ASSIGNMENT

1. What advantages does Seaborn have over other plotting libraries when it comes to statistical visualization?

Seaborn is specifically designed for statistical visualization, and it offers several advantages over other plotting libraries like Matplotlib:

Advantage	Explanation
High-Level Abstraction	Seaborn simplifies the creation of complex statistical plots like regression plots, boxplots, and violin plots.
Built-in Statistical Tools	It includes tools to automatically compute and display statistical values (e.g., regression lines, confidence intervals).
Aesthetic Defaults	Seaborn provides beautiful, publication-quality plots with minimal configuration.
Integration with Pandas	It seamlessly works with Pandas DataFrames, making it easier to handle and visualize data directly.
Categorical Plots	Specialized for handling categorical data with plots like bar plots, count plots, and swarm plots.
Built-in Themes	Offers built-in themes (darkgrid, whitegrid, etc.) for better plot appearance.
Faceting	Facilitates the creation of multi-plot grids using FacetGrid to visualize subsets of data.
Data Aggregation	Automatically aggregates data (e.g., mean, median) for categorical plots like barplot or pointplot.

2. How do you deal with outliers in Seaborn? Please elaborate on some methods.

Outliers are data points significantly different from others. Handling them depends on the context and purpose of analysis. Seaborn provides several methods to identify and visualize outliers.

Methods to Handle Outliers

1. **Visualization** Use visualizations to detect outliers:
    - **Boxplots**: Outliers appear as points outside the whiskers.
    - **Swarmplots**: Outliers appear as isolated points.

```
sns.boxplot(data=tips, x='total_bill')

plt.show()
```

2.Capping and Flooring Replace outliers with thresholds (e.g., 5th and 95th percentiles).

```
import numpy as np

# Define thresholds

lower_bound = tips['total_bill'].quantile(0.05)

upper_bound = tips['total_bill'].quantile(0.95)


# Cap and floor the data

tips['total_bill'] = np.clip(tips['total_bill'], lower_bound, upper_bound)
```

3.Removing Outliers Remove rows containing outliers.

```
# Calculate IQR

Q1 = tips['total_bill'].quantile(0.25)

Q3 = tips['total_bill'].quantile(0.75)

IQR = Q3 - Q1


# Define bounds

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR


# Filter data

filtered_tips = tips[(tips['total_bill'] >= lower_bound) & (tips['total_bill'] <= upper_bound)]
```

4.Transformations

Use transformations like log or square root to reduce the impact of outliers.

```
ips['log_total_bill'] = np.log(tips['total_bill'])

sns.boxplot(data=tips, x='log_total_bill')

plt.show()
```

5.Z-Score Method

Use statistical thresholds (e.g., Z-scores > 3) to detect and handle outliers.

```
from scipy.stats import zscore


tips['z_score'] = zscore(tips['total_bill'])

filtered_tips = tips[tips['z_score'].abs() < 3]
```