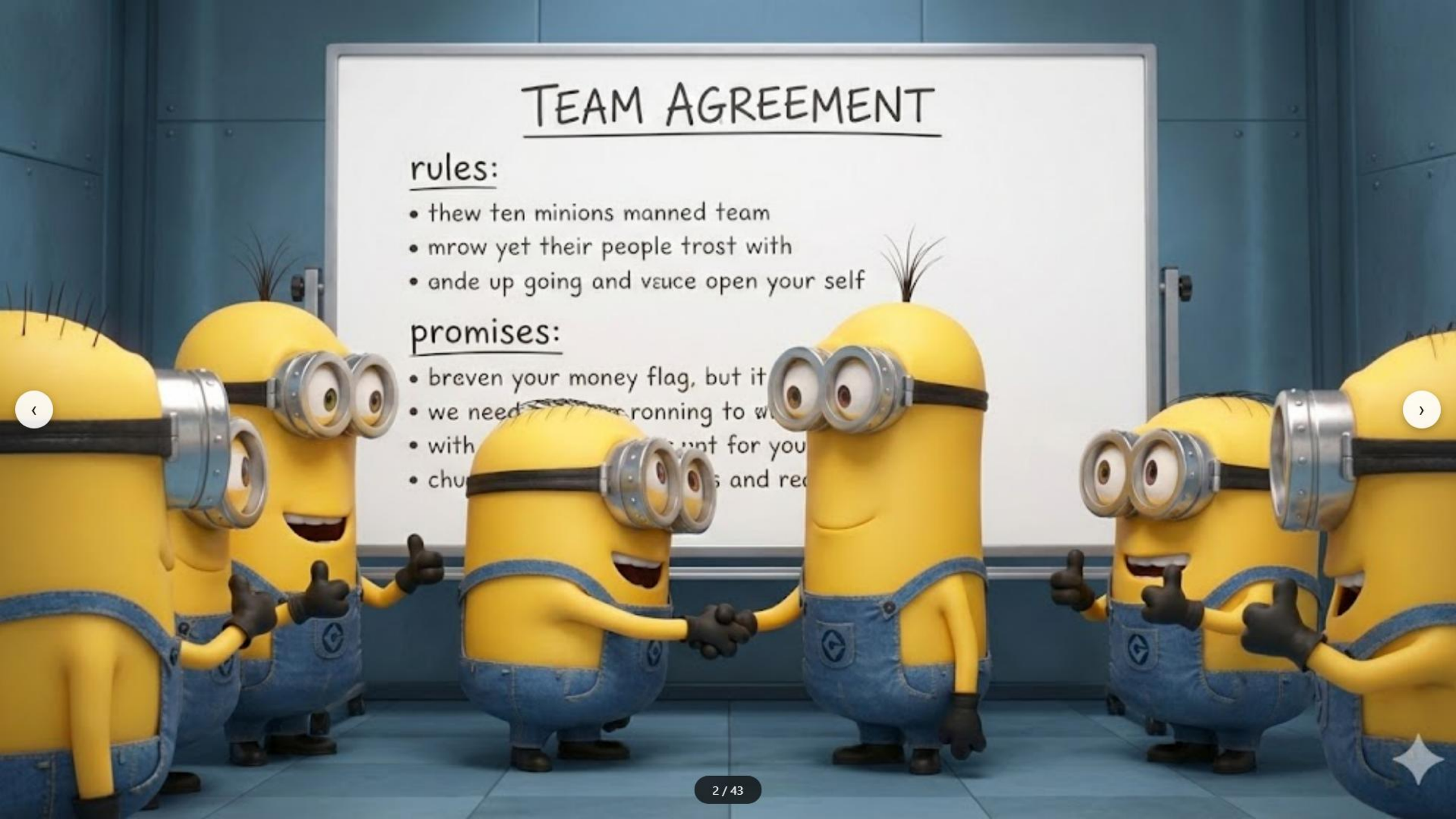
WELCOME!


TEAM AGREEMENT
rules:
• thew ten minions manned team
• mrow yet their people trost with
• ande up going and vauce open your self
promises:
• breven your money flag, but it
• we need ronning to w
• with nt for you
• chu s and re

WORKSHOP MENU:
CODE & CHAOS
APPETIZERS (Intro)
Hello World
MAIN COURSES (ETL & Viz)
1. The Great Merger
2. The Detective
3. The Messy Survey
4. The Manager's Chart
DESSERTS (Auto & Bonus)
5. The Report Generator
6. The Professional Polish
7. The File Organizer
8. The Big Data Stress Test

INTRO - Intro: Hello World - The First Flight
ETL 1 - Exercise: The Great Merger
ETL 2 - Exercise: ETL 2 - The Detective
ETL 3 - Exercise: ETL 3 - The Messy Survey (The Human Factor)
VIZ 1 - Exercise: The Manager's Chart
VIZ 2 - Exercise: The Report Generator (Viz 2)
AUTO 1 - Exercise: The Professional Polish ("The Bridge")
AUTO 2 - Exercise: The automated File Organizer
BONUS - Exercise: The Big Data Stress Test
Bonus Exercises
Break Time
Finish Session

INTRO

Intro: Hello World - The First Flight

Intro: Hello World - The First Flight

The scenario

It's your first day. You have installed VS Code and Python, but it looks like a cockpit of a plane you don't know how to fly. You need to prove to your boss (and yourself) that your setup is working.

Your mission

Verify that your "cockpit" (VS Code + Python) is fully operational. You will write your first lines of code to print a message to the screen, perform a business calculation, and automate a repetitive task using a loop.

- **The Environment:** Create a new file named `hello.py`.
- **The First Prompt:** Use Copilot to ask: "How do I print 'Hello World' in Python?". Run the code.
- **The Calculator:** Ask Copilot: "Create a variable for revenue = 1000 and cost = 400. Calculate profit and print it."
- **The Loop:** Ask Copilot: "Write a loop that prints numbers 1 to 5."

Expected result

You have printed "Hello World", calculated profit, and printed numbers 1 to 5 — your setup is working.

ETL 1

Exercise: The Great Merger

Exercise: The Great Merger

The scenario

Your boss has emailed you three separate Excel files containing quarterly sales data (Q1, Q2, and Q3). She needs a single "Year-To-Date" report by lunchtime. Copy-pasting manually is slow and prone to errors.

