



Data Visualizations using Seaborn

- **Author:** Basel Amr Barakat
- **Email:** baselamr52@gmail.com
- **Date:** 2024-12-25
- **Task Name:** Data Visualizations using Seaborn
- **Task Number:** 23
- **Part:** Seaborn Statistical Visualization
- **Module:** Python Programming Language for AI / ML
- **Submit Number:** 1

Description:

Students will use Seaborn to create distribution, correlation, and categorical plots, learning to customize and interpret these visualizations for better data insights.

Requirements:

1. Requirement 1: Basic Plotting with Seaborn.

◦ Description:

Description: Use Seaborn to create a line plot and scatter plot, applying a suitable Seaborn theme and color palette. Customize the plots with titles, labels, and appropriate markers.

2. Requirement 2: Distribution and Categorical Data Visualization

◦ Description:

Visualize the distribution of a numerical column using a box plot and a violin plot. Compare the two plots in terms of the insights they provide. Create a bar plot and a count plot to represent categorical data, and analyze the differences. Arrange of the subplots with gridlines, legends, and different colors.

3. Requirement 3: Correlation Visualization with Heatmap.

◦ Description:

Generate a correlation matrix heatmap using Seaborn to visualize relationships between numerical features in the dataset. Annotate the heatmap with correlation values and interpret the insights gained.



1. Introduction & Objective

This notebook demonstrates how to use Seaborn for data visualization, focusing on distribution, correlation, and categorical plots.

The main objectives are to:

- Use Seaborn for basic plotting (line plot, scatter plot).
- Visualize distribution and categorical data (box plot, violin plot, bar plot, and count plot).
- Create a heatmap to visualize correlations between numerical features.

✓ 2. Setting Up the Environment

```
1 # Importing necessary libraries
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import pandas as pd
6 # Ignore all the warnings
7 import warnings
8 warnings.filterwarnings("ignore")
```

✓ 3. Requirement 1

Task Name : Basic Plotting with Seaborn.

Description : Use Seaborn to create a line plot and scatter plot, applying a suitable Seaborn theme and color palette.

✓ 3.1 Set Seaborn theme and color palette for consistency in visualizations

```
1 # Set Seaborn theme and color palette for consistency in visualizations
2 sns.set_theme(style="darkgrid", palette="pastel")
3 sns.set_context("notebook")
4 sns.set_palette("coolwarm")
```

