Seaborn Exercises

Instructions:

• Complete the tasks using Python (pandas, numpy, seaborn).

• Include code cells, resulting plots, and short interpretations where requested.

• Submit a single notebook with all answers and save final figures as PNG files.

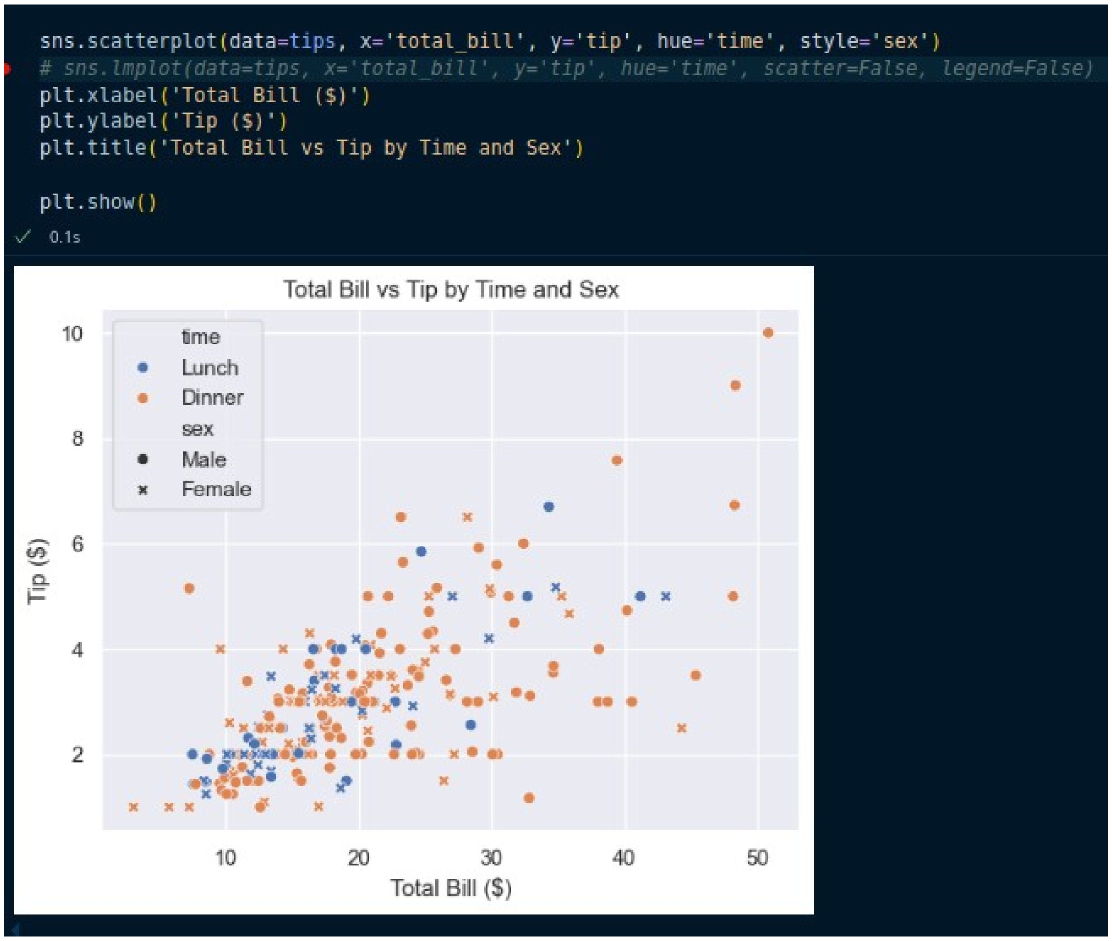
• This worksheet contains 10 exercises arranged from easy → hard.

1. **1) Load & visualize: `tips` scatter**

   Dataset: sns.load_dataset('tips')

   Task: Create a scatter plot of total_bill vs tip. Color points by time and use different markers for sex. Add axis labels and a title.

