Bar Chart

```
sns.barplot(df, x="Product", y="Sales")
# estimator="sum"
# errorbar = None
# hue="DayofWeek"
# order=["C", "A", "B"]
# hue_order=[...]
# orient = "h"
# err_kws={"linewidth": 5, "color": "red", linestyle="--"}
# ax = axes
# legend=False


plt.show()
```

Line Chart

```
sns.lineplot(df, x="Month", y="Sales")
# linestyle, color, linewidth
# hue
# palette={"A": "red", "B": "green", "C": "blue"} # line color in hue
# markers  (but specify style = "Product" to apply)
# dashes: "" ➔ solid line; (4, 2) ➔ dashed; (3, 1, 1, 1) ➔ Dashed dot


plt.show()
```

Scatter Plot

```
sns.scatterplot(df, x="Revenue", y="Sales")
# hue, style (different marker)
# size
# markers
# edgecolor
# linewidth ➔ Thickness of edge
plt.show()
```

```
tips = sns.load_dataset("tips")

sns.scatterplot(data=tips, x="total_bill", y="tip", hue="day", size="time",
alpha=0.7, edgecolor="black", linewidth=1.2,
```

Box Plot

```
tips = sns.load_dataset("tips")

sns.boxplot(data=tips,
    x="day", y="total_bill", # Category on X, Numeric on Y
    hue="sex", order=["Thur", "Fri", "Sat", "Sun"],
    hue_order=["Male", "Female"],
    width=0.6,  # box width
    dodge=True,  # separate hue boxes side-by-side
    fliersize=5,  # size of outlier points
    linewidth=1.2,  # outline thickness
    whis=1.5,  # whisker length (IQR multiplier)
    showcaps=True,  # show the caps at ends of whiskers
    boxprops={"alpha": 0.7})

plt.show()
```

Histogram

- Explore and Plot histogram from your end
- What is use case of KDE parameter in sns historgram plot?

Heat Map

```
df = sns.load_dataset("flights")

# Convert to pivot table for heatmap
pivot_flights = df.pivot_table(index="month", columns="year", values="passengers")

sns.heatmap(data=pivot_flights,
    annot=True,  # show numbers in cells
    fmt=".1f",  # value formatting
    cmap="Blues",  # color map (YlGnBu)
    linewidths=0.5,  # lines between cells
    linecolor="white",  # line color
    square=False,  # keep rectangular cells
    robust=True, # 2nd to 98th percentile
    annot_kws={"size": 8},  # annotation text size)

plt.title("Flight Passengers Heatmap", fontsize=14)
```

Count Plot

- Count occurrence of each category in specified column
- Useful for frequency visualization

```
tips = sns.load_dataset("tips")

# Count plot of number of records per day
sns.countplot(data=tips,
    x="day",           # categorical variable to count (try: smoker)
    hue="sex",         # split counts by category (optional)
    hue_order=["Female", "Male"],
    palette={"Male": "black", "Female": "red"},
)

plt.title("Count of Records per Day by Sex")
plt.show()
```

Relational Plot - Scatter

- For scatter / line plot. Manages figure and axes for us

```
tips = sns.load_dataset("tips")

sns.relplot(
    data=tips,
    x="total_bill",
    y="tip",
    hue="sex",
    col="time",  # create separate plots for lunch/dinner
    row="smoker",
    kind="scatter")

# col_wrap = number of column in a plot if more category (Try with day as col)

plt.show()
```

Relational Plot - Line

```
tips = sns.load_dataset("tips")

sns.relplot(
    data=tips,
    x="total_bill",
    y="tip",
    hue="sex",
    col="time",  # create separate plots for lunch/dinner
    row="smoker",
    kind="line")

# col_wrap = number of column in a plot if more category (Try with day as col)
# style, plot, dashed parameters of line plot is valid in relplot

plt.show()
```

Add Label

For Facet Grid use:

```
g = sns.catplot(….)
g.figure.suptitle(“Name”)  # Set figure title
g.figure.subplots_adjust(top=0.9) # title above subplot
```

```
g.set(xlabel="x-axis label", ylabel="y-axis label")
```

xlim=(min, max), ylim, xticks=[], yticks, xticklabels=[],

- For axes plot use customization with ax as in matplotlb

Categorical Plot

- For visualizing categorical variable
- Same advantages as that of relplot
- Can create subplot with row and col parameters
- Kind: 'strip', 'swarm', 'box', 'violin', 'boxen', 'point', 'bar', 'count'

Categorical Plot

kind	Plot type	Best for	Example visualization
"strip"	Scatterplot for categorical data	Showing all individual data points, quick distribution check	
"swarm"	Scatterplot with non-overlapping points	Showing all points clearly when there's overlap	Points are jittered to avoid overlap
"box"	Box-and-whisker plot	Summarizing median, quartiles, and outliers	Statistical summary of distribution
"violin"	Violin plot (box + KDE)	Showing distribution shape and spread	Good for visualizing multi-modal distributions
"boxen"	Letter-value plot	Showing distribution tail behavior for large datasets	More detailed than box plot
"point"	Point plot	Showing mean values with confidence intervals	Emphasizes trend across categories
"bar"	Bar plot (mean + CI)	Showing average values and uncertainty	Aggregated summary per category
"count"	Count plot	Showing counts per category	Frequency of each category