✓ 3.2 Use Seaborn to create a line plot

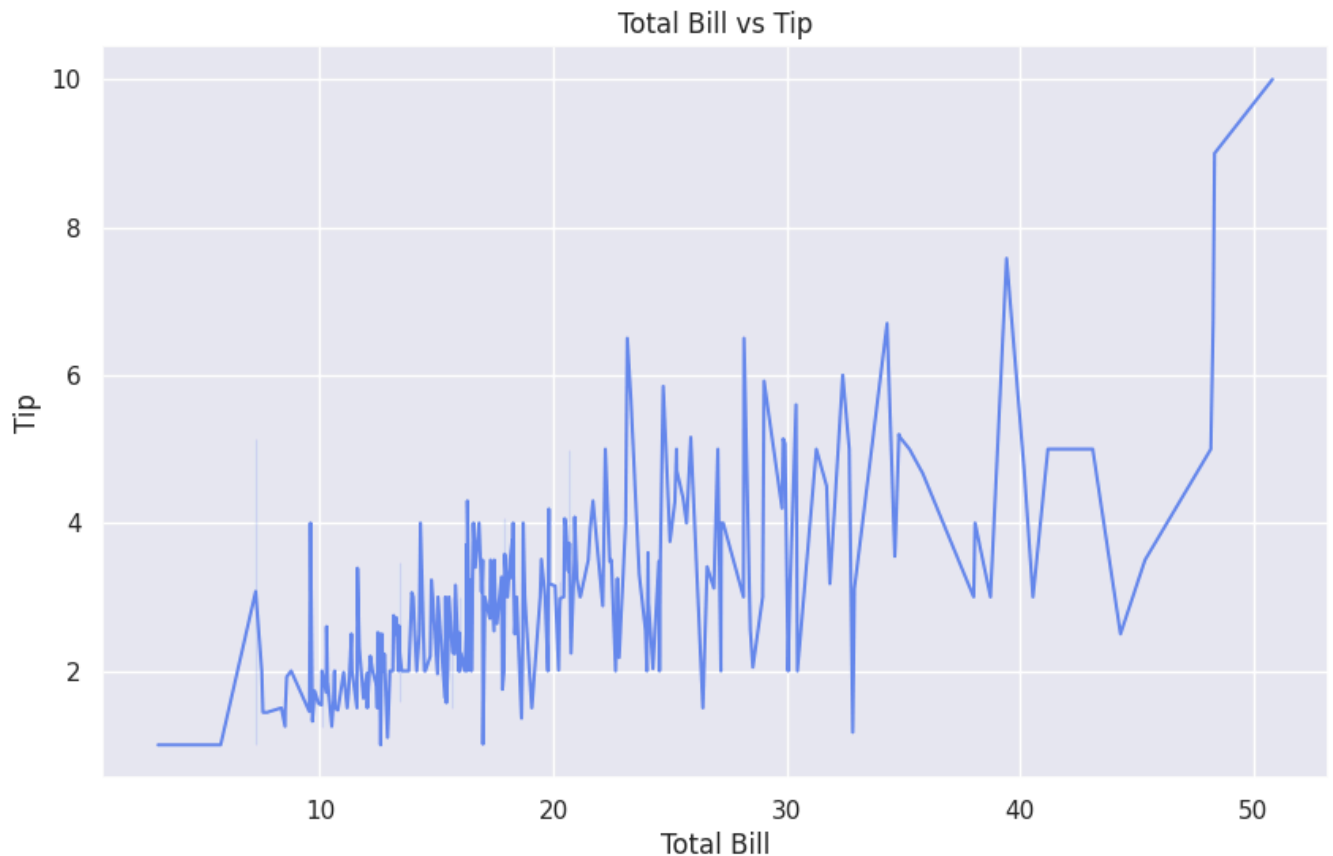
```
1 # Sample dataset for plotting
2 data = sns.load_dataset("tips")
3 display(data.head())
4 # Get the number of records and features in the dataset
5 print(f"Number of records: {data.shape[0]}")
6 print(f"Number of features: {data.shape[1]}")
7 # Line Plot
8 plt.figure(figsize=(10, 6))
9 sns.lineplot(data=data, x="total_bill", y="tip", markers=True)
10 plt.title('Total Bill vs Tip')
11 plt.xlabel('Total Bill')
12 plt.ylabel('Tip')
13 plt.savefig('01.02_line_plot.png', format="png", dpi=800)
14 plt.grid(True)
15 plt.show()
```



	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	

Number of records: 244

Number of features: 7



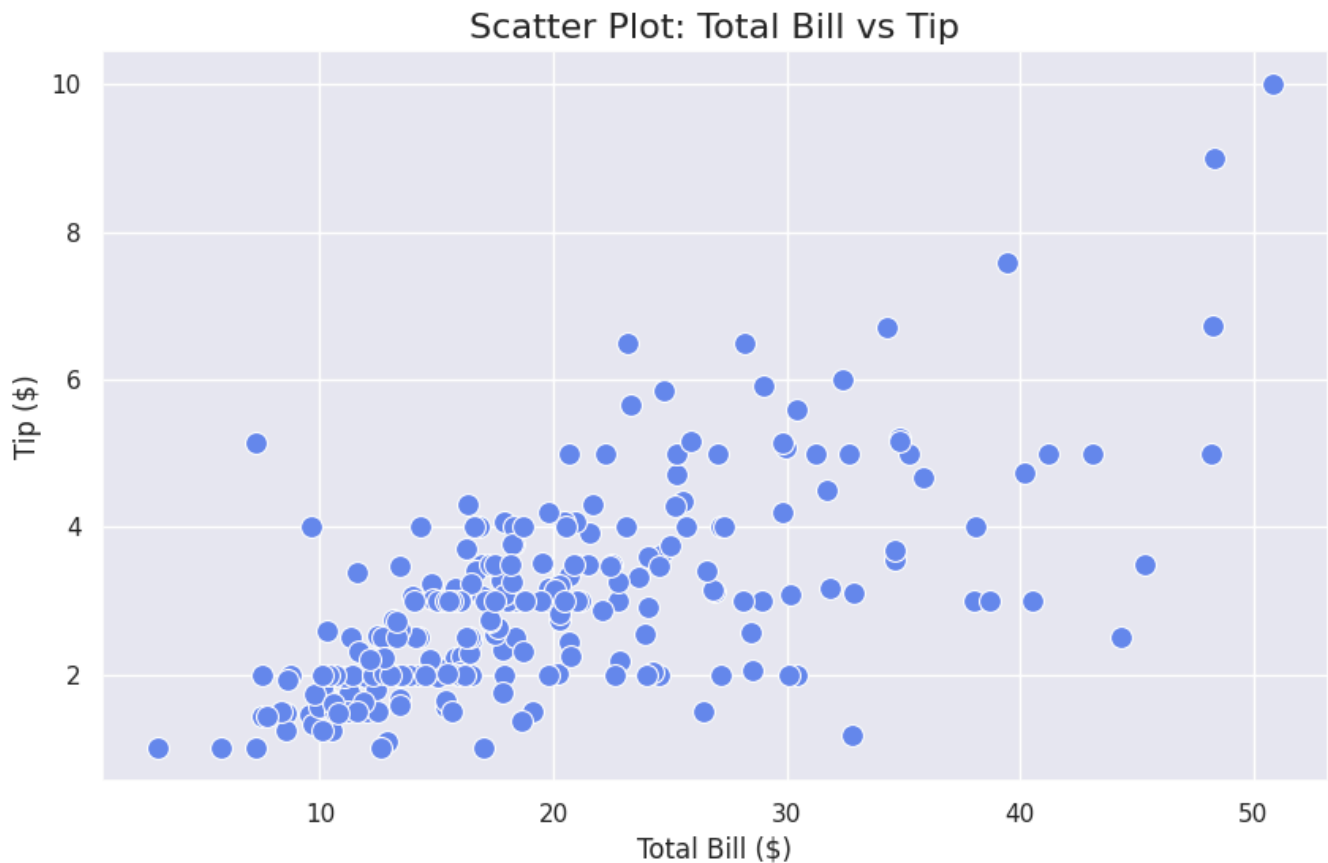
✓ 3.3 Use Seaborn to create a scatter plot

```

1 # Scatter Plot
2 plt.figure(figsize=(10, 6))
3 sns.scatterplot(x="total_bill", y="tip", data=data, marker="o", s=100)
4 plt.title("Scatter Plot: Total Bill vs Tip", fontsize=16)