   Bonus: Add a regression line for each time (lunch/dinner) on the same axes.



```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='time', style='sex')
# sns.lmplot(data=tips, x='total_bill', y='tip', hue='time', scatter=False, legend=False)
plt.xlabel('Total Bill ($)')
plt.ylabel('Tip ($)')
plt.title('Total Bill vs Tip by Time and Sex')

plt.show()
```

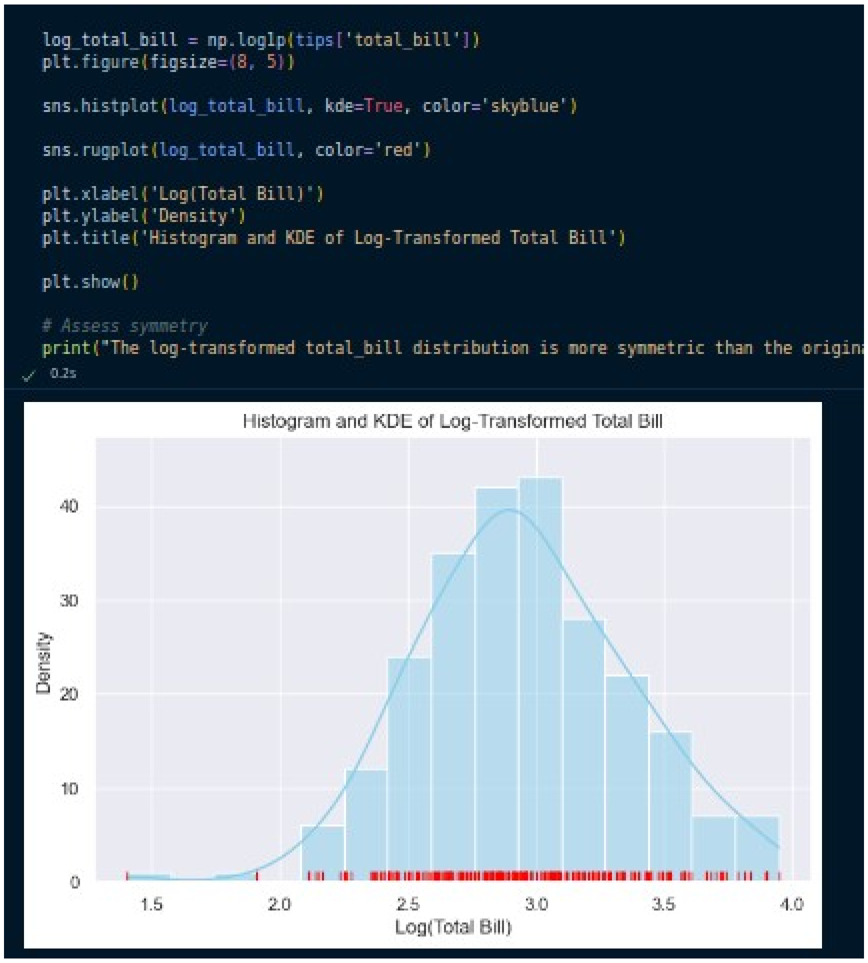
```
✓ 0.1s
```

2. **2) Distribution + NumPy: histogram & KDE**

   Dataset: tips (use total_bill)

   Task: Using numpy, compute a log-transformed version of total_bill (np.log1p). Plot its histogram and KDE on the same axes. Briefly state whether the transform made the distribution more symmetric.

   Bonus: Overlay a rug plot.



