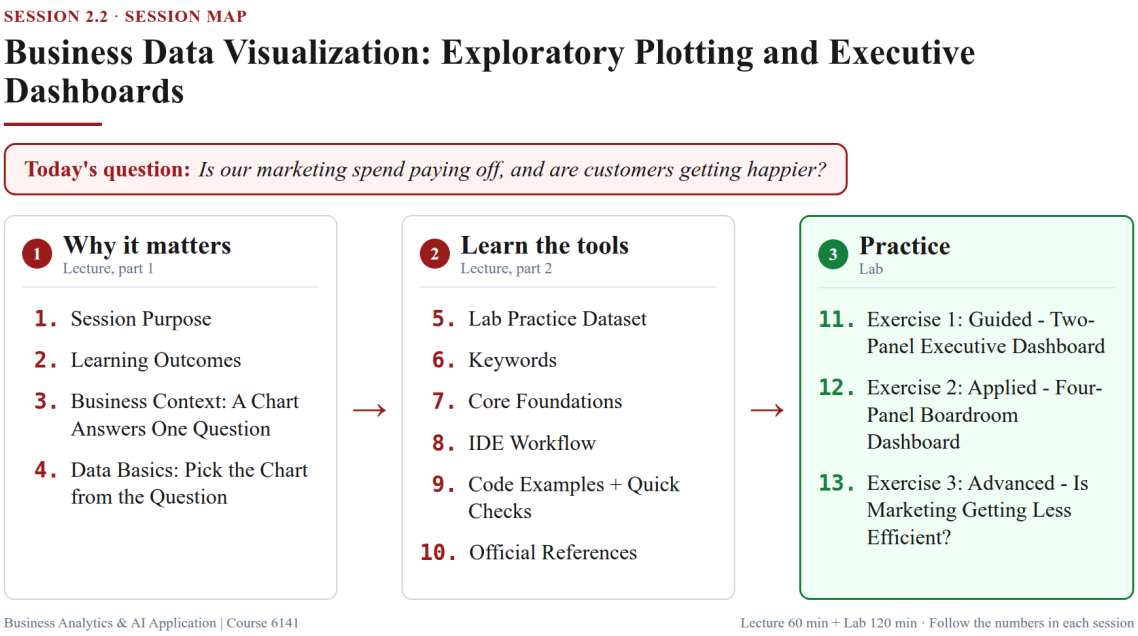
Session 2.2: Business Data Visualization: Exploratory Plotting and Executive Dashboards

Course 6141: Business Analytics & AI Application | Reading for Week 6



Schedule: Week 6 | In-person classes (1:2 Delivery Ratio: 60 Min Lecture + 120 Min Lab)

1. Session Purpose

- The marketing team keeps 12 months of results in one table: revenue, marketing spend, cost per new customer, and customer loyalty
- Executives often have only a minute or two per chart, and a table of 72 numbers hides the trend
- The CMO (Chief Marketing Officer) asks: is our marketing spend paying off, and are customers getting happier? Today, Python turns the table into charts that answer this at a glance

2. Expected Learning Outcomes (By the end of this session, you will be able to)

1. "Is revenue growing month by month?": draw a line chart with Matplotlib (Python's main charting library) using `fig, ax = plt.subplots()` (`fig` is the whole image; `ax` is one chart inside it)
2. "In which months did we spend the most on marketing?": draw a bar chart with Seaborn (a library built on Matplotlib for quick, clean statistical charts)

3. "Which numbers move together?": read a correlation heatmap (a colored grid of correlations, each from -1 to +1), and fix an `IndexError` in a 2x2 dashboard

3. Business Context: A Chart Answers One Question

- A table makes the reader hunt for the pattern; a chart shows it. In our table, revenue rises from `12.4` to `25.8` (USD millions), but the jump in November and December is easy to miss among 72 numbers.
- Five rules for charts that executives trust:
  - One message per chart, and the title states it, for example "Revenue More Than Doubled from Jan to Dec"
  - Label every axis with its unit, such as "USD M" (millions of U.S. Dollars)
  - Bar charts start at zero; a cut-off axis makes small differences look huge
  - No 3D effects or decoration; every mark on the chart should carry data
  - Use colors that still work in black-and-white print and for color-blind readers