```

```
5 plt.xlabel("Total Bill ($)", fontsize=12)
6 plt.ylabel("Tip ($) ", fontsize=12)
7 plt.grid(True)
8 plt.show()
```



✓ 4. Requirement 2

Task Name : Distribution and Categorical Data Visualization

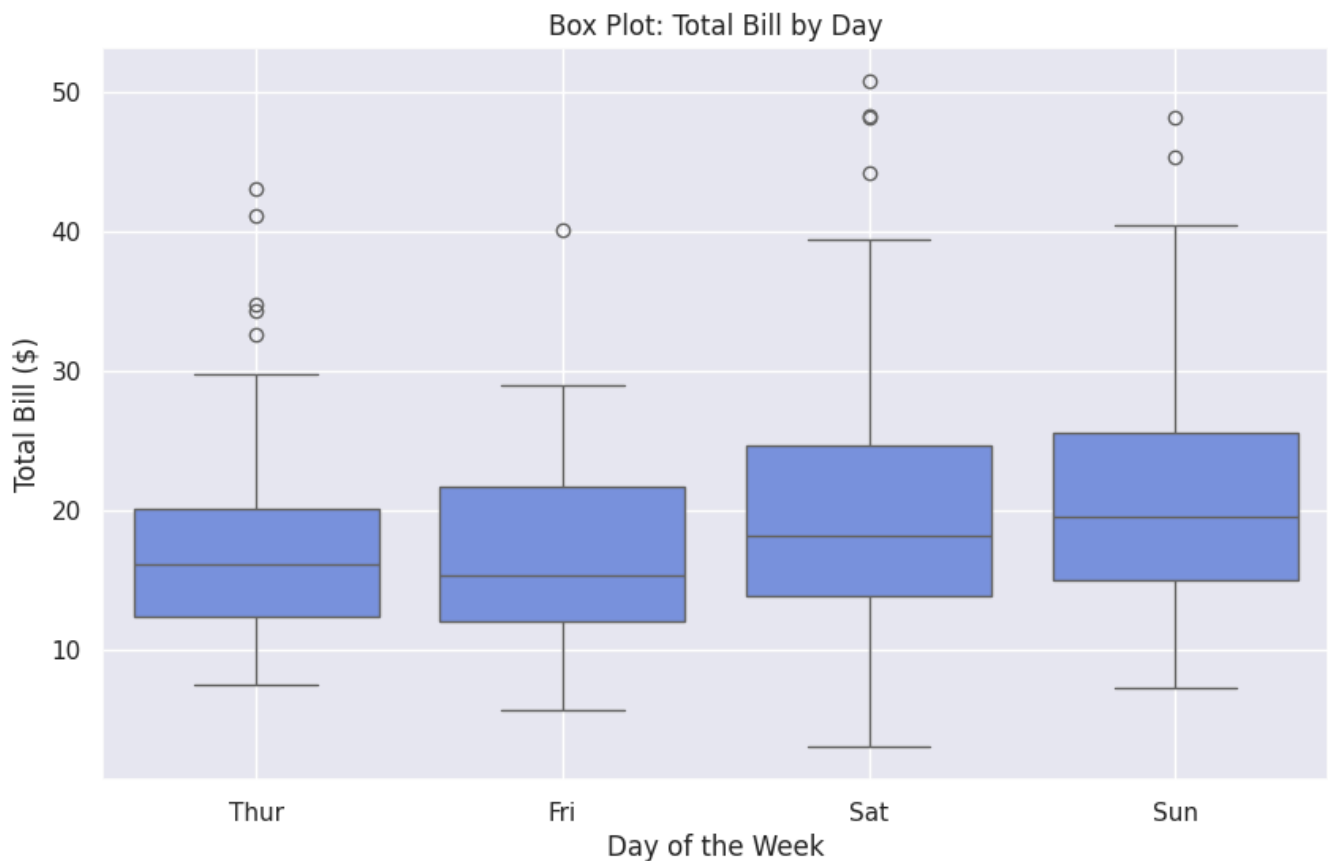
Description :