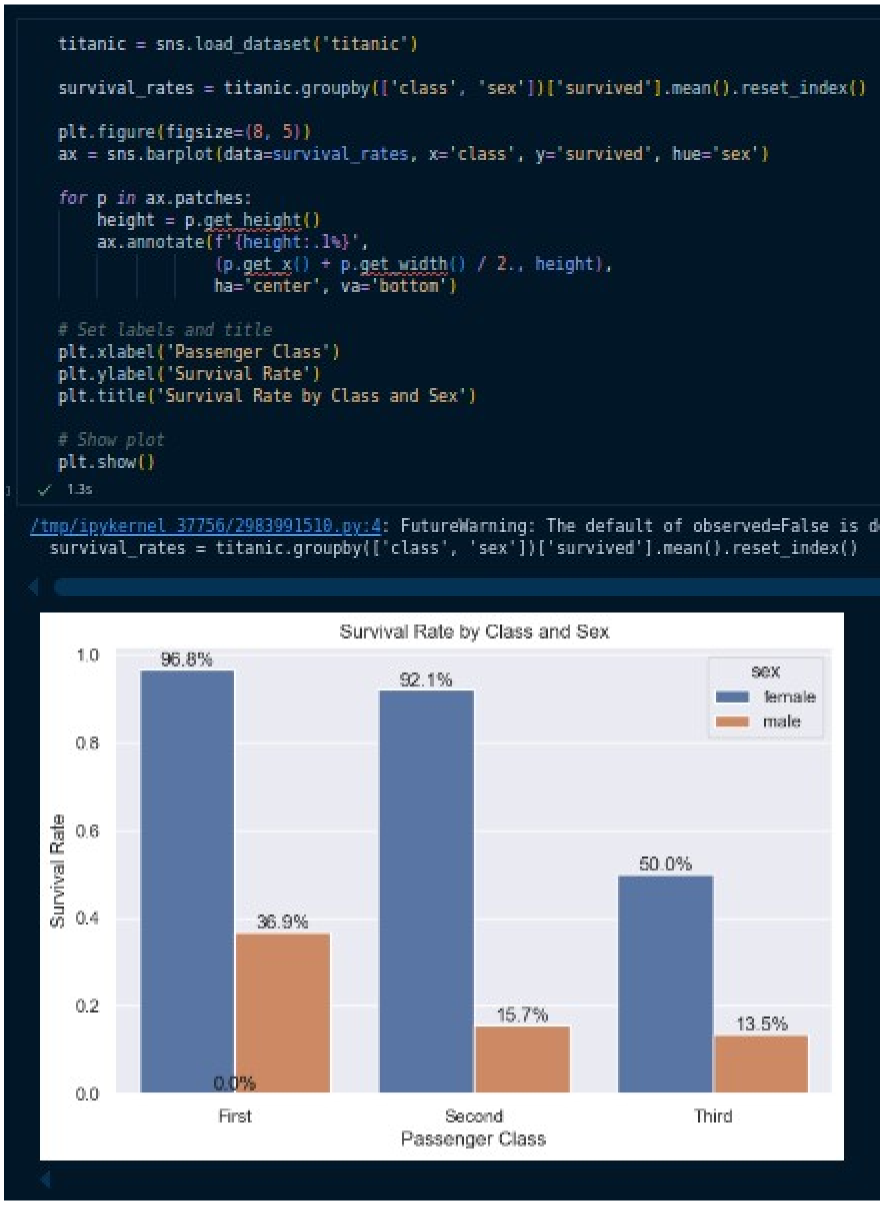
```
log_total_bill = np.log1p(tips['total_bill'])
plt.figure(figsize=(8, 5))

sns.histplot(log_total_bill, kde=True, color='skyblue')

sns.rugplot(log_total_bill, color='red')

plt.xlabel('Log(Total Bill)')
plt.ylabel('Density')
plt.title('Histogram and KDE of Log-Transformed Total Bill')

plt.show()

# Assess symmetry
print("The log-transformed total_bill distribution is more symmetric than the origina
```

3. **3) Categorical aggregation: barplot with pandas groupby**

   Dataset: titanic

   Task: Using pandas, group by class and compute the survival rate; show a bar plot of survival rate per class. Annotate bars with percentages.

   Bonus: Split bars by sex using hue.