4. Data Basics: Pick the Chart from the Question

Business question	Chart	Python
How does a number change over time?	Line chart	`ax.plot()`
Which category is biggest?	Bar chart	`sns.barplot()` or `ax.bar()`
Do two numbers move together?	Scatter plot	`ax.scatter()`
Which of many numbers move together?	Correlation heatmap	`sns.heatmap()`

- **Correlation** measures how closely two columns move together, on a scale from -1 to +1:
  - `+1`: when one goes up, the other always goes up
  - `0`: no straight-line pattern
  - `-1`: when one goes up, the other always goes down
- Correlation is not causation. In our data, marketing spend and revenue have a correlation of `0.996`, but both also simply grew through the year. A high correlation alone does not prove that the spending caused the revenue.

5. Lab Practice Dataset

- File Link: `data/session_02_02_dataset.csv`
- Audited 12-row monthly performance dataset (fictional teaching data), one row per month:
  - `Month_Index`: Month number, 1 to 12.
  - `Month_Name`: Three-letter month name, `Jan` through `Dec`.
  - `Organic_Revenue_USD_M`: Organic revenue in millions of U.S. Dollars. "Organic" means revenue from the company's own operations, not from buying other companies.
  - `Paid_Marketing_Spend_USD_M`: Paid advertising spend in millions of U.S. Dollars.

  - `Customer_Acquisition_Cost_USD`: CAC (Customer Acquisition Cost): the average marketing cost to win one new customer, in U.S. Dollars.
  - `Net_Promoter_Score`: NPS (Net Promoter Score): a customer loyalty score from -100 to +100, based on how likely customers are to recommend the company. Our values range from `58` to `74`.

6. Keywords (Core Programming Concepts)

- `Matplotlib` and `Seaborn`:
  - What: Python charting libraries; Seaborn adds quick, clean statistical charts on top of Matplotlib
  - Why: they turn a DataFrame into a chart image with a few lines of code
- `Figure` and `Axes`:
  - What: the `Figure` is the whole image; each `Axes` is one chart inside it
  - Why: you draw on a specific `Axes`, so a dashboard can hold several charts
- `Line Chart`:
  - What: points joined in time order
  - Why: shows trends, such as monthly revenue
- `Bar Chart`:
  - What: one bar per category; the bar length is the value
  - Why: compares sizes, such as spend by month
- `Scatter Plot`:
  - What: one dot per row, with one column on each axis
  - Why: shows whether two numbers move together
- `Correlation Heatmap`:
  - What: a grid of correlations, colored from -1 to +1
  - Why: spots which of many numbers move together, in one picture

7. Core Programming & Computational Foundations

- **7.1 Figure and Axes: the canvas and the charts on it**
  - *The idea:* Matplotlib draws in two layers. The **Figure** is the whole image, like a blank sheet of paper. An **Axes** is one chart drawn on that sheet, with its own x-axis, y-axis, title, and data. One Figure can hold one Axes or many. `fig, ax = plt.subplots()` creates both at once and gives you a name for each, so every later instruction says exactly which chart it changes.
  - *See it on real data:* the revenue chart in Micro-Example 1.

Instruction	Goes to	Effect
`fig, ax = plt.subplots(figsize=(10, 5))`	creates both	a 10 x 5 inch image with one chart area

Instruction	Goes to	Effect
`ax.plot(df["Month_Name"], df["Organic_Revenue_USD_M"])`	the chart	draws 12 points joined by a line
`ax.set_title("Revenue More Than Doubled from Jan to Dec")`	the chart	the chart's title
`ax.set_ylabel("Organic Revenue (USD M)")`	the chart	the y-axis label
`fig.savefig("data/revenue_trend.png")`	the whole image	writes the image to a file

  - *Common mistake:* mixing the two styles. Many online examples write `plt.plot()` and `plt.title()`, which draw on "whatever chart is active". With several charts, that is often the wrong one. Always draw on a named `ax`.
  - *More examples:*

Code	Result	What it tells you
`fig, ax = plt.subplots()`	one `Figure`, one `Axes`	the most common start for one chart
`fig, (ax1, ax2) = plt.subplots(1, 2)`	2 charts side by side	1 row, 2 columns; each gets its own name
`fig, axes = plt.subplots(2, 2)`	`axes.shape` is `(2, 2)`	4 charts, stored as a 2 x 2 table
`len(axes.flatten())`	`4`	`.flatten()` lines the 4 charts up in one row, for a loop
`ax.set_xlabel("")`	no x-axis label	remove a label that adds nothing

  - *Remember:* the Figure is the page; each Axes is a chart; draw on a named `ax`.

