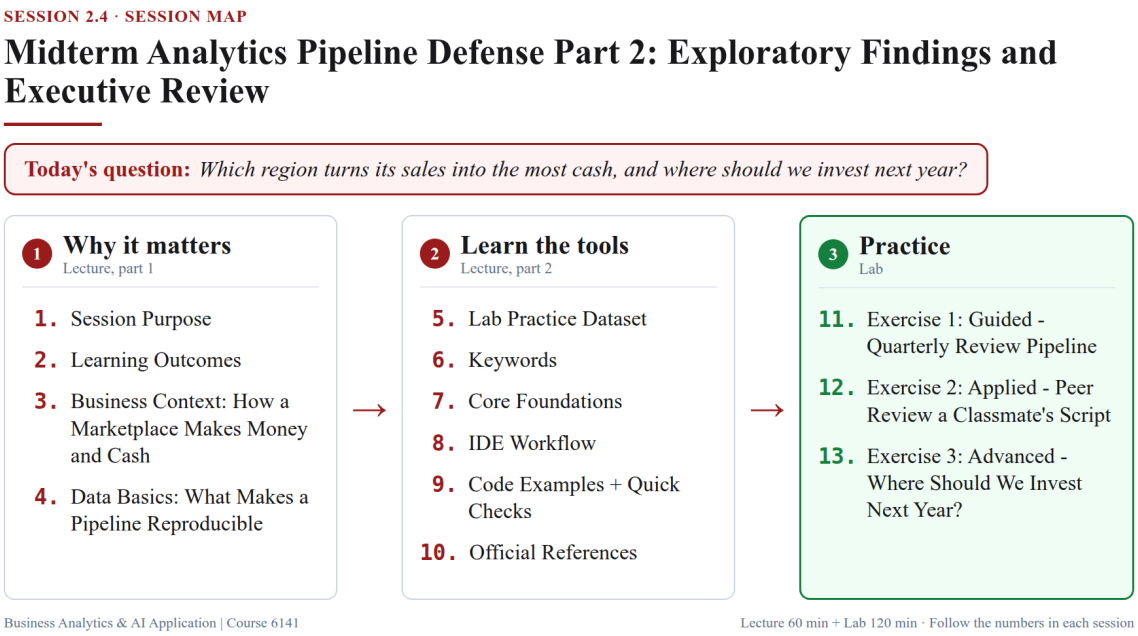
Session 2.4: Midterm Analytics Pipeline Defense Part 2: Exploratory Findings and Executive Review

Course 6141: Business Analytics & AI Application | Reading for Week 8



Schedule: Week 8 | In-person classes (1:2 Delivery Ratio: 60 Min Lecture + 120 Min Lab)

1. Session Purpose

- An online marketplace earns a fee on every sale its sellers make, in three regions: North America, Europe, and Asia Pacific
- Big sales are not the same as cash in the bank: the region that sells the most may not be the one that generates the most cash
- The CFO (Chief Financial Officer) asks: which region turns its sales into the most cash, and where should we invest next year? Today, you answer with one script that anyone can rerun, and present your midterm findings in the same way

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

1. "Can someone else get the same numbers?": build a reproducible pipeline (one script that goes from the raw file to the final report with no manual steps), organized into functions

2. “Which region turns sales into the most cash?”: calculate revenue and the operating cash flow margin by region and quarter
3. “How do I explain it to the CFO?”: write findings as finding, evidence, and recommendation; review a classmate’s script; and fix a `TypeError` (mixing text and numbers)

3. Business Context: How a Marketplace Makes Money and Cash

- Four terms, one example (North America, 2024-Q1):

Term	Meaning	Example
GMV (Gross Merchandise Value)	the total value of everything sold on the platform	`145.2` USD M
Take rate	the share of GMV the platform keeps as its fee	`12.5`%
Revenue	GMV x take rate / 100: the platform’s own income	`18.15` USD M
OCF (Operating Cash Flow)	the cash the business actually generated after paying its operating costs	`5.6` USD M

