Session 4.3: AI Applications in Business Decision-Making Part 2: Pricing Optimization and Revenue Management

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

Price Elasticity of Demand (PED)

The percentage change in quantity demanded resulting from a 1% change in product price, quantifying consumer price sensitivity.

CONCEPT 02

Core Keyword

Revenue Management Optimization

Programmatically identifying the price point that maximizes total gross profit (`(Price - Unit_Cost) * Quantity`) subject to market demand curves.

CONCEPT 03

Core Keyword

Gross Margin Simulation

Simulating projected sales volume and financial profit across discrete pricing scenarios (+5%, +10%, -10%).

CONCEPT 04

Core Keyword

Competitor Price Gap Sensitivity

Modeling how the premium or discount relative to competitor catalog prices influences customer purchase conversion.

Computational Foundations & Business Operations

PILLAR 01 Architecture

**Core Programming & Computational Foundations

**

PILLAR 02 Architecture

Microeconomic Demand Curves in Empirical Machine Learning

Translating economic theory into executable code. Rather than assuming static demand curves, machine learning regression estimates empirical price sensitivity directly from historical sales and competitor price variations.

PILLAR 03 Architecture

Revenue Optimization vs. Margin Maximization

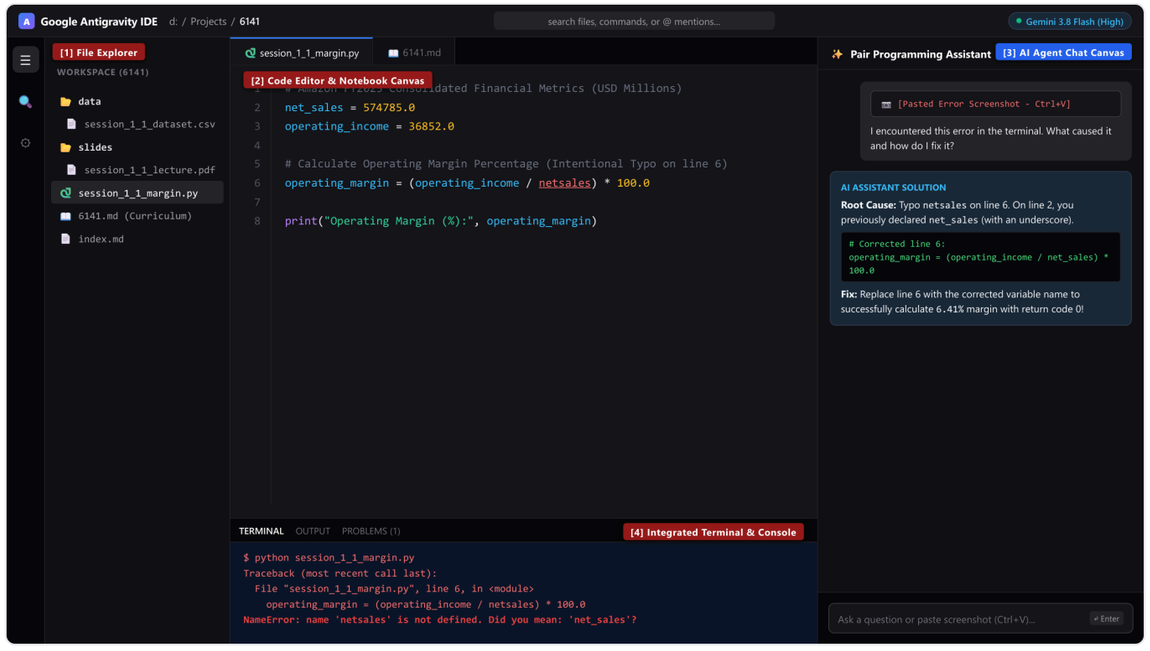
Maximizing total revenue does not always maximize operating profit. Lowering prices may increase unit sales while eroding gross margins. Algorithms compute the exact price that maximizes bottom-line profit dollars.

PILLAR 04 Architecture

Dynamic Response to Competitor Shifts

Consumer e-commerce purchase decisions depend heavily on relative price differences. Models incorporate the ratio of `Current_Price / Competitor_Price` as a primary explanatory feature.

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_4_3_demo_1.py — Python 3.10

```
"kw">class="kw">import pandas "kw">class="kw">as pd
    "kw">class="kw">import numpy "kw">class="kw">as np

    df = pd."fn">read_csv("data/session_4_3_dataset.csv")

    # Percentage premium relative to competitor
    df["Price_Premium_Pct"] = ((df["Current_Price_USD"] - df["Competitor_Price_
    # Empirical Price Elasticity: % Change "kw">class="kw">in Sales / % Change
    base_price = df["Current_Price_USD"]."fn">mean()
    base_sales = df["Historical_Daily_Sales_Units"]."fn">mean()

    # Simulated elasticity: For every 10% price premium, daily sales drop by 1
    elasticity_coefficient = -1.5
```

```
$ python session_4_3_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_4_3_demo_2.py — Python 3.10

```
"kw">class="kw">def simulate_pricing_scenario(data, price_adjustment_pct):
        simulated = data.copy()
        simulated["Simulated_Price_USD"] = simulated["Current_Price_USD"] * (1
        # Quantity adjusted by elasticity
        simulated["Simulated_Daily_Units"] = simulated["Historical_Daily_Sales_
        simulated["Simulated_Gross_Profit_USD"] = (simulated["Simulated_Price_U
        "kw">class="kw">return simulated["Simulated_Gross_Profit_USD"]."fn">sum

    current_profit = simulate_pricing_scenario(df, 0.0)
    plus_5_profit = simulate_pricing_scenario(df, 0.05)
    minus_5_profit = simulate_pricing_scenario(df, -0.05)

    "fn">print(f"Current Daily Gross Profit: ${current_profit:,.2f}")
    "fn">print(f"Daily Gross Profit at +5% Price: ${plus_5_profit:,.2f}")
```

```
$ python session_4_3_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_4_3_demo_3.py — Python 3.10

```python
# Days of inventory remaining at current sales velocity
    df["Days_Of_Inventory_Remaining"] = df["Inventory_Units_Count"] / df["Histo
    df["Markdown_Recommended"] = df["Days_Of_Inventory_Remaining"].apply(lambda
    "fn">print(df[["Product_ID", "Inventory_Units_Count", "Days_Of_Inventory_Re
```

```
$ python session_4_3_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_4_3_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 & 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.