Session 2.3: Midterm Applied Analytics Formulation Part 1: Problem Structuring and Data Audit

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

Analytical Hypothesis Framing

Translating ambiguous managerial problems into falsifiable, testable quantitative statements expressed in executable code.

CONCEPT 02 — Core Keyword

Exploratory Data Analysis (EDA)

Systematic statistical inspection of dataset distributions, correlations, and anomalies to inform analytical design.

CONCEPT 03 — Core Keyword

Data Hygiene Audit

Automated scanning for missing records, duplicate primary keys, unexpected negative numbers, and data type inconsistencies.

CONCEPT 04 — Core Keyword

Baseline Metric Architecture

Calculating baseline benchmarks (e.g., historical contract renewal rate) against which analytical improvements are measured.

Computational Foundations & Business Operations

PILLAR 01 Architecture

**Core Programming & Computational Foundations

**

PILLAR 02 Architecture

Translating Ambiguous Directives into Code

Senior executives often state broad challenges (e.g., "our enterprise client churn is increasing"). Data analysts must translate this into concrete quantitative questions: *"Is client churn correlated with Service Level Agreement (SLA) breaches or billing discrepancies?"*

PILLAR 03 Architecture

Data Cleanliness Audit Protocols

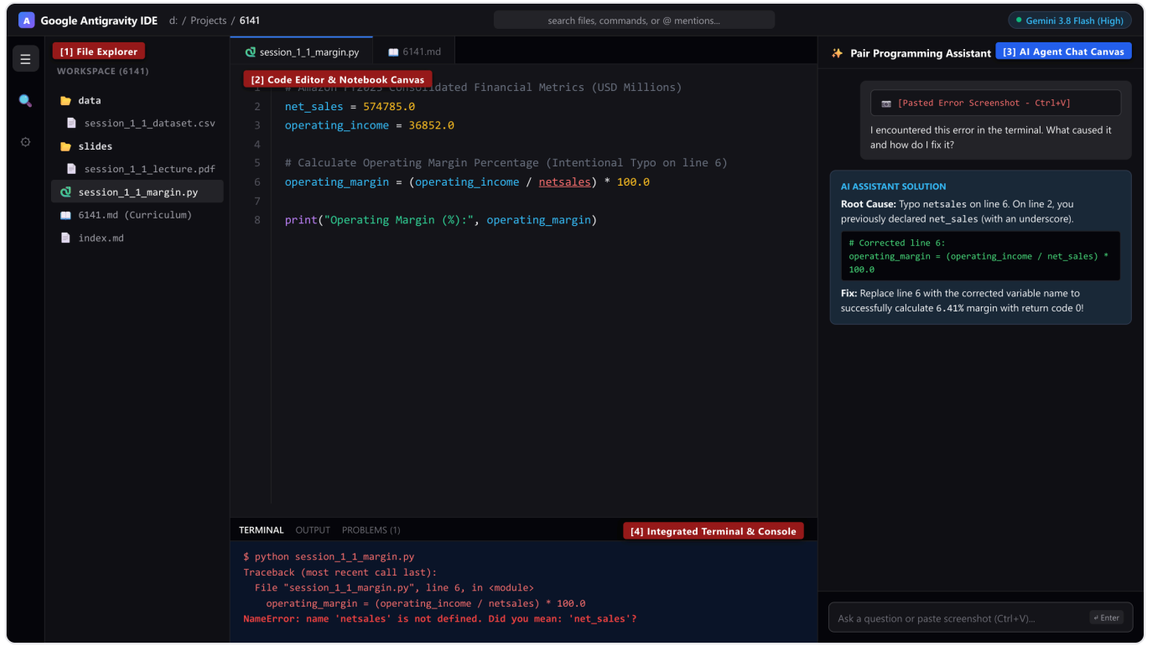
In enterprise analytics, raw corporate data is frequently incomplete or dirty. Before building machine learning models, analysts must run programmatic audit checks to verify completeness, uniqueness, and validity.

PILLAR 04 Architecture

Establishing Audit Baselines

Before proposing automated optimization, analysts must measure historical performance baselines. Without a baseline, proving return on investment is impossible.

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_3_demo_1.py — Python 3.10

```python
import pandas as pd

df = pd.read_csv("data/session_2_3_dataset.csv")

def audit_dataset_hygiene(data):
    return {
        "Total_Rows": len(data),
        "Missing_Values_Total": int(data.isnull().sum().sum()),
        "Duplicate_Accounts": int(data["Account_ID"].duplicated().sum
        "Min_SLA_Compliance": float(data["SLA_Compliance_Pct"].min()),
        "Total_Contract_Value_USD": float(data["Contract_Value_USD"].s
    }

audit_report = audit_dataset_hygiene(df)
```

```
$ python session_2_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_2_3_demo_2.py — Python 3.10

```
1  sla_breaches = df[df["SLA_Compliance_Pct"] < 95.0]
2      "fn">print(f"Accounts Breaching SLA (<95%): {">len(sla_breaches)}")
3      "fn">print(sla_breaches[["Account_ID", "Enterprise_Client", "SLA_Compliance_
```

```
$ python session_2_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_2_3_demo_3.py — Python 3.10

```
risk_summary = df."fn">groupby("Client_Retention_Risk")."fn">agg(
        Total_Contract_Value_USD=("Contract_Value_USD", "sum"),
        Account_Count=("Account_ID", "count"),
        Mean_Billing_Discrepancies=("Billing_Discrepancy_Count", "mean")
    )
    "fn">print("Client Retention Financial Exposure Summary:\\n", risk_summary)
```

```
$ python session_2_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_2_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 & 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.