Data & Task Abstraction

A practical framework for translating domain questions into defensible visual designs.

Marc Reyes
Professional Lecturer · marc.reyes@dlsu.edu.ph
DATA101 — De La Salle University

Today's Plan

01 · SETUP

Why abstraction matters

Avoid the chart-first trap.

02 · DATA

Dataset + attribute types

What you have, what it means.

03 · TASKS

Goals + actions + targets

What your user needs to do.

04 · DESIGN

From abstraction → charts

Views + interactions you can defend.

05 · PRACTICE

Exercises + exit ticket + Python assignment

Write abstractions like a practitioner (then implement in pandas).

Learning Outcomes

DATA

Identify dataset structure

Table, time series, spatial, network, hybrid...

DATA

Label attribute types

Categorical, ordinal, quantitative, temporal.

TASKS

Write task statements

Action + Target + Constraints + Output.

DESIGN

Justify chart + interaction

Design decisions that map to data + tasks.

Warm-Up (3 minutes)

PROMPT

“The Dean wants to know if students are struggling more this term.”

METRIC

Scores? pass rate? attendance? drop rate?

BASELINE

Last term? last year? another section?

OUTPUT

Which groups? when? how big? how confident?

Abstraction = translation

From domain language → to general structures that visualization methods can support.

The Chart-First Trap (and How to Avoid It)

COMMON FAILURE MODE

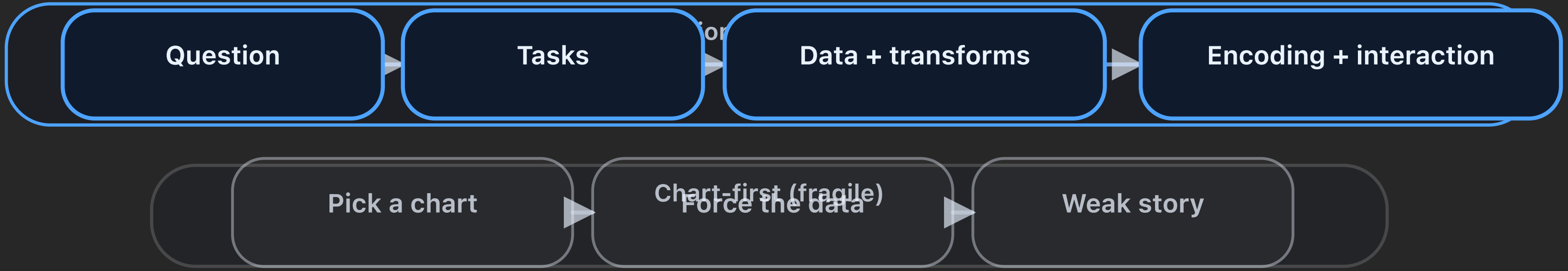
Chart-first thinking

“Make a bar chart” is a solution, not a problem statement.

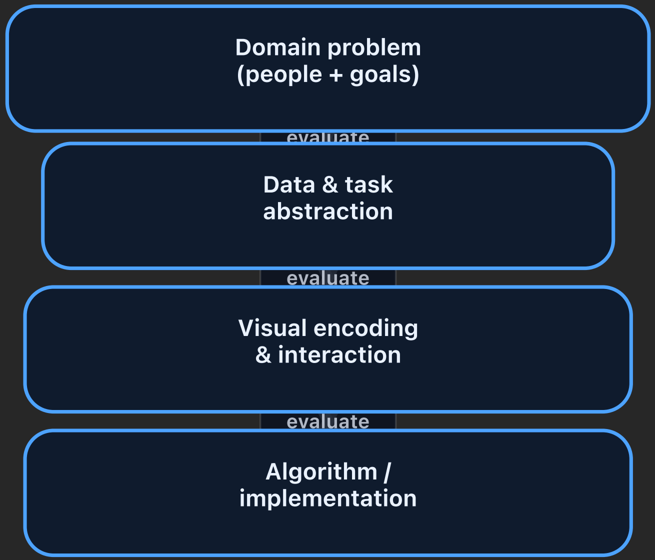
PROFESSIONAL WORKFLOW

Abstraction-first thinking

Question → tasks → data needs → transforms → design.



Munzner’s Nested Model (Where Abstraction Lives)

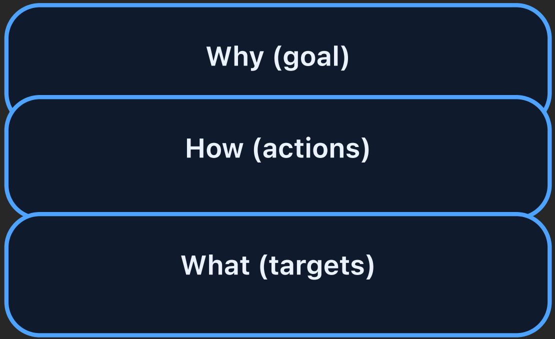


Two Outputs You Should Be Able to Write

Before picking charts: write the **task spec** and **data spec**.

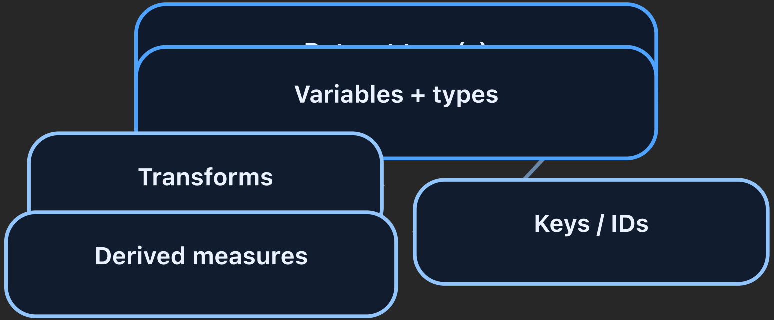
TASK ABSTRACTION

Why → How → What



DATA ABSTRACTION

Types → Variables → Transforms



Running Example (We'll Use This All Lecture)

Question: "Are students struggling more this term?"

- Possible data sources: weekly quizzes, attendance logs, LMS activity, advising records
- Possible unit of analysis: student, section, program, college
- Possible time scale: week, month, midterms/finals phases

What a “Good Answer” Looks Like

SUCCESS CRITERIA

TARGET

Who?

Which sections/programs are struggling?

TIME

When?

Which weeks; before/after which event?

MAGNITUDE

How much?

Show distributions, not just averages.

BASELINE

Compared to what?

Last term, target, or benchmark.

PART 1 · DATA

Data Abstraction

From domain data → dataset types + attribute types + transformations

Data Abstraction: What You Produce

