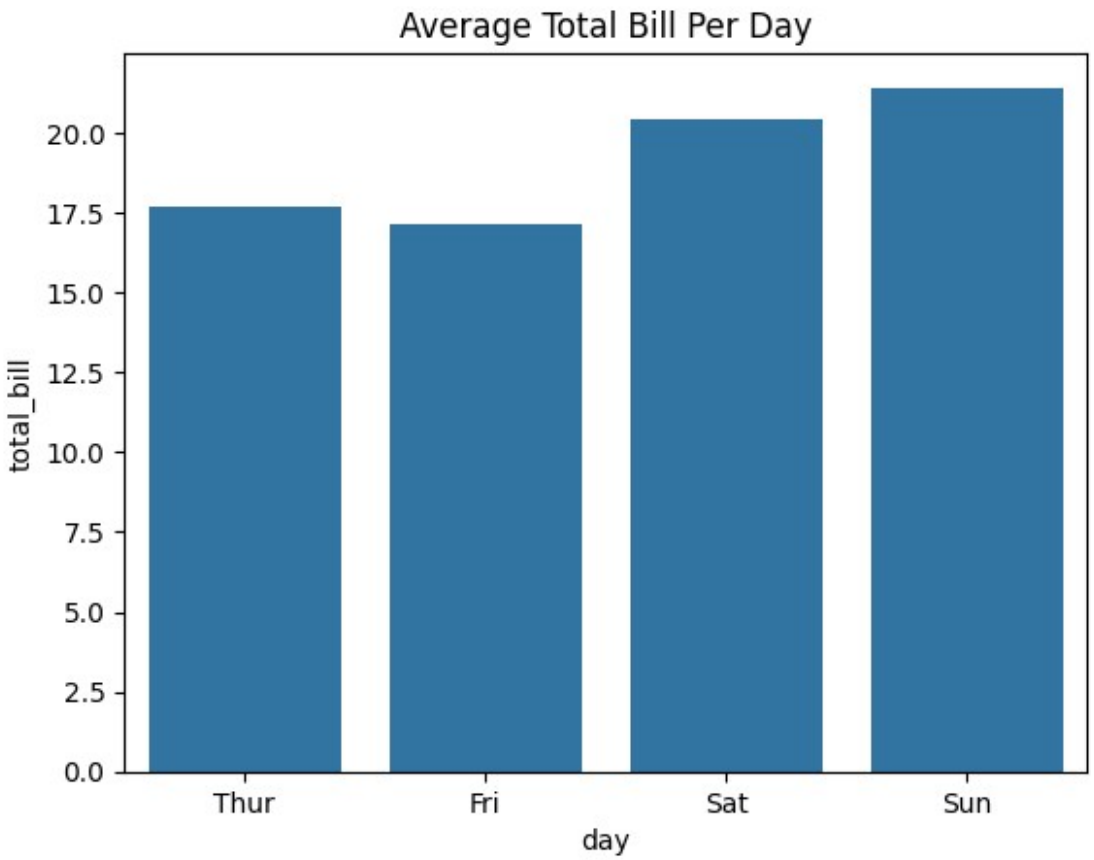
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female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

Bar Plot

What: A bar plot is used to show the distribution of categorical data with rectangular bars where the length represents the value.

Why: Use it when you want to compare quantities across categories (e.g., total bill across different days).

```
# Bar plot: Average total bill per day
sns.barplot(x="day", y="total_bill", data=tips, errorbar=None)
plt.title("Average Total Bill Per Day")
plt.show()
```

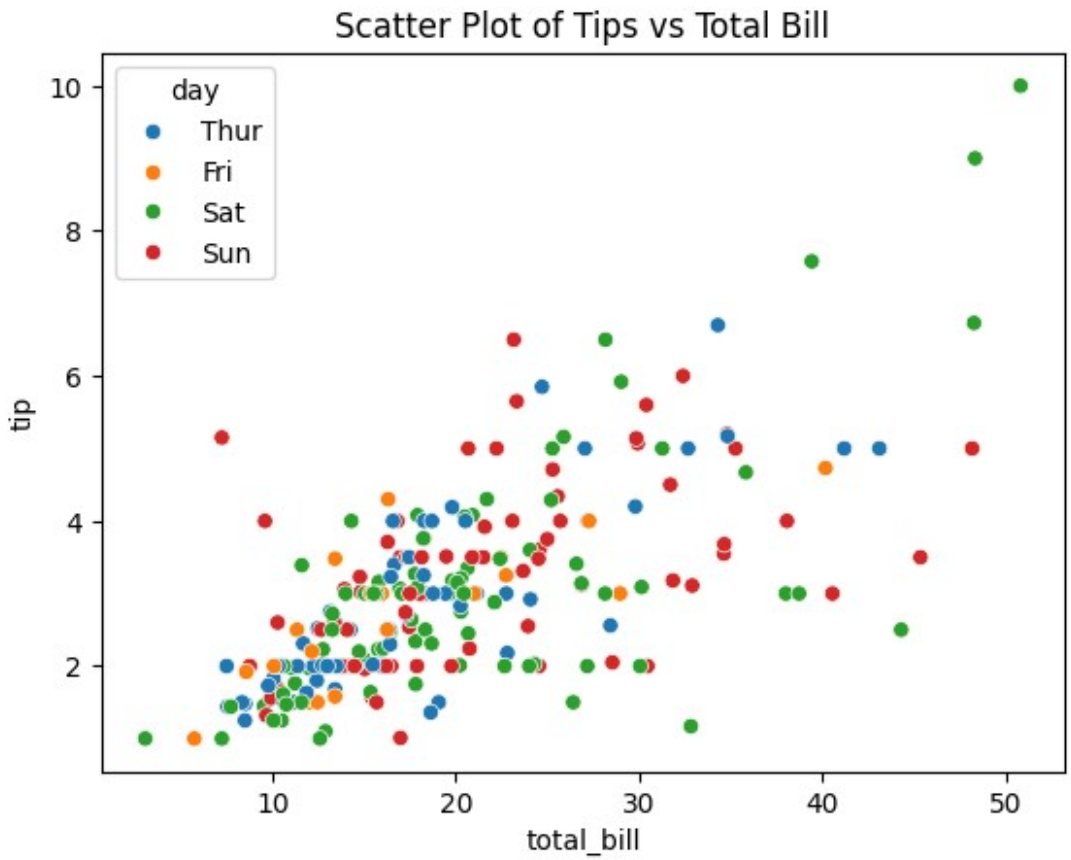


Scatter Plot

What: A scatter plot shows the relationship between two continuous variables, with each point representing a pair of values.

Why: Use it to identify correlations, trends, or clusters in data.

```
# Scatter plot: Relationship between total bill and tip
sns.scatterplot(x="total_bill", y="tip", hue="day", data=tips)
plt.title("Scatter Plot of Tips vs Total Bill")
plt.show()
```

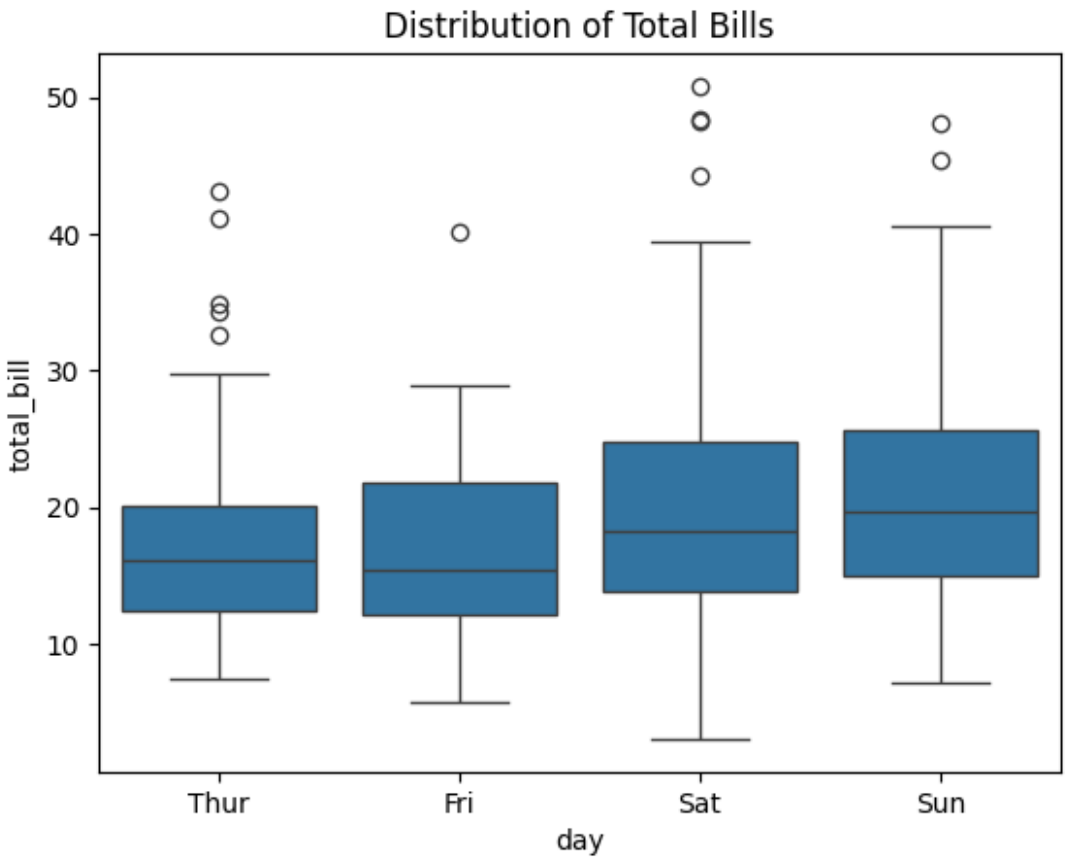


Box Plot

What: A box plot visualizes the distribution of data based on five summary statistics: minimum, first quartile (Q1), median, third quartile (Q3), and maximum.