- Visualize the distribution of a numerical column using a box plot and a violin plot. Compare the two plots in terms of the insights they provide.
- Create a bar plot and a count plot to represent categorical data, and analyze the differences.

✓ 4.1 Box Plot for Distribution Visualization

- Provides a summary of data distribution, highlighting median, quartiles, and outliers.
- Useful for identifying data spread.

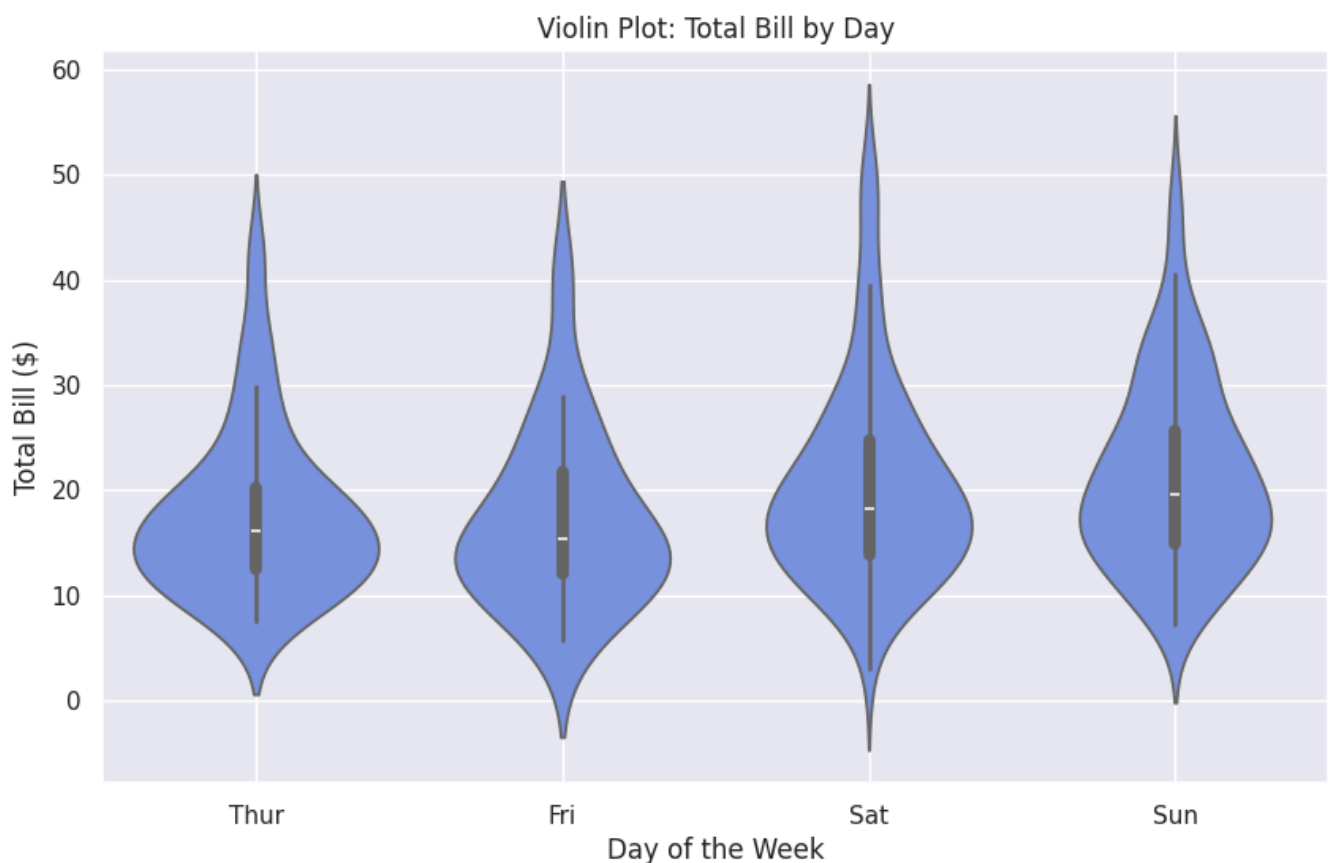
```
1 # Plotting BoxPlot for Distribution Visualization
2 plt.figure(figsize=(10, 6))
3 sns.boxplot(x="day", y="total_bill", data=data)
4 plt.title("Box Plot: Total Bill by Day")
5 plt.xlabel("Day of the Week")
6 plt.ylabel("Total Bill ($)")
7 plt.grid(True)
8 plt.savefig('02.01_BoxPlot.png', format="png", dpi=800)
9 plt.show()
```