- **7.2 Choose the chart from the question**
  - *The idea:* Each chart type answers one kind of question well. Start from the question the manager asks, then pick the chart; do not start from the chart you like.

Question	Chart	Example from our data
How does a number change over time?	line (`ax.plot()`)	revenue from Jan (`12.4`) to Dec (`25.8`)
Which category is biggest?	bar (`sns.barplot()`, `ax.bar()`)	marketing spend peaks in Dec (`6.4`)
Do two numbers move together?	scatter (`ax.scatter()`)	cost per new customer vs loyalty
Which of many numbers move together?	heatmap (`sns.heatmap()`)	all 4 measures at once

  - *Common mistake:* a line chart for categories that have no order (such as regions), or a pie chart with many slices. A line suggests a trend from one point to the next; it only makes sense when the x-axis has an order, such as months.
  - *More examples:*

Question	Best chart	Why
Did revenue grow every month?	line	the eye follows the slope from month to month
Which 3 months had the most spend?	bar, sorted	lengths are easy to compare
Does a higher CAC go with a higher NPS?	scatter	one dot per month shows the pattern
Is any measure moving against the others?	heatmap	a blue cell would stand out among red ones

  - *Remember:* question first, chart second.

- **7.3 Correlation: how closely two columns move together**
  - *The idea:* The correlation of two columns is one number from `-1` to `+1`. Near `+1`, when one goes up the other goes up too; near `-1`, when one goes up the other goes down; near `0`, there is no straight-line pattern. `df["A"].corr(df["B"])` computes it for two columns, and `df[cols].corr()` for every pair at once.
  - *See it on real data:*

Pair of columns	Correlation	Meaning
marketing spend and revenue	`0.996`	they rose almost perfectly together
month number and revenue	`0.946`	revenue grew through the year
CAC and NPS	`0.976`	months with costlier customers also had happier customers
month number and revenue per marketing dollar	`-0.95`	each dollar of marketing brought in less as the year went on

  - *Common mistake:* reading correlation as cause. Spend and revenue both grew month by month, so they correlate strongly, but the data cannot tell whether the spend **caused** the revenue, or whether both simply followed the holiday season.
  - *More examples:*

Code	Result	What it tells you
`pd.Series([1, 2, 3]).corr(pd.Series([2, 4, 6]))`	`1.0`	a perfect straight-line rise
`pd.Series(range(1, 13)).corr(-pd.Series(range(1, 13)))`	`-1.0`	a perfect opposite movement
`df["Month_Index"].corr(df["Net_Promoter_Score"])`	`0.963`	loyalty also rose through the year
`df[cols].corr()`	a 4 x 4 table	every pair at once; the diagonal is always `1.0`

  - *Remember:* correlation measures moving together, not cause and effect.

- **7.4 Saving a chart: size and sharpness**
  - *The idea:* `fig.savefig(file, dpi=...)` writes the chart to an image file. The final size in pixels is the figure size in inches times the dpi (dots per inch). More dots mean a sharper image and a larger file.
  - *See it on real data:* a `figsize=(10, 5)` chart saved three ways.

`dpi`	Image size in pixels	Good for
`100`	1000 x 500	a quick look
`150`	1500 x 750	screens and slides
`300`	3000 x 1500	printed reports

  - *Other settings:* `bbox_inches="tight"` trims empty white borders. The file type follows the name: `.png` for images, `.pdf` for documents that stay sharp at any zoom.
  - *Common mistake:* calling `plt.savefig()` after `plt.show()`. In many setups, showing the chart clears it, so the saved file is empty. Save first.
  - *More examples:*

Settings	Result	What it tells you
`figsize=(6, 4)`, `dpi=100`	600 x 400 pixels	a small image for a quick check
`figsize=(6, 4)`, `dpi=300`	1800 x 1200 pixels	the same chart, sharp enough for print
`figsize=(10, 5)`, `dpi=150`, `bbox_inches="tight"`	1233 x 674 pixels	smaller than 1500 x 750, because the empty border was trimmed
`fig.savefig("data/chart.pdf")`	a PDF file	stays sharp at any zoom, good for reports

  - *Remember:* inches x dpi = pixels; save before you show.

