

DATA VISUALIZATION

Matplotlib · Seaborn · Plotly — Quick Reference



- 1. Matplotlib Core — Subplots & Styling
- 2. Seaborn — Distribution & Comparison Plots
- 3. Seaborn — Relationship & Correlation Plots
- 4. Plotly Basics — Interactive Dashboards
- 5. Decision Guide — Which Plot Answers Which Question

Phase 2 · Practical EDA Notes · Not Theory-Heavy

Matplotlib Core — Subplots & Styling

Matplotlib = the base layer everything else (Seaborn, Pandas .plot()) is built on. Know it for full control: multi-panel figures, custom styling, saving high-res images for reports.

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(10, 4)) # 1 row, 2 plots side by side

axes[0].plot(x, y, color="steelblue", linewidth=2)
axes[0].set_title("Trend")
axes[0].set_xlabel("Time"); axes[0].set_ylabel("Value")

axes[1].bar(categories, counts, color="coral")
axes[1].set_title("Counts")

plt.tight_layout() # fixes overlapping labels
plt.savefig("chart.png", dpi=150, bbox_inches="tight")
plt.show()
```

► Key building blocks

Figure = whole canvas



Axes = one subplot



plot/bar/scatter = draws on that axes

Need	Use
Multiple charts in a grid	<code>plt.subplots(rows, cols)</code>
Fix overlapping labels/titles	<code>plt.tight_layout()</code>
Save a high-res chart for a report/deck	<code>plt.savefig(..., dpi=150)</code>
Quick one-off single chart	<code>plt.plot()</code> / <code>plt.bar()</code> directly (no subplots needed)

🔗 Common Interview Questions

- Figure vs Axes — what's the difference? → Figure is the entire canvas/window; Axes is one individual plot inside it. A figure can contain multiple axes (subplots).
- Why use `plt.subplots()` instead of multiple `plt.figure()` calls? → Lets you place multiple related charts in one figure/grid for side-by-side comparison — standard for EDA reports.
- matplotlib vs seaborn — when do you reach for plain matplotlib? → When you need fine-grained custom control (annotations, custom layouts, publication-style figures) that Seaborn's high-level API doesn't expose directly.

Seaborn — Distribution & Comparison Plots

histplot — Distribution of ONE numeric column

WHEN: check skewness, spot multi-modal data, decide mean vs median for imputation.

```
import seaborn as sns

sns.histplot(data=df, x="age", bins=30, kde=True)    # distribution + smooth curve
```

boxplot — Compare a numeric column ACROSS categories

WHEN: spot outliers fast, compare medians/IQR across groups (e.g. salary by dept).

```
sns.boxplot(data=df, x="dept", y="salary")          # compare spread + outliers
across groups
```

violinplot — boxplot + shape of the distribution

WHEN: same as boxplot, but you also care about whether a group is bimodal — boxplot alone would hide that.

```
sns.violinplot(data=df, x="dept", y="salary")       # boxplot + full
distribution shape
```

🔗 Common Interview Questions

- boxplot vs violinplot — when would you pick one over the other? → boxplot is cleaner for quick outlier/median comparison across many categories; violinplot adds the full distribution shape, useful when you suspect bimodality that a boxplot would hide.
- How do you spot skewness from a histplot? → Long tail on one side; if the tail is on the right, use median (not mean) for imputation since mean gets pulled by the tail.
- How does a boxplot define an outlier? → Points beyond $1.5 \times \text{IQR}$ from $Q1/Q3$ ($\text{IQR} = Q3 - Q1$) are plotted as individual dots outside the whiskers.

Seaborn — Relationship & Correlation Plots

scatterplot — Relationship between TWO numeric columns

WHEN: check linear/non-linear relationship, spot clusters, color by a category (hue) to add a 3rd dimension.

```
sns.scatterplot(data=df, x="salary", y="score", hue="dept") # relationship between
2 numeric vars
```

heatmap — Correlation matrix across ALL numeric columns

WHEN: quickly spot multicollinearity before modeling, decide which features might be redundant.

```
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f") # correlation matrix at
a glance
```

pairplot — Every numeric column vs every other, in one grid

WHEN: fast first-look EDA on a new dataset with few numeric columns (slow if too many columns — pick key ones with vars=).

```
sns.pairplot(df, hue="target", vars=["age", "salary", "score"])
# scatter for every numeric pair + histogram on the diagonal, colored by class
```

🕒 Common Interview Questions

- How do you detect multicollinearity visually before modeling? → A correlation heatmap — look for cells close to +1/-1 between feature pairs (not target); consider dropping/combining one of the pair.
- Correlation heatmap shows correlation, not causation — how would you explain that in an interview? → Two variables can move together due to a third confounding factor or coincidence; correlation alone never proves one causes the other.
- Why does pairplot get slow/unreadable with many columns? → It's $O(n^2)$ subplots — for a dataset with 15+ numeric columns, restrict to the most relevant ones via vars= instead of plotting everything.

Plotly Basics — Interactive Dashboards

Plotly Express (px) = high-level API, same mental model as Seaborn but renders interactive charts (hover, zoom, filter by legend click) — used for dashboards and stakeholder-facing reports instead of static EDA.

```
import plotly.express as px

fig = px.scatter(df, x="salary", y="score", color="dept",
                 hover_data=["name"], title="Salary vs Score")
fig.show()  # interactive: zoom, pan, hover tooltips

fig2 = px.bar(df, x="dept", y="salary", color="dept", barmode="group")
fig2.write_html("dashboard.html")  # shareable interactive file, no server needed
```

Aspect	Matplotlib/Seaborn	Plotly
Output	Static image	Interactive (zoom/hover/filter)
Best for	Quick EDA, notebooks, reports/papers	Dashboards, stakeholder presentations, web apps
Speed to plot	Fast	Fast (similar px.* syntax)

🕒 Common Interview Questions

- When would you choose Plotly over Seaborn/Matplotlib? → When the audience needs to explore the data themselves (hover for exact values, zoom into a region, filter via legend) — e.g. a dashboard for stakeholders, not a static notebook plot.
- How do you share a Plotly chart with someone who doesn't have Python? → `fig.write_html('file.html')` — opens in any browser, fully interactive, no server needed.
- Is Plotly a replacement for Matplotlib in day-to-day EDA? → Not usually — Matplotlib/Seaborn are faster for quick, throwaway exploration; Plotly is worth the extra step when interactivity adds real value.

Decision Guide — Which Plot Answers Which Question

► "What does ONE column look like?" → Distribution

Question	Plot
Shape/spread of a numeric column	histplot (+ kde)
Median, IQR, outliers of a numeric column	boxplot
Frequency of a categorical column	countplot / bar chart

► "How do TWO numeric columns relate?" → Relationship

Question	Plot
Linear/non-linear relationship, clusters	scatterplot
Relationship + trend line	sns.regplot / lmplot
Interactive exploration of relationship	Plotly scatter

► "How does a metric differ ACROSS groups?" → Comparison

Question	Plot
Compare spread/outliers across categories	boxplot
Compare spread + full shape across categories	violinplot
Compare totals/counts across categories	bar chart

► "How do MANY variables correlate?" → Correlation

Question	Plot
Correlation strength between all numeric features	heatmap (df.corr())
First-look at every pairwise relationship + class separation	pairplot (hue=target)

Quick mental shortcut: 1 variable → histplot/boxplot, 2 variables → scatterplot, 1 numeric across groups → boxplot/violinplot, many variables at once → heatmap/pairplot.