✓ 4.2 Violin Plot for Distribution Visualization

- Combines aspects of box plot with density estimation.
- Reveals distribution shape and frequency of values across different categories.

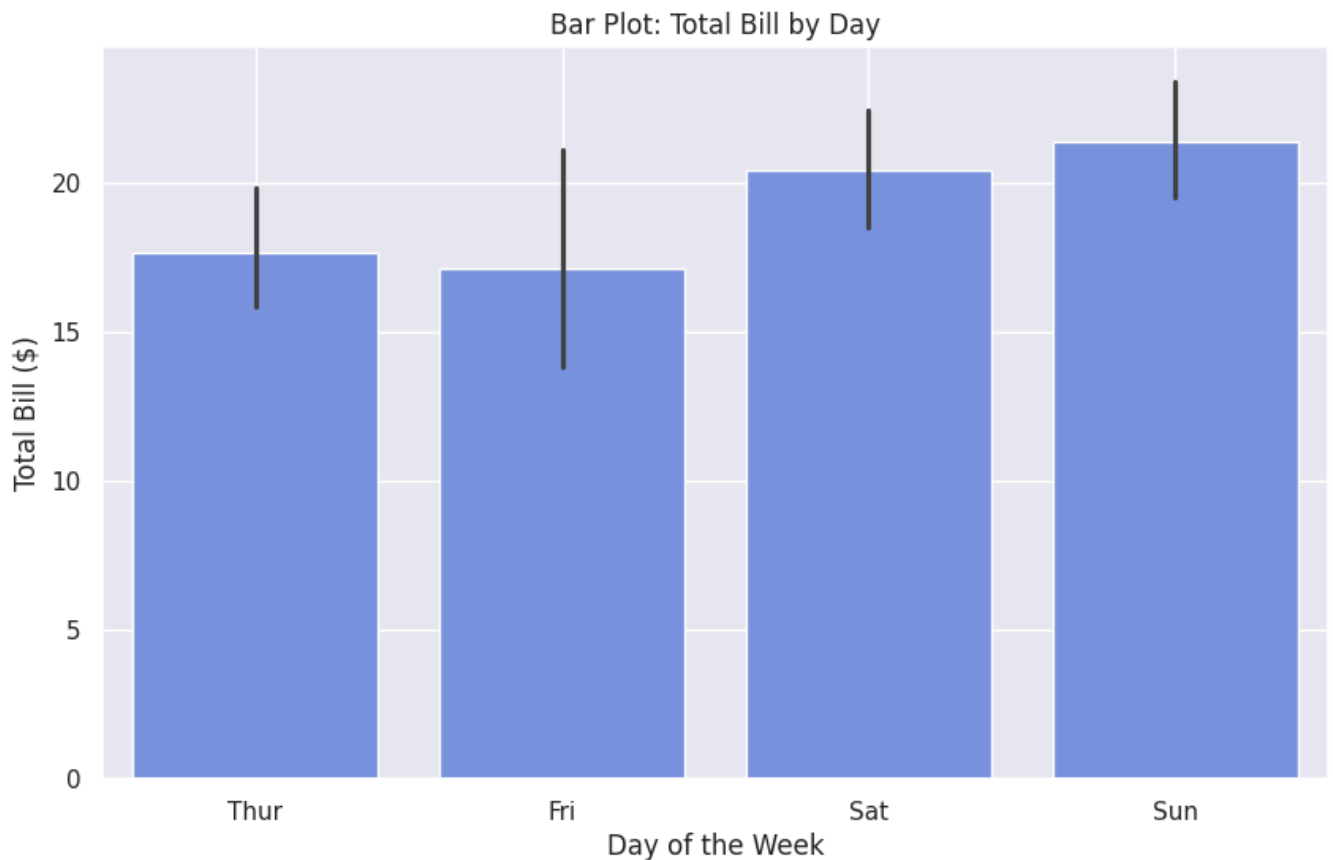
```
1 # Plot Violin Plot for Distribution Visualization
2 plt.figure(figsize=(10, 6))
3 sns.violinplot(x="day", y="total_bill", data=data)
4 plt.title("Violin Plot: Total Bill by Day")
5 plt.xlabel("Day of the Week")
6 plt.ylabel("Total Bill ($)")
7 plt.grid(True)
8 plt.savefig('02.02_ViolinPlot.png', format="png", dpi=800)
9 plt.show()
```



