Sprints

– Task 23 –

Data Visualizations using Seaborn

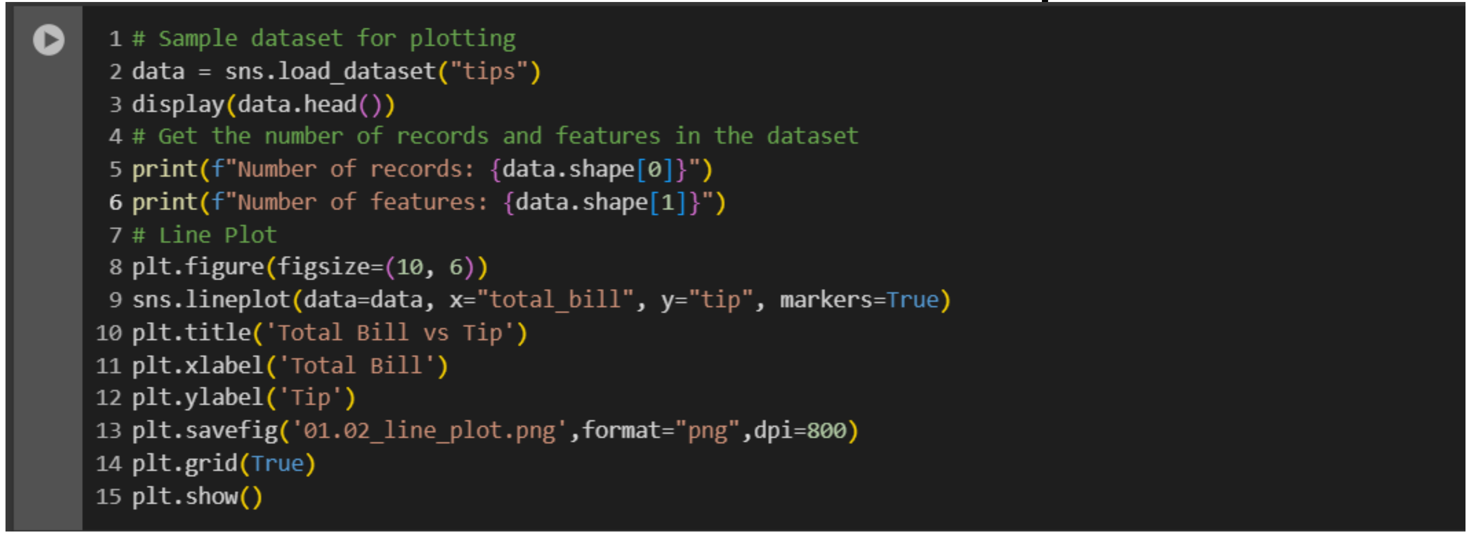
Name: Basel Amr Barakat
Email: baselamr52@gmail.com

Requirement One

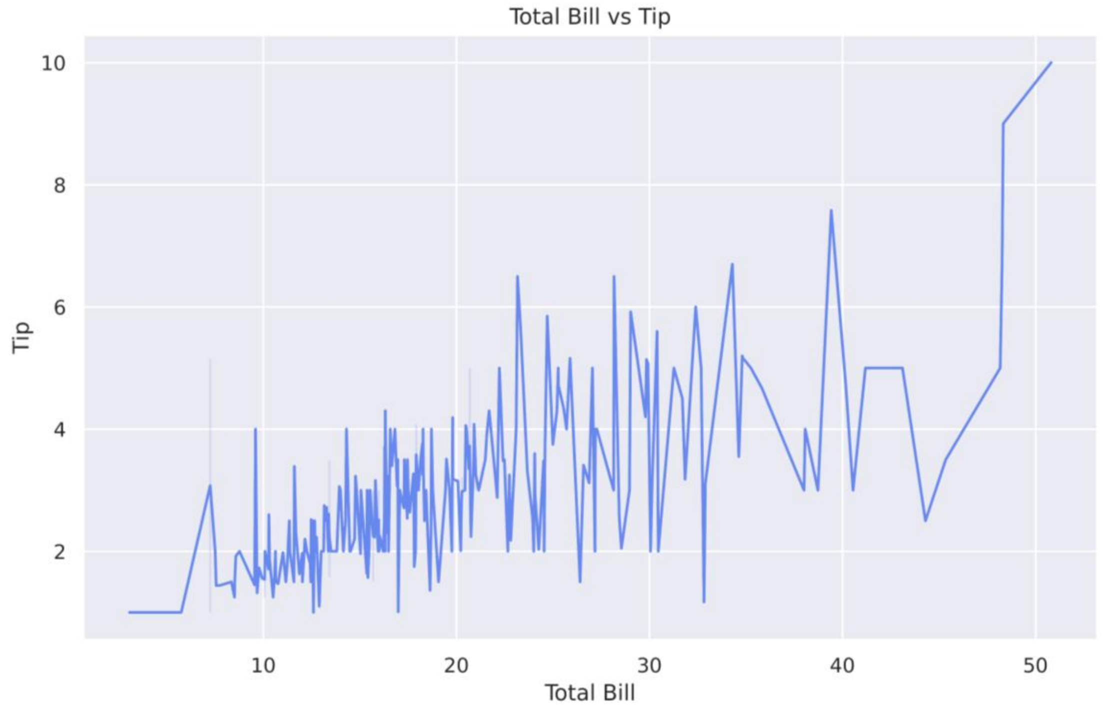
Basic Plotting with Seaborn.