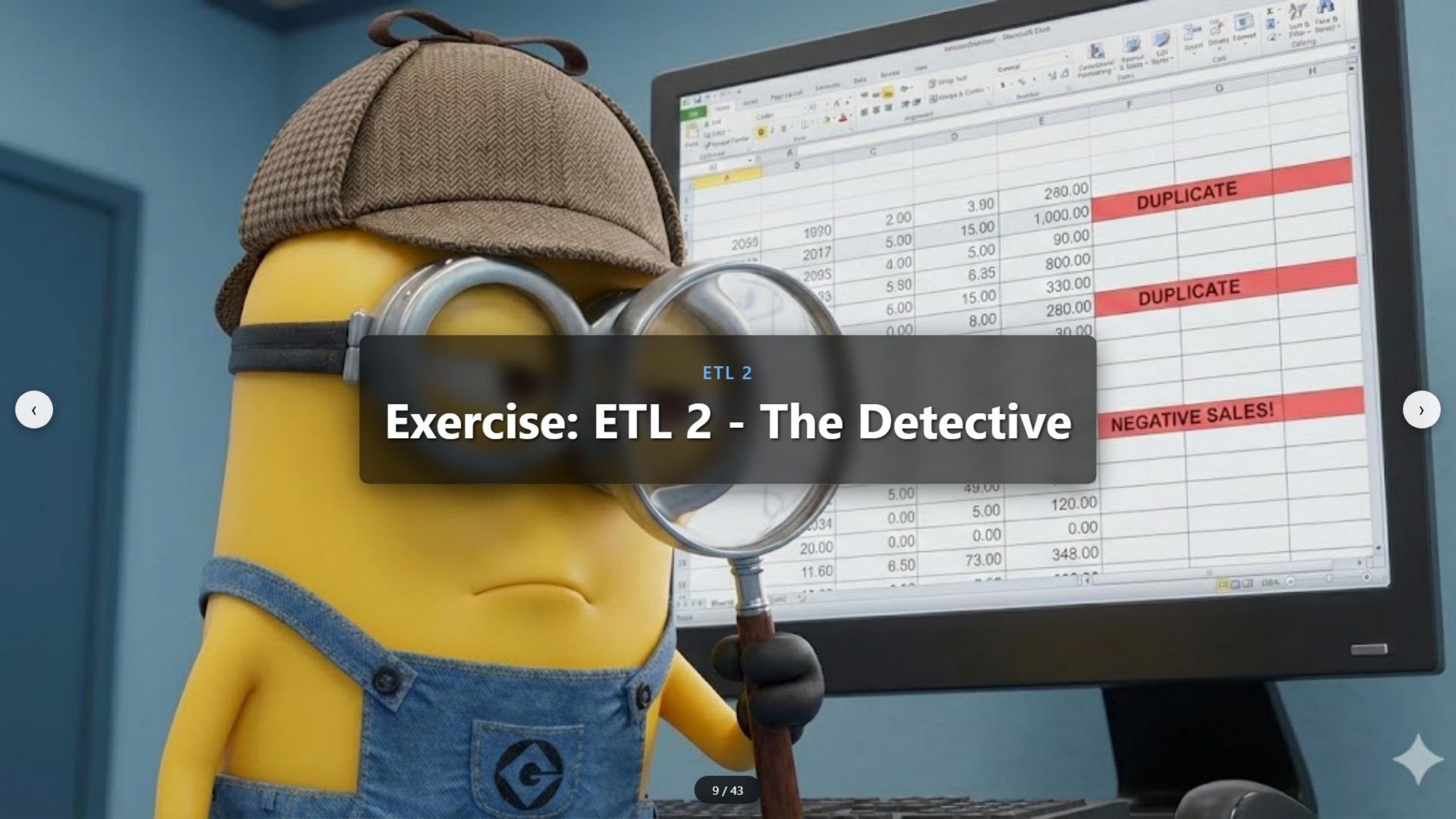
Your mission

Use Python to automate this common reporting task: load the three files, stack them into one master table, verify the data count, and export the result to a CSV file.

- Load the three files.
- Stack them into one master table.
- Verify the data count.
- Export the result to a CSV file.

Expected result

A single file named `combined_sales.csv` containing all records from the three quarterly files.



ETL 2

Exercise: ETL 2 - The Detective

Exercise: ETL 2 - The Detective

The scenario

You are a Data analyst at *Global Widgets Inc.* You successfully merged the sales files in the previous project. However, the Finance team has rejected your report. "The numbers don't add up," they said. "We see the same Order ID appearing twice, and why do we have negative sales? Are we paying people to take our products?" Your job is to act as a detective: separate the good data from the bad.

Your mission

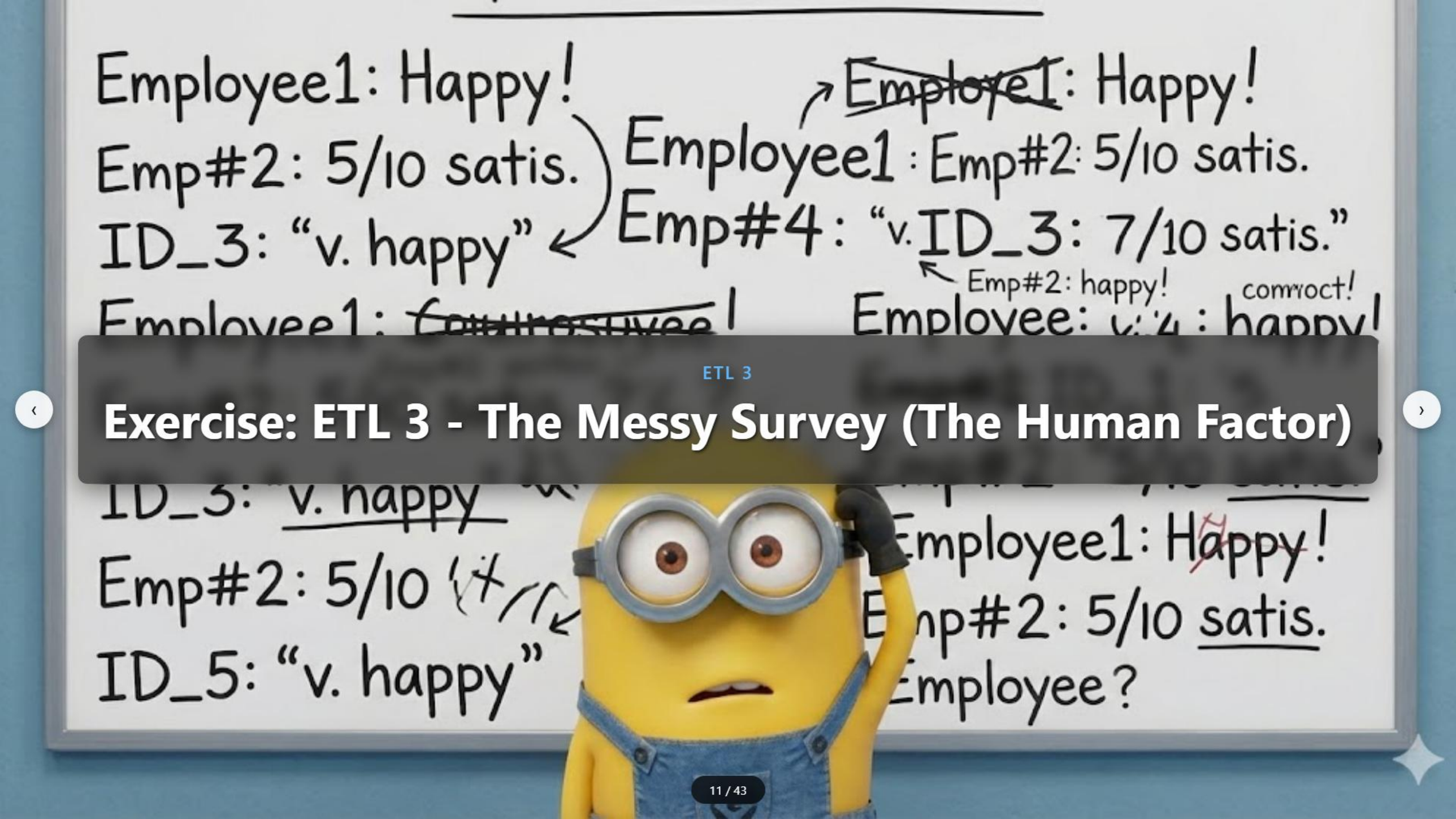
- Load the `combined_sales.csv` file.
- Find and Remove duplicate `Order_ID`s (glitch in the system).
- Filter out negative numbers in the `Revenue` column (erroneous refunds).
- Save the clean dataset as `clean_sales_data.csv`.