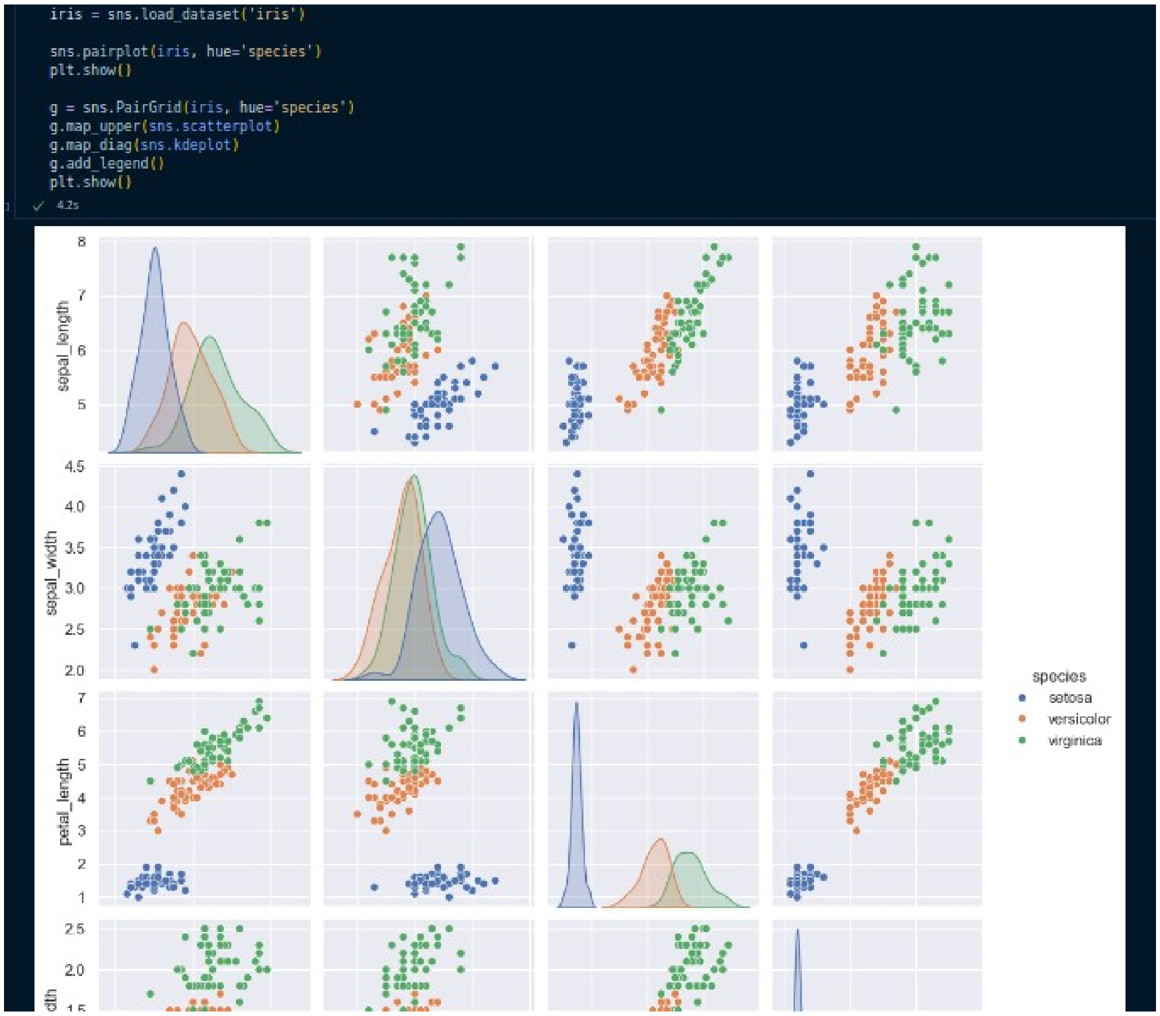
```
titanic = sns.load_dataset('titanic')

survival_rates = titanic.groupby(['class', 'sex'])['survived'].mean().reset_index()

plt.figure(figsize=(8, 5))
ax = sns.barplot(data=survival_rates, x='class', y='survived', hue='sex')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.1%}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom')

# Set labels and title
plt.xlabel('Passenger Class')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Class and Sex')

# Show plot
plt.show()
```

```
/tmp/ipykernel_37756/2983991510.py:4: FutureWarning: The default of observed=False is d
  survival_rates = titanic.groupby(['class', 'sex'])['survived'].mean().reset_index()
```

4. **4) Pairwise relationships: pairplot vs PairGrid**

Dataset: iris

Task:
 a) Produce sns.pairplot(iris, hue='species').
 b) Create the same using PairGrid with scatterplots in the upper triangle and KDEs on the diagonal. Explain one advantage of PairGrid.