- **7.5 Grids of charts: rows and columns start at `0`**
  - *The idea:* `fig, axes = plt.subplots(2, 2)` makes 4 charts, and `axes` is a small table of them with 2 rows and 2 columns. You reach one chart with its row number and column number, both counted from `0`, just like `.iloc[]` in Session 1.3.

Position	Code
top left	`axes[0, 0]`
top right	`axes[0, 1]`
bottom left	`axes[1, 0]`
bottom right	`axes[1, 1]`

  - *Common mistake:* `axes[2]` asks for a third row, which does not exist, so Python stops with `IndexError: index 2 is out of bounds for axis 0 with size 2`. "Axis 0" means the rows; "size 2" means there are only rows `0` and `1`.
  - *Spacing:* `plt.tight_layout()` moves the charts apart so that titles and axis labels do not overlap.
  - *More examples:*

Code	Result	What it tells you
`axes[1, 0]`	bottom-left chart	row 1 (second row), column 0 (first column)
`axes[-1, -1]`	bottom-right chart	`-1` means "the last", for rows and columns alike
`axes.flatten()[2]`	bottom-left chart	after flattening, the charts are numbered 0 to 3, row by row
`fig, axes = plt.subplots(3, 1)`	`axes.shape` is `(3,)`	one column of 3 charts: now `axes[2]` exists
`axes[2]` with a 2 x 2 grid	`IndexError`	there is no third row

  - *Remember:* `axes[row, column]`, both from `0`.
- **7.6 Honest charts: bars start at zero**
  - *The idea:* A bar's length is read as its size. If the y-axis starts above zero, the bars are cut off and small differences look huge.
  - *See it on real data:* marketing spend in Oct (`4.1`) and Nov (`5.2`).

y-axis starts at	Nov bar compared with Oct bar	What a reader believes
`0`	`1.27` times as long	Nov spend was about a quarter higher (true)
`4.0`	`12.0` times as long	Nov spend was 12 times higher (false)

  - *Other rules:* one message per chart, stated in the title; every axis labeled with its unit; no 3D effects; colors that still work in black-and-white print.

– *More examples:*

Chart	What the reader sees	Honest?
Jan (`2.1`) vs Dec (`6.4`), axis from `0`	Dec bar `3.05` times as long	true: spend tripled
the same, axis from `2.0`	Dec bar `44.0` times as long	false: the cut axis exaggerates fifteenfold
revenue line chart, axis from `12`	the line fills the chart	acceptable: a line shows change, not size, so its axis need not start at `0`
a 3D bar chart	bars at the back look smaller	avoid 3D: depth distorts the lengths

– *Remember:* if the axis is cut, the bar chart lies.

8. Development Workflow in Google Antigravity IDE

- `[1]` File Explorer: create your script with your name and the session code (for example, `Reacher_S22_demo_1.py`), and check the dataset is in `data/`. Saved charts appear here as `.png` files; click one to open it.
- `[2]` Code Editor: write a few lines, run, repeat. Small steps keep errors small.
- `[3]` AI Agent Chat: on an error, capture the traceback (Python's error report) with `Win + Shift + S` (macOS: `Cmd + Shift + 4`), paste it with `Ctrl + V`, and ask why it failed.
- `[4]` Terminal: run `python` followed by your script name. Return code `0` means the script finished with no errors. First time only: `pip install matplotlib seaborn`.

9. Hands-On Coding Micro-Examples

- Every example has three parts: **1. Input**, **2. Process**, and **3. Output**. Name each file with your own name and the session code, for example `Reacher_S22_demo_1.py` (pattern: `YourName_S22_demo_1.py`).
- The examples save each chart as a `.png` file in `data/`. Open the file in Panel `[1]` to see the chart.
- *Micro-Example 1 (Line Chart: Revenue Trend):* `YourName_S22_demo_1.py`

```
# 1. Input: load the monthly results
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("data/session_02_02_dataset.csv")

# 2. Process: one figure with one chart, revenue by month
fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(df["Month_Name"], df["Organic_Revenue_USD_M"],
        color="#991B1B", marker="o")
ax.set_title("Revenue More Than Doubled from Jan to Dec")
```

```
ax.set_ylabel("Organic Revenue (USD M)")

# 3. Output: save the chart as an image file
fig.savefig("data/revenue_trend.png", dpi=150, bbox_inches="tight")
print("Saved: data/revenue_trend.png")
```