- OCF margin = OCF / revenue x 100. It answers “of every dollar of revenue, how many cents became cash?” Example: `5.6 / 18.15 x 100 = 30.9`%.
- **Midterm Project, Step 2 of 2: present your findings.** In a short presentation, walk through:
    1. Your Question Card from Session 2.3 (worry, question, hypothesis, data, baseline)
    2. Your audit result: what you checked and what you fixed
    3. Your findings, each written in three parts:
        - **Finding:** one sentence a manager can repeat. “North America turns the most revenue into cash.”
        - **Evidence:** the number behind it. “Its OCF margin is 31.0%, against 24.7% in Europe and 18.5% in Asia Pacific.”
        - **Recommendation:** what to do. “Study North America’s cost structure before expanding in Asia Pacific.”
    4. A live run of your script, from the raw file to the final report

4. Data Basics: What Makes a Pipeline Reproducible

- A reproducible pipeline runs from top to bottom and produces the same numbers on any computer. It has four stages:

Stage	What it does	Example in today's script
Load	Read the raw file	`pd.read_csv("data/session_02_04_dataset.csv")`
Check	Stop if the data breaks a rule	`assert (df["Gross_Merchandise_Value_USD_M"] > 0).all()`
Calculate	Add the measures	revenue and OCF margin
Report	Print and save the result	`summary.to_csv("data/region_summary.csv")`

- Three habits that make a script reproducible:
  - Use relative paths such as `data/...`, never paths that only exist on your computer, such as `C:/Users/Reacher/Desktop/...`
  - Calculate every number from the data; never type a result into the code by hand
  - Before presenting, delete your output files and run the script again from the start

5. Lab Practice Dataset

- File Link: `data/session_02_04_dataset.csv`
- Audited 12-row quarterly marketplace dataset (fictional teaching data): 3 regions x 4 quarters, one row per region and quarter:
  - `Quarter_Code`: Reporting quarter, `2024-Q1` through `2024-Q4`.
  - `Market_Segment`: Region: `North_America`, `Europe`, or `Asia_Pacific`.
  - `Gross_Merchandise_Value_USD_M`: GMV (Gross Merchandise Value) in millions of U.S. Dollars.
  - `Platform_Take_Rate_Pct`: The platform's fee as a percentage of GMV.
  - `Operating_Cash_Flow_USD_M`: OCF (Operating Cash Flow) in millions of U.S. Dollars.

6. Keywords (Core Programming Concepts)

- `Reproducible Pipeline`:
  - What: one script that goes from the raw file to the final report, with no manual steps
  - Why: anyone can rerun it and check your numbers
- `Function` (`def` / `return`):
  - What: a named block of code that takes inputs and gives back a result
  - Why: each pipeline stage gets a clear name, and you can reuse it
- `Relative Path`:
  - What: a file path that starts from the project folder, such as `data/session_02_04_dataset.csv`
  - Why: works on every computer that has the project folder
- `Data Type` and `TypeError`:

  - What: text (`str`) and numbers (`float`, `int`) are different types; `"Total: " + 204.3` raises a `TypeError`
  - Why: printing a result is where this mistake happens most

- `Executive Finding`:
  - What: a finding, its evidence, and a recommendation
  - Why: managers act on clear statements backed by numbers, not on code

- `Peer Review`:
  - What: a classmate checks your script against a short checklist
  - Why: a second reader catches mistakes the author no longer sees

7. Core Programming & Computational Foundations

- **7.1 Functions: name a step, then reuse it**
  - *The idea:* A function is a named block of code. You give it inputs (parameters), it does its work, and `return` hands back the result. Once a step is a function, you can run it again with one line, on any data, and a reader can understand the pipeline from the function names alone.

    ```
    def revenue_usd_m(gmv, take_rate_pct):
        return gmv * take_rate_pct / 100
    ```

    `def` starts the function, the name follows, then the parameters in brackets and a colon. The indented lines are the body.
  - *See it on real data:*

Call	Result	What happened
`revenue_usd_m(145.2, 12.5)`	`18.15`	North America, Q1: `145.2 x 12.5 / 100`
`revenue_usd_m(160.8, 13.0)`	`20.9` (rounded)	North America, Q4: same function, new inputs
`revenue_usd_m(df["Gross_Merchandise_Value_USD_M"], df["Platform_Take_Rate_Pct"])`	12 values	the same function works on whole columns

  - *Common mistake 1, forgetting `return`:* a function without `return` gives back `None`, Python's "nothing". The calculation happens, but the result is thrown away.
  - *Common mistake 2, using an inside name outside:* a variable created inside a function exists only there. After `def f(x): y = x * 2`, the line `print(y)` outside the function raises `NameError: name 'y' is not defined`. Use `return y` and store the result: `y = f(3)`.
  - *More examples:*