Why: Use it to detect outliers, understand the spread, and compare distributions across categories.

```
# Box plot: Distribution of total bills per day
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("Distribution of Total Bills")
plt.show()
```

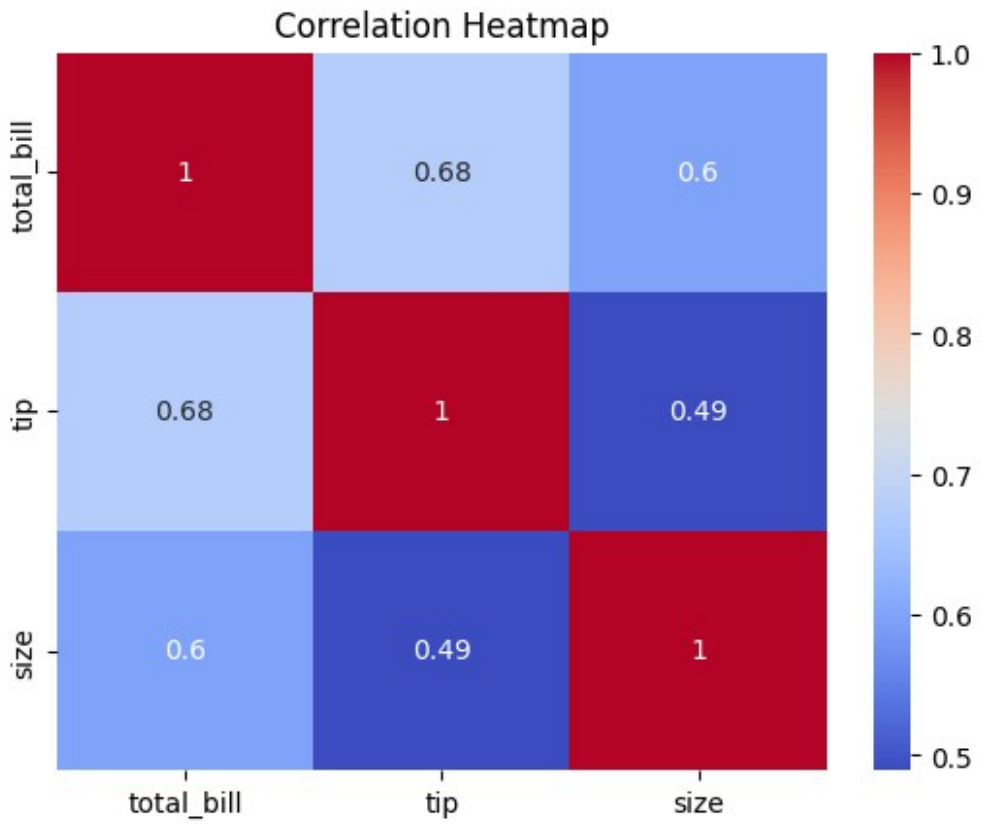


Heatmap

What: A heatmap displays data in matrix format, where values are represented by colors, useful for visualizing correlations or intensity.

Why: Use it to observe patterns, correlations, or relationships between numerical variables.

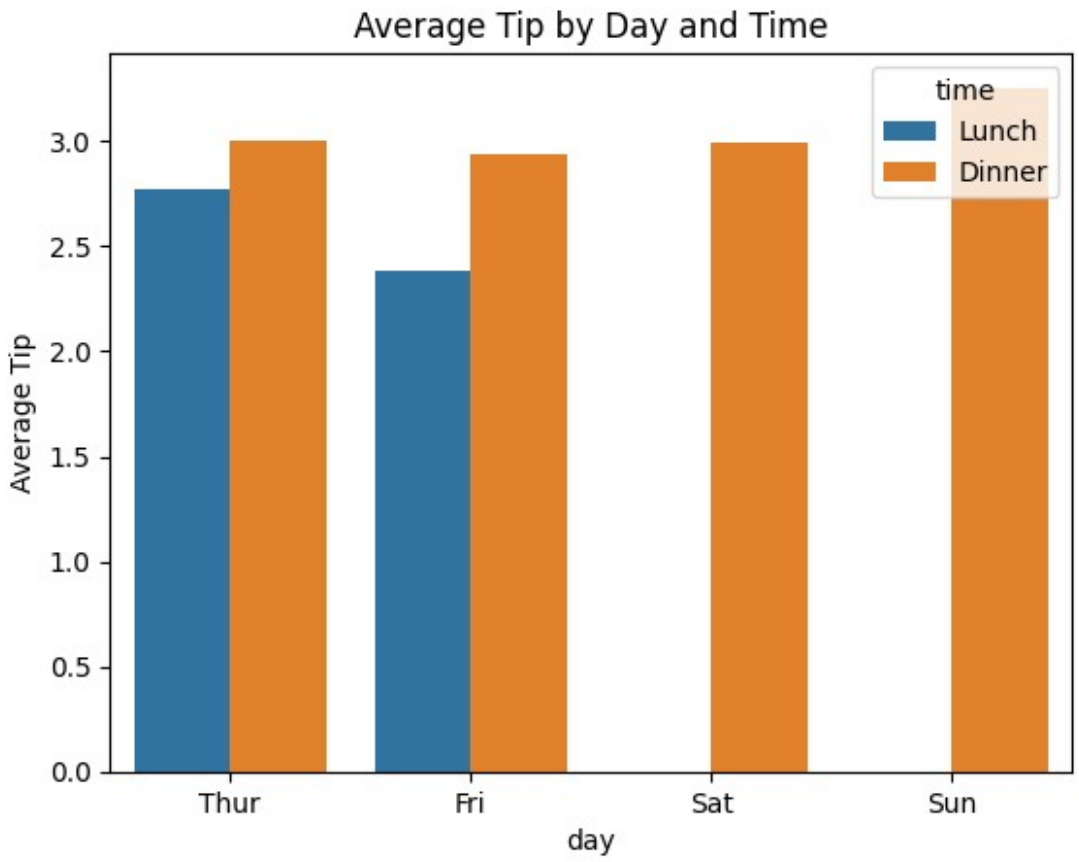
```
# Select numeric columns from the dataset
numeric_tips = tips.select_dtypes(include=["float64", "int64"])

# Compute the correlation matrix
correlation = numeric_tips.corr()

# Display the heatmap
sns.heatmap(correlation, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



Bar Plot: Average Tip by Day and Time

Bar plots are great for summarizing and comparing values.

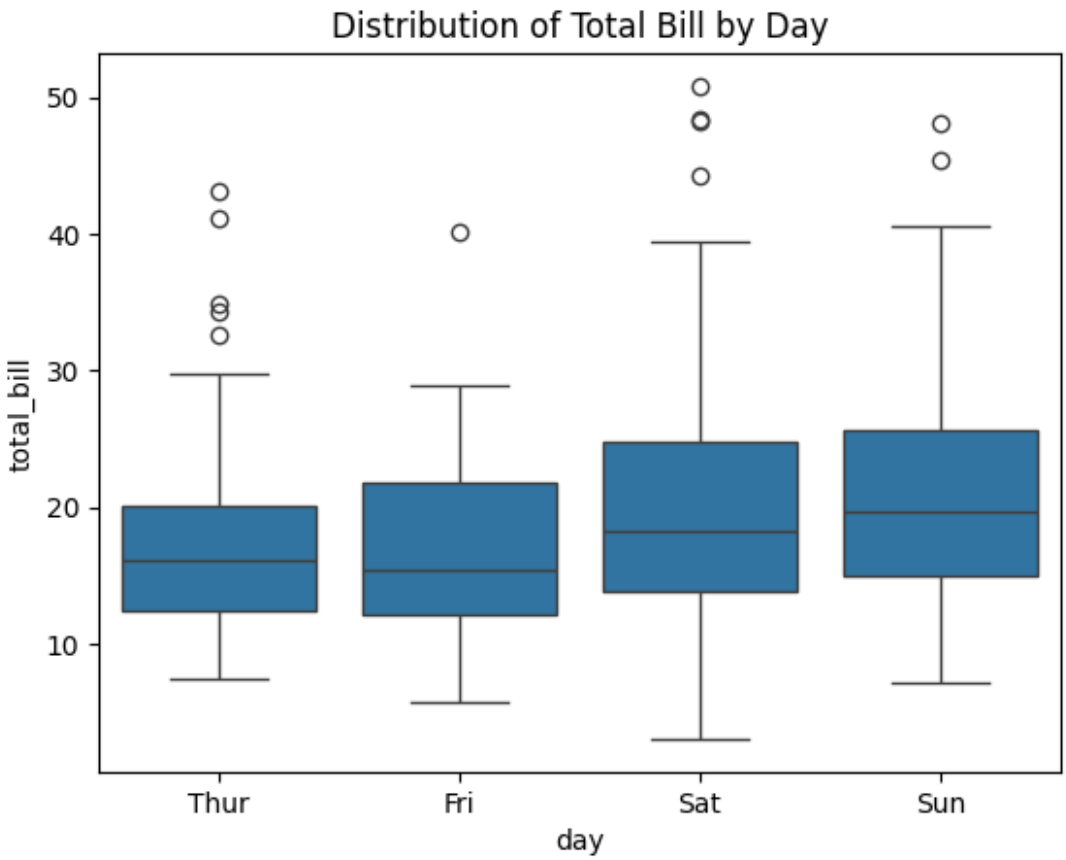
```
import seaborn as sns
import matplotlib.pyplot as plt

