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

PEDAGOGICAL AGENDA

1:2 Delivery Rhythm

Session Learning Architecture

Structured 3-hour applied computing session designed for business students with zero prior coding experience.

- **Phase 1 (60 Min Lecture):** Computational foundations, business vocabulary, and syntactic mechanics.
- **Phase 2 (120 Min Lab):** Guided hands-on implementation, error diagnosis, and AI co-pilot resolution.
- **Zero Fluff Mandate:** Focus directly on executable Python code, tabular data wrangling, and commercial intelligence.

STRATEGIC OUTCOME

Operational Autonomy

Business Analytical Competency

Transforming passive spreadsheet users into autonomous programmatic decision makers.

- **Speed & Scale:** Processing datasets exceeding spreadsheet limits with instantaneous vectorized execution.
- **Repeatable Pipelines:** Replacing manual copy-paste workflows with fully automated Python scripts.
- **AI Debugging Literacy:** Leveraging modern IDEs to diagnose and resolve errors independently.

Core Computational Keywords & Business Meaning

CONCEPT 01 — Core Keyword

Visual Encoding Hierarchy

Mapping quantitative commercial data to spatial coordinates, lengths, and colors following cognitive visual perception principles.

CONCEPT 02 — Core Keyword

Figure & Axes Architecture

The object-oriented Matplotlib canvas model where a top-level `Figure` container manages one or more individual `Axes` plotting subplots.

CONCEPT 03 — Core Keyword

Time-Series Revenue Curves

Continuous line plots visualizing revenue, marketing spend, and customer acquisition costs across fiscal months.

CONCEPT 04 — Core Keyword

Categorical Distribution Plots

Clean bar plots and box plots created with Seaborn to compare performance metrics across corporate business units.

Computational Foundations & Business Operations

PILLAR 01 Architecture

**Core Programming & Computational Foundations

**

PILLAR 02 Architecture

Boardroom Visual Hygiene

Executive leadership teams make multi-million dollar decisions based on quantitative charts. Visualizations must eliminate chartjunk, 3D effects, and misleading truncated axes, prioritizing clear data ink ratios.

PILLAR 03 Architecture

Object-Oriented Matplotlib Hierarchy

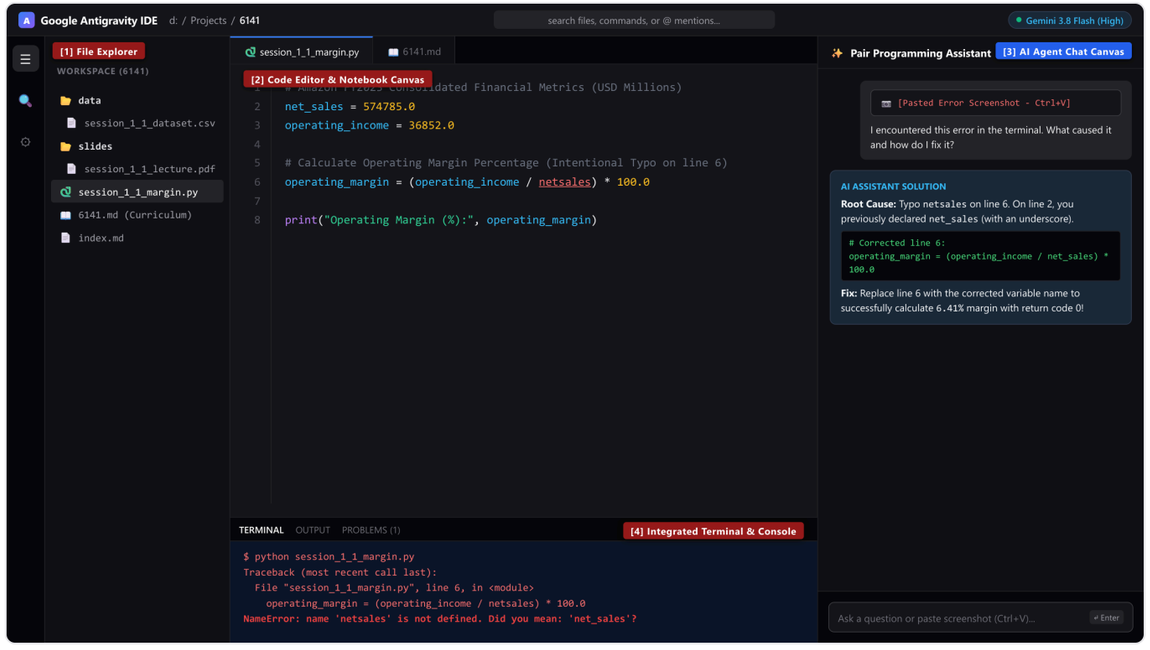
Avoid procedural plotting (`plt.plot()`) in favor of explicit object-oriented architecture (`fig, ax = plt.subplots()`). This enables precise control over tick marks, dual y-axes, titles, and export resolutions.

PILLAR 04 Architecture

Statistical Density & Outlier Detection

Using distribution visualizations (box plots, histograms) to identify revenue skews and extreme transaction outliers that distort simple arithmetic averages.

Google Antigravity IDE Architecture & AI Debugging Protocol



[1] File Explorer

Project Hierarchy

Manage datasets in `data/`, scripts in root, and exported artifacts.

