Data Visualization and Chart Selection

Section 1: Chart Selection Framework

Choosing the right chart depends on the message you want to convey. Follow these guidelines:

- **Trends over Time:** When you need to show how data changes over a continuous period (days, months, years), a Line Chart is the best choice. For showing cumulative totals over time, use an Area Chart.
- **Comparison between Categories:** To compare different items or groups, use a Bar Chart (horizontal) or a Column Chart (vertical). Use a Bar Chart if the category names are long or if there are more than 10 categories.
- **Relationships:** To show the correlation between two numeric variables (e.g., height vs. weight), use a Scatter Plot. If you have a third variable, represent it with the size of the dots in a Bubble Chart.
- **Data Distribution:** To see how data points are distributed or to identify outliers, use a Histogram (for frequency) or a Box Plot (for statistical summary).
- **Part-to-Whole Composition:** To show how individual parts make up a whole, use a Pie Chart (only if you have 5 or fewer categories). For more complex comparisons of parts within totals, use a Stacked Bar Chart.

Section 2: Design Principles for Effective Visualization

Follow these professional standards to ensure clarity:

1. **Zero Baseline:** Always start the Y-axis at zero for Bar and Column charts to avoid misleading the audience.
2. **Minimize Clutter:** Maintain a high **Data-Ink Ratio**. Remove unnecessary gridlines, borders, and background colors.
3. **Color Consistency:** Use colors purposefully. Avoid "rainbow" effects. Use the same color for the same category across multiple charts.
4. **Order Matters:** Sort bar charts in ascending or descending order by value unless there is a natural order (like dates).

Section 3: Python Implementation Reference (Matplotlib & Seaborn)

Use the following functions to generate charts in Python:

- **Line Plot:** `plt.plot(x, y)`
- **Bar Chart:** `plt.bar(x, height)` or `sns.barplot(x=x, y=y)`
- **Histogram:** `plt.hist(data, bins=30)` or `sns.histplot(data)`
- **Scatter Plot:** `plt.scatter(x, y)` or `sns.scatterplot(x=x, y=y)`
- **Heatmap:** `sns.heatmap(data.corr(), annot=True)`