✓ 4.3 Bar Plot for Categorical Data Visualization

- Displays aggregated data (e.g., average total bill) per category.
- Useful for comparing numerical summaries across categories.

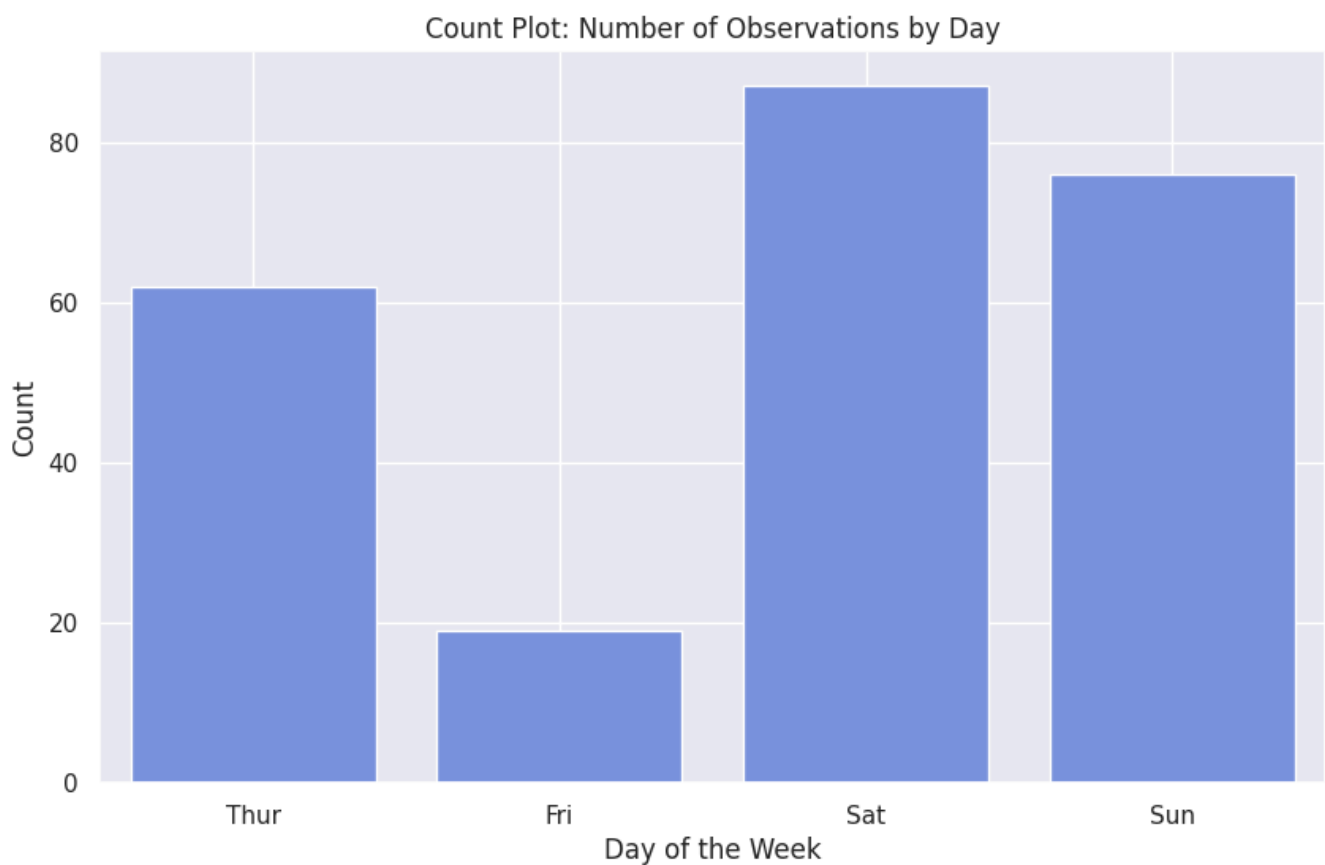
```
1 # Bar Plot for Categorical Data Visualization
2 plt.figure(figsize=(10, 6))
3 sns.barplot(x="day", y="total_bill", data=data)
4 plt.title("Bar Plot: Total Bill by Day")
5 plt.xlabel("Day of the Week")
6 plt.ylabel("Total Bill ($)")
7 plt.grid(True)
8 plt.savefig('02.03_BarPlot.png', format="png", dpi=800)
9 plt.show()
```