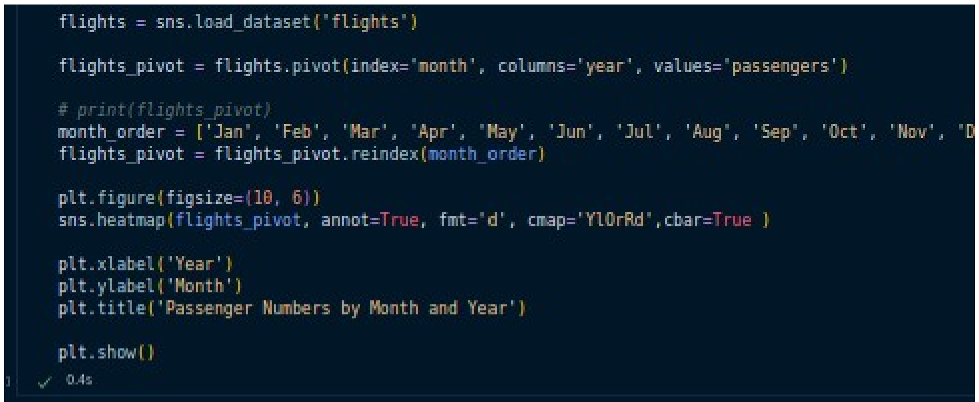
```
iris = sns.load_dataset('iris')

sns.pairplot(iris, hue='species')
plt.show()

g = sns.PairGrid(iris, hue='species')
g.map_upper(sns.scatterplot)
g.map_diag(sns.kdeplot)
g.add_legend()
plt.show()
```

5. **5) Time-series & heatmap: flights pivot**

   Dataset: flights (year, month, passengers)

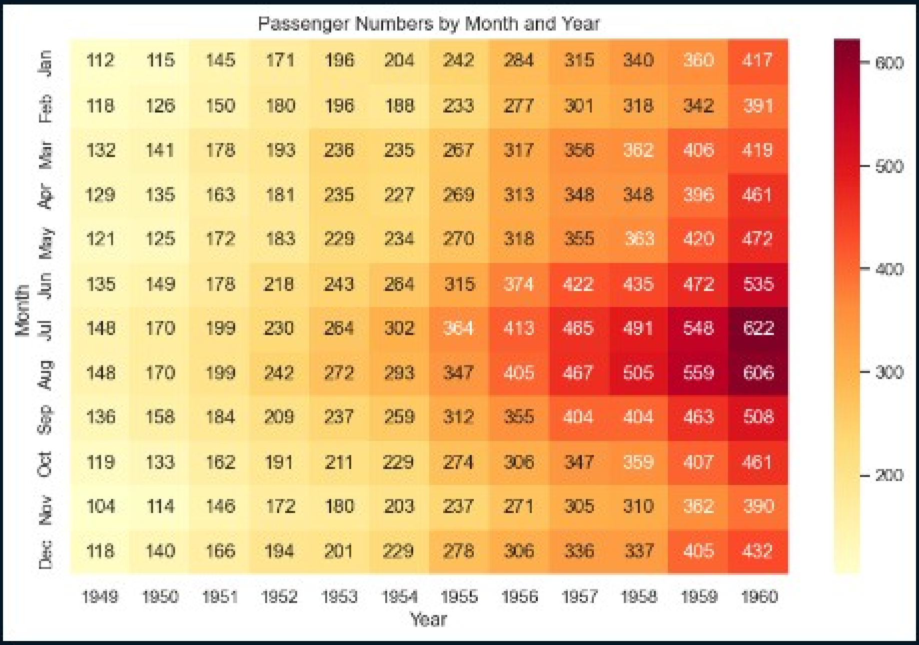
   Task: Pivot into a month x year matrix and plot a heatmap with annotations and colorbar. Interpret the main pattern you observe.