Function	Call	Result
`def ocf_margin(ocf, revenue): return round( ocf / revenue * 100, 1)`	`ocf_margin(5.6, 18.15)`	`30.9`
`def load_data(path): return pd.read_csv(path)`	`load_data("data/session_02_04_dataset.csv")`	the 12-row DataFrame
`def is_weak(margin, limit=20): return margin < limit`	`is_weak(18.6)`	`True`; `limit` uses its default value, `20`
the same	`is_weak(18.6, limit=15)`	`False`; a different limit passed in

  - *Remember:* `def` to define, `return` to hand back, and store the result when you call it.

- **7.2 A reproducible pipeline: load, check, calculate, report**
  - *The idea:* A pipeline is one script that goes from the raw file to the final report with no manual step in between. "Reproducible" means anyone can run it again, on any computer, and get exactly the same numbers. That is what lets a manager trust your midterm results.

Stage	Function in Micro-Example 2	Why it is separate
Load and check	`load_data(path)`	bad data stops here, before any number is calculated
Calculate	`add_metrics(df)`	one place for every formula
Summarize	`summarize(df)`	the table the manager reads
Report	`print(summary)` and `.to_csv()`	the result on screen and in a file

  - *Three habits:* use relative paths such as `data/...` (a path like `C:/Users/Reacher/Desktop/...` only exists on one computer); calculate every number from the data, never type a result by hand; before you present, delete the output files and run the script again from the top.
  - *Common mistake:* running cells or lines out of order while developing, so the script works on your screen but fails when run from the start. The test of a pipeline is a clean run, top to bottom.
  - *More examples:*

Path or habit	Works on	Verdict
`"data/session_02_04_dataset.csv"`	every computer with the project folder	relative path: use this
`"C:/Users/Reacher/Desktop/data.csv"`	only Reacher's computer	absolute path: breaks for everyone else
a total typed into the code as `204.3`	only until the data changes	hard-coded result: calculate it instead
output files left from an earlier run	hides missing steps	delete outputs, then rerun from the top

- *Remember:* one script, top to bottom, same numbers every time.

- **7.3 Text and numbers do not mix: `TypeError`**
  - *The idea:* Python keeps text (`str`) and numbers (`int`, `float`) apart. `+` adds two numbers or joins two texts, but it refuses to combine a text with a number, because it cannot know whether you want to add or to join.
  - *See it on real data:*

Code	Result
`"Total: " + 204.3`	`TypeError: can only concatenate str ( not "float") to str`
`"Rows: " + len(df)`	`TypeError: can only concatenate str ( not "int") to str`
`"Total: " + str(204.3)`	`Total: 204.3`
`f"Total: {204.3}"`	`Total: 204.3`

    "Concatenate" means "join texts together".
  - *Formatting numbers inside f-strings:* after the value, a colon and a format code control how the number looks.

Code	Result	Format
`f"{204.3049:.2f}"`	`204.30`	2 decimals
`f"{1234567.8:,.0f}"`	`1,234,568`	thousands separators, no decimals
`f"{0.2467:.1%}"`	`24.7%`	a share shown as a percentage

  - *More examples:*

Code	Result	What it tells you
`str(12) + " stores"`	`12 stores`	convert the number to text first
`f"{12} rows"`	`12 rows`	an f-string converts for you
`"Q" + str(4)`	`Q4`	build a label from text and a number
`int("12") + 1`	`13`	convert text to a number before adding
`float("18.15") * 2`	`36.3`	decimal text becomes a `float`
`"5" * 3`	`555`	a surprise: text times a number repeats the text

  - *Remember:* put numbers into text with an f-string, and format them for the reader.

- **7.4 Hard-coded numbers: believable but wrong**
  - *The idea:* A hard-coded number is a value typed directly into the code, such as `0.125` for the take rate. It is risky because it looks reasonable and produces believable results, even when the real values differ from row to row.

- *See it on real data:* revenue for the year, by region, with the real take rate of each row vs a fixed `12.5%`.

Region	With each row's take rate	With a fixed `0.125`	Error
North America	`65.57`	`67.16`	`+1.59`
Europe	`68.34`	`69.28`	`+0.94`
Asia Pacific	`70.39`	`70.49`	`+0.10`

Our take rates range from `10.5%` to `13.5%`, so no single number is right. Nothing in the output warns you: the error is easy to miss.