# Load the tips dataset
tips = sns.load_dataset("tips")

# Bar plot: Average tip by day and time
sns.barplot(x="day", y="tip", hue="time", data=tips, errorbar=None)
plt.title("Average Tip by Day and Time")
plt.ylabel("Average Tip")
plt.show()
```



Box Plot: Distribution of Total Bill by Day

Box plots help visualize the spread and outliers in the data.

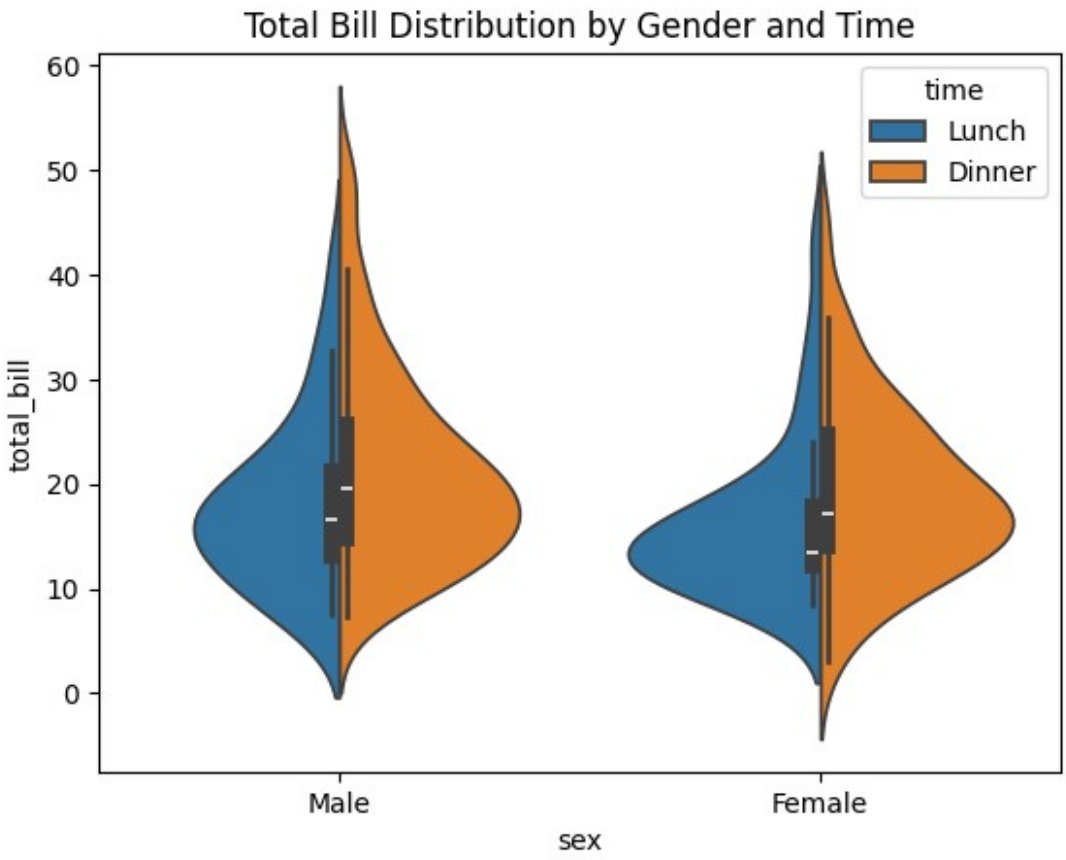
```
# Box plot: Total bill by day
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("Distribution of Total Bill by Day")
plt.show()
```



Violin Plot: Total Bill by Gender and Time

Violin plots show the distribution and density of the data.

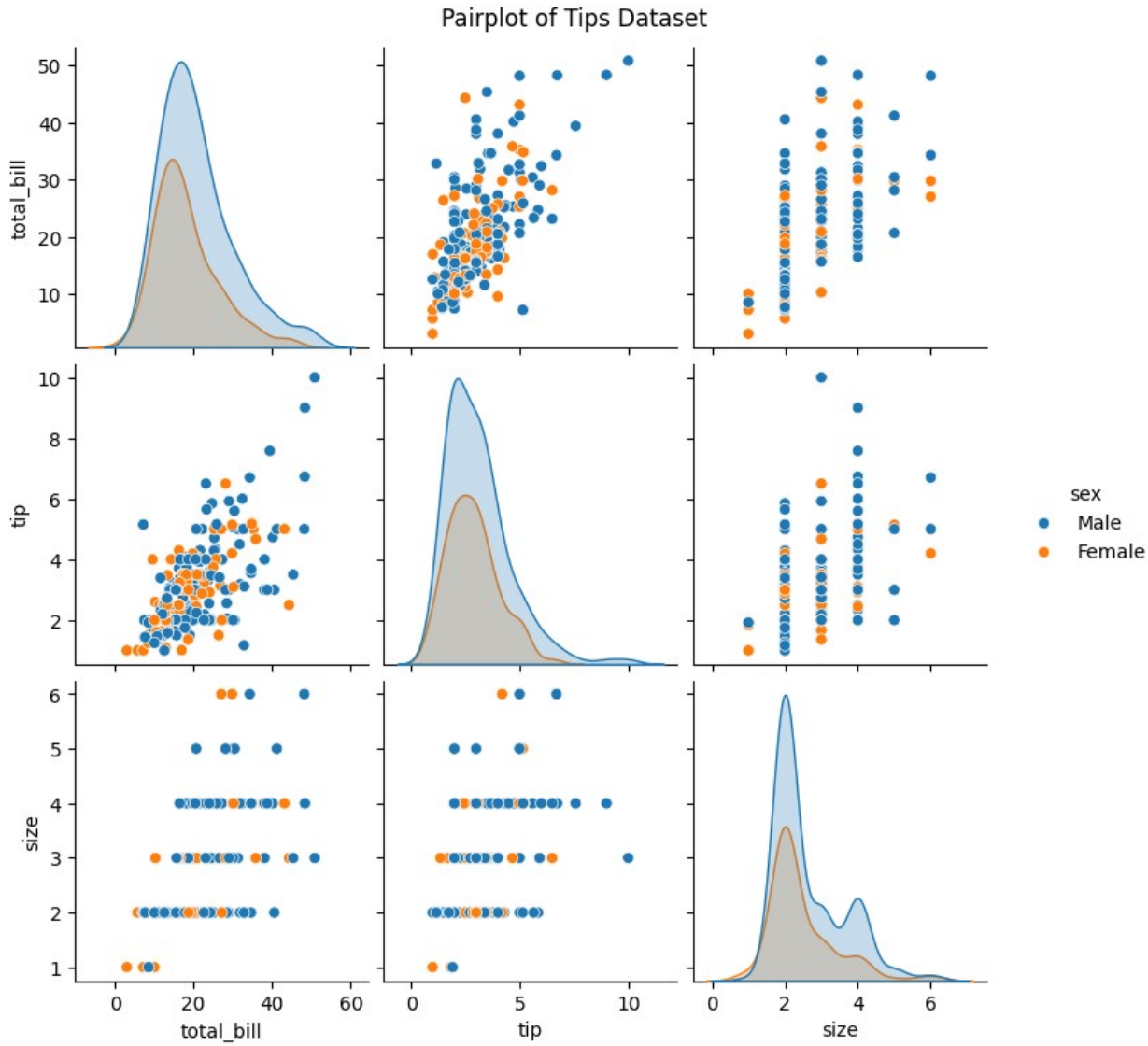
```
# Violin plot: Total bill by gender and time
sns.violinplot(x="sex", y="total_bill", hue="time", data=tips,
split=True)
plt.title("Total Bill Distribution by Gender and Time")
plt.show()
```



Pairplot: Relationships Between Variables

Pair plots are ideal for exploring relationships in datasets with numerical and categorical features.

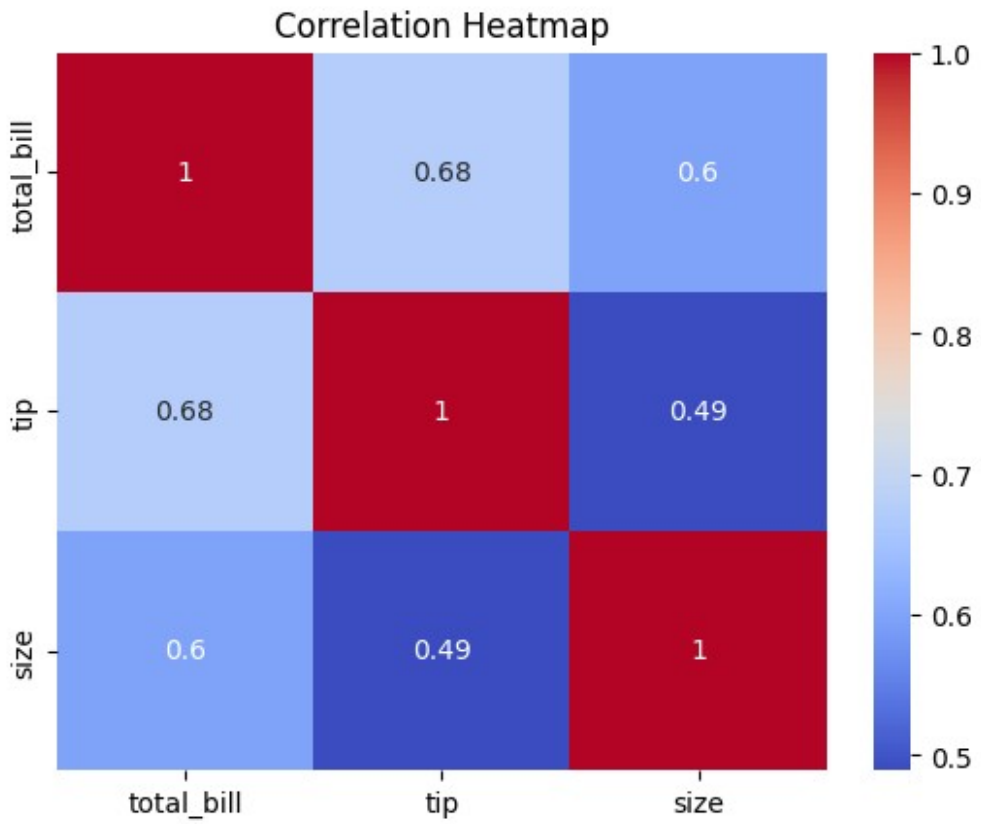
```
# Pairplot for numerical relationships in tips
sns.pairplot(tips, hue="sex")
plt.suptitle("Pairplot of Tips Dataset", y=1.02)
plt.show()
```



Heatmap: Correlation Between Numerical Variables

A heatmap can help identify patterns and strong correlations.