(Optional) Tier 2 Challenge: Identify "Suspiciously High" revenue transactions (outliers) that are statistically impossible (3 standard deviations from the mean).

Expected result

A clean file `clean_sales_data.csv` with no duplicate Order IDs and no negative revenue — ready for reporting.

Exercise: ETL 3 - The Messy Survey (The Human Factor)



Exercise: ETL 3 - The Messy Survey (The Human Factor)

The scenario

While you are finalizing the sales data, HR pings you. They have an "Employee Satisfaction Survey" (~1000 rows) and want to know if happy sales reps sell more. But their data is a disaster: Inconsistent Data (some typed "Male", others "M", "m", or " F "), Mixed Dates (join dates in every format imaginable), and Unstructured Text (comments with hidden gems like emails or complaint keywords).

Your mission

Clean this dataset so it can eventually be accurately joined with your Sales data and used for analysis.

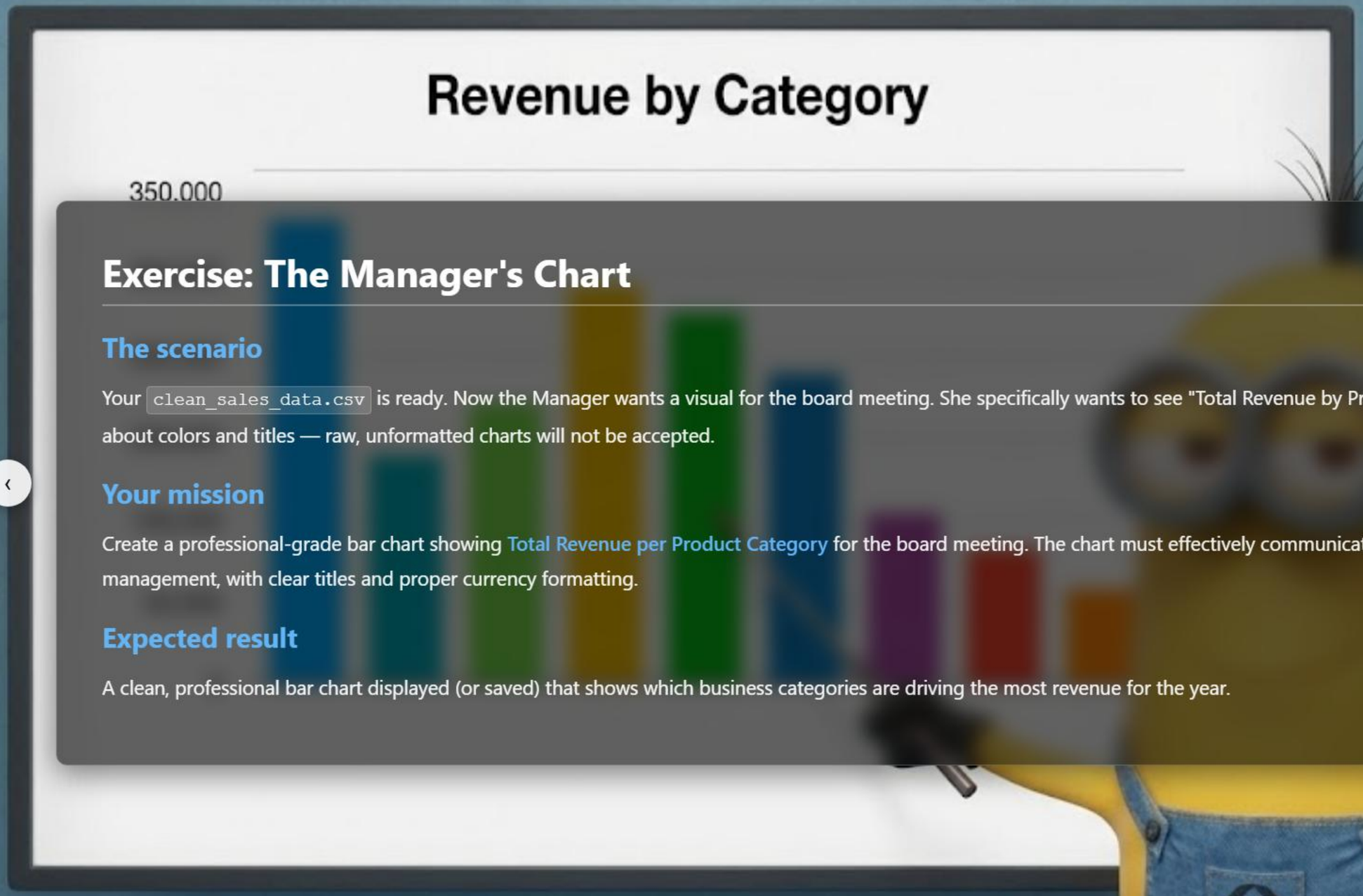
- Standardize the `Gender` column (Male/Female/Other).
- Standardize `Join_Date` (Datetime objects).
- Add a new flag column `Compensation_Issue` (True/False) based on keywords.
- (Optional) Add a new column `extracted_email` pulled from the comments using Regex.

Expected result

A clean DataFrame (and CSV) with standardized Gender, Join_Date, and the new columns — ready to join with Sales data.