Sprints

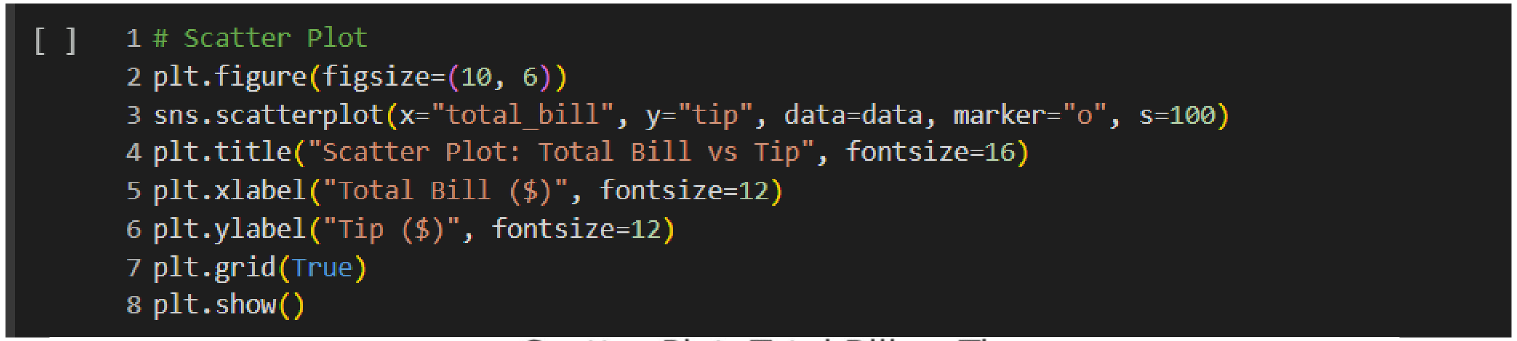
1.1 Use Seaborn to create a line plot



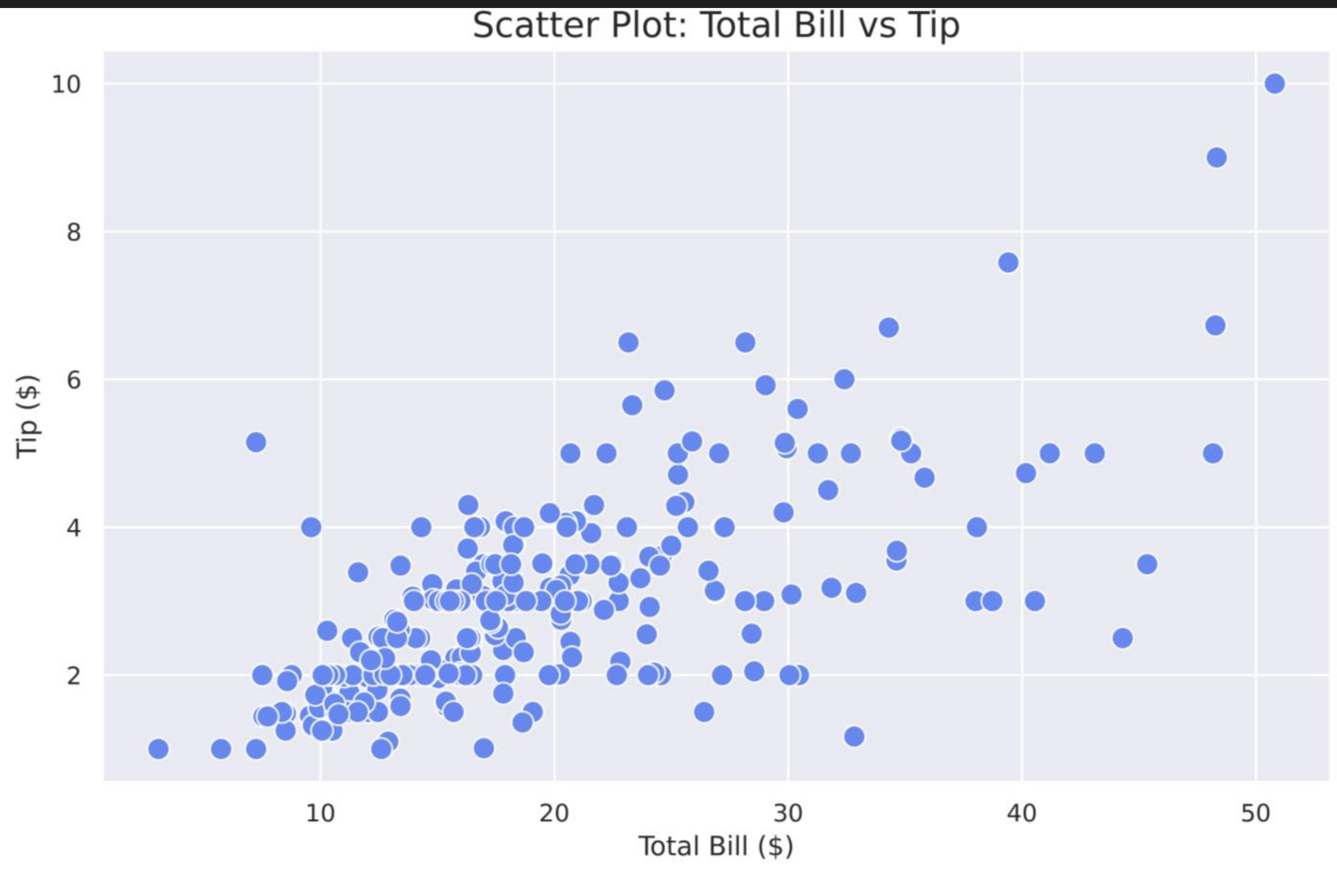
```python
# Sample dataset for plotting
data = sns.load_dataset("tips")
display(data.head())
# Get the number of records and features in the dataset
print(f"Number of records: {data.shape[0]}")
print(f"Number of features: {data.shape[1]}")
# Line Plot
plt.figure(figsize=(10, 6))
sns.lineplot(data=data,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



1.2 Use Seaborn to create a scatter