Strip

```
# Show all data point for categorical variable

tips = sns.load_dataset("tips")

sns.catplot(data=tips, x="day", y="total_bill",
    hue="sex", kind="strip", jitter=True,

)

plt.show()
```

- Play around with col, col_wraps, row parameter

Swarm

- Show all data point for categorical variable
- Avoids overlap of points.

```
tips = sns.load_dataset("tips")

sns.catplot(data=tips, x="day", y="total_bill", hue="sex", kind="swarm")

plt.show()
```

- Try using plt.subplots to visualize strip on one and swarm on other

Box

```
tips = sns.load_dataset("tips")

sns.catplot(data=tips, x="day", y="total_bill", hue="sex", kind="box")

plt.show()
```

Boxen

```
tips = sns.load_dataset("tips")

sns.catplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="sex",
    kind="boxen",
)
plt.show()
```

Violin

- Distribution shape along with summary stat

```
tips = sns.load_dataset("tips")

sns.catplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="sex",
    kind="violin",
    split=True,  # Split by hue
    inner="quartile",  # Show quartiles
)
plt.show()
```

Point Plot

- Show mean value and confidence interval

```
sns.catplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="sex",
    kind="point",
    join=False,
    markers=["o", "s"],  # Custom markers
    linestyles=["-", "--"], # Custom line styles
)
plt.show()
```

- estimator, capsize, errorbar=None

Bar Plot

- Show averages value and confidence interval

```
sns.catplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="sex",
    kind="bar",
    height=5,
    aspect=1.3
)
plt.show()
```

Count Plot

```
sns.catplot(
    data=tips,
    x="day",
    hue="sex",
    kind="count",
    height=5,
    aspect=1.3
)
plt.show()
```

Distribution Plot

- To visualize numerical distribution of variable

Example

- `sns.displot(data=tips, x="total_bill", bins=20, col="sex")`

pass kde=True above and see difference

kind: hist, kde, ecdf

stat: count, frequency, density, probability

Regression Plot

- Bi-variate. Scatter plot along with regression line
- Shows confidence interval around line

Example

```
sns.regplot(data=tips, x="total_bill", y="tip", color="blue")
# order=2 (attempt to fit second order polynomial)
# x_estimator = np.mean
```

lmplot

- Same as regplot but its figure level
- We can use parameters like row, col, col_wrap

Residplot

- Axis level plot for showing residual of regression
- Difference in y and y_pred from regression
- Should be randomly scattered around 0

Example:

```
sns.residplot(data=tips, x="total_bill", y="tip", color="blue")
```

Pairplot

- Visualize pairwise relationship of numeric columns in dataset
- Plots grid of scatter plot for each pair with dist plot on diagonal

Example:

```
sns.pairplot(data=tips,  vars=['total_bill', 'tip', 'size'], hue='sex',
kind='scatter', diag_kind='kde')

# kind='reg' for regplot
```

All chart at once

```
plot_funcs = { "strip": sns.stripplot, "swarm": sns.swarmplot,
    "box": sns.boxplot, "violin": sns.violinplot,
    "boxen": sns.boxenplot, "point": sns.pointplot,
    "bar": sns.barplot, "count": sns.countplot,}

fig, axes = plt.subplots(2, 4, figsize=(18, 8))
axes = axes.flatten()

for ax, (kind, func) in zip(axes, plot_funcs.items()):
    if kind == "count":
        func(data=tips, x="day", hue="sex", ax=ax)
    else:
        func(data=tips, x="day", y="total_bill", hue="sex", ax=ax)
    ax.set_title(kind.capitalize())
    ax.set_xlabel("Day")
    ax.set_ylabel("Total Bill")

plt.tight_layout()
```

Changing Palette

- Apply default color for categorical plot
- Apply consistent color theme across all plots
- Work with pre-defined palette name or custom color list
- Controlled by sns.set_palette({palette})
- Pre-defined: pastel (soft color), deep (strong), bright, dark, colorblind
- Custom: Blues, RdBu, [‘Red’, ‘Green’, ‘Blue’] # Can be hex value list
- View Palette Color:

```
sns.set_palette("Blues")
sns.palplot(sns.color_palette("colorblind"))
```

Changing Scale

- Scale of plot elements: font, line-width, marker sizes can be adjusted
- Use case: Different size plot for presentation, paper, notebook
- Controlled by: sns.set_context({name})

Context	Best for	Effect
"paper"	Small plots for tight spaces (e.g., research papers)	Smallest elements
"notebook" (default)	Interactive environments	Medium elements
"talk"	Slides and presentations	Larger text and markers
"poster"	Posters and large displays	Largest scaling