[2] Code Editor

Syntax Workspace

Modern syntax highlighting, lint diagnostics, and line-by-line script authoring.

[3] AI Agent Chat

AI Pair Programmer

Paste error screenshots directly to inspect root causes and get verified fixes.

[4] Terminal

Execution Engine

Run Python scripts (`python script.py`) and view console logs and tracebacks.

Zero-Friction AI Error Resolution Protocol

1. **Capture:** Press `Win + Shift + S` to clip terminal error traceback.
2. **Paste:** Press `Ctrl + V` into AI Chat [3] with "Explain root cause & fix".
3. **Apply:** Review the explanation and apply verified fix into Editor [2].

Script Demonstration 1: Script Demonstration 1

session_2_2_demo_1.py — Python 3.10

```
"kw">class="kw">import matplotlib.pyplot "kw">class="kw">as plt
    "kw">class="kw">import pandas "kw">class="kw">as pd

    df = pd."fn">read_csv("data/session_2_2_dataset.csv")

    fig, ax1 = plt.subplots(figsize=(10, 5))
    ax1.plot(df["Month_Name"], df["Organic_Revenue_USD_M"], color="#991B1B", m
    ax1.set_ylabel("Organic Revenue (USD M)", color="#991B1B")

    ax2 = ax1.twinx()
    ax2.plot(df["Month_Name"], df["Customer_Acquisition_Cost_USD"], color="#47
    ax2.set_ylabel("Customer Acquisition Cost (USD)", color="#475569")
    plt.title("Revenue Growth vs. Acquisition Cost Efficiency")
    plt.savefig("data/dual_axis_trend.png", dpi=150, bbox_inches="tight")
```

```
$ python session_2_2_demo_1.py [Exit 0]
```

1 Syntactic Structure

Importing production libraries and defining clean analytical data structures.

2 Vectorized Transformation

Applying deterministic operations across the full dataset without fragile cell references.

3 Console Verification

Emitting formatted business KPIs to the terminal for auditability and compliance.

Script Demonstration 2: Script Demonstration 2

session_2_2_demo_2.py — Python 3.10

```
"kw">class="kw">import seaborn "kw">class="kw">as sns

    fig, ax = plt.subplots(figsize=(8, 4))
    sns.barplot(data=df, x="Month_Name", y="Paid_Marketing_Spend_USD_M", color=
    ax.set_title("Paid Marketing Budget Allocation by Month")
    ax.set_ylabel("Marketing Spend (USD M)")
    plt.savefig("data/marketing_spend_bar.png", dpi=150, bbox_inches="tight")
```

```
$ python session_2_2_demo_2.py [Exit 0]
```

1 Syntactic Structure

Importing production libraries and defining clean analytical data structures.

2 Vectorized Transformation

Applying deterministic operations across the full dataset without fragile cell references.

3 Console Verification

Emitting formatted business KPIs to the terminal for auditability and compliance.

Script Demonstration 3: Script Demonstration 3

session_2_2_demo_3.py — Python 3.10

```python
numeric_cols = ["Organic_Revenue_USD_M", "Paid_Marketing_Spend_USD_M", "Customer
    corr_matrix = df[numeric_cols].corr()

    fig, ax = plt.subplots(figsize=(6, 5))
    sns.heatmap(corr_matrix, annot=True, cmap="Reds", fmt=".2f", ax=ax)
    ax.set_title("Commercial KPI Correlation Matrix")
    plt.savefig("data/kpi_correlation_heatmap.png", dpi=150, bbox_inches="tight
```

```
$ python session_2_2_demo_3.py [Exit 0]
```

1 Syntactic Structure

Importing production libraries and defining clean analytical data structures.

2 Vectorized Transformation

Applying deterministic operations across the full dataset without fragile cell references.

3 Console Verification

Emitting formatted business KPIs to the terminal for auditability and compliance.

Interactive Two-Tier Lab Protocol & AI Diagnosis

EXERCISE 1

Guided Lab

Foundational Implementation

Step-by-step guided practice establishing baseline syntax mastery in Antigravity IDE.

- **Step 1:** Ingest dataset from `data/session_2_2_dataset.csv`.
- **Step 2:** Implement core analytical functions and compute primary business metrics.
- **Step 3:** Format console outputs and verify calculations against baseline ledger totals.
- **Step 4:** Run in Terminal [4] and confirm clean execution with exit code 0.

EXERCISE 2+

Coding Challenge

Advanced Scripting & AI Debugging

Applied programming challenge expanding pipeline logic and resolving intentional exceptions.

- **Task 1 (Pipeline Expansion):** Add secondary business unit logic and multi-tier calculations.
- **Task 2 (Deliberate Error & AI Capture):** Introduce intentional syntax or type error, capture traceback with `Win + Shift + S`, and diagnose via AI Chat [3].
- **Task 3 (Verified Production Run):** Apply verified fix, export audit output, and confirm zero errors with return code 0.

Interactive Lab Protocol & 1-2 Delivery Rhythm

Hands-On Code Execution

The Pedagogical Contract

Computational mastery develops through active problem-solving. By creating intentional errors and using AI co-pilots to resolve them, students build lifelong debugging autonomy.