```
# Select numeric columns from the dataset
numeric_tips = tips.select_dtypes(include=["float64", "int64"])

# Compute the correlation matrix
correlation = numeric_tips.corr()

# Display the heatmap
sns.heatmap(correlation, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



Plotly

- A library for creating interactive visualizations.
- Allows for dynamic updates, hover functionality, zooming, and exporting graphs.
- Offers integration with Dash for creating interactive web apps. Best for line plots, scatter plots, 3D plots, and dashboards.

Importing Plotly and Dataset

```
import plotly.express as px

# Load example data
df = px.data.iris()
print(df.head(10))
```

```
   sepal_length  sepal_width  petal_length  petal_width species
species_id
0           5.1          3.5           1.4          0.2  setosa
1
1           4.9          3.0           1.4          0.2  setosa
1
2           4.7          3.2           1.3          0.2  setosa
1
```

```
3            4.6           3.1            1.5           0.2  setosa
1
4            5.0           3.6            1.4           0.2  setosa
1
5            5.4           3.9            1.7           0.4  setosa
1
6            4.6           3.4            1.4           0.3  setosa
1
7            5.0           3.4            1.5           0.2  setosa
1
8            4.4           2.9            1.4           0.2  setosa
1
9            4.9           3.1            1.5           0.1  setosa
1
```

Line Plot

```
# Line plot: Sepal length across samples
fig = px.line(df, x=df.index, y="sepal_length", title="Sepal Length
Trend")
fig.show()

# Line plot: Petal length across samples
fig = px.line(df, x=df.index, y="petal_length", title="Petal Length
Trend")
fig.show()
```

Scatter Plot

```
# Scatter plot: Sepal vs Petal Length
fig = px.scatter(df, x="sepal_length", y="petal_length",
color="species", title="Sepal vs Petal Length")
fig.show()
```

Bar Plot

```
# Bar plot: Average sepal width per species
fig = px.bar(df, x="species", y="sepal_width", title="Average Sepal
Width by Species")
fig.show()

# Bar plot: Average petal width per species
fig = px.bar(df, x="species", y="petal_width", title="Average Petal
Width by Species")
fig.show()
```

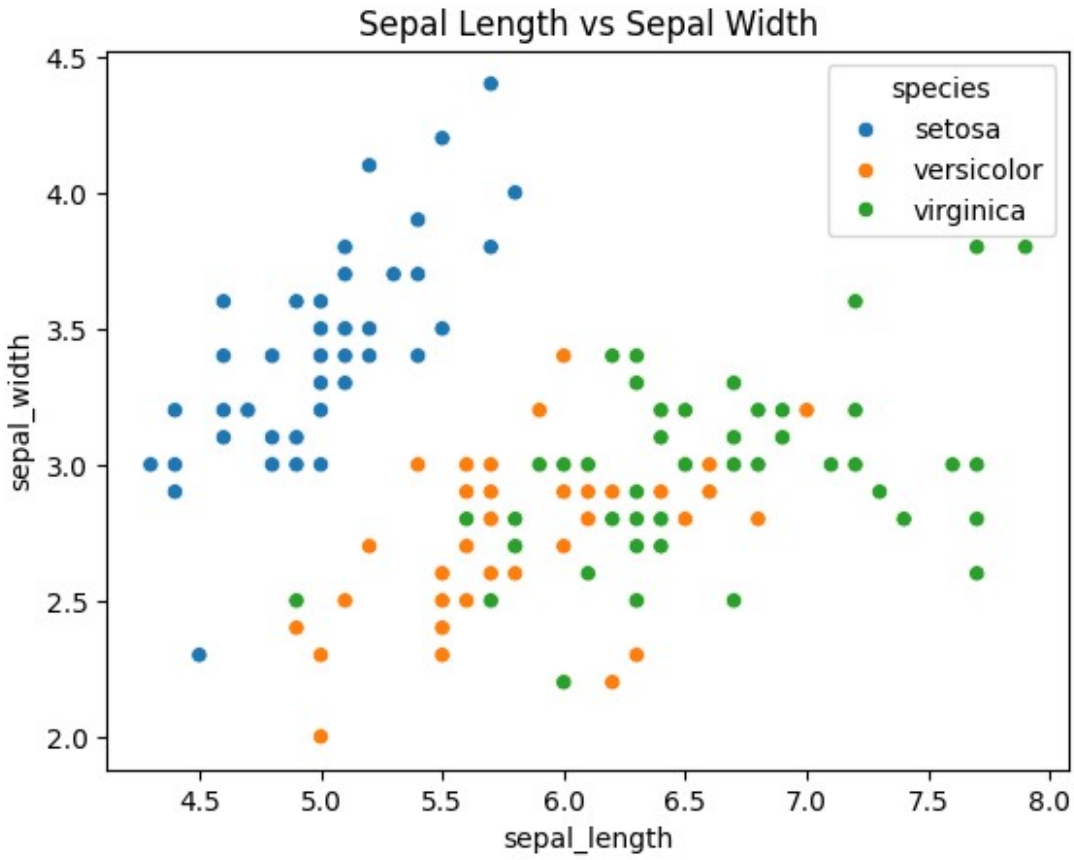
Pie Chart

```
# Pie chart: Species distribution
fig = px.pie(df, names="species", title="Species Distribution")
fig.show()
```

Scatter Plot: Sepal Length vs Sepal Width

Scatter plots show relationships between two numerical variables.

```
# Load the iris dataset
iris = sns.load_dataset("iris")

# Scatter plot: Sepal length vs sepal width
sns.scatterplot(x="sepal_length", y="sepal_width", hue="species",
data=iris)
plt.title("Sepal Length vs Sepal Width")
plt.show()
```

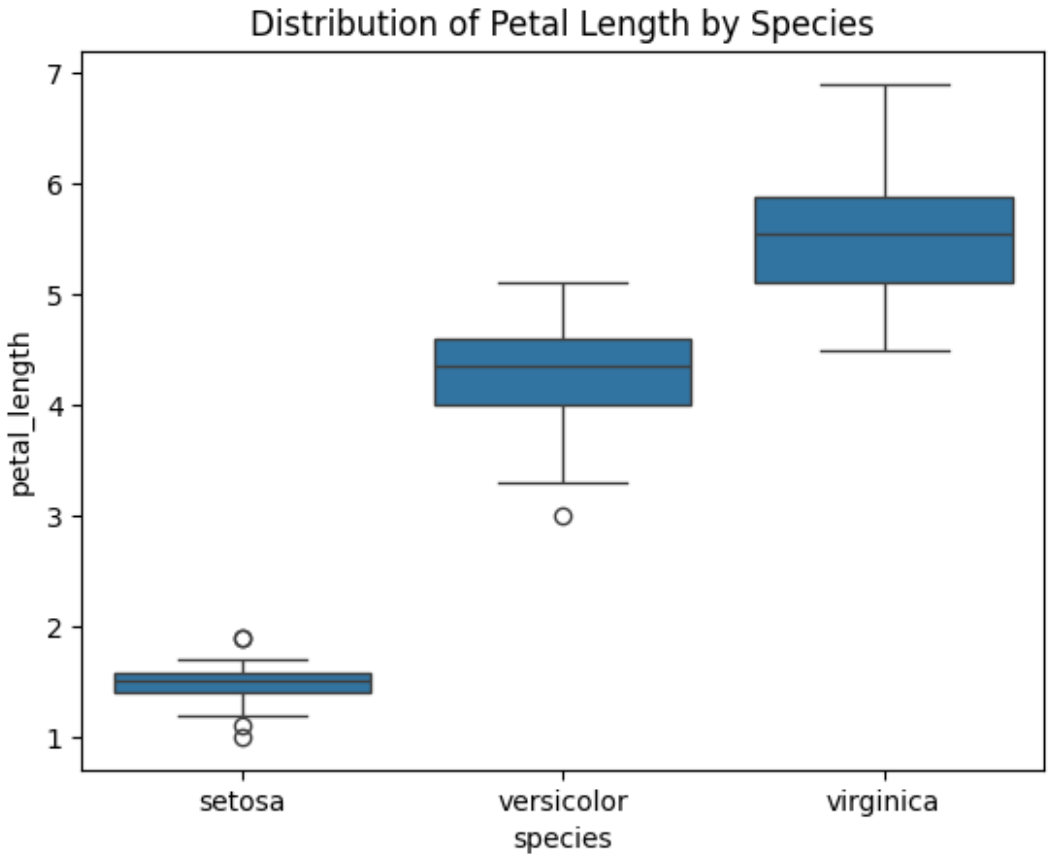


Box Plot: Distribution of Petal Length

Box plots help analyze distributions of numerical data.

```
# Box plot: Petal length by species
sns.boxplot(x="species", y="petal_length", data=iris)
```

