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

THE BUSINESS SITUATION

- An IT (Information Technology) services company holds 15 enterprise client contracts worth $10.7 million a year
- The account director has a vague worry: "some clients are unhappy and may leave", but a worry is not something data can test
- The account director asks: which contracts are at risk of leaving, and how much revenue is at stake? Today, you turn that worry into a testable question, check that the data can be trusted, and measure the starting point. This is step 1 of your midterm project

Today's flow: 60 min lecture, then 120 min hands-on lab

EXPECTED OUTCOMES

By the end of today, you will be able to:

1. ***"What exactly should we test?"***
   Turn a vague business problem into a testable hypothesis (a clear statement that data can support or prove wrong)
2. ***"Can we trust this data?"***
   Audit a dataset for missing values (`.isnull()`), duplicates (`.duplicated()`), and impossible values, such as a percentage above 100
3. ***"How do we stop bad data automatically?"***
   Write `assert` checks (rules the data must pass) and handle an `AssertionError` with `try` / `except`

Session Map: Follow the Numbers

Today's question: *Which contracts are at risk of leaving, and how much revenue is at stake?*

1 Why it matters
Lecture, part 1

1. **Session Purpose**
   answer today's question
2. **Learning Outcomes**
   3 skills you will show today
3. **Business Context**
   From a Vague Worry to a Testable Question
4. **Data Basics**
   Three Checks Before Any Analysis

→

2 Learn the tools
Lecture, part 2

5. **Lab Dataset**
   2 CSVs
6. **Keywords**
   6 terms
7. **Core Foundations**
   5 ideas, one at a time
8. **IDE Setup**
   name files, run in the Terminal
9. **Code Examples**
   3 demos + Quick Checks
10. **References**
    3 official docs

→

3 Practice
Lab

11. **Exercise 1: Guided**
    Hypothesis Test Script
12. **Exercise 2: Applied**
    Data Gatekeeper
13. **Exercise 3: Advanced**
    How Much Did Bad Data Hide?

From a Vague Worry to a Testable Question

KEY POINTS

- Managers rarely hand you a clean question.
- SLA (Service Level Agreement): the service promise in a contract, such as "the system is available 99% of the time".
- Baseline: today's value of the measure you care about.
- **Midterm Project, Step 1 of 2: your Question Card.** Choose a business problem with a dataset you can audit (for example, any dataset from Sessions 1.3 to 2.2), and fill in five lines:
  - Vague worry: what the manager is concerned about
  - Business question: what decision the answer supports
  - Hypothesis: one sentence the data can support or prove wrong
  - Data and audit: the file you use, and the result of the three checks in section 4
  - Baseline: one number that describes today
- Bring your completed Question Card and your audit script to Session 2.4, where you present your findings.

SEE IT ON REAL DATA

Step	Example
Vague worry	"Some clients are unhappy and may leave."
Business question	"Is poor service linked to clients at risk of leaving?"
Testable hypothesis	"Accounts below 95% SLA compliance have more billing disputes and a higher retention risk than other accounts."
Measure today (baseline)	"Revenue in High-risk accounts today: $930,000."

Three Checks Before Any Analysis

KEY POINTS

- Bad data produces confident but wrong answers ("garbage in, garbage out").
- `NaN` ("Not a Number") is how pandas shows an empty cell.
- A rule for validity comes from business logic: a percentage must be between 0 and 100, and a contract value must be above 0.

AT A GLANCE

Check	Question	pandas
Completeness	Are any values missing?	`df.isnull().sum()`
Uniqueness	Does each ID appear only once?	`df["Account_ID"].duplicated().sum()`
Validity	Are all values possible?	`(df["SLA_Compliance_Pct"] > 100).sum()`

The Data for Today

FILES

- `data/session_02_03_dataset.csv`
- `data/session_02_03_raw.csv`

Columns (the same in both files):

COLUMNS

`Account_ID`	Unique client account code (e.g., `ACC-301`).
`Enterprise_Client`	Client company name (Titan Tech, Beacon Health, Vanguard Retail, and others). One company can hold several accounts.
`Contract_Value_USD`	Yearly contract value in U.S. Dollars.
`SLA_Compliance_Pct`	Share of the service promise delivered, in percent.
`Billing_Discrepancy_Count`	Number of billing disputes (invoices the client challenged) during the contract.
`Client_Retention_Risk`	The account team's rating of how likely the client is to leave: `Low`, `Medium`, or `High`.