  - *How it works, line by line:*
    - `import matplotlib.pyplot as plt` loads Matplotlib's charting tools under the short name `plt`.
    - `plt.subplots(figsize=(10, 5))` creates the Figure (10 inches wide, 5 tall) and one Axes, named `fig` and `ax` (section 7.1).
    - `ax.plot(x, y, color="#991B1B", marker="o")` draws the line: month names along the x-axis, revenue up the y-axis. `color` is a color code (`#991B1B` is a dark red) and `marker="o"` puts a dot on each month.
    - `ax.set_title(...)` writes the message of the chart, not just its topic: a reader knows the conclusion before looking at the line.
    - `fig.savefig(..., dpi=150, bbox_inches="tight")` writes a 1500 x 750 pixel image and trims the white border (section 7.4).
  - *Read the output:*

    ```
    Saved: data/revenue_trend.png
    ```

    - The Terminal only confirms the file; the chart itself is in `data/revenue_trend.png`. Open it from Panel `[1]`.
    - The line climbs from `12.4` in Jan to `25.8` in Dec, `2.08` times higher, with the steepest rise in Nov and Dec.

- *Quick Check 1 (`ax.grid()`):*
  - New tool: `ax.grid(True, linestyle=":")` adds light dotted guide lines behind the chart. Why: readers can line up a point with its value on the axis.
  - Task: add one line at the end of part **2. Process**: `ax.____(True, linestyle=":")`
  - Check: open `data/revenue_trend.png`; dotted grid lines now sit behind the revenue line.

- *Micro-Example 2 (Bar Chart: Marketing Spend by Month):* `YourName_S22_demo_2.py`

```
# 1. Input: load the monthly results
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("data/session_02_02_dataset.csv")

# 2. Process: one bar per month
fig, ax = plt.subplots(figsize=(10, 4))
sns.barplot(data=df, x="Month_Name", y="Paid_Marketing_Spend_USD_M",
            color="#475569", ax=ax)
ax.set_title("Marketing Spend Peaks in November and December")
ax.set_xlabel("")
```

```
ax.set_ylabel("Paid Marketing Spend (USD M)")

# 3. Output: save the chart
fig.savefig("data/marketing_spend_bar.png", dpi=150, bbox_inches="tight")
print("Saved: data/marketing_spend_bar.png")
```

  - *How it works, line by line:*
    - `import seaborn as sns` loads Seaborn, which draws common statistical charts with less code than Matplotlib alone.
    - `sns.barplot(data=df, x="Month_Name", y="Paid_Marketing_Spend_USD_M", ...)` reads the columns by name from the DataFrame and draws one bar per month. `ax=ax` tells Seaborn to draw on our named chart.
    - Seaborn's bars start at zero, so their lengths can be compared honestly (section 7.6).
    - `ax.set_xlabel("")` removes the automatic x-axis label (`Month_Name`), because the month names already explain the axis.
  - *Read the output:*

    ```
    Saved: data/marketing_spend_bar.png
    ```

    - In the image, the bars grow through the year, and the last two are clearly the tallest: Nov `5.2` and Dec `6.4`, against `2.1` in Jan.
    - The title states this finding, so the chart can go straight into a report.

- *Quick Check 2 (`.nlargest()`):*
  - New tool: `df.nlargest(3, "Column_Name")` returns the 3 rows with the largest values in that column. Why: confirms the chart's message with exact numbers.
  - Task: add one line to part **3. Output**: `print(df._____(3, "Paid_Marketing_Spend_USD_M")[["Month_Name", "Paid_Marketing_Spend_USD_M"]])`
  - Check: the Terminal lists `Dec` (`6.4`), `Nov` (`5.2`), and `Oct` (`4.1`).