   Bonus: Ensure months appear in chronological order.



```python
flights = sns.load_dataset('flights')

flights_pivot = flights.pivot(index='month', columns='year', values='passengers')

# print(flights_pivot)
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'D
flights_pivot = flights_pivot.reindex(month_order)

plt.figure(figsize=(10, 6))
sns.heatmap(flights_pivot, annot=True, fmt='d', cmap='YlOrRd',cbar=True )

plt.xlabel('Year')
plt.ylabel('Month')
plt.title('Passenger Numbers by Month and Year')

plt.show()
```

0.4s



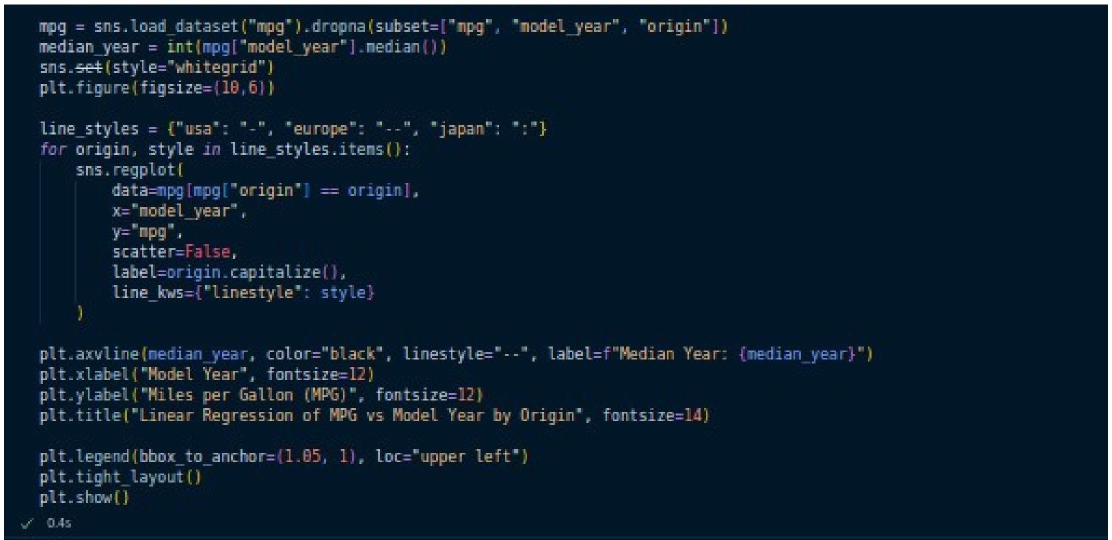
	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

6. **6) Regression with groups & matplotlib tweak**

   Dataset: mpg (model_year, mpg, origin)

