Advanced Matplotlib (Subplots & Styling)

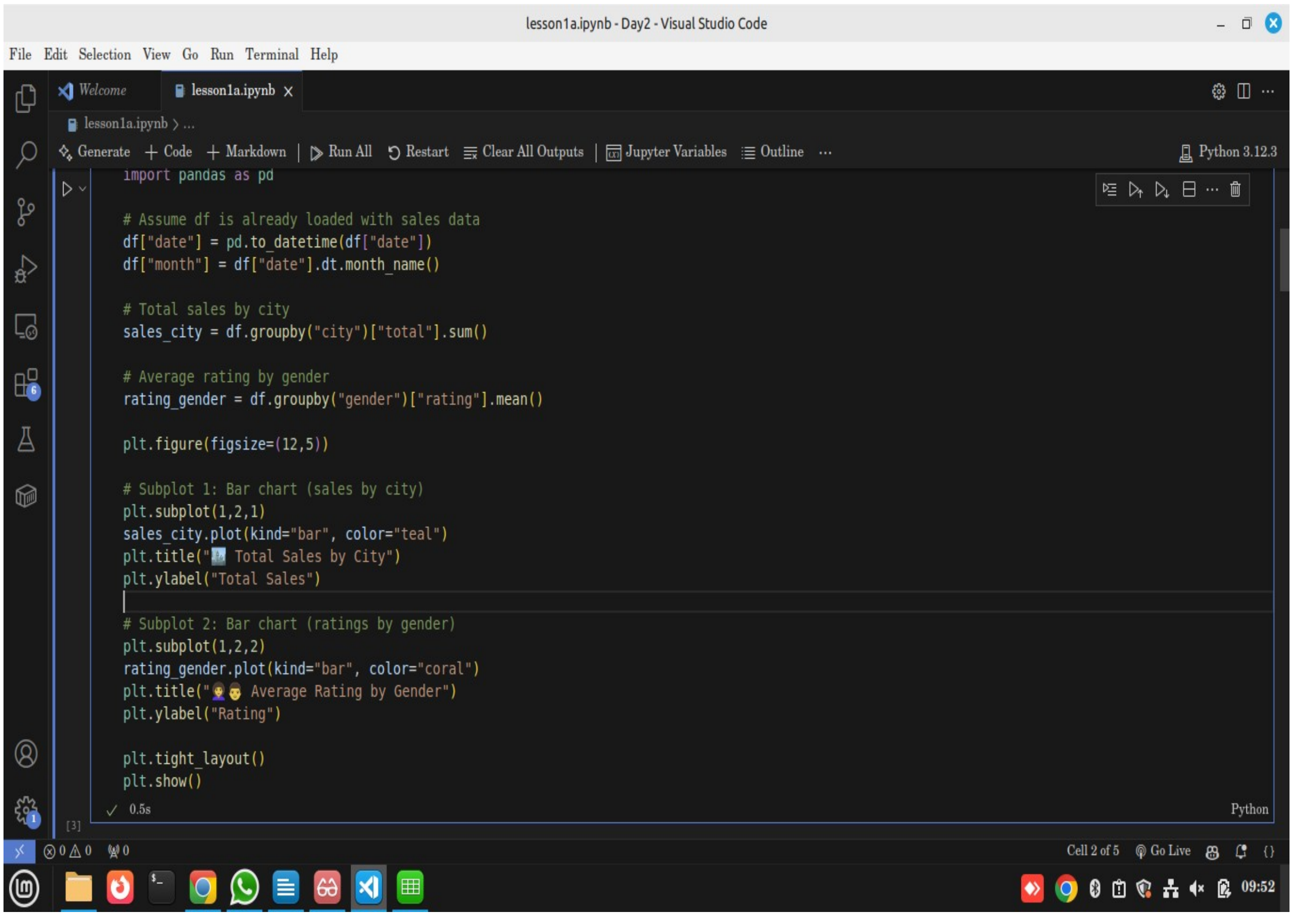
Reminder

- Yesterday we drew **basic plots** (line, bar, histogram).
- Today → we learn how to make plots **look professional** using **subplots & styling**.

Subplots & Styling

- **Subplots** = multiple charts in one figure (compare side by side).
- **Styling** = customizing fonts, colors, grids, backgrounds, legends.
- Why important in data science?
  👉 Clean, attractive visuals = easier insights & better storytelling.