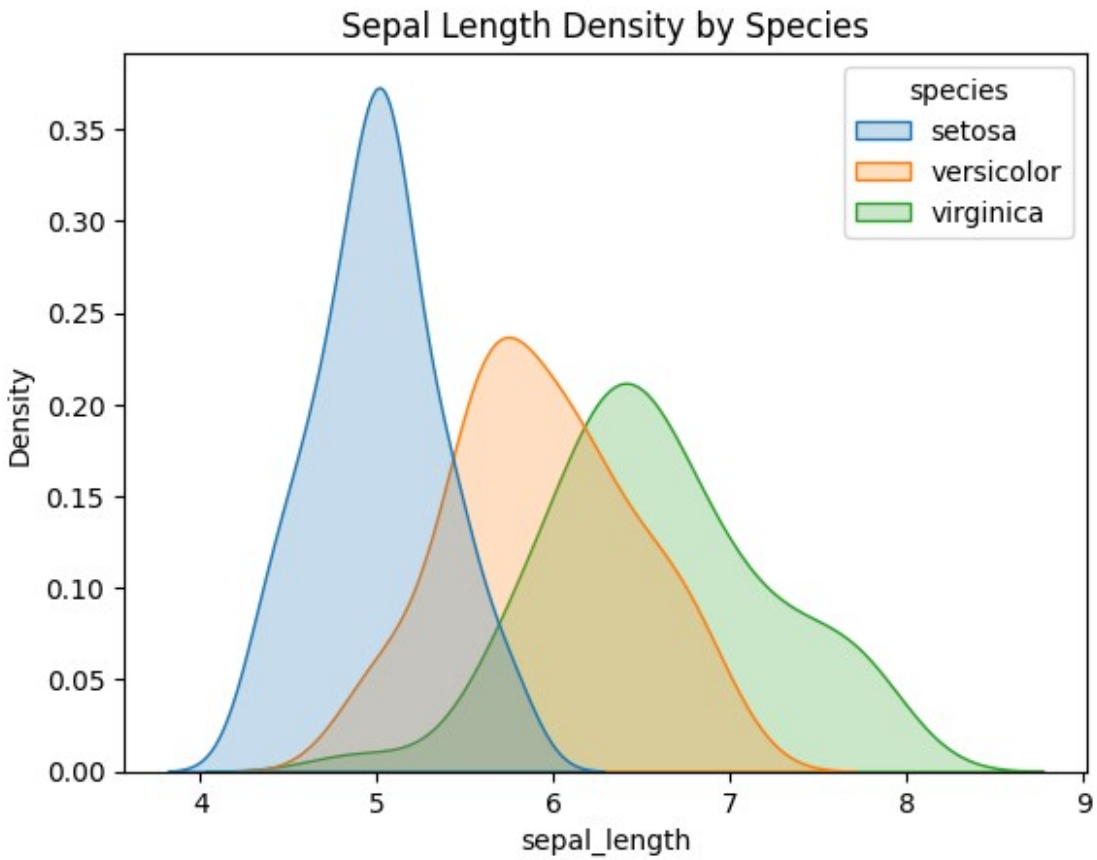
```
plt.title("Distribution of Petal Length by Species")
plt.show()
```



KDE Plot: Sepal Length Density

Kernel Density Estimation (KDE) plots visualize the probability density.

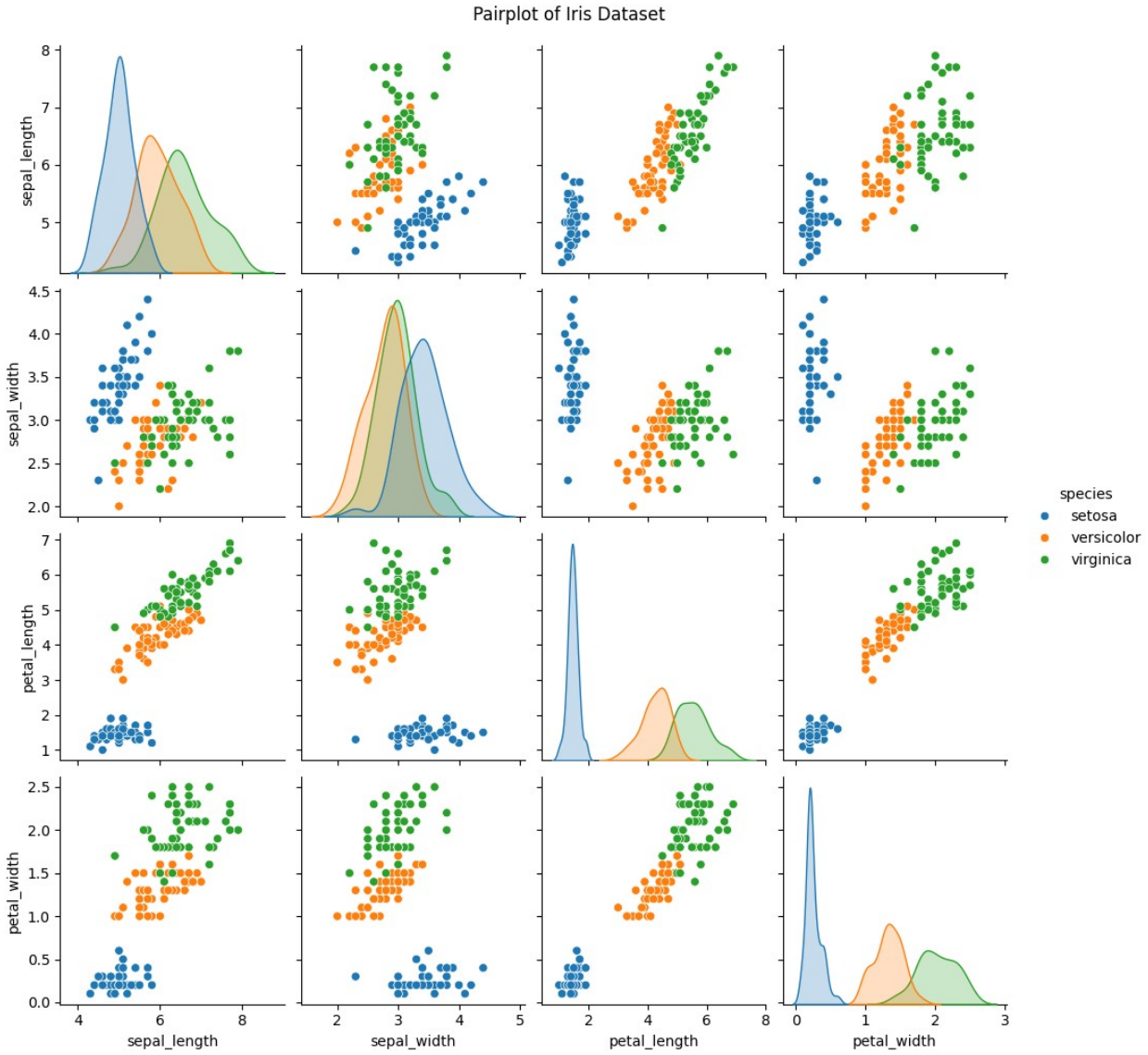
```
# KDE plot for Sepal length by species
sns.kdeplot(data=iris, x="sepal_length", hue="species", fill=True)
plt.title("Sepal Length Density by Species")
plt.show()
```



Pairplot: Relationships in Iris Dataset

Pair plots allow a quick overview of relationships among variables.

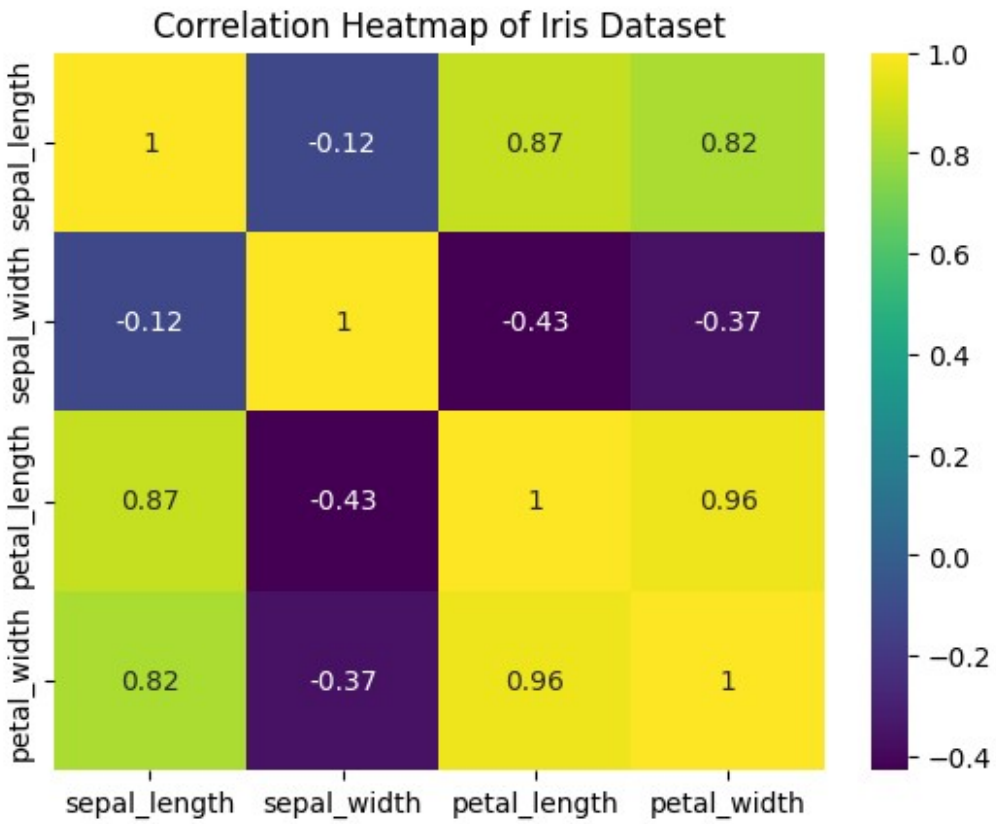
```
# Pairplot of iris dataset
sns.pairplot(iris, hue="species")
plt.suptitle("Pairplot of Iris Dataset", y=1.02)
plt.show()
```



Heatmap: Correlation Between Numerical Variables

This shows numerical relationships and their strength.

```
# Correlation matrix
correlation_iris = iris.drop("species", axis=1).corr()

# Heatmap
sns.heatmap(correlation_iris, annot=True, cmap="viridis")
plt.title("Correlation Heatmap of Iris Dataset")
plt.show()
```



Student Guidelines

- Use Comments: Add comments to explain your code.
- Experiment: Try changing the axes, colors, and other parameters to make the plots more informative.
- Combine Seaborn and Plotly: Use both libraries for the same dataset to compare their features.
- Submit Code and Outputs: Provide the Python script and screenshots of your outputs.