```
[ ]
# Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x="total_bill", y="tip", data=data, marker="o", s=100)
plt.title("Scatter Plot: Total Bill vs Tip", fontsize=16)
plt.xlabel("Total Bill ($)", fontsize=12)
plt.ylabel("Tip ($)", fontsize=12)
plt.grid(True)
plt.show()
```

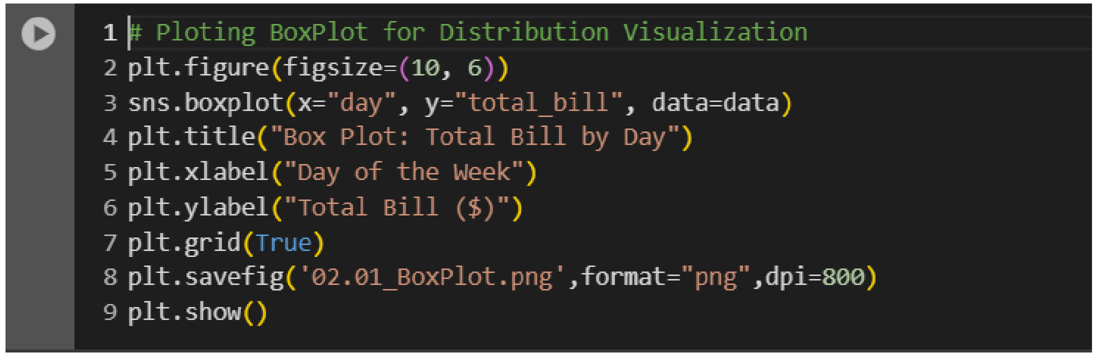


Sprints

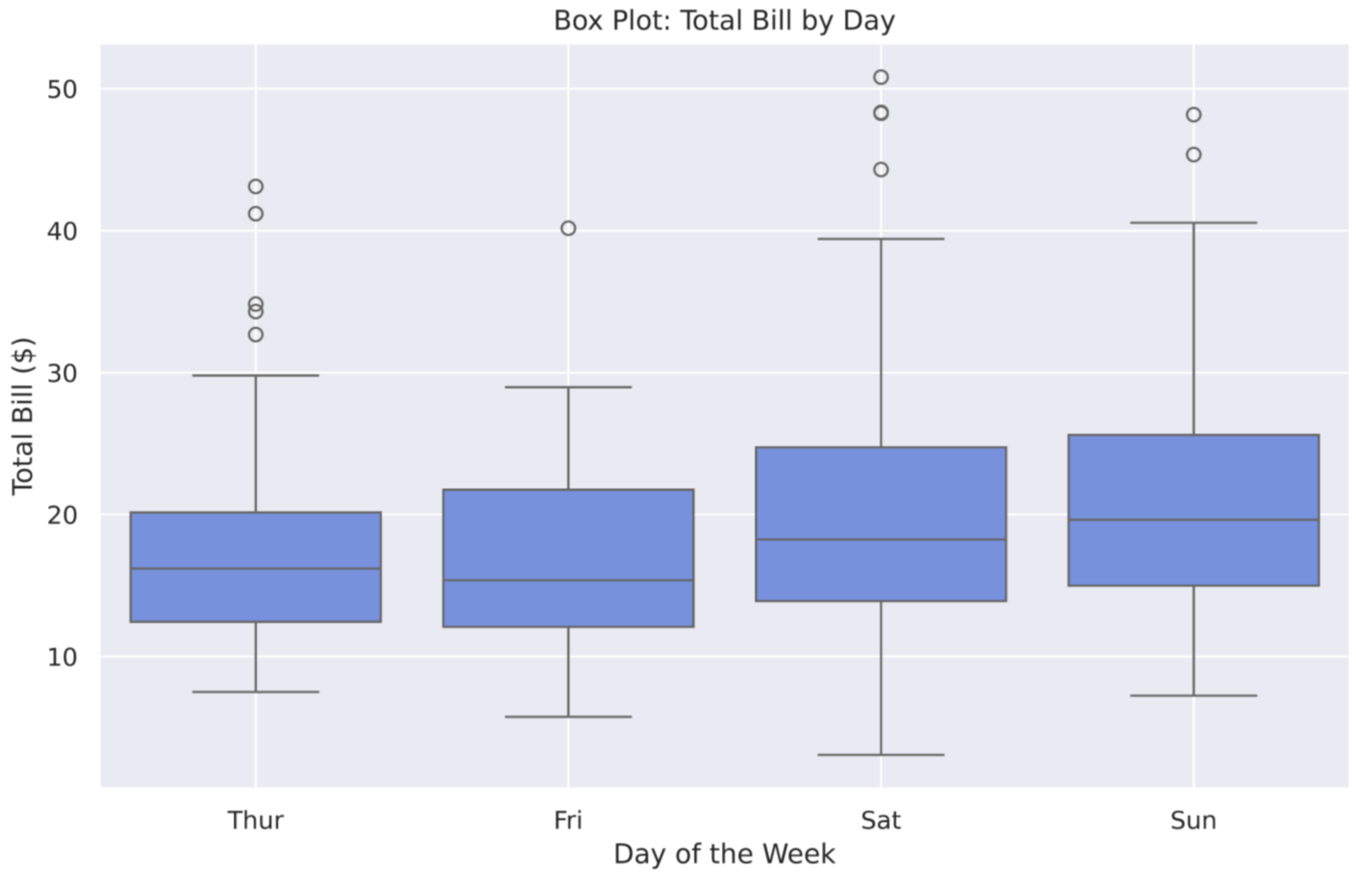
Requirement Two

Distribution and Categorical Data Visualization

2.1 Box Plot for Distribution Visualization

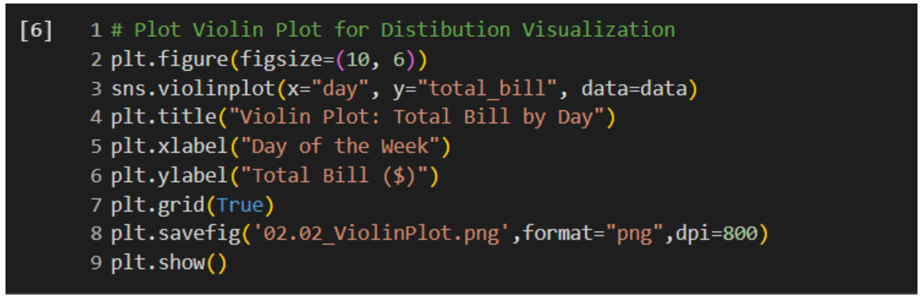


```python
# Ploting BoxPlot for Distribution Visualization
plt.figure(figsize=(10, 6))
sns.boxplot(x="day", y="total_bill", data=data)
plt.title("Box Plot: Total Bill by Day")
plt.xlabel("Day of the Week")
plt.ylabel("Total Bill ($)")
plt.grid(True)
plt.savefig('02.01_BoxPlot.png',format="png",dpi=800)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