- *Micro-Example 3 (Correlation Heatmap):* `YourName_S22_demo_3.py`

```
# 1. Input: load the monthly results
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("data/session_02_02_dataset.csv")

# 2. Process: correlation between every pair of 4 columns
cols = ["Organic_Revenue_USD_M", "Paid_Marketing_Spend_USD_M",
        "Customer_Acquisition_Cost_USD", "Net_Promoter_Score"]
corr = df[cols].corr().round(2)
fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap="RdBu_r", vmin=-1, vmax=1, ax=ax)
ax.set_title("All Four Measures Rose Together in 2024")

# 3. Output: print the numbers, then save the chart
```

```
print(corr)
fig.savefig("data/kpi_correlation_heatmap.png", dpi=150,
            bbox_inches="tight")
```

  - *How it works, line by line:*
    - `cols = [...]` lists the 4 measures to compare. A list keeps the code short and makes it easy to add a column later.
    - `df[cols].corr()` computes the correlation of every pair, a 4 x 4 table (section 7.3). `.round(2)` keeps 2 decimals.
    - `sns.heatmap(corr, annot=True, cmap="RdBu_r", vmin=-1, vmax=1, ax=ax)` colors each cell: red for positive, blue for negative, white near zero. `annot=True` writes the number in each cell, and `vmin` / `vmax` fix the color scale from `-1` to `+1`, so a pale cell really means a weak correlation.
    - `print(corr)` shows the same numbers in the Terminal.
  - *Read the output:*

```
                            Organic_Revenue_USD_M  ...  Net_Promoter_Score
Organic_Revenue_USD_M                        1.00  ...                0.98
Paid_Marketing_Spend_USD_M                   1.00  ...                0.97
Customer_Acquisition_Cost_USD                1.00  ...                0.98
Net_Promoter_Score                           0.98  ...                1.00
```

    - The `...` hides middle columns that do not fit the Terminal width; the image shows all 16 cells.
    - Every value is between `0.97` and `1.00`: all four measures rose together through the year, so the heatmap is almost entirely dark red.
    - This is a reminder of section 7.3: when everything grows over time, everything correlates, and correlation alone cannot say what caused what.

- *Quick Check 3 (`.corr()` on two columns):*
  - New tool: `column_a.corr(column_b)` returns one correlation number for two columns. Why: answers a single question without a whole heatmap.
  - Task: add one line to part **3. Output**: `print(round(df["Paid_Marketing_Spend_USD_M"].__ __(df["Organic_Revenue_USD_M"]), 3))`
  - Check: the Terminal shows `0.996`.

10. Official Python Documentation & References

- Matplotlib Official Documentation: User Guide.
- Seaborn Official Documentation: Statistical Data Visualization.
- Edward Tufte: The Visual Display of Quantitative Information (Principles of Information Design).

11. Exercise 1: Guided Practice - Two-Panel Executive Dashboard

- *Step 1: Open Script:* In Google Antigravity IDE, create a new script named `YourName_S22_dashboard.py`, replacing `YourName` with your own name (for example, `Reacher_S22_dashboard.py`).
- *Step 2: Build Two Charts Side by Side:* Type the following code:

```
# 1. Input: load the monthly results
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("data/session_02_02_dataset.csv")

# 2. Process: one figure, two charts side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Left chart: revenue trend
ax1.plot(df["Month_Name"], df["Organic_Revenue_USD_M"],
         color="#991B1B", marker="s", linewidth=2)
ax1.set_title("Monthly Revenue (USD M)")
ax1.grid(True, linestyle=":")

# Right chart: cost per new customer vs customer loyalty
ax2.scatter(df["Customer_Acquisition_Cost_USD"],
            df["Net_Promoter_Score"], color="#1E293B", s=80)
ax2.set_title("Loyalty (NPS) vs Cost per New Customer (CAC)")
ax2.set_xlabel("CAC (USD)")
ax2.set_ylabel("Net Promoter Score")
ax2.grid(True, linestyle=":")

# 3. Output: save the dashboard
plt.tight_layout()
fig.savefig("data/executive_dashboard_2panel.png", dpi=200)
print("Dashboard saved: data/executive_dashboard_2panel.png")
```

  - `plt.subplots(1, 2)` makes 1 row with 2 charts, and `(ax1, ax2)` names them left and right.
  - `ax.scatter(x, y)` draws one dot per month; `s=80` sets the dot size.

- *Step 3: Inspect Output:* Open `data/executive_dashboard_2panel.png` in Panel `[1]` and check that titles and axis labels do not overlap.
- *Step 4: Execute in Terminal:* Run `python YourName_S22_dashboard.py` in Panel `[4]` and confirm return code `0`.
- *Success Check:* the image has 2 charts. The right chart's dots climb from bottom-left to top-right: months with a higher cost per new customer also had higher loyalty, which matches their correlation of `0.98`.