VIZ 1

Exercise: The Manager's Chart



Exercise: The Manager's Chart

The scenario

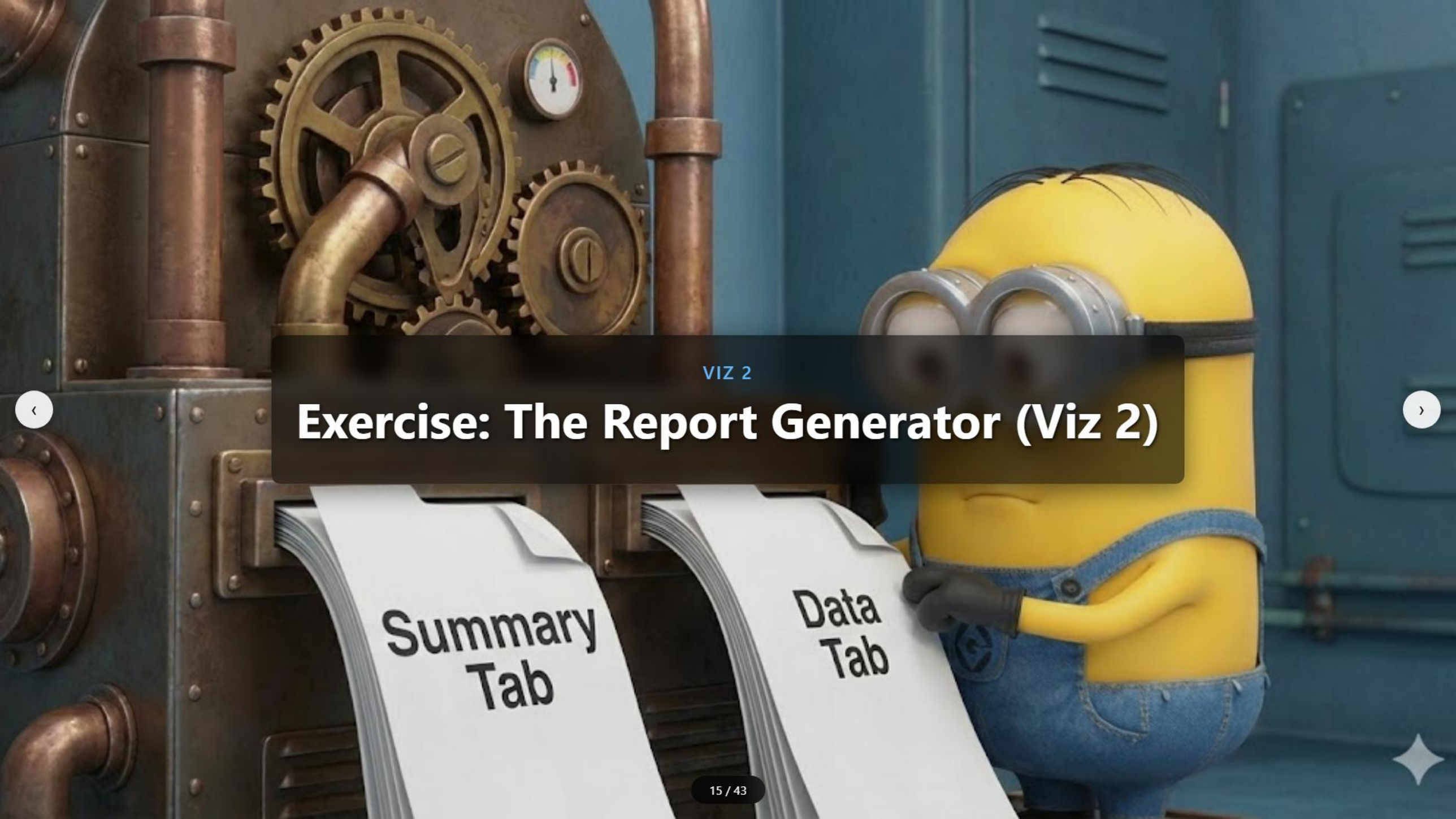
Your `clean_sales_data.csv` is ready. Now the Manager wants a visual for the board meeting. She specifically wants to see "Total Revenue by Product Category." She is picky about colors and titles — raw, unformatted charts will not be accepted.

Your mission

Create a professional-grade bar chart showing Total Revenue per Product Category for the board meeting. The chart must effectively communicate sales performance to management, with clear titles and proper currency formatting.

Expected result

A clean, professional bar chart displayed (or saved) that shows which business categories are driving the most revenue for the year.



VIZ 2

Exercise: The Report Generator (Viz 2)

Exercise: The Report Generator (Viz 2)

The scenario

The chart is great, but the Manager also wants the raw numbers in Excel to play with. She asks for *one* Excel file containing an executive summary and the full backing data — the kind of deliverable that bridges your analysis and the corporate report.

Your mission

Take the cleaned sales data and generate a single professional Excel file (`final_report.xlsx`) with two tabs:

- "Executive Summary": A high-level table showing Total Revenue by Category.
- "Backing Data": The full, detailed dataset for auditing purposes.

Use Python's `pandas` library to act as the bridge between raw data and the final corporate report.

Expected result

One file `final_report.xlsx` with two sheets: Executive Summary and Backing Data — ready to share with management.

AUTO 1

Exercise: The Professional Polish ("The Bridge")

Exercise: The Professional Polish ("The Bridge")

The scenario

You sent `final_report.xlsx` to the boss. She says: *"Can you make the headers bold and blue? And the columns are too narrow, I can't read the product names."* You don't want to do this manually every quarter.

Your mission

Write a Python script that takes the raw data (or the existing Excel file) and applies professional formatting automatically:

- **Professional Headers:** Bold text with a nice blue background.
- **Readable Columns:** All columns should auto-fit to the content size.
- **Highlights:** Any Revenue over $10,000 should be highlighted in Green.

Expected result

A new Excel file named `polished_report.xlsx` that is formatted perfectly — headers bold and blue, columns readable, high revenue highlighted in green.

AUTO 2

Exercise: The automated File Organizer

Exercise: The automated File Organizer

The scenario

Your "Downloads" folder and project workspace have become a chaotic mess of files. There are CSVs mixed with PDFs, Excel reports, and random text files. It's time to clean up before you clock out.

Your mission

Write a Python script that automatically organizes these files into neat subfolders based on their type (e.g., all `.csv` files go into a `Data` folder, `.pdf` into `Docs`).

When you run your script, the files in the `data` folder should be moved into:

- Spreadsheets/ (for `.xlsx`, `.csv`)
- Documents/ (for `.pdf`, `.txt`, `.docx`)
- Images/ (for `.png`, `.jpg`)
- Others/ (for anything else)

Expected result

Files in the workspace are sorted into the folders above — no more chaos.

BONUS

Exercise: The Big Data Stress Test

Exercise: The Big Data Stress Test

The scenario

Fast forward two years. Your company has grown, and the "Sales Data" is no longer just 3 simple files — it's gigabytes of server logs. Traditional scripts try to load everything into RAM at once and crash. CSV files are slow; text-based logs take up too much space.

Your mission

Process these simulated large datasets without crashing your computer. Move from memory-heavy tools (like Pandas) to an "Out-of-Core" processing engine that handles data larger than your RAM.

- Experience why the "old way" fails.
- Use a new tool (DuckDB) to query data *without* loading it all first.
- Convert bulky CSVs into the efficient Parquet format.

Expected result

You have run queries on large data using DuckDB and exported (or converted) data to Parquet — without running out of memory.

BONUS MENU
API Pipelines
D3.js
PDF Reports
Geo Heatmaps
Interactive Dashboards
Seaborn Stats
BONUS EXERCISES

Bonus Exercises

BONUS - Bonus: Auto PDF Report (The Executive Deliverable)

BONUS - Bonus: ETL API Pipeline (From Web to Warehouse)

BONUS - Bonus: D3 Bar Chart (The Animated Bar Race)

BONUS - Bonus: D3 Force Network (The Relationship Map)

BONUS - Bonus: Viz Geo Heatmap (The Global View)

BONUS - Bonus: Interactive Dashboard (The Wow Factor)

BONUS - Bonus: Seaborn Scatter (The Executive Scatter)

BONUS - Bonus: Seaborn Statistical (The Statistical Dashboard)

← Back to Menu | Break Time | Finish Session

BONUS

Bonus: Auto PDF Report (The Executive Deliverable)

Bonus: Auto PDF Report (The Executive Deliverable)