- *More examples:*

What you need	From the data	The hard-coded risk
total revenue for the year	`204.31`	`206.92` with a fixed `0.125`
number of quarters	`len(df)` reads `12`	typing `12` breaks when a 13th row arrives
list of regions	`df["Market_Segment"].unique()`	a typed list misses a new region
a business rule	`MARGIN_LIMIT = 20` at the top of the script	a named rule is visible and easy to change

- *Remember:* read every input from the data; type a number into code only when it is a rule, and give it a name, such as `MARGIN_LIMIT = 20`.

- **7.5 Writing findings for managers**
  - *The idea:* Managers act on clear statements, not on code. Write each finding in three parts: the **finding** (one sentence a manager can repeat), the **evidence** (the number behind it), and the **recommendation** (what to do).
  - *See it on real data:*

Part	Weak version	Strong version
Finding	"Margins vary by region."	"North America turns the most revenue into cash."
Evidence	"See the table."	"Its OCF margin is 31.0%, against 24.7% in Europe and 18.5% in Asia Pacific."
Recommendation	"Look into it."	"Study North America's cost structure before expanding in Asia Pacific."

  - *Common mistake:* averaging percentages. The 4 quarterly margins of Asia Pacific average `18.1%`, but its true margin for the year, total OCF / total revenue, is `18.5%`. Always compute a group's ratio from its totals.

– *More examples:*

Finding	Evidence	Recommendation
Europe has a mid-level cash margin.	`24.7`%, between North America and Asia Pacific	Keep Europe's investment level unchanged.
Asia Pacific is improving fast.	its margin rose from `15.0`% in Q1 to `22.0`% in Q4: `+7.0` points	Keep investing, with a margin target for next year.
Q4 was the strongest quarter.	GMV `524.4` and OCF `17.9`, both the year's highest	Plan capacity for next year's Q4 peak.

– *Remember:* finding, evidence, recommendation, and every number from the data.

8. Development Workflow in Google Antigravity IDE

- `[1]` File Explorer: create your script with your name and the session code (for example, `Reacher_S24_demo_1.py`), and check the dataset is in `data/`.
- `[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.

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_S24_demo_1.py` (pattern: `YourName_S24_demo_1.py`).
- *Micro-Example 1 (From GMV to Cash, Row by Row):* `YourName_S24_demo_1.py`

```python
# 1. Input: load the quarterly results
import pandas as pd
df = pd.read_csv("data/session_02_04_dataset.csv")

# 2. Process: revenue, then OCF margin, for every row
df["Revenue_USD_M"] = (
    df["Gross_Merchandise_Value_USD_M"] * df["Platform_Take_Rate_Pct"] / 100
).round(2)
df["OCF_Margin_Pct"] = (
    df["Operating_Cash_Flow_USD_M"] / df["Revenue_USD_M"] * 100
).round(1)

# 3. Output: the 3 rows with the lowest margin
lowest = df.sort_values("OCF_Margin_Pct").head(3)
print(lowest[["Quarter_Code", "Market_Segment", "OCF_Margin_Pct"]])
```

  - *How it works, line by line:*
    - The first calculation multiplies each row's GMV by that row's take rate and divides by 100: revenue in USD millions. No take rate is typed by hand (section 7.4).
    - The second divides each row's OCF by its revenue: the OCF margin, the share of revenue that became cash. `.round(1)` keeps 1 decimal.
    - `df.sort_values("OCF_Margin_Pct")` orders the rows from the lowest margin to the highest, and `.head(3)` keeps the 3 lowest.
  - *Read the output:*

```
  Quarter_Code Market_Segment  OCF_Margin_Pct
8      2024-Q1   Asia_Pacific            15.0
5      2024-Q2   Asia_Pacific            16.8
2      2024-Q3   Asia_Pacific            18.6
```

    - The 3 weakest quarters all belong to Asia Pacific: in Q1, only `15.0` cents of every revenue dollar became cash.
    - The margins rise from Q1 to Q3: Asia Pacific is weak but improving. Quick Check 1 shows Q4 too.