Seaborn Classwork

Dataset: penguins

The penguins dataset contains information about penguin species, their sizes, and island habitats.

Tasks:

- Bar Plot: Create a bar plot showing the average body mass for each penguin species.
- Pairplot: Create a pair plot showing relationships between numerical features of the penguins dataset, categorized by species.

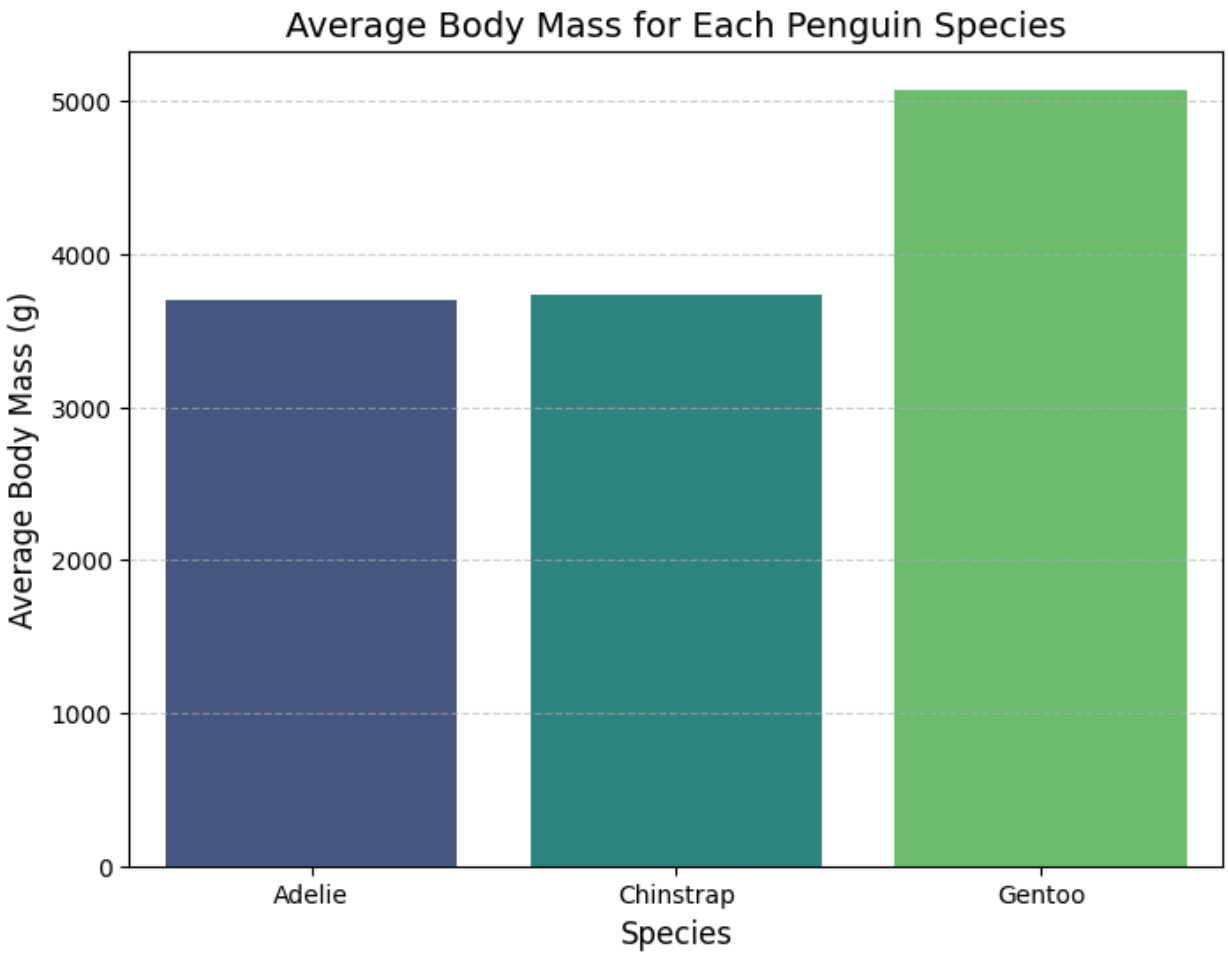
- Box Plot: Create a box plot showing the distribution of flipper length for each species.

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

# Load the penguins dataset
penguins = sns.load_dataset("penguins")

penguins
```

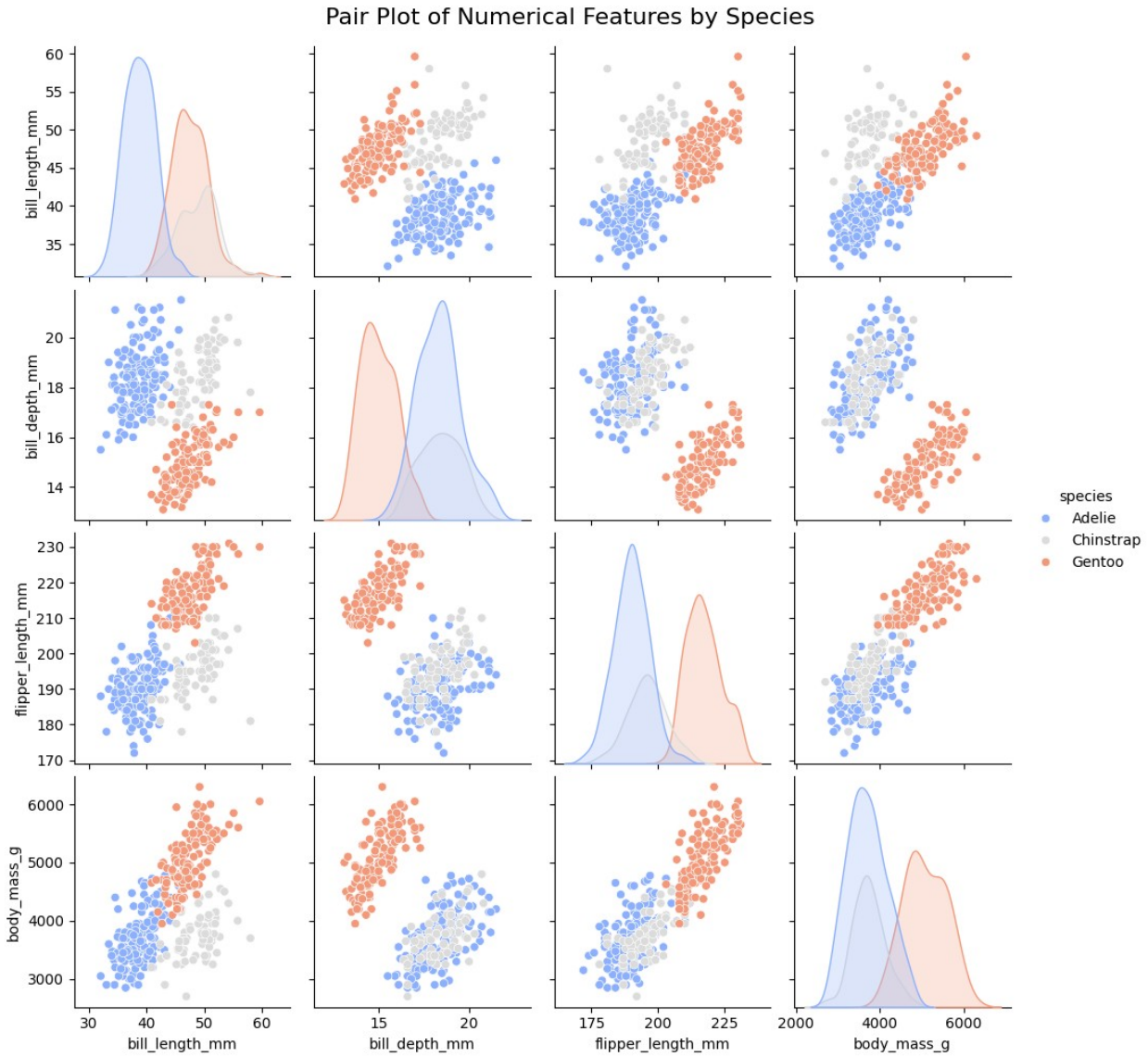

```
{"summary":"{\n  \"name\": \"penguins\",\n  \"rows\": 344,\n
\"fields\": [\n    {\n      \"column\": \"species\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Adelie\",\n          \"Chinstrap\",\n          \"Gentoo\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"island\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Torgersen\",\n          \"Biscoe\",\n
\"Dream\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"bill_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5.459583713926532,\n        \"min\":
32.1,\n        \"max\": 59.6,\n        \"num_unique_values\": 164,\n
\"samples\": [\n          48.2,\n          49.8,\n          45.1\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"bill_depth_mm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.9747931568167816,\n        \"min\": 13.1,\n        \"max\": 21.5,\n
\"num_unique_values\": 80,\n        \"samples\": [\n          16.9,\n
18.7,\n          18.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flipper_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.061713679356894,\n        \"min\":
172.0,\n        \"max\": 231.0,\n        \"num_unique_values\": 55,\n
\"samples\": [\n          201.0,\n          180.0,\n          212.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"body_mass_g\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
801.9545356980954,\n        \"min\": 2700.0,\n        \"max\":
6300.0,\n        \"num_unique_values\": 94,\n        \"samples\": [\n
4350.0,\n          4150.0,\n          3525.0\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}","type":"dataframe","variable_name":"penguins"}
```