6 Terms You Need Today

TERM 1

`Hypothesis`

What: one clear statement that data can support or prove wrong

Why: turns a vague worry into a test with a yes or no answer

TERM 2

`Exploratory Data Analysis (EDA)`

What: a first look at the data with counts, averages, and charts before any formal test

Why: shows what the data can answer, and where it looks strange

TERM 3

`Data Audit`

What: the three checks: completeness, uniqueness, and validity

Why: one bad row can change a total by millions

TERM 4

`Baseline`

What: today's value of the measure you care about

Why: the starting point for proving any later improvement

TERM 5

`assert`

What: a line that stops the script with an `AssertionError` when a rule is broken, such as `assert (df["Contract_Value_USD"] > 0).all()`

Why: bad data is caught at the door, before it reaches a report

TERM 6

`try` / `except`

What: runs code in `try`; if a chosen error happens, the code in `except` runs instead of stopping the script

Why: the script can report the problem clearly and keep going

How It Works

7.1

From a worry to a hypothesis you can test

Remember: a good hypothesis names the measure, the groups, and the expected difference.

7.2

The three data checks

Remember: complete, unique, valid; check all three before you calculate.

7.3

Rules the data must pass

Remember: one `assert` per rule, always with a message.

7.4

Handling a failure

Remember: `try` the check, `except` the expected error, and report it clearly.

7.5

The baseline

Remember: audit first, then record the baseline.

7.1 From a worry to a hypothesis you can test: On Real Data

SEE IT ON REAL DATA

Part of the hypothesis	In our data
What is measured	`Billing_Discrepancy_Count`
The two groups	SLA below 95% (`4` accounts) vs 95% or above (`11` accounts)
The expected difference	the first group has more disputes
The result	`4.75` vs `0.64` disputes on average: supported

Remember: a good hypothesis names the measure, the groups, and the expected difference.

Audit the Raw File

YourName_S23_demo_1.py — Input → Process → Output

```python
# 1. Input: the raw file, as it arrived
import pandas as pd
raw = pd.read_csv("data/session_02_03_raw.csv")

# 2. Process: the three checks, as counts
missing_values = raw.isnull().sum().sum()
duplicate_ids = raw["Account_ID"].duplicated().sum()
sla_above_100 = (raw["SLA_Compliance_Pct"] > 100).sum()
value_not_positive = (raw["Contract_Value_USD"] <= 0).sum()

# 3. Output: one line per check
print("Rows:", len(raw))
print("Missing values:", missing_values)
print("Duplicate account IDs:", duplicate_ids)
print("SLA above 100%:", sla_above_100)
print("Contract value of 0 or less:", value_not_positive)
```

HOW IT WORKS

- The script reads the **raw** file, as it came from the billing system, before anything is cleaned.
- `raw.isnull()` marks every empty cell `True`.
- `raw["Account_ID"].duplicated()` marks a row `True` when its ID already appeared above; `.sum()` counts them.

READ THE OUTPUT

```
Rows: 16
Missing values: 1
Duplicate account IDs: 1
...
```

- The file has 16 rows, but only 15 real contracts: one row is a copy.

Quick Check 1: `.drop_duplicates()`

NEW TOOL

`.drop_duplicates()`

`raw.drop_duplicates()` returns the table without rows that exactly repeat an earlier row. Why: a copied row counts the same contract twice.

YOUR TASK

Fill in the Blank

Add one line to part **3. Output**: `print("Rows after removing duplicates:", len(raw._____()))`

Check: the Terminal shows `Rows after removing duplicates: 15`.