- *Quick Check 1 (`.pivot_table()`):*
  - New tool: `df.pivot_table(index="A", columns="B", values="C")` makes a grid with one row per value of `A` and one column per value of `B`, like an Excel pivot table. Why: shows all 12 margins in one readable grid.
  - Task: add one line to part **3. Output**: `print(df._____(index="Market_Segment", columns="Quarter_Code", values="OCF_Margin_Pct"))`
  - Check: the `Asia_Pacific` row reads `15.0`, `16.8`, `18.6`, `22.0`: its margin improved every quarter.

- *Micro-Example 2 (The Same Work as a Pipeline of Functions):* `YourName_S24_demo_2.py`

```python
# 1. Input: a function that loads and checks the raw file
import pandas as pd

def load_data(path):
    df = pd.read_csv(path)
    assert (df["Gross_Merchandise_Value_USD_M"] > 0).all(), "GMV must be above 0"
    return df

# 2. Process: functions that add measures and summarize by region
def add_metrics(df):
    df["Revenue_USD_M"] = (
        df["Gross_Merchandise_Value_USD_M"] * df["Platform_Take_Rate_Pct"] / 100
    ).round(2)
    return df

def summarize(df):
    summary = df.groupby("Market_Segment").agg(
        Revenue_USD_M=("Revenue_USD_M", "sum"),
```

```
        OCF_USD_M=("Operating_Cash_Flow_USD_M", "sum"),
    )
    summary["OCF_Margin_Pct"] = (
        summary["OCF_USD_M"] / summary["Revenue_USD_M"] * 100
    ).round(1)
    return summary

# 3. Output: run the stages in order, then print and save
df = load_data("data/session_02_04_dataset.csv")
df = add_metrics(df)
summary = summarize(df)
print(summary)
summary.to_csv("data/region_summary.csv")
```

  - *How it works, line by line:*
    - `load_data(path)` reads the file and checks with `assert` that every GMV is above 0; bad data stops the pipeline here (section 7.2).
    - `add_metrics(df)` adds the revenue column and returns the table.
    - `summarize(df)` groups by region, sums revenue and OCF, and computes each region's margin **from its totals** (section 7.5).
    - The last 5 lines run the stages in order. Read them alone and you know what the whole script does: load, add metrics, summarize, print, save.
    - `summary.to_csv("data/region_summary.csv")` saves the result so the numbers can be checked later.
  - *Read the output:*

```
                Revenue_USD_M  OCF_USD_M  OCF_Margin_Pct
Market_Segment
Asia_Pacific            70.39       13.0            18.5
Europe                  68.34       16.9            24.7
North_America           65.57       20.3            31.0
```

    - Asia Pacific earns the most revenue (`70.39`) but keeps the least as cash (`18.5`%).
    - North America earns the least revenue (`65.57`) but generates the most cash (`20.3`), with a margin of `31.0`%.
    - Bigger is not the same as better: the CFO's question needs both columns.

- *Quick Check 2 (Docstring):*
  - New tool: a docstring, a text in triple quotes on the first line inside a function. `help(function_name)` prints it. Why: a reviewer learns what each stage does without reading its code.
  - Task: add one line directly under `def add_metrics(df):`, indented like the line below it: `"""Add revenue in USD M from GMV and the take rate."""`. Then add `help(add_metrics)` to part **3. Output**.
  - Check: the Terminal prints a help page that includes `Add revenue in USD M from GMV and the take rate.`

- *Micro-Example 3 (Write the Findings for the CFO):* `YourName_S24_demo_3.py`

```
# 1. Input: load the data and build the region summary
import pandas as pd
df = pd.read_csv("data/session_02_04_dataset.csv")
df["Revenue_USD_M"] = (
    df["Gross_Merchandise_Value_USD_M"] * df["Platform_Take_Rate_Pct"] / 100
).round(2)
summary = df.groupby("Market_Segment").agg(
    Revenue_USD_M=("Revenue_USD_M", "sum"),
    OCF_USD_M=("Operating_Cash_Flow_USD_M", "sum"),
)

# 2. Process: margin, then the best region by margin
summary["OCF_Margin_Pct"] = (
    summary["OCF_USD_M"] / summary["Revenue_USD_M"] * 100
).round(1)
ranked = summary.sort_values("OCF_Margin_Pct", ascending=False)
best_region = ranked.index[0]
best_margin = ranked["OCF_Margin_Pct"].iloc[0]

# 3. Output: finding, evidence, recommendation
print(f"Finding: {best_region} turns the most revenue into cash.")
print(f"Evidence: its OCF margin is {best_margin}%.")
print("Recommendation: study its cost structure before expanding elsewhere.")
```