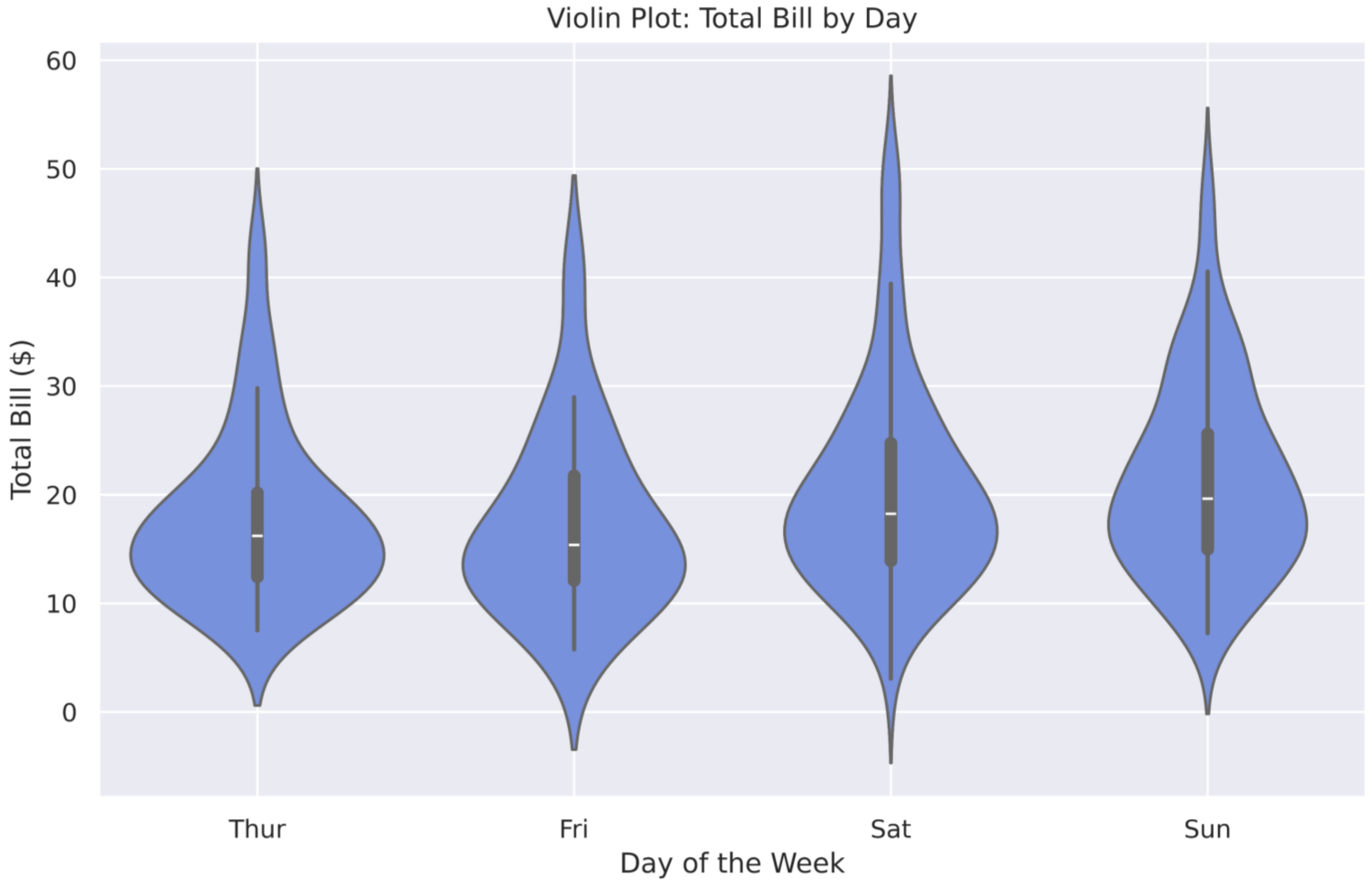


Sprints

2.2 Violin Plot for Distribution Visualization

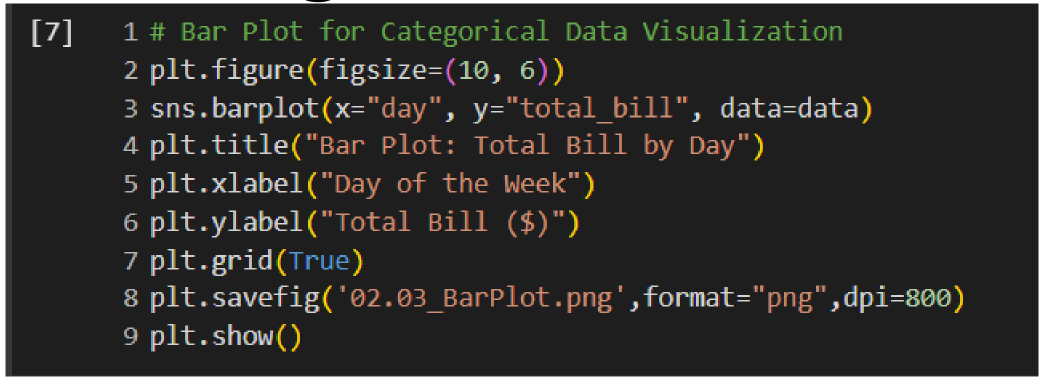


```
[6]
# Plot Violin Plot for Distibution Visualization
plt.figure(figsize=(10, 6))
sns.violinplot(x="day", y="total_bill", data=data)
plt.title("Violin Plot: Total Bill by Day")
plt.xlabel("Day of the Week")
plt.ylabel("Total Bill ($)")
plt.grid(True)
plt.savefig('02.02_ViolinPlot.png',format="png",dpi=800)
plt.show()
```