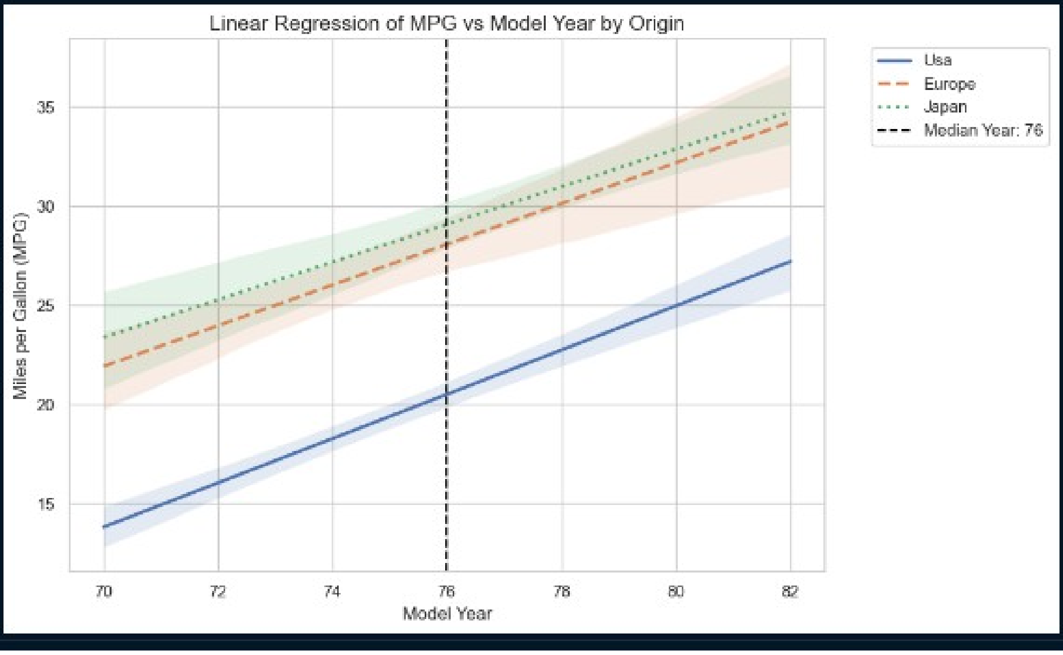
   Task: For each origin, plot a linear regression (mpg vs model_year) on the same axes with different line styles. Add a legend outside the plot and a vertical dashed line at the median model year.

   Bonus: Compute and display the slope for each origin as text on the plot.



```
mpg = sns.load_dataset("mpg").dropna(subset=["mpg", "model_year", "origin"])
median_year = int(mpg["model_year"].median())
sns.~~set~~(style="whitegrid")
plt.figure(figsize=(10,6))

line_styles = {"usa": "-", "europe": "--", "japan": ":"}
for origin, style in line_styles.items():
    sns.regplot(
        data=mpg[mpg["origin"] == origin],
        x="model_year",
        y="mpg",
        scatter=False,
        label=origin.capitalize(),
        line_kws={"linestyle": style}
    )

plt.axvline(median_year, color="black", linestyle="--", label=f"Median Year: {median_year}")
plt.xlabel("Model Year", fontsize=12)
plt.ylabel("Miles per Gallon (MPG)", fontsize=12)
plt.title("Linear Regression of MPG vs Model Year by Origin", fontsize=14)

plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left")
plt.tight_layout()
plt.show()
```

✓ 0.4s



7. **7) Faceting & custom aggregation with FacetGrid**

   Dataset: tips

   Task: Create a FacetGrid faceted by day (columns) and smoker (rows). In each facet show a violin plot of total_bill and overlay the mean point (white dot with black edge). Annotate each facet with the mean value.

8. **8) Joint distributions + conditional coloring**

   Dataset: penguins (or substitute)

   Task: Create a jointplot of bill_length_mm vs bill_depth_mm with hex bins. Color hexes by the average species-encoded-as-integer inside each bin (compute binned-statistic). Explain your binning approach.

   Bonus: Provide a legend mapping species encoding back to names.

9. **9) Build a custom seaborn helper function**

   Task: Implement pretty_violin(df, x, y, hue=None, title=None) that:
   - sets a publication-style seaborn theme
   - draws a violin plot with inner quartiles
   - overlays swarm points (max 200 per group, sample if needed)
   - adds a text box with mean, median, std for each group

   Demonstrate on a synthetic dataset (3 groups).

   Bonus: Return fig, ax.

10. **10) Exploratory mini-project**

    Dataset: choose one (titanic, penguins, mpg, or tips)

    Deliverables: For the chosen dataset produce cells that:
    1. Clean missing data and describe strategy.
    2. Show a correlation heatmap and explain three strongest associations.
    3. Compare a numerical outcome across two categorical variables.
    4. Create a multi-panel facet visualization that reveals an interaction effect.
    5. Create one advanced visualization (annotated regression + residuals OR hexbin + marginals OR clustermap).
    6. Summarize your 3 most important findings in bullet points.

    Bonus: Add an interactive widget to choose hue/col (optional).