  - *How it works, line by line:*
    - Part 1 rebuilds the region summary, as in Micro-Example 2.
    - `summary.sort_values("OCF_Margin_Pct", ascending=False)` puts the best margin first.
    - `ranked.index[0]` reads the first row label, here the name of the best region. After `groupby()`, the region names are the row labels.
    - `ranked["OCF_Margin_Pct"].iloc[0]` reads the first value of the margin column by position.
    - The three `print()` lines write the finding, the evidence, and the recommendation (section 7.5), with the numbers filled in by f-strings.
  - *Read the output:*

```
Finding: North_America turns the most revenue into cash.
Evidence: its OCF margin is 31.0%.
Recommendation: study its cost structure before expanding elsewhere.
```

    - The region and the margin come from the data, not from typing: if next year's file changes, the sentence changes with it.
    - The recommendation line is still written by a person. The script supplies the evidence; the judgement stays with the analyst.

- *Quick Check 3 (`.idxmax()`):*
  - New tool: `column.idxmax()` returns the row label of the largest value, in one step. Why: finds "which region?" without sorting.

- Task: add one line to part **3. Output**: `print("Most revenue:", summary["Revenue_USD_M"].______())`
- Check: the Terminal shows `Most revenue: Asia_Pacific`. The region with the most revenue is not the one with the best margin.

10. Official Python Documentation & References

- Python Tutorial: Defining Functions.
- PEP 257: Docstring Conventions.
- Pandas API Reference: `DataFrame.pivot_table`.
- Cookiecutter Data Science: A Logical, Reasonably Standardized Project Structure.

11. Exercise 1: Guided Practice - Quarterly Review Pipeline

- *Step 1: Open Script:* In Google Antigravity IDE, create a new script named `YourName_S24_defense.py`, replacing `YourName` with your own name (for example, `Reacher_S24_defense.py`).
- *Step 2: Build the Quarterly Review:* Type the following code:

```
# 1. Input: load the quarterly results
import pandas as pd
df = pd.read_csv("data/session_02_04_dataset.csv")

# 2. Process: revenue per row, then totals per quarter
df["Revenue_USD_M"] = (
    df["Gross_Merchandise_Value_USD_M"] * df["Platform_Take_Rate_Pct"] / 100
).round(2)
quarterly = df.groupby("Quarter_Code").agg(
    GMV_USD_M=("Gross_Merchandise_Value_USD_M", "sum"),
    Revenue_USD_M=("Revenue_USD_M", "sum"),
    OCF_USD_M=("Operating_Cash_Flow_USD_M", "sum"),
)
quarterly["OCF_Margin_Pct"] = (
    quarterly["OCF_USD_M"] / quarterly["Revenue_USD_M"] * 100
).round(1)

# 3. Output: one row per quarter
print("Quarterly review:\n", quarterly)
```

- *Step 3: Write Two Findings:* Under the code, write two comment lines in the three-part form (finding, evidence, recommendation): one about the quarter with the highest GMV, and one about how the total OCF margin changed during the year.
- *Step 4: Execute in Terminal:* Run `python YourName_S24_defense.py` in Panel `[4]` and confirm return code `0`.
- *Success Check:* `2024-Q4` has the highest GMV (`524.4`) and OCF (`17.9`); Q2 has the lowest GMV (`312.4`). The total OCF margin is `23.7`% in Q1, `23.7`% in Q2, `24.6`% in Q3, and `25.8`% in Q4.

12. Exercise 2: Applied Practice - Peer Review a Classmate's Script

- *Coding Challenge:* A classmate sends you this script for their midterm. Copy it into `YourName_S24_practice.py`, review it, and fix it.

```
# Classmate's script: this script has mistakes on purpose
import pandas as pd
df = pd.read_csv("data/session_02_04_dataset.csv")
df["Revenue_USD_M"] = df["Gross_Merchandise_Value_USD_M"] * 0.125
total_revenue = df["Revenue_USD_M"].sum()
print("Total revenue (USD M): " + total_revenue)
```