The scenario

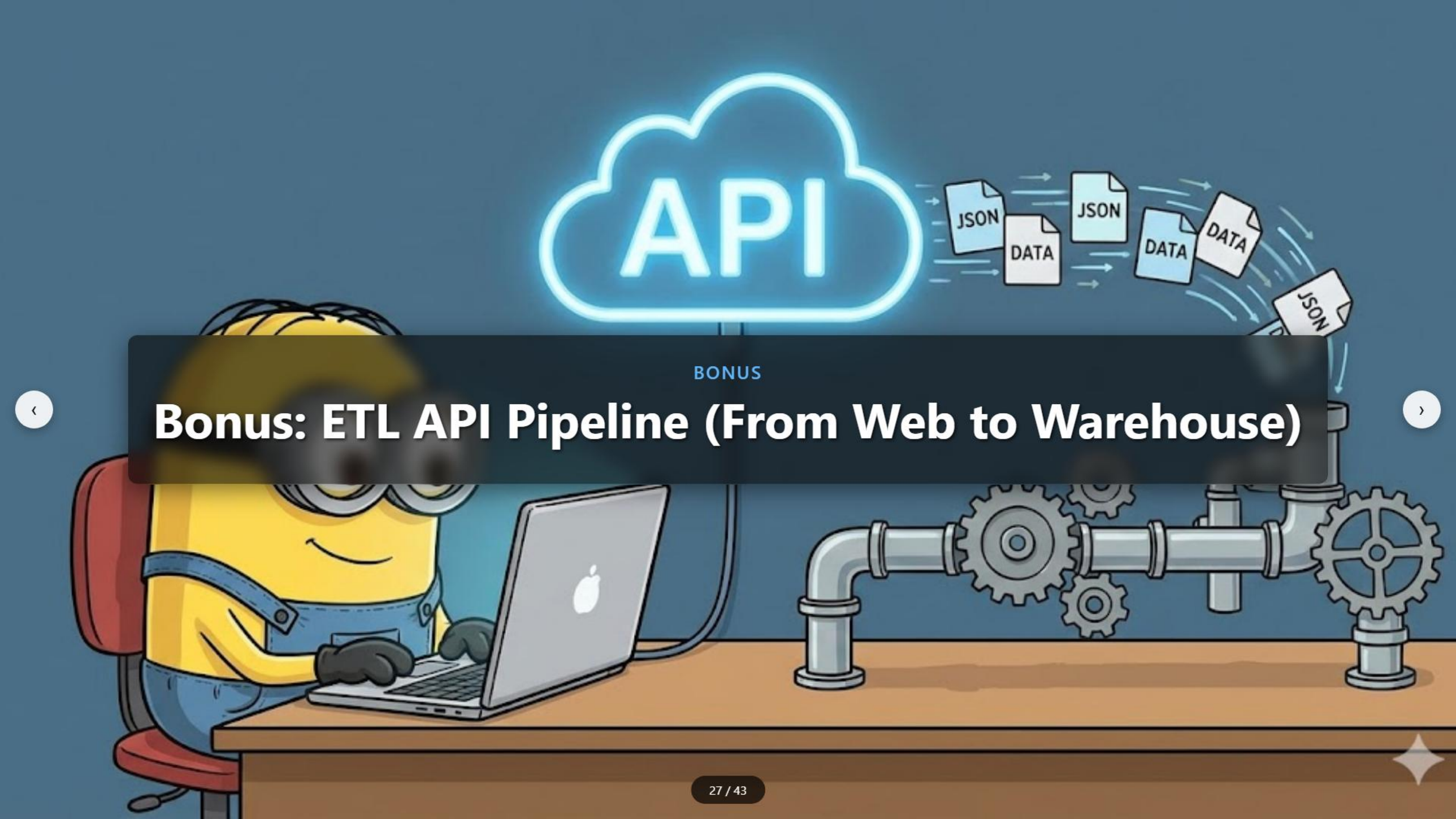
The CFO doesn't use Excel. She doesn't open HTML files. She wants a PDF — clean, branded, with charts and tables embedded — that she can print, annotate, and forward to the board. Every quarter, you've been manually screenshotting charts, pasting them into Word, and exporting to PDF. No more.

Your mission

Build a Python script that automatically generates a multi-page PDF report: a cover page with title and branding, a KPI summary section, embedded bar and pie charts (Revenue by Category), a formatted data table (top 10 transactions), and page footers with page numbers.

Expected result

A polished `quarterly_report.pdf` that looks professionally designed — but was generated entirely by Python in seconds. Run it once, and it works every quarter with fresh data.



BONUS

Bonus: ETL API Pipeline (From Web to Warehouse)

Bonus: ETL API Pipeline (From Web to Warehouse)

The scenario

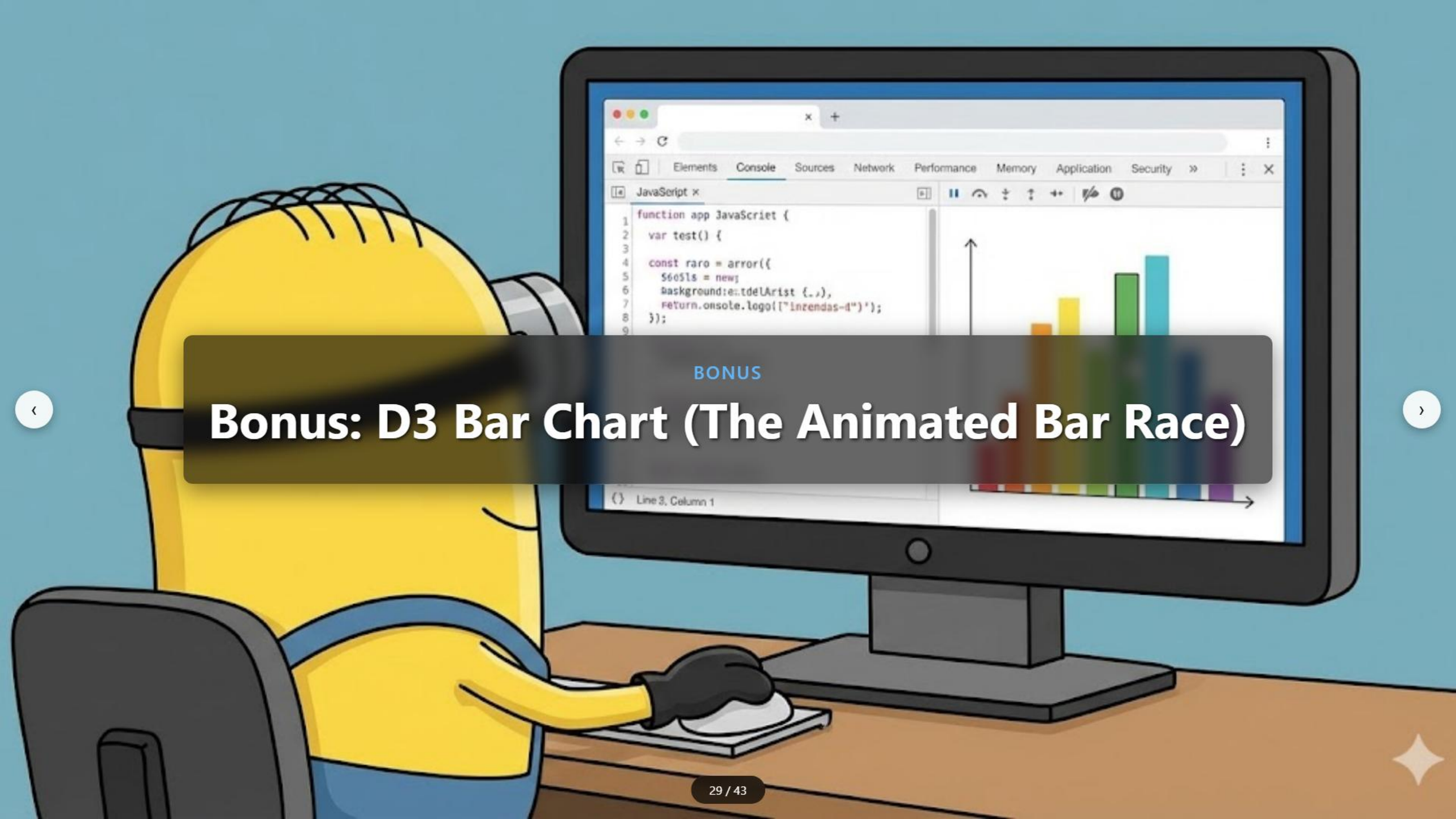
Your company has adopted a new CRM system that exposes customer and order data via a REST API (JSON format). Management wants a single, flat Excel report combining customer profiles with their order history — but the API returns deeply nested JSON with inconsistent fields.