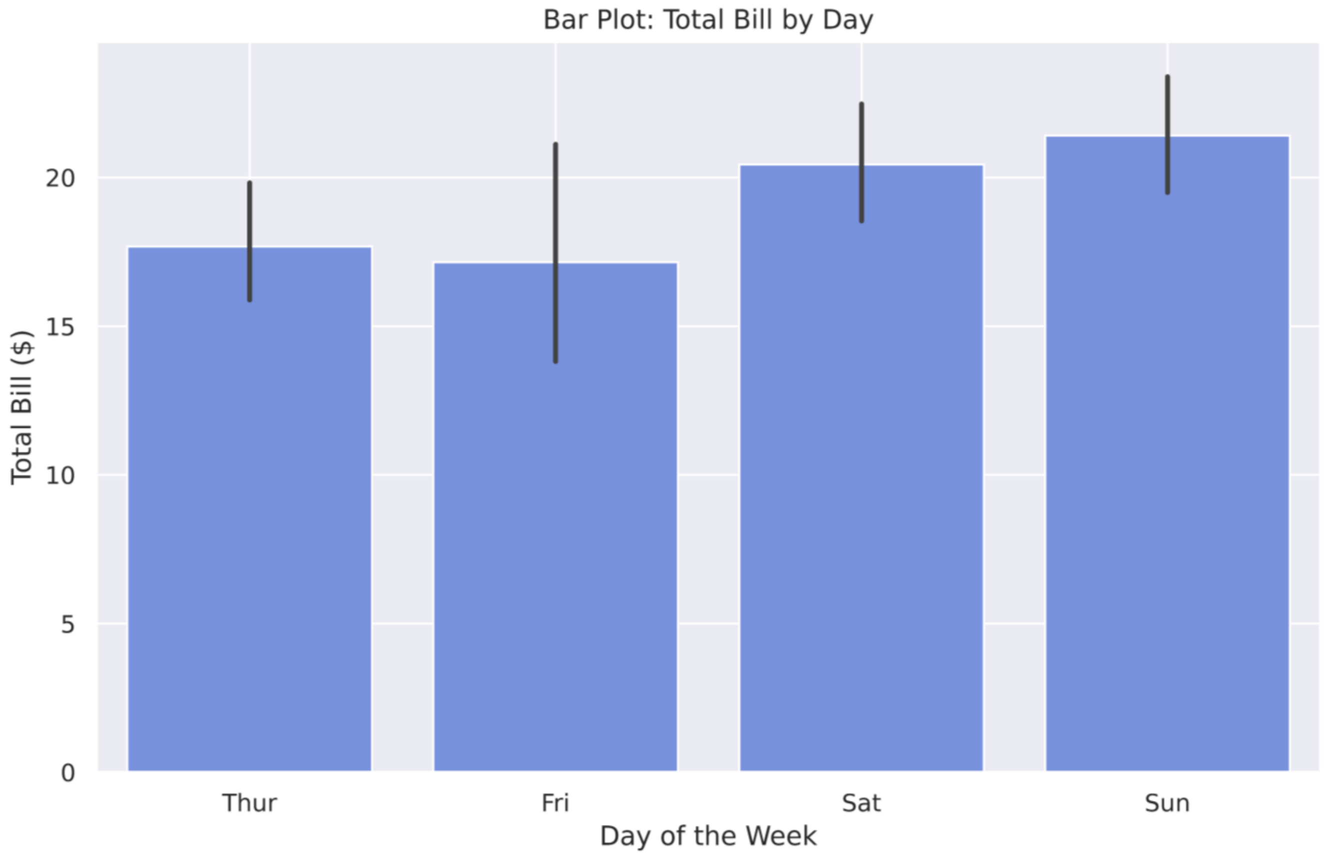


Sprints

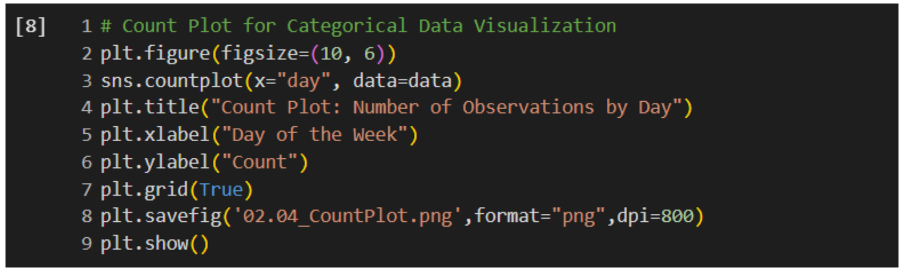
2.3 Bar Plot for Categorical Data Visualization



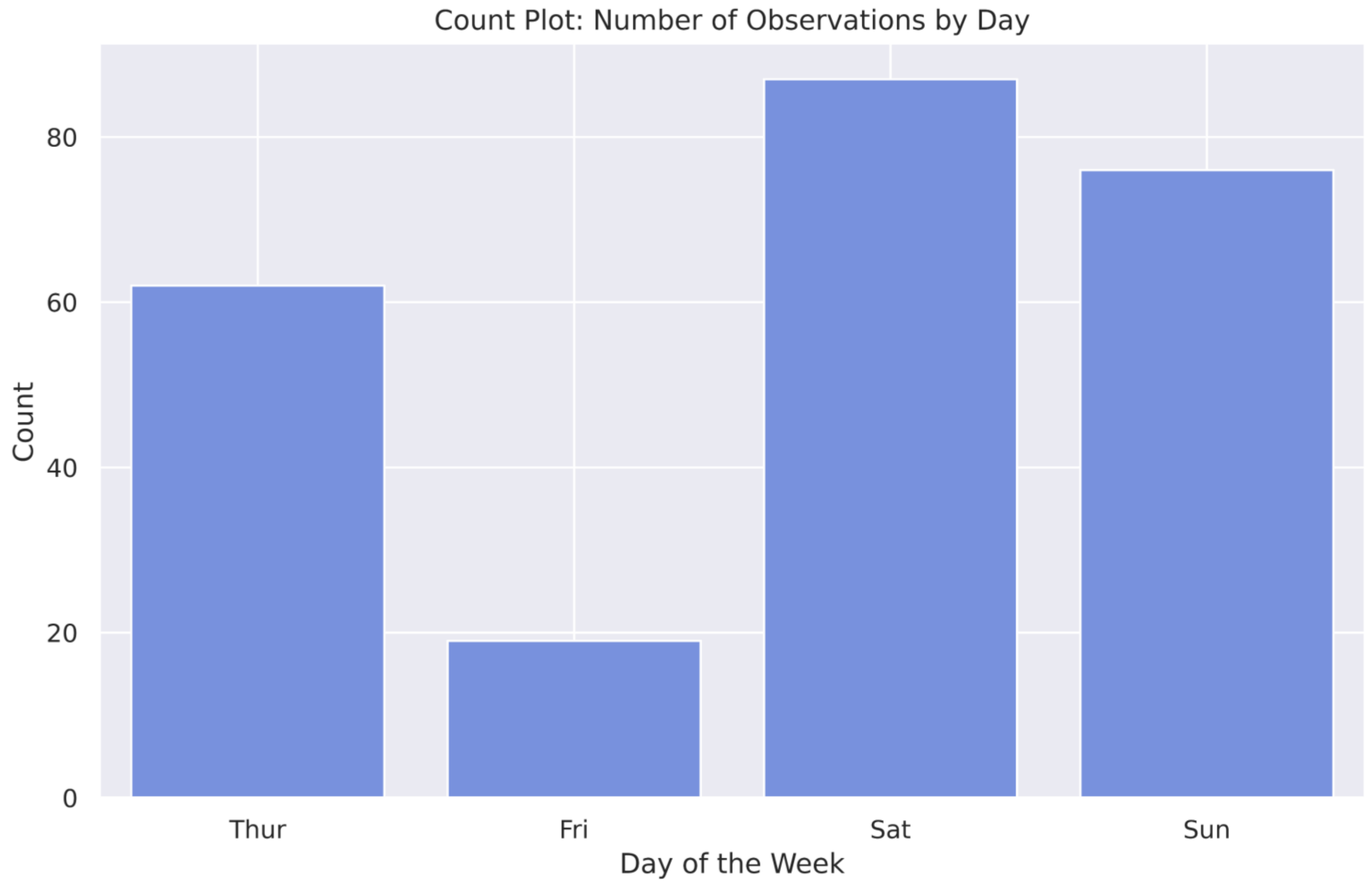
```
[7]
# Bar Plot for Categorical Data Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x="day", y="total_bill", data=data)
plt.title("Bar Plot: Total Bill by Day")