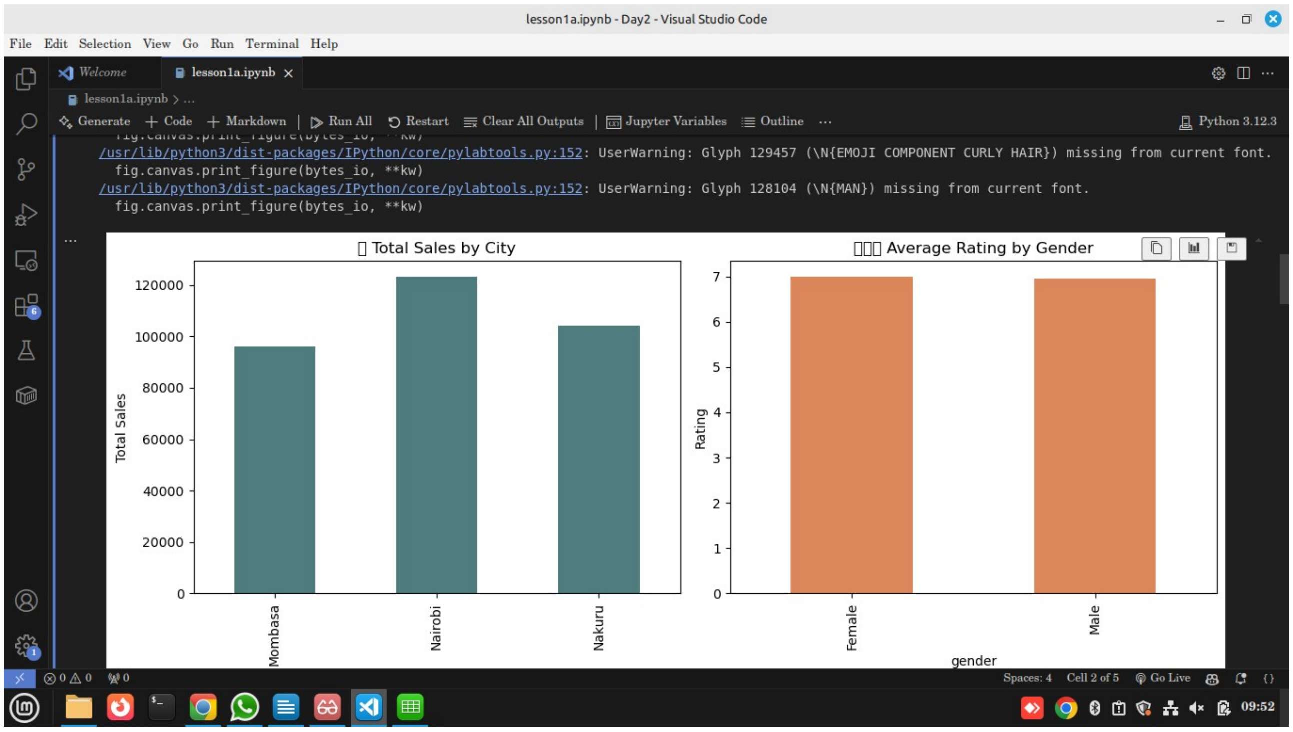
Practical 1: Subplots


lesson1a.ipynb - Day2 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Welcome
lesson1a.ipynb
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline
Python 3.12.3
Cell 2 of 5
Go Live
09:52

```python
import pandas as pd

# Assume df is already loaded with sales data
df["date"] = pd.to_datetime(df["date"])
df["month"] = df["date"].dt.month_name()

# Total sales by city
sales_city = df.groupby("city")["total"].sum()

# Average rating by gender
rating_gender = df.groupby("gender")["rating"].mean()

plt.figure(figsize=(12,5))

# Subplot 1: Bar chart (sales by city)
plt.subplot(1,2,1)
sales_city.plot(kind="bar", color="teal")
plt.title("🏙 Total Sales by City")
plt.ylabel("Total Sales")

# Subplot 2: Bar chart (ratings by gender)
plt.subplot(1,2,2)
rating_gender.plot(kind="bar", color="coral")
plt.title("👩‍🦱👨 Average Rating by Gender")
plt.ylabel("Rating")

plt.tight_layout()
plt.show()
```