```
# 1. Bar Plot: Average body mass for each penguin species
plt.figure(figsize=(8, 6))
sns.barplot(x="species", y="body_mass_g", data=penguins, ci=None,
palette="viridis")
plt.title("Average Body Mass for Each Penguin Species", fontsize=14)
plt.xlabel("Species", fontsize=12)
plt.ylabel("Average Body Mass (g)", fontsize=12)
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```

```
<ipython-input-9-ec2067d8dd52>:3: FutureWarning:

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

  sns.barplot(x="species", y="body_mass_g", data=penguins, ci=None,
palette="viridis")
<ipython-input-9-ec2067d8dd52>:3: FutureWarning:

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

  sns.barplot(x="species", y="body_mass_g", data=penguins, ci=None,
palette="viridis")
```



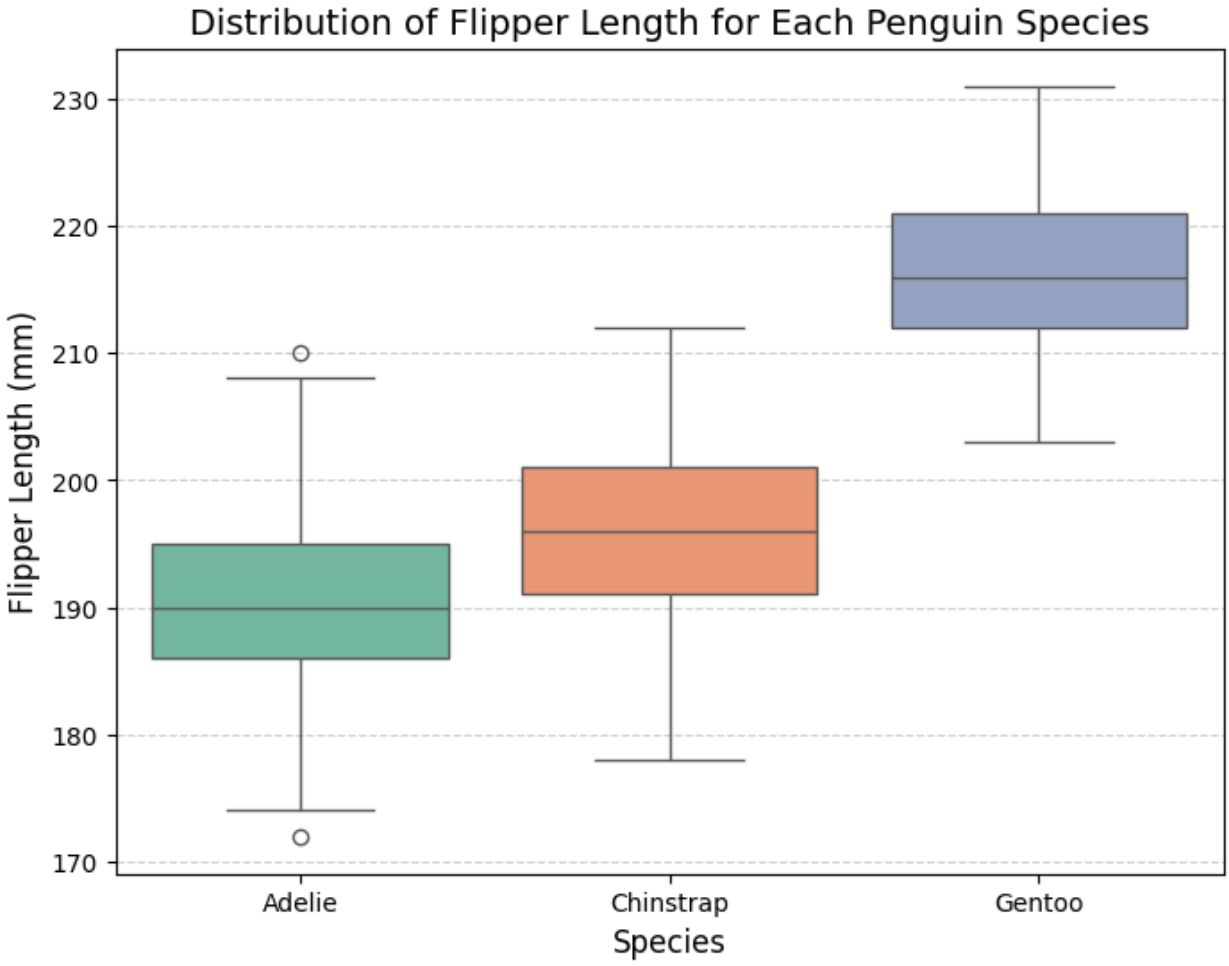

```
# 2. Pair Plot: Relationships between numerical features categorized
by species
sns.pairplot(penguins, hue="species", palette="coolwarm",
diag_kind="kde", height=2.5)
plt.suptitle("Pair Plot of Numerical Features by Species", y=1.02,
fontsize=16)
plt.show()
```




```
# 3. Box Plot: Distribution of flipper length for each species
plt.figure(figsize=(8, 6))
sns.boxplot(x="species", y="flipper_length_mm", data=penguins,
palette="Set2")
plt.title("Distribution of Flipper Length for Each Penguin Species",
fontsize=14)
plt.xlabel("Species", fontsize=12)
plt.ylabel("Flipper Length (mm)", fontsize=12)
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```

```
<ipython-input-11-d5363b22e09e>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

```
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x="species", y="flipper_length_mm", data=penguins,
palette="Set2")
```



Plotly Classwork

Dataset: gapminder

The gapminder dataset contains information about countries, their GDP, life expectancy, and population over time.

Tasks:

- Scatter Plot: Create an interactive scatter plot showing the relationship between GDP per capita and life expectancy for the year 2007.
- Bar Plot: Create a bar chart showing the total population for each continent in 2007.

- 3D Plot: Create a 3D scatter plot to visualize GDP per capita, life expectancy, and population for the year 2007.
- Line Plot: Create a line plot showing the evolution of life expectancy over time for a selected country (e.g., India,UAE,Germany,China,Australia,Russia).

```
import plotly.express as px

# Load gapminder dataset
gapminder = px.data.gapminder()

gapminder
```


```
{"summary":"{\n  \"name\": \"gapminder\",\n  \"rows\": 1704,\n
\"fields\": [\n    {\n      \"column\": \"country\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 142,\n        \"samples\": [\n
\"Turkey\",\n          \"Cameroon\",\n          \"Mauritius\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"continent\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Europe\",\n          \"Oceania\",\n          \"Africa\"\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\": 12.917107415241183,\n        \"min\": 23.599,\n
\"max\": 82.603,\n        \"num_unique_values\": 1626,\n
\"samples\": [\n          66.66199999999999,\n          51.445,\n
62.4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"pop\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 106157896,\n        \"min\": 60011,\n        \"max\":
1318683096,\n        \"num_unique_values\": 1704,\n
\"samples\": [\n          23634436,\n          2878220,\n
2156814\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"gdpPercap\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 9857.454542541445,\n        \"min\":
241.1658765,\n        \"max\": 113523.1329,\n
\"num_unique_values\": 1704,\n        \"samples\": [\n
388.0,\n          5599.077872,\n          6650.195573\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"iso_alpha\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 141,\n        \"samples\": [\n
```

```
\"KEN\",\n          \"SAU\",\n          \"HRV\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"iso_num\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 248,\n
\"min\": 4,\n        \"max\": 894,\n        \"num_unique_values\":
141,\n        \"samples\": [\n          404,\n          682,\n
191\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"gapminder"}

# Filter data for the year 2007
gapminder_2007 = gapminder[gapminder["year"] == 2007]

# 1. Scatter Plot: GDP per capita vs Life Expectancy for the year 2007
scatter_fig = px.scatter(
    gapminder_2007,
    x="gdpPercap",
    y="lifeExp",
    size="pop",
    color="continent",
    hover_name="country",
    log_x=True,
    title="GDP per Capita vs Life Expectancy (2007)",
    labels={"gdpPercap": "GDP per Capita", "lifeExp": "Life
Expectancy"},
)
scatter_fig.show()

# 2. Bar Plot: Total Population by Continent in 2007
bar_fig = px.bar(
    gapminder_2007.groupby("continent", as_index=False).sum(),
    x="continent",
    y="pop",
    color="continent",
    title="Total Population by Continent (2007)",
    labels={"pop": "Total Population"},
)
bar_fig.show()

# 3. 3D Scatter Plot: GDP per Capita, Life Expectancy, and Population
in 2007
scatter_3d_fig = px.scatter_3d(
    gapminder_2007,
    x="gdpPercap",
    y="lifeExp",
    z="pop",
    color="continent",
    size="pop",
    hover_name="country",
    title="3D Scatter Plot: GDP per Capita, Life Expectancy, and
```