Test the Hypothesis on the Clean File

YourName_S23_demo_2.py — Input → Process → Output

```python
# 1. Input: the clean file
import pandas as pd
df = pd.read_csv("data/session_02_03_dataset.csv")

# 2. Process: mark SLA breaches, then compare the two groups
df["SLA_Breach"] = df["SLA_Compliance_Pct"] < 95.0
comparison = df.groupby("SLA_Breach").agg(
    Accounts=("Account_ID", "count"),
    Avg_Disputes=("Billing_Discrepancy_Count", "mean"),
).round(2)

# 3. Output: breach (True) vs no breach (False)
print(comparison)
```

HOW IT WORKS

- The script reads the **clean** file: analysis is done on data that passed the checks.
- `df["SLA_Compliance_Pct"] < 95.0` marks each account `True` (breach) or `False`, and stores the result as a new column, `SLA_Breach`.
- `df.groupby("SLA_Breach")` splits the accounts into the 2 groups (Session 1.4).

READ THE OUTPUT

```
            Accounts  Avg_Disputes
SLA_Breach
False             11          0.64
True               4          4.75
```

- The 4 accounts below the 95% promise average `4.75` billing disputes; the other 11 average `0.64`, about 7 times fewer.

Quick Check 2: `pd.crosstab()`

NEW TOOL

`pd.crosstab()`

`pd.crosstab(column_a, column_b)` counts the rows for every pair of values, such as breach yes or no against risk Low, Medium, or High. Why: tests the second half of the hypothesis in one table.

YOUR TASK

Fill in the Blank

Add one line to part **3. Output**:

```
print(pd._____(df["SLA_Breach"], df["Client_Retention_Risk"]))
```

Check: in the `True` row, `High` is `3` and `Medium` is `1`: 3 of the 4 breached accounts are rated High risk, and no non-breached account is.

Measure the Baseline

YourName_S23_demo_3.py — Input → Process → Output

```
# 1. Input: the clean file
import pandas as pd
df = pd.read_csv("data/session_02_03_dataset.csv")

# 2. Process: contract value by risk rating, and the High-risk share
value_by_risk = df.groupby("Client_Retention_Risk")["Contract_Value_USD"].sum()
high_risk_value = value_by_risk["High"]
high_risk_share = high_risk_value / df["Contract_Value_USD"].sum() * 100

# 3. Output: today's baseline
print(value_by_risk)
print("High-risk share of revenue (%):", round(high_risk_share, 1))
```

HOW IT WORKS

- `df.groupby("Client_Retention_Risk")["Contract_Value_USD"].sum()` adds up contract value for each risk rating.
- `value_by_risk["High"]` reads one value from that list, by its label.
- The share divides the High-risk total by all contract value and multiplies by 100.

READ THE OUTPUT

```
Client_Retention_Risk
High       930000.0
Low       7770000.0
...
```

- `$930,000` a year, `8.7` % of revenue, sits in accounts rated High risk.

Quick Check 3: f-string number format

NEW TOOL

f-string number format

`f"${value:,.0f}"` writes a number with thousands separators and no decimals, such as `$930,000`. Why: managers read money faster in this form.

YOUR TASK

Fill in the Blank

Add one line to part **3. Output**: `print(f"Revenue at High risk: ${high_risk_value:___}")`

Check: the Terminal shows `Revenue at High risk: $930,000`.

Where to Look Things Up

REFERENCE

Python Tutorial: Errors and Exceptions (try / except)

https://docs.python.org/3/tutorial/errors.html

REFERENCE

Python Reference: The assert Statement

https://docs.python.org/3/reference/simple_stmts.html#the-assert-statement

REFERENCE

Pandas User Guide: Working with Missing Data

https://pandas.pydata.org/docs/user_guide/missing_data.html

Three Exercises

EXERCISE 1: GUIDED

Hypothesis Test Script

Goal: Test two hypotheses and measure the revenue at risk

File:
`YourName_S23_hypothesis.py`

Check: correlation `-0.982`; baseline `$930,000`

EXERCISE 2: APPLIED

Data Gatekeeper

Goal: Write checks that stop bad data, then clean the file

File:
`YourName_S23_practice.py`

Check: `clean` has `12` rows and passes all 3 checks

EXERCISE 3: ADVANCED

How Much Did Bad Data Hide?

Goal: Show in numbers how bad data distorts the risk report

File:
`YourName_S23_challenge.py`

Check: raw file: 4 problems; clean file: `0`

Full steps for all three exercises: sections 11-13 of the course page.