Your mission

Build a Python ETL pipeline that: reads paginated JSON (simulating API responses), flattens nested structures into clean DataFrames, joins customer data with orders, handles missing values and type inconsistencies, and exports the consolidated dataset to CSV and Excel.

Expected result

A clean `customer_orders_consolidated.csv` with one row per order enriched with customer details, plus a `pipeline_report.xlsx` with "Orders Detail" and "Customer Summary" sheets.



BONUS

Bonus: D3 Bar Chart (The Animated Bar Race)

Bonus: D3 Bar Chart (The Animated Bar Race)

The scenario

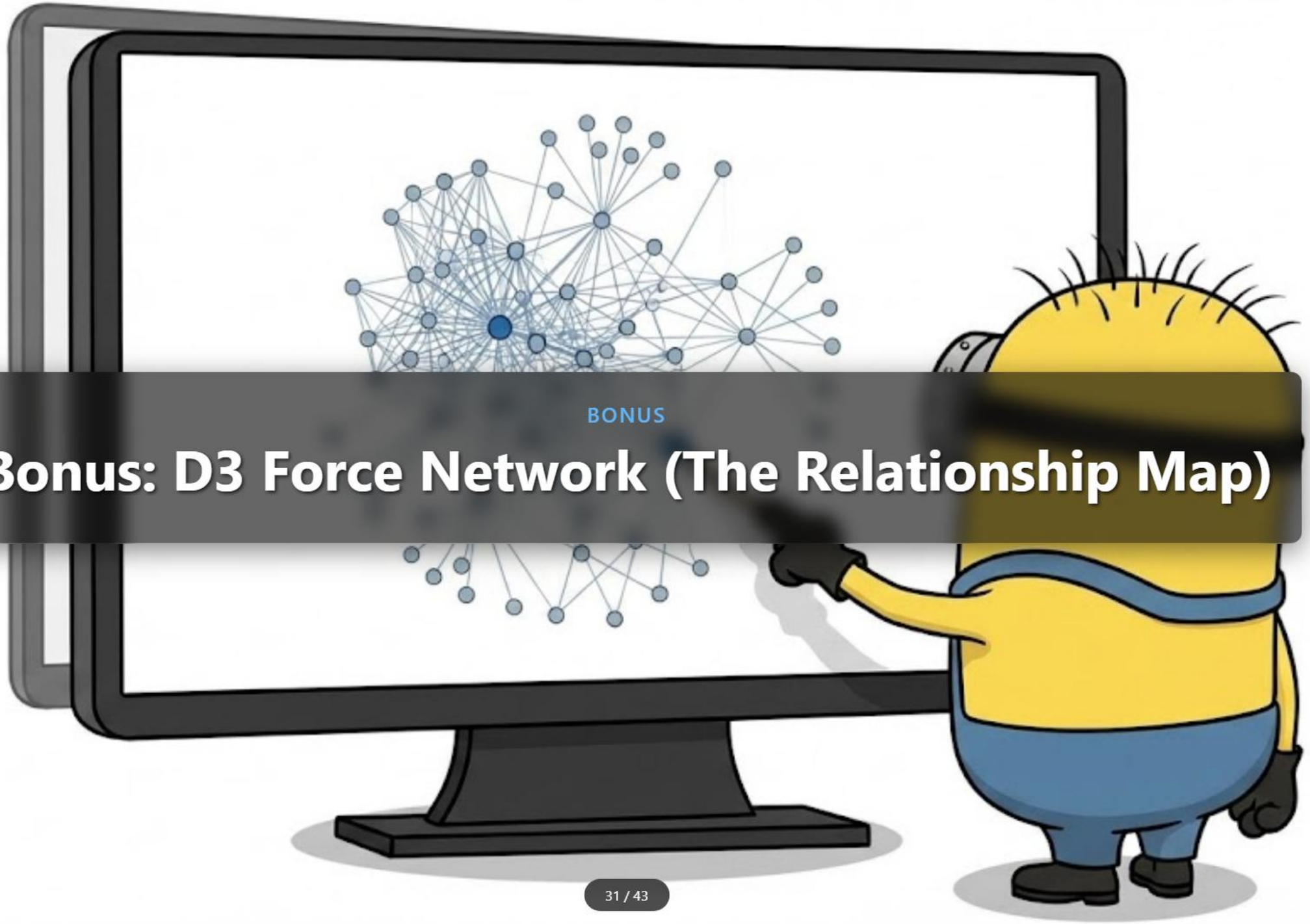
The Marketing Director saw an animated "bar chart race" on social media showing brand rankings over time and said: *"I want one of those for our sales data."* Static charts can't show how product rankings shift month-by-month. You need a chart that moves.

Your mission

Using Python to prepare the data and generate an HTML file with embedded D3.js, build an interactive, animated bar chart that shows the top 10 products by revenue at each month, animates through time (bars grow, shrink, reorder), has smooth transitions, and includes a play/pause button with a month indicator.

Expected result

A self-contained `bar_chart_race.html` file that you open in a browser and hit "Play" to watch the rankings animate through 12 months of data — built with your company's actual data.



BONUS

Bonus: D3 Force Network (The Relationship Map)

Bonus: D3 Force Network (The Relationship Map)