✓ 4.4 Count Plot for Categorical Data Visualization

- Displays the frequency/count of each category.
- Useful for understanding category distribution.

```
1 # Count Plot for Categorical Data Visualization
2 plt.figure(figsize=(10, 6))
3 sns.countplot(x="day", data=data)
4 plt.title("Count Plot: Number of Observations by Day")
5 plt.xlabel("Day of the Week")
6 plt.ylabel("Count")
7 plt.grid(True)
8 plt.savefig('02.04_CountPlot.png', format="png", dpi=800)
9 plt.show()
```



4.5 Comparison: Box Plot vs Violin Plot

Aspect	Box Plot	Violin Plot
Purpose	Displays summary statistics (median, quartiles, outliers).	Combines box plot summary with data density visu
Key Insights	Effective for identifying outliers and overall spread of data.	Provides detailed insight into the distribution's sha
Best Use Case	Comparing statistical summaries across categories.	Understanding distribution density and patterns in
Visual Complexity	Simple and clean.	Slightly more complex but informative.
Limitation	Does not show fine details of the distribution.	Can be harder to interpret with large datasets.
Ideal Scenario	When focusing on statistical summaries.	When distribution density is a key focus.

4.6 Comparison: Bar Plot vs Count Plot

Aspect	Bar Plot	Count Plot
Purpose	Represents aggregated numerical data (e.g., mean, sum).	Represents frequency/count of categories.
Key Insights	Highlights statistical summaries per category.	Shows raw counts per category.
Best Use Case	Comparing average or total values across categories.	Comparing occurrences or frequency of categories.
Visual Complexity	Moderate.	Very simple and direct.
Limitation	Can be misleading if data aggregation isn't clear.	Cannot display numerical aggregates.
Ideal Scenario	When numerical aggregation is required.	When raw frequency counts are the focus.

✓ 

5. Requirement 3

Task Name : Correlation Visualization with Heatmap.


Description :

- Generate a correlation matrix heatmap using Seaborn to visualize relationships between numerical features in the dataset.
- Annotate the heatmap with correlation values and interpret the insights gained




✓

5.1 Generate a colloration matrix

```
1 # Generate a correlation matrix
2 corr_matrix = data.corr(numeric_only=True)
3 display(corr_matrix)
```



	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



Next
steps:

Generate code
with `corr_matrix`

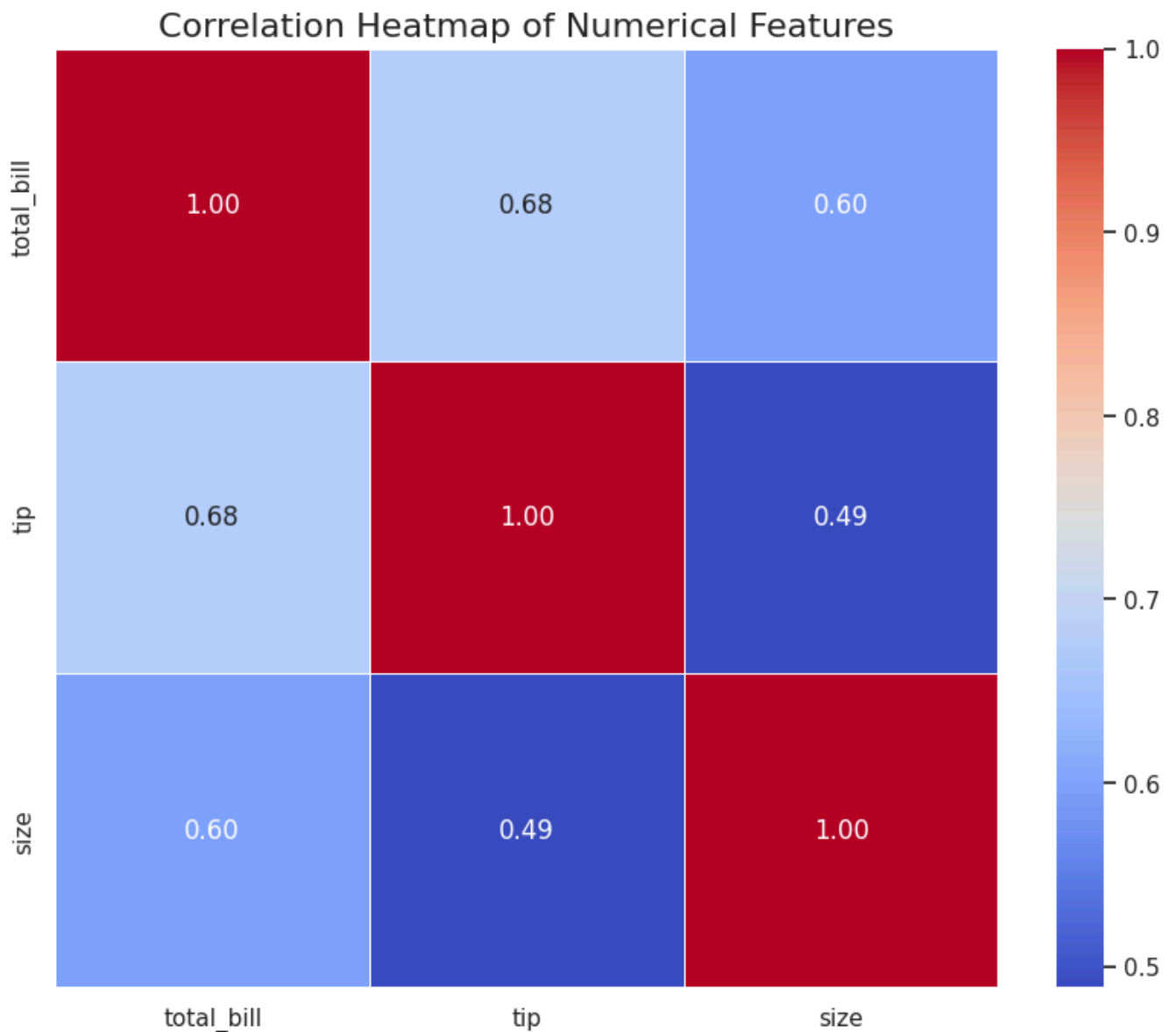


View recommended
plots

New interactive
sheet

5.2 Annotate the heatmap with correlation values and interpret the insights gained

```
1 # Create a heatmap
2 plt.figure(figsize=(10, 8))
3 sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
4 plt.title("Correlation Heatmap of Numerical Features", fontsize=16)
5 plt.savefig('03_CorrelationHeatmap.png', format="png", dpi=800)
6 plt.show()
```



✓ 6. Requirement 4 "Extra"

Task Name : Advanced Pair Plots and Joint Plots

```
1 # Pair Plot
2 print("Pair Plot")
3 sns.pairplot(data[["total_bill", "tip", "size"]])
4 plt.suptitle("Pair Plot of Numerical Features", fontsize=16)
5 plt.savefig('04.01_PairPlot.png', format="png", dpi=800)
6 plt.show()
```

```
7
8 # Joint Plot
9 print("Joint Plot")
10 sns.jointplot(x="total_bill", y="tip", data=data, kind="scatter", color="blue")
11 plt.suptitle("Joint Plot: Total Bill vs Tip", fontsize=16)
12 plt.savefig('04.02_JointPlot.png',format="png",dpi=800)
13 plt.show()
```

Pair Plot