plt.xlabel("Day of the Week")
plt.ylabel("Total Bill ($)")
plt.grid(True)
plt.savefig('02.03_BarPlot.png',format="png",dpi=800)
plt.show()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



2.4 Count Plot for Categorical Data Visualization

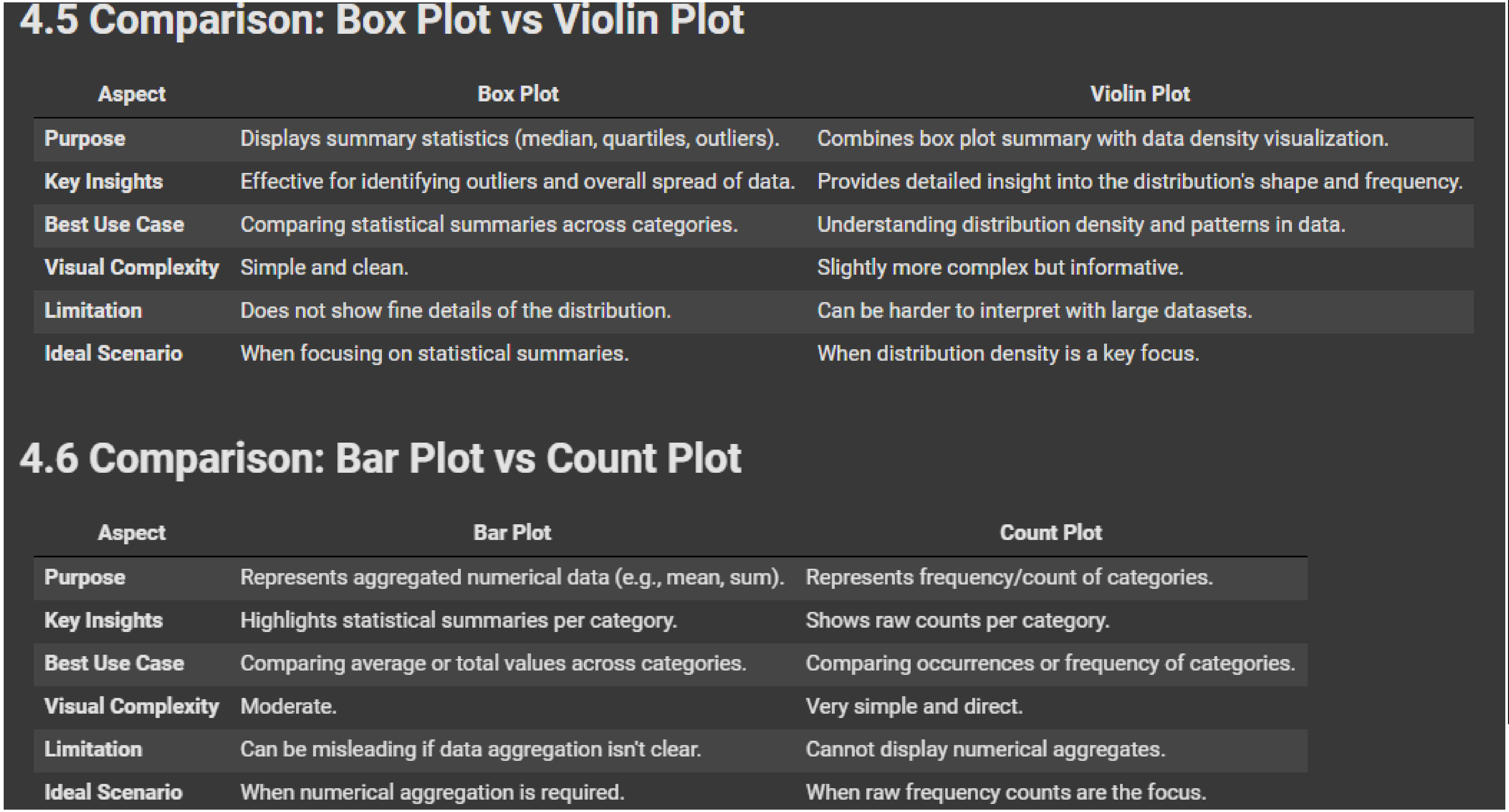