✓ 0.5s

Output



```
/usr/lib/python3/dist-packages/IPython/core/pylabtools.py:152: UserWarning: Glyph 129457 (\N{EMOJI COMPONENT CURLY HAIR}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/lib/python3/dist-packages/IPython/core/pylabtools.py:152: UserWarning: Glyph 128104 (\N{MAN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```

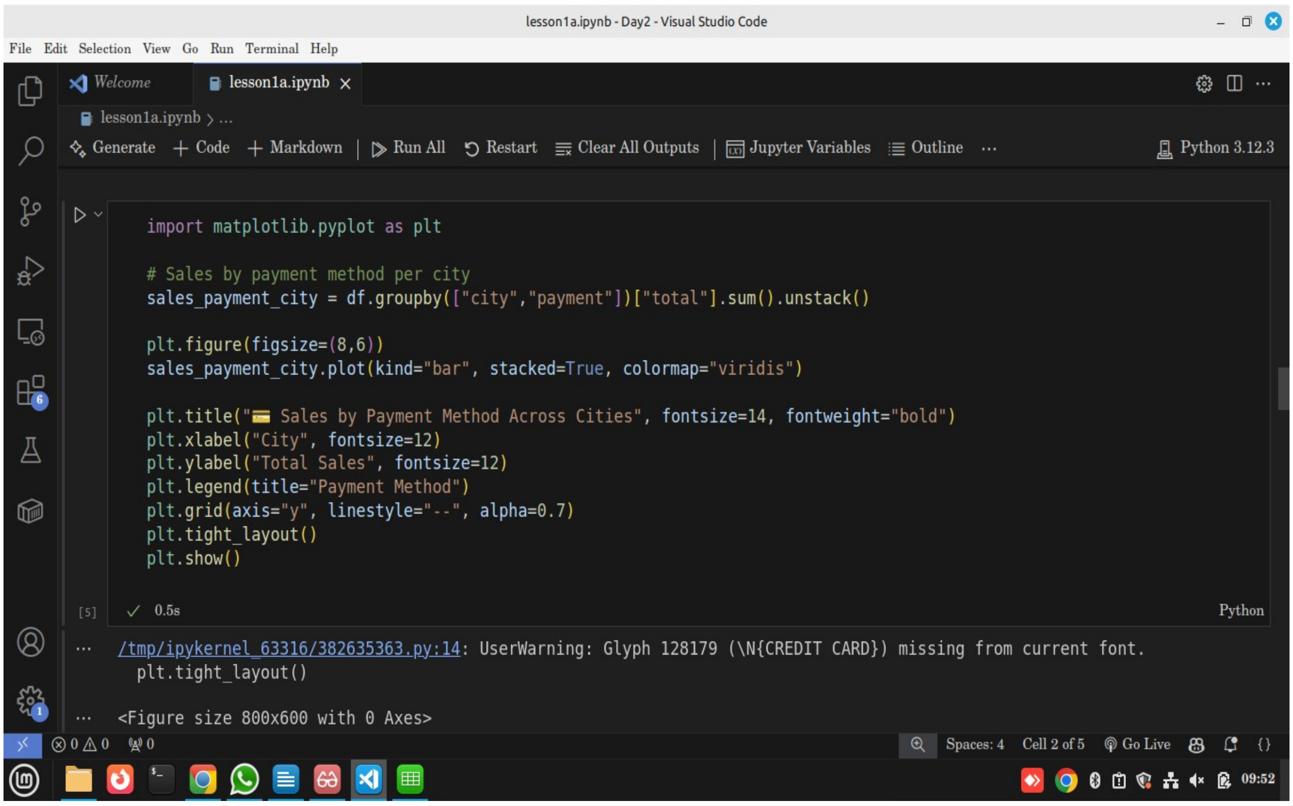
Explanation

- `plt.subplot(1,2,1)` → 1 row, 2 columns, 1st chart.
- `plt.subplot(1,2,2)` → 2nd chart.
- `tight_layout()` avoids overlap.

Use in Data Science

Subplots allow comparing **two metrics** in one glance (sales vs customer satisfaction).

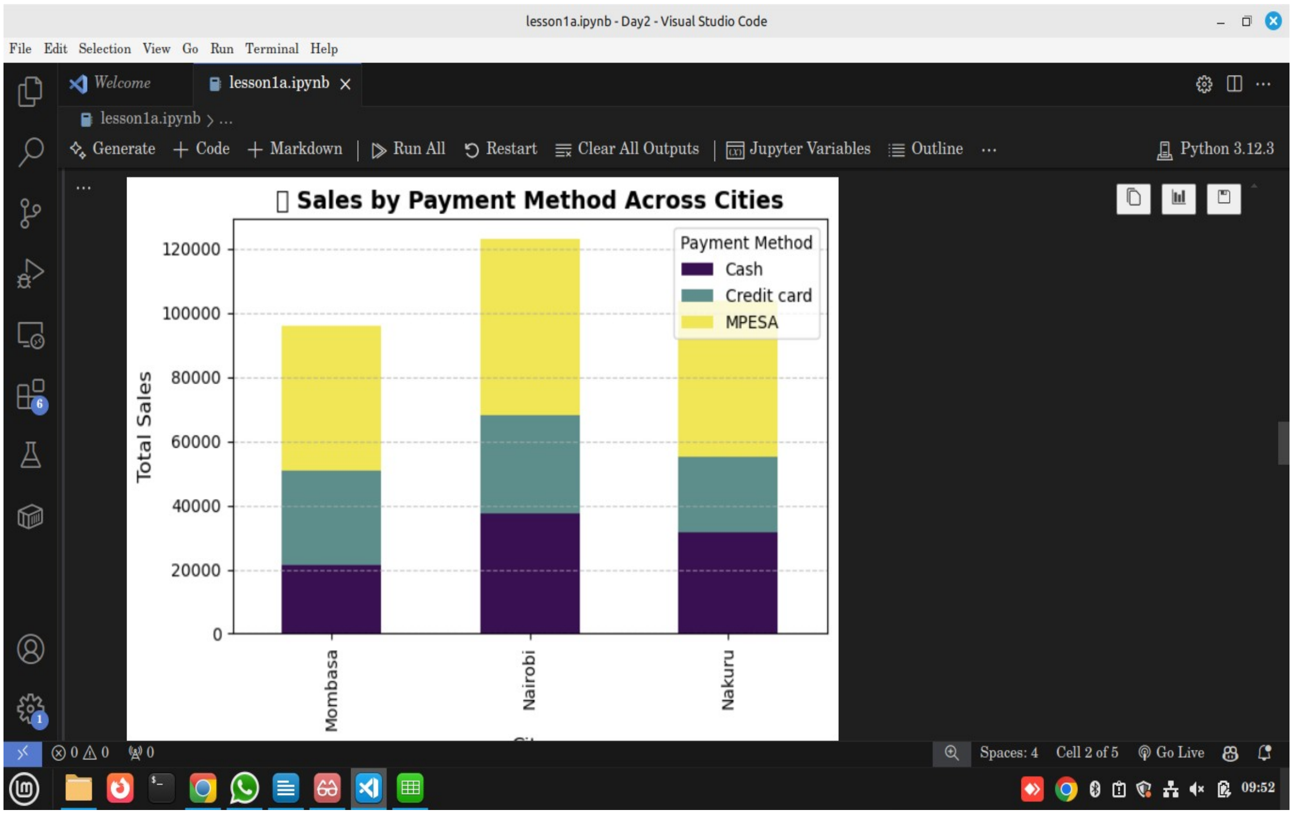
Practical 2: Styling (Fonts, Colors, Grids, Legends)



```
import matplotlib.pyplot as plt

# Sales by payment method per city
sales_payment_city = df.groupby(["city","payment"])["total"].sum().unstack()

plt.figure(figsize=(8,6))
sales_payment_city.plot(kind="bar", stacked=True, colormap="viridis")

plt.title("💳 Sales by Payment Method Across Cities", fontsize=14, fontweight="bold")
plt.xlabel("City", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.legend(title="Payment Method")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_63316/382635363.py:14: UserWarning: Glyph 128179 (\N{CREDIT CARD}) missing from current font.
  plt.tight_layout()

<Figure size 800x600 with 0 Axes>
```

Output



Explanation

- `stacked=True` → payments stacked inside each city bar.
- `colormap="viridis"` → modern gradient palette.
- `grid(alpha=0.7)` → improves readability.

Use in Data Science
Styling makes charts **clearer for presentations/reports**.

Assignment
(see assignment)

Reflection

- Why are **subplots** useful when comparing metrics?
- How does **styling** improve storytelling with data?