The scenario

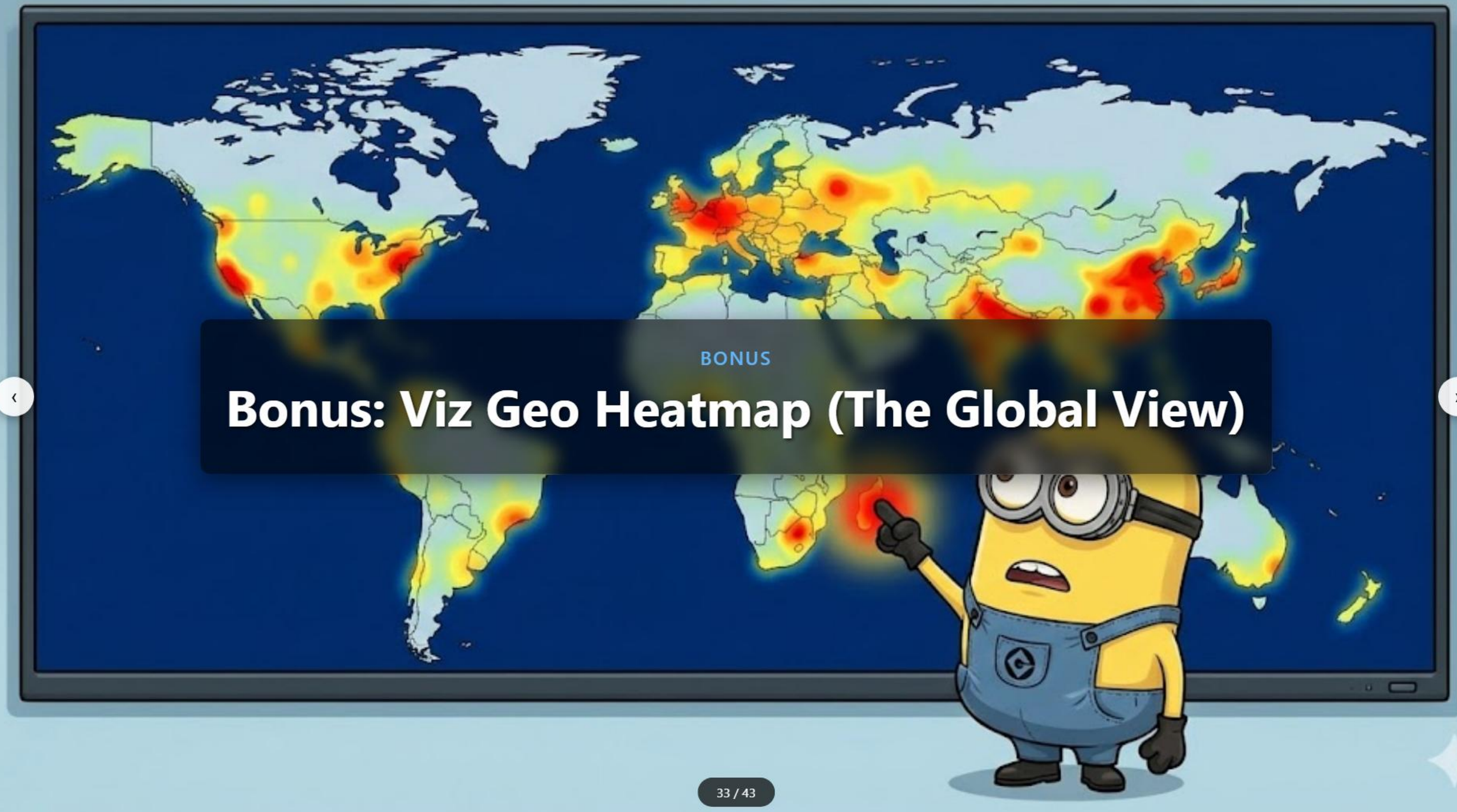
The VP of Partnerships wants to understand how departments, products, and clients are connected. Who buys what? Which departments share the same top clients? Which products are frequently purchased together? Spreadsheets can't show relationships — you need a network graph.

Your mission

Using Python to prepare the data and D3.js in the browser, build an interactive force-directed network graph: nodes for departments, products, and clients; edges for purchase relationships; node color by entity type; node size by revenue; drag to explore clusters; tooltips on hover.

Expected result

A self-contained `relationship_network.html` file that opens in a browser and shows a physics-based network that self-organizes into clusters. Drag nodes, hover for details, zoom in/out — and reveal hidden patterns like which clients are central to the business.



BONUS

Bonus: Viz Geo Heatmap (The Global View)

Bonus: Viz Geo Heatmap (The Global View)

The scenario

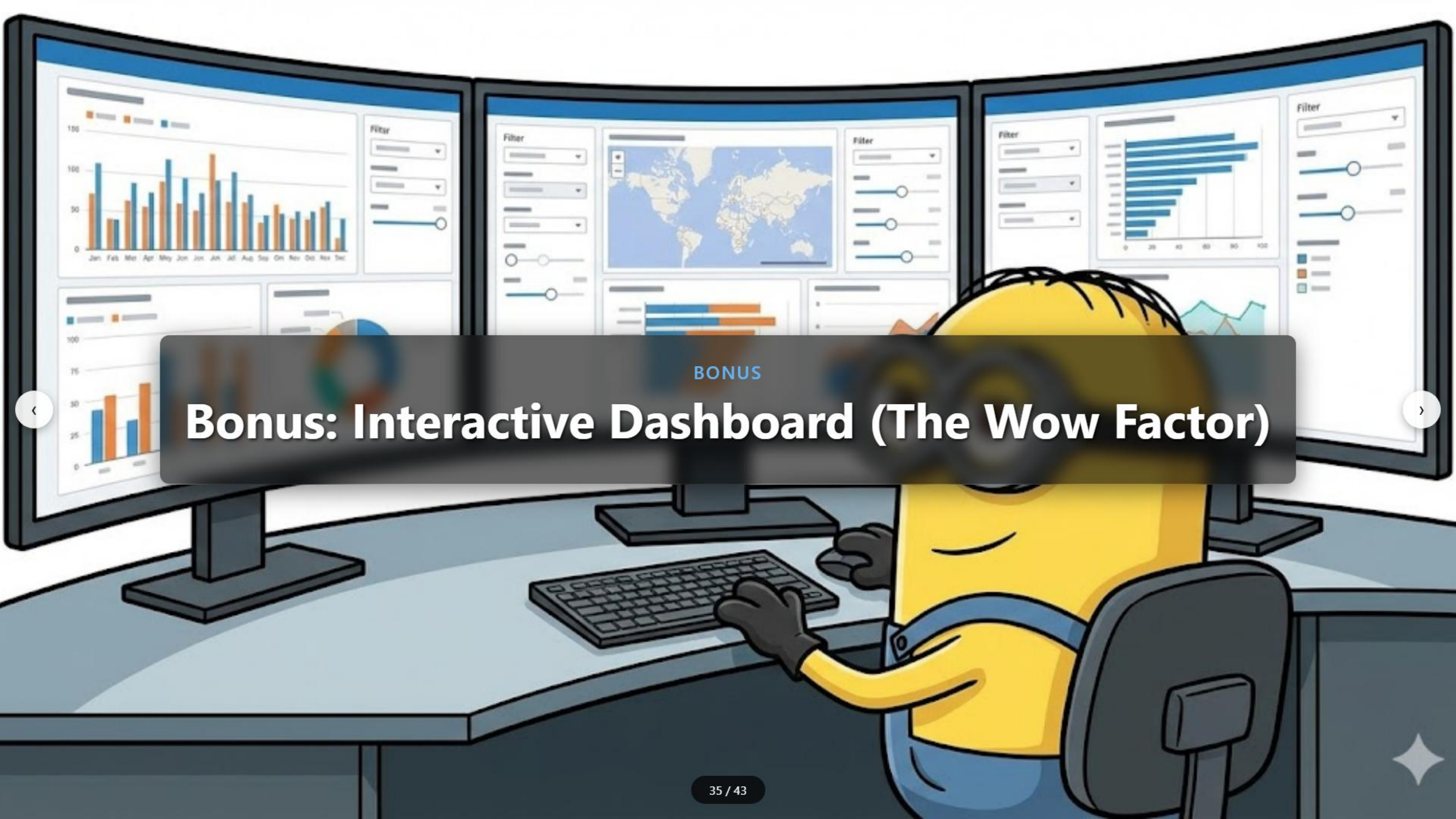
The VP of International Sales wants to see *where* the money is coming from — not in a table, but on an actual map. She wants to open a single file and instantly see hotspots of sales activity across the globe, with the ability to click on cities for details.