12. Exercise 2: Applied Practice - Four-Panel Boardroom Dashboard

- *Coding Challenge:* In `YourName_S22_practice.py`, build a 2x2 dashboard with 4 charts: revenue trend (line), marketing spend by month (bar), cost per new customer by month (line), and the correlation heatmap.
- *Task 1 (2x2 Layout):* Create the grid with `fig, axes = plt.subplots(2, 2, figsize=(16, 10))`. Draw each chart on its own position: `axes[0, 0]`, `axes[0, 1]`, `axes[1, 0]`, and `axes[1, 1]`. Pass `ax=axes[1, 1]` to `sns.heatmap()`.
- *Task 2 (Deliberate IndexError & AI Diagnosis):* Change one position to `axes[2].plot(...)`. Run the script and read `IndexError: index 2 is out of bounds for axis 0 with size 2`. Capture the traceback with `Win + Shift + S`, paste it into Panel `[3]` (AI Agent Chat) with `Ctrl + V`, and review the AI's explanation of row and column numbers.
- *Task 3 (Fix and Export):* Fix the position, call `plt.tight_layout()`, and save the dashboard to `data/boardroom_dashboard.png` with `dpi=300`.
- *Success Check:* `data/boardroom_dashboard.png` shows 4 charts with no overlapping titles, and the script ends with return code `0`.

13. Exercise 3: Advanced Practice - Is Marketing Getting Less Efficient?

- *Goal:* Answer the CMO's question with a new measure and one honest chart: does each marketing dollar still bring in as much revenue at the end of the year as at the start?
- *What you need:*
  - File: `data/session_02_02_dataset.csv` (described in **5. Lab Practice Dataset**).
  - Script name: `YourName_S22_challenge.py`.
  - Tools you already know: column math, `.round()`, `.mean()`, `.sort_values()`, a `for` loop with `if` / `else` (Session 1.2), `fig, ax = plt.subplots()`, and `fig.savefig()`.
- *Requirements (follow them in order):*
  1. Add `Revenue_per_Marketing_USD` = `Organic_Revenue_USD_M` / `Paid_Marketing_Spend_USD_M`, rounded to 2 decimal places. It shows how many dollars of revenue came with each dollar of paid marketing in that month.
  2. Compute the 12-month average of that column and round it to 2 decimal places.
  3. Build a list of 12 colors with a `for` loop: `"#991B1B"` (red) for months **below** the average, `"#94A3B8"` (grey) for all others.
  4. Draw a bar chart with `ax.bar()`: month names on the x-axis, `Revenue_per_Marketing_USD` as the bar heights, and your color list.
  5. Add a dashed horizontal line at the average, then a title that states your finding and a y-axis label with its unit ("USD of revenue per USD of marketing").
  6. Save the chart to `data/marketing_efficiency.png` with `dpi=150`.
  7. Print the average, then the best and the worst month with their values.
- *New tools for this exercise:*
  - `ax.bar(x_values, heights, color=color_list)`: Matplotlib's own bar chart. When `color` is a list, bar 1 gets color 1, bar 2 gets color 2, and so on.
  - `ax.axhline(y=5.08, linestyle="--", color="black")`: draws a horizontal line across the chart at height `y`, useful for marking an average or a target.

  - Building a list in a loop: start with `colors = []`, then `colors.append("#991B1B")` adds one item to the end.

- *Hints:*
  - Loop over the column itself: `for value in df["Revenue_per_Marketing_USD"]:`.
  - For the best and worst month, sort the DataFrame by the new column and read the first and last rows, for example with `.head(1)` and `.tail(1)`.

- *Self-Check (expected results):*
  - Average: `5.08`. Best month: `Jan` (`5.9`). Worst month: `Dec` (`4.03`).
  - The 6 red bars are `Jul` through `Dec`: every month in the second half of the year is below average.
  - Business meaning: spending rose faster than revenue. Revenue doubled, but each marketing dollar brought in about 32% less revenue in December than in January, and the cost per new customer rose from `48.5` to `68.0`. The CMO should ask whether the November and December campaigns were worth their extra cost.