- *Task 1 (Run It & AI Diagnosis):* Run the script. It stops with `TypeError: can only concatenate str (not "numpy.float64") to str`. Capture the traceback with `Win + Shift + S`, paste it into Panel `[3]` (AI Agent Chat) with `Ctrl + V`, and ask why text and a number cannot be joined with `+`. (`numpy.float64` is the number type that pandas returns; it behaves like a normal `float`.)
- *Task 2 (Fix the TypeError):* Change the last line to an f-string, rounding to 2 decimals: `f"Total revenue (USD M): {total_revenue:.2f}"`.
- *Task 3 (Apply the Peer Review Checklist):* Check the script against these five questions, and fix every "no":

  1. Does it run from top to bottom without errors?
  2. Does it use a relative path (`data/...`)?
  3. Is every number calculated from the data, not typed in by hand?
  4. Are outputs labeled with units?
  5. Would a manager understand the printed result?

- *Success Check:* the hard-coded `0.125` fails question 3, because each row has its own take rate. With `df["Platform_Take_Rate_Pct"] / 100`, the script prints `Total revenue (USD M): 204.31`. The hard-coded version prints `206.93`: a believable number that is wrong.

13. Exercise 3: Advanced Practice - Where Should We Invest Next Year?

- *Goal:* Build the region report the CFO needs for next year's budget: size, cash efficiency, and trend for every region, plus a list of weak quarters.
- *What you need:*
  - File: `data/session_02_04_dataset.csv` (described in **5. Lab Practice Dataset**).
  - Script name: `YourName_S24_challenge.py`.
  - Tools you already know: functions with `def` / `return`, `groupby()` + `.agg()`, `.pivot_table()`, `.loc[]`, `.sort_values()`, f-strings, and `.to_csv()`.
- *Requirements (follow them in order):*

  1. Write `load_data(path)` that reads the file and checks with `assert` that GMV, take rate, and OCF are all above 0.
  2. Write `add_metrics(df)` that adds `Revenue_USD_M` (2 decimals) and `OCF_Margin_Pct` (1 decimal) to every row.

3. Write `region_report(df)` that returns one row per region with `GMV_USD_M`, `Revenue_USD_M`, `OCF_USD_M` (sums), `OCF_Margin_Pct` (from the sums, 1 decimal), and `Margin_Change_Pts`: the region's Q4 margin minus its Q1 margin, in percentage points, 1 decimal.
4. Write `weak_quarters(df, limit)` that returns the rows with `OCF_Margin_Pct` below `limit`, sorted from lowest to highest. Call it with `limit=20`.
5. Sort the region report by `OCF_Margin_Pct`, highest first. Print it, print the weak quarters, and save the report to `data/region_investment_report.csv`.
6. Print one recommendation line built with an f-string from the report's numbers (not typed in by hand).

- *New tools for this exercise:*
  - Percentage points: the difference between two percentages. A margin moving from `15.0`% to `22.0`% rose by `7.0` percentage points (pts), not by 7%.
  - Reading one cell of a pivot table: `margins = df.pivot_table(index="Market_Segment", columns="Quarter_Code", values="OCF_Margin_Pct")`, then `margins["2024-Q4"] - margins["2024-Q1"]` gives the change for every region at once.
  - Adding a column that lines up by row label: `report["Margin_Change_Pts"] = (margins["2024-Q4"] - margins["2024-Q1"]).round(1)` works because both use the region names as row labels.
  - A function parameter with a value: `weak_quarters(df, limit=20)` passes `20` into `limit`.
- *Hints:*
  - Build and print each function's result before you write the next function.
  - The margin of a region must come from its total OCF divided by its total revenue, not from averaging the 4 quarterly margins.
- *Self-Check (expected results):*
  - Report order: `North_America` (margin `31.0`, change `+0.2` pts), `Europe` (`24.7`, `-1.1` pts), `Asia_Pacific` (`18.5`, `+7.0` pts).
  - Weak quarters (margin below 20%): `Asia_Pacific` in `2024-Q1` (`15.0`), `2024-Q2` (`16.8`), and `2024-Q3` (`18.6`).
  - Business meaning: Asia Pacific is the biggest region by GMV (`563.9`) and revenue, but the least cash-efficient, and also the fastest-improving (`+7.0` pts). North America is the cash engine. A sound recommendation: keep investing in Asia Pacific's growth, tied to a target such as reaching North America's cost structure, and protect North America's margin.