Your mission

Build an interactive geographic visualization using Folium (or Leaflet/Plotly): a heatmap layer showing sales intensity by city, circle markers sized by revenue with popup details, a layer control to toggle views, and a legend or title overlay.

Expected result

A self-contained `sales_heatmap.html` file that opens in any browser showing an interactive world map with sales data overlaid. Clicking on a city reveals total revenue, number of orders, and top product — the kind of deliverable that gets projected in a board room.



BONUS

Bonus: Interactive Dashboard (The Wow Factor)

Bonus: Interactive Dashboard (The Wow Factor)

The scenario

The CEO loved your static charts, but now she wants something she can *interact* with — filter by region, hover over data points, and share with the board via a simple link. Static PNGs won't cut it anymore.

Your mission

Build a stunning, interactive HTML dashboard using Plotly (or Streamlit/Dash): a KPI summary row (Total Revenue, Orders, Avg Order Value), a sunburst chart (Revenue by Region → Category → Product), a time-series line chart filterable by region, and a top 10 products bar chart with hover tooltips.

Expected result

A single self-contained `dashboard.html` file that opens in any browser — no server required. It should look polished, professional, and interactive. This is the kind of deliverable that makes people say "*You did this with Python?!*"

BONUS

Bonus: Seaborn Scatter (The Executive Scatter)

Bonus: Seaborn Scatter (The Executive Scatter)

The scenario

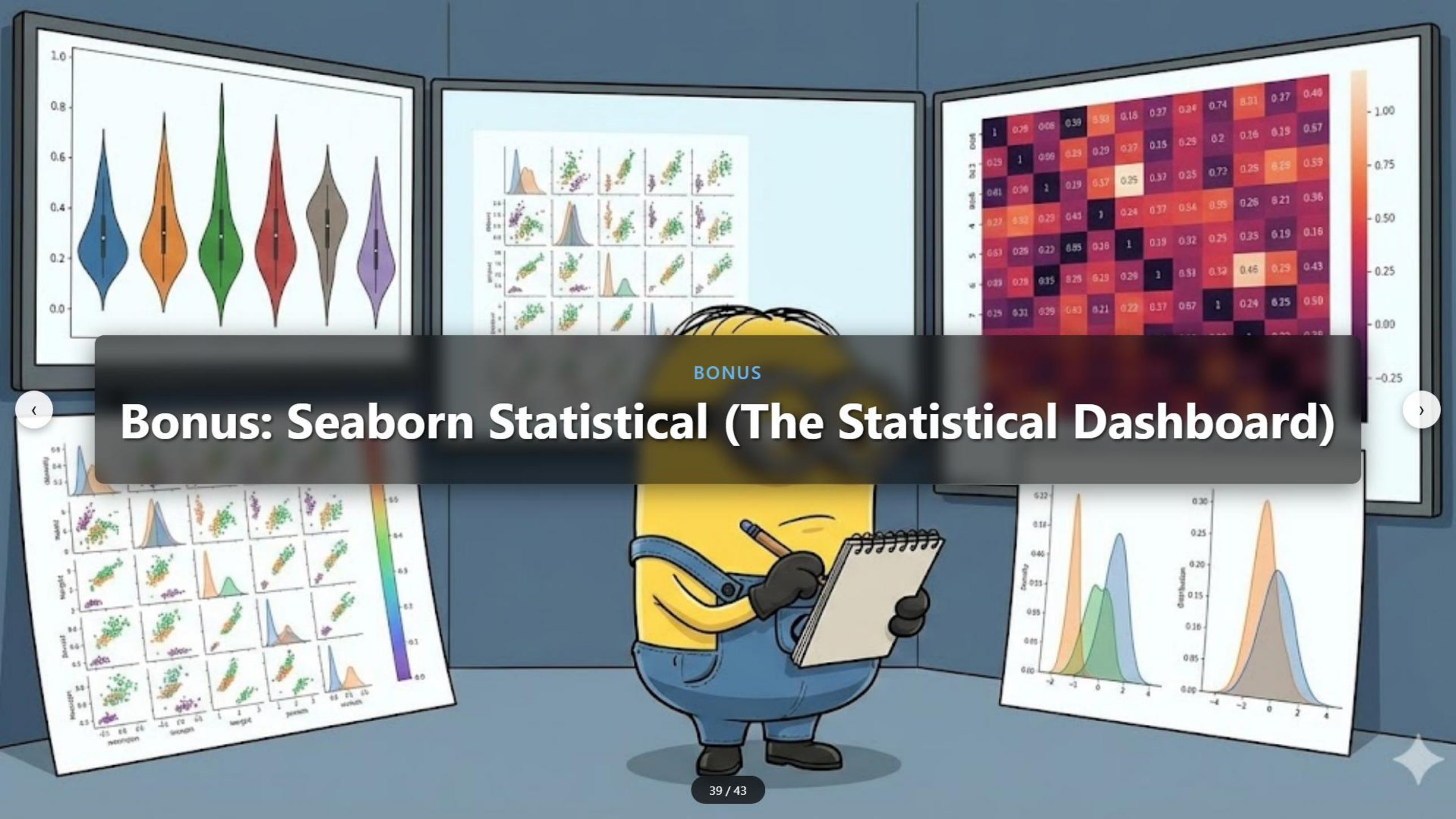
The Head of Strategy wants to see the *relationship* between deal size and quantity sold — but broken down by product category and region simultaneously. A simple bar chart won't cut it; she needs a chart where every data point tells a story: position shows deal economics, color shows category, size shows volume.

Your mission

Build a publication-quality scatter plot using Seaborn that encodes four dimensions: X-axis (Unit Price), Y-axis (Revenue), Color (Product Category), Size (Quantity sold — bigger bubbles = bigger volume). Include clear title, axis labels with currency formatting, a legend, and a professional theme. Optional: regression trend lines per category.

Expected result

A polished `executive_scatter.png` that looks like it belongs in a Harvard Business Review article — with clear encoding, legend, and no default gray.



BONUS

Bonus: Seaborn Statistical (The Statistical Dashboard)

Bonus: Seaborn Statistical (The Statistical Dashboard)

The scenario

The Data Science team is presenting at the quarterly all-hands meeting. They need a single image — a multi-panel statistical dashboard — that tells the full story of employee performance and satisfaction in one glance. Each panel should show a different statistical perspective using the best chart type for that data shape.

Your mission

Build a polished, multi-panel figure using Seaborn: a correlation heatmap (relationships between revenue, satisfaction, experience), a violin plot (satisfaction across departments), a pair plot (scatter matrix colored by department), and a box + strip combo (revenue by seniority with overlaid points).

Expected result

Publication-quality PNG images: `correlation_heatmap.png`, `satisfaction_violins.png`, `metrics_pairplot.png`, `revenue_by_seniority.png`. These are the kind of charts that make a data team look world-class.

BREAK
TIME

SUGGESTION
BOX

CONGRATULATIONS!