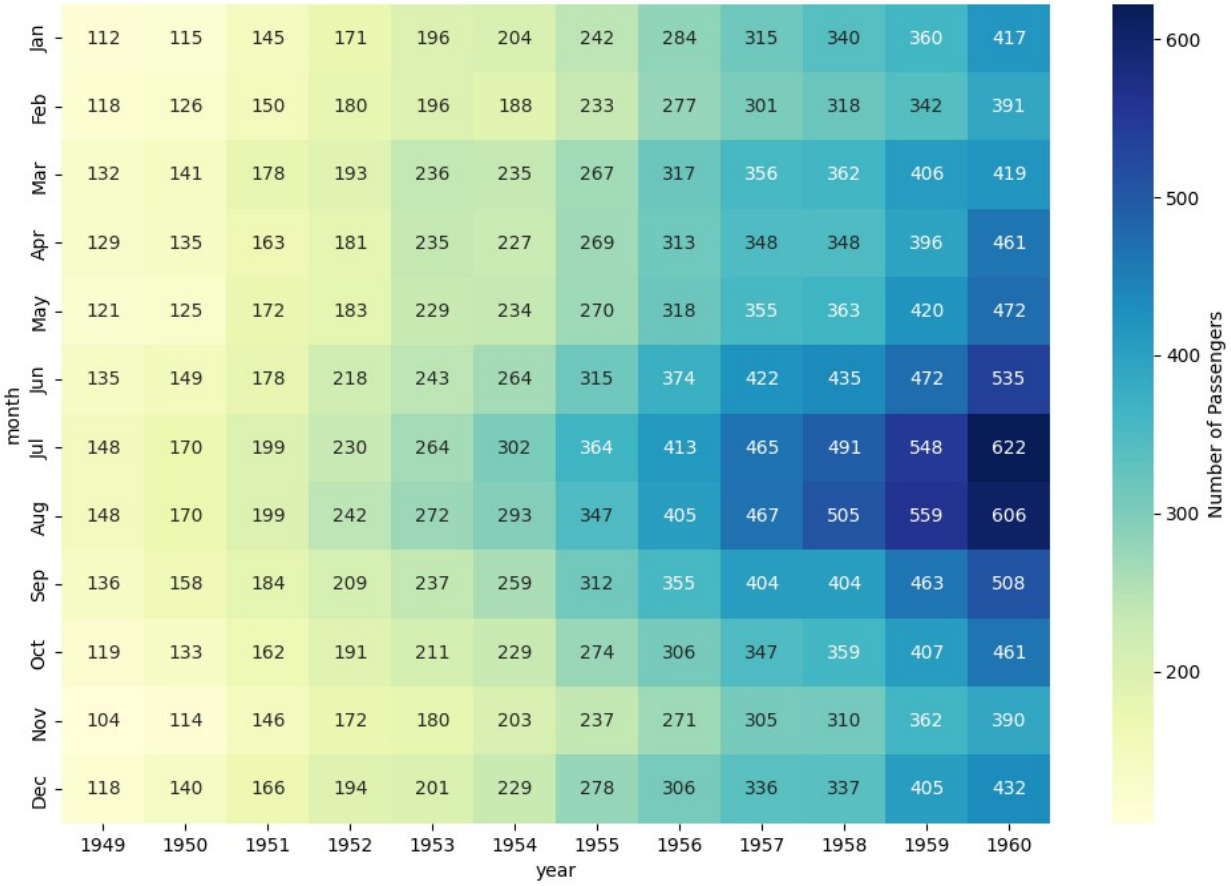
```
Population (2007)",
    labels={"gdpPercap": "GDP per Capita", "lifeExp": "Life
Expectancy", "pop": "Population"},
)
scatter_3d_fig.show()

# 4. Line Plot: Evolution of Life Expectancy Over Time for Selected
Countries
selected_countries = ["India", "UAE", "Germany", "China", "Australia",
"Russia"]
gapminder_selected =
gapminder[gapminder["country"].isin(selected_countries)]
line_fig = px.line(
    gapminder_selected,
    x="year",
    y="lifeExp",
    color="country",
    title="Evolution of Life Expectancy Over Time",
    labels={"lifeExp": "Life Expectancy", "year": "Year"},
)
line_fig.show()
```

Load the **flights** dataset from Seaborn. Create a heatmap to show the number of passengers for each month and year. Customize the heatmap with labels and a color bar.

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

# Load the flights dataset
flights = sns.load_dataset('flights')

flights
```


```
{"summary":"{\n  \"name\": \"flights\",\n  \"rows\": 144,\n
\"fields\": [\n    {\n      \"column\": \"year\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 1949,\n        \"max\": 1960,\n
\"num_unique_values\": 12,\n        \"samples\": [\n          1959,\n
1958,\n          1949\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"month\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 12,\n        \"samples\": [\n
\"Nov\",\n          \"Oct\",\n          \"Jan\"\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"passengers\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
119,\n        \"min\": 104,\n        \"max\": 622,\n
\"num_unique_values\": 118,\n        \"samples\": [\n          293,\n
340,\n          121\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"flights"}

# Pivot the data to create a matrix format suitable for a heatmap
flights_pivot = flights.pivot(index="month", columns="year",
values="passengers")

# Create the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(
    flights_pivot,
    annot=True,  # Annotate cells with the data values
    fmt="d",  # Format annotations as integers
    cmap="YlGnBu",  # Colormap for the heatmap
    cbar_kws={"label": "Number of Passengers"},  # Customize the color
bar
)

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




```
# Add labels and title
plt.title("Monthly Number of Passengers (1949-1960)", fontsize=16)
plt.xlabel("Year", fontsize=12)
plt.ylabel("Month", fontsize=12)

Text(0, 0.5, 'Month')
```


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

```
<Figure size 640x480 with 0 Axes>
```

Load the **tips** dataset. First, create a Seaborn boxplot to show the distribution of total_bill across different days of the week. Then, create an interactive Plotly pie chart showing the percentage contribution of each day to the total number of records.

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

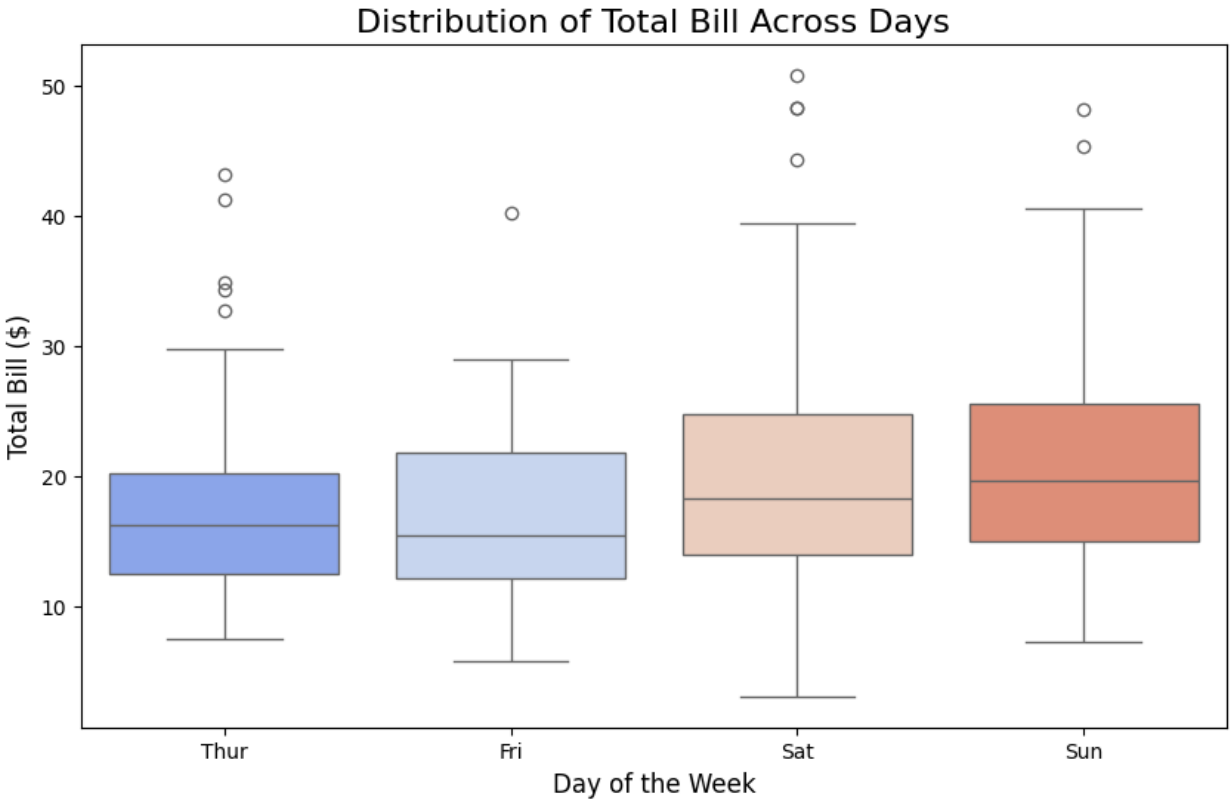
```
import pandas as pd

# Load the tips dataset
tips = sns.load_dataset('tips')

tips
```


```
{"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"}
```


```
# Seaborn boxplot for total_bill distribution across days
plt.figure(figsize=(10, 6))
sns.boxplot(x="day", y="total_bill", data=tips, palette="coolwarm")
plt.title("Distribution of Total Bill Across Days", fontsize=16)
plt.xlabel("Day of the Week", fontsize=12)
```

```
plt.ylabel("Total Bill ($)", fontsize=12)
plt.show()
```

```
<ipython-input-35-b8bd058d4bc5>:3: FutureWarning:


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




```
# Create a DataFrame with counts for each day
day_counts = tips["day"].value_counts().reset_index()
day_counts.columns = ["day", "count"]

# Plotly pie chart for percentage contribution of each day
fig = px.pie(
    day_counts,
    names="day",
    values="count",
    title="Percentage Contribution of Each Day to Total Records",
    color_discrete_sequence=px.colors.sequential.Blues,
    hole=0.4  # For a donut chart look
```

```
)
fig.update_traces(textinfo="percent+label")
fig.show()
```