```
[8]
# Count Plot for Categorical Data Visualization
plt.figure(figsize=(10, 6))
sns.countplot(x="day", data=data)
plt.title("Count Plot: Number of Observations by Day")
plt.xlabel("Day of the Week")
plt.ylabel("Count")
plt.grid(True)
plt.savefig('02.04_CountPlot.png',format="png",dpi=800)
plt.show()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



Sprints

2.5 Comparison



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 visualization.
Key Insights	Effective for identifying outliers and overall spread of data.	Provides detailed insight into the distribution's shape and frequency.
Best Use Case	Comparing statistical summaries across categories.	Understanding distribution density and patterns in data.
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.

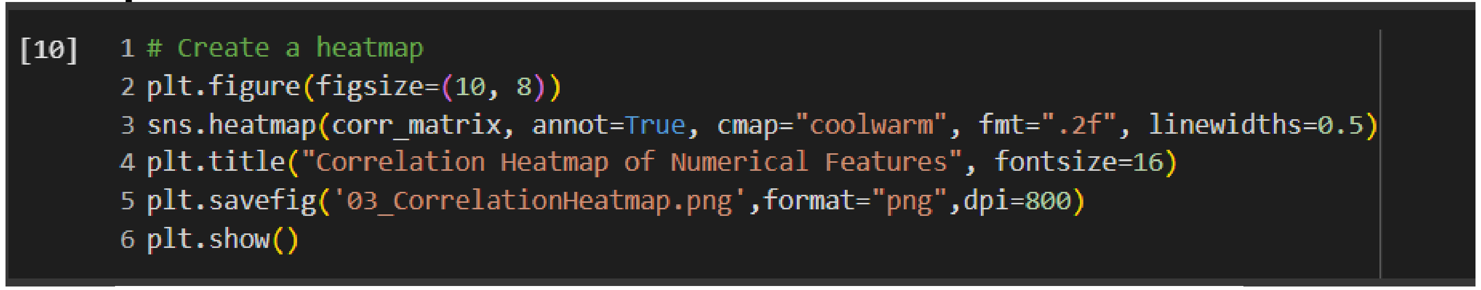
Sprints

Requirement Three

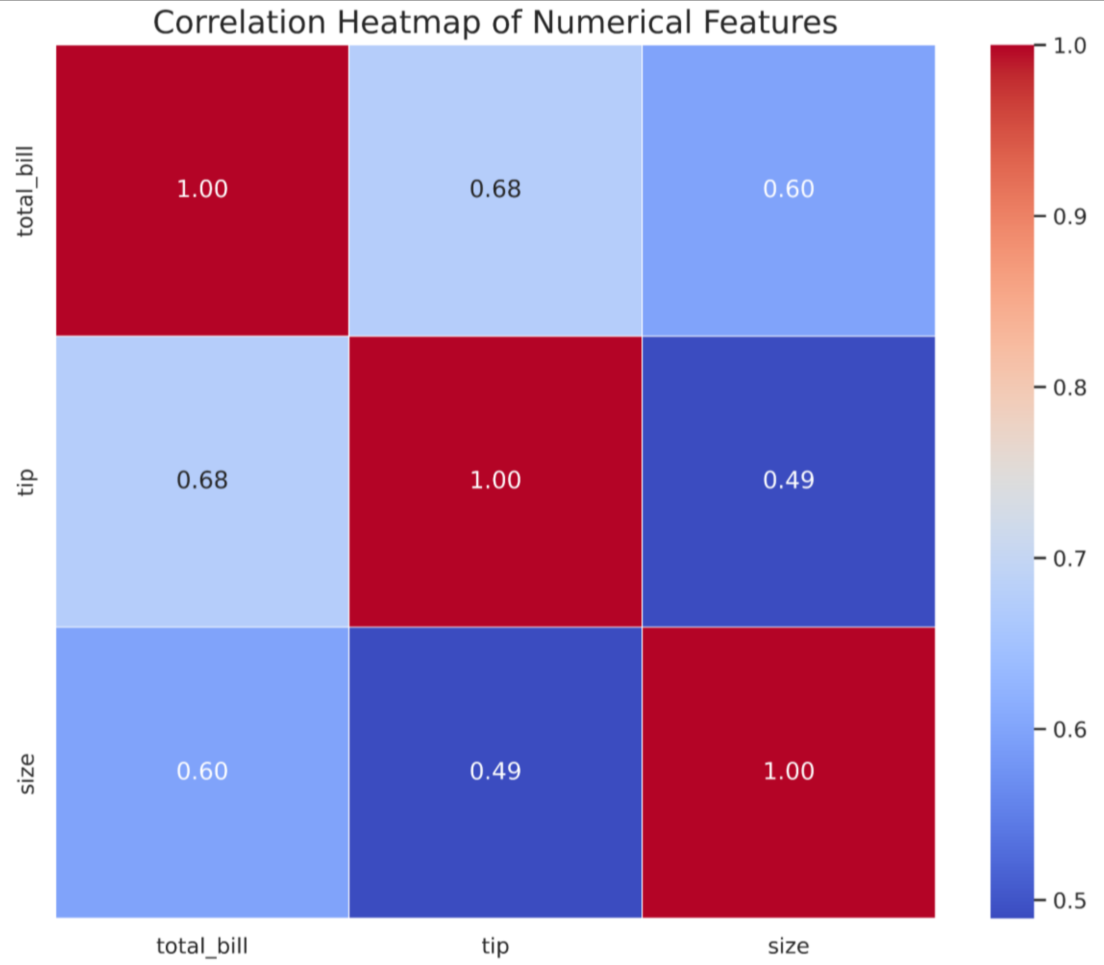
Correlation Visualization with Heatmap.

Sprints

3.1 Heatmap with Correlation



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



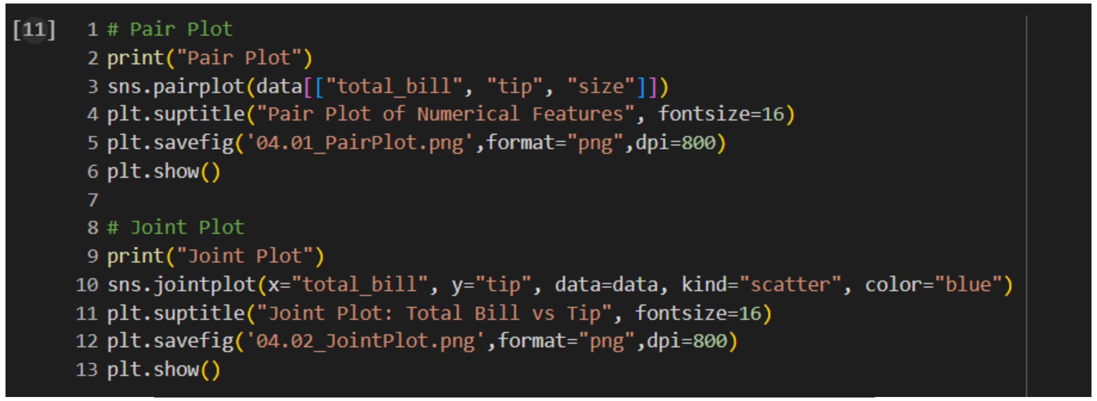
	total_bill	tip	size
total_bill	1.00	0.68	0.60
tip	0.68	1.00	0.49
size	0.60	0.49	1.00

Sprints

Requirement Four “Extra”

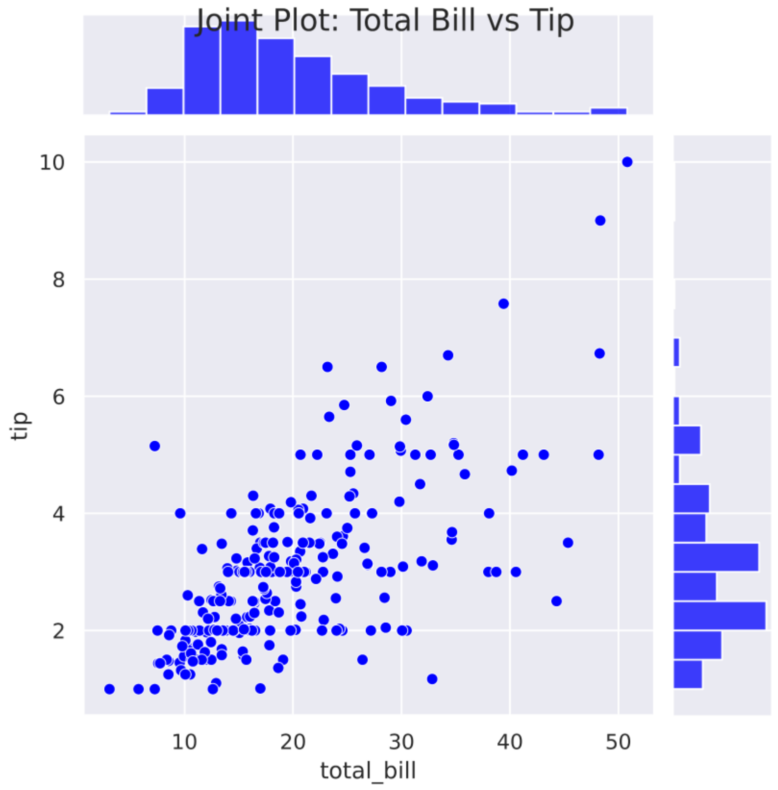
Advanced Pair Plots and Joint Plots

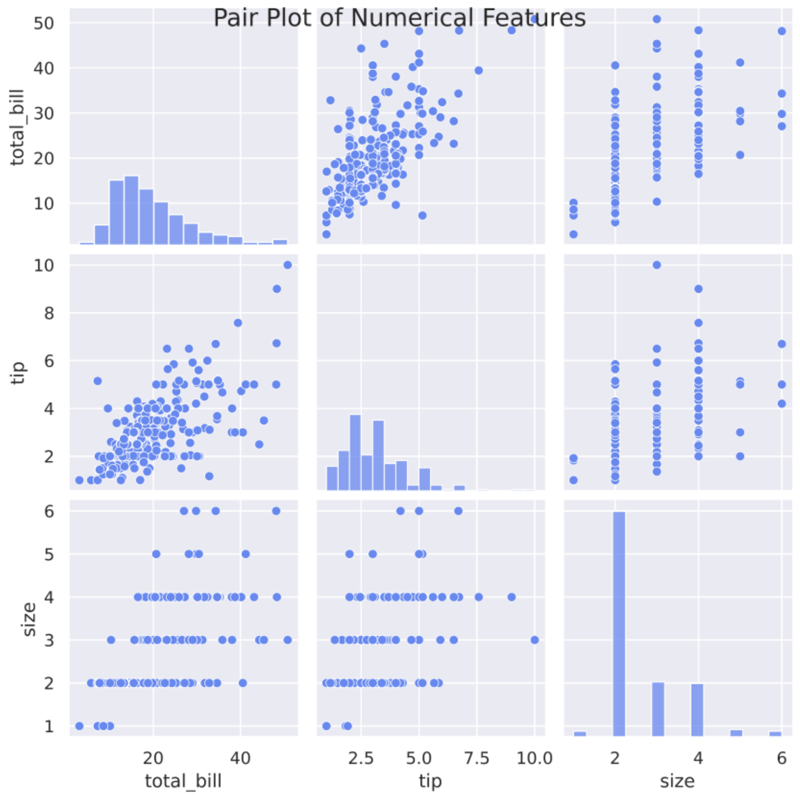
4.1 Advanced Pair and Joint Plot



```
[11]
# Pair Plot
print("Pair Plot")
sns.pairplot(data[["total_bill", "tip", "size"]])
plt.suptitle("Pair Plot of Numerical Features", fontsize=16)
plt.savefig('04.01_PairPlot.png',format="png",dpi=800)
plt.show()

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



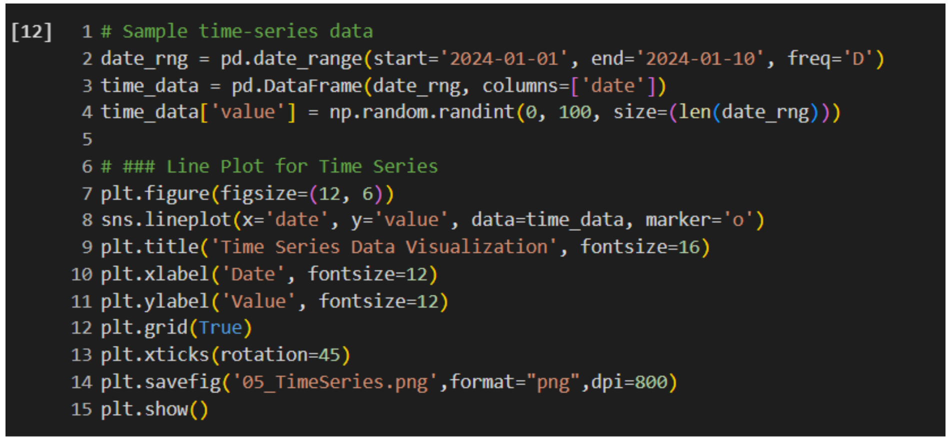


Sprints

Requirement Five “Extra”

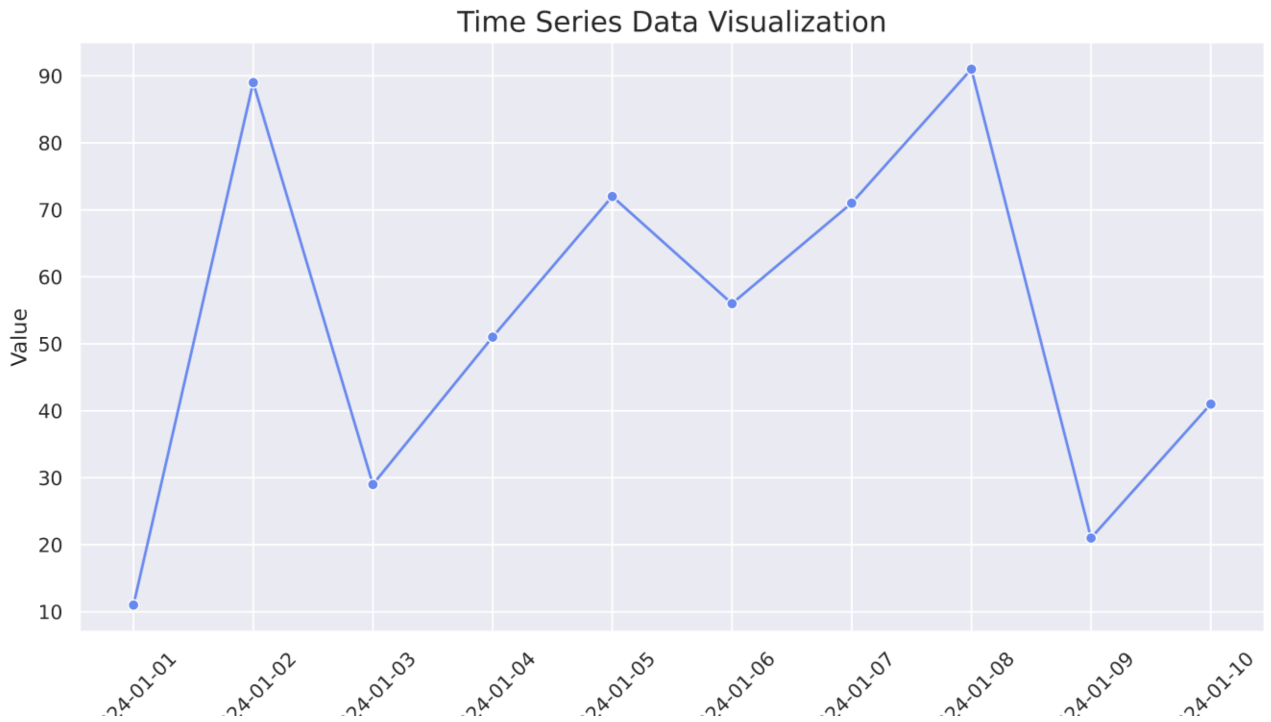
Time Series Data Visualization

4.1 Advanced Pair and Joint Plot



```
[12]
# Sample time-series data
date_rng = pd.date_range(start='2024-01-01', end='2024-01-10', freq='D')
time_data = pd.DataFrame(date_rng, columns=['date'])
time_data['value'] = np.random.randint(0, 100, size=(len(date_rng)))

# ### Line Plot for Time Series
plt.figure(figsize=(12, 6))
sns.lineplot(x='date', y='value', data=time_data, marker='o')
plt.title('Time Series Data Visualization', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Value', fontsize=12)
plt.grid(True)
plt.xticks(rotation=45)
plt.savefig('05_TimeSeries.png',format="png",dpi=800)
plt.show()
```


Time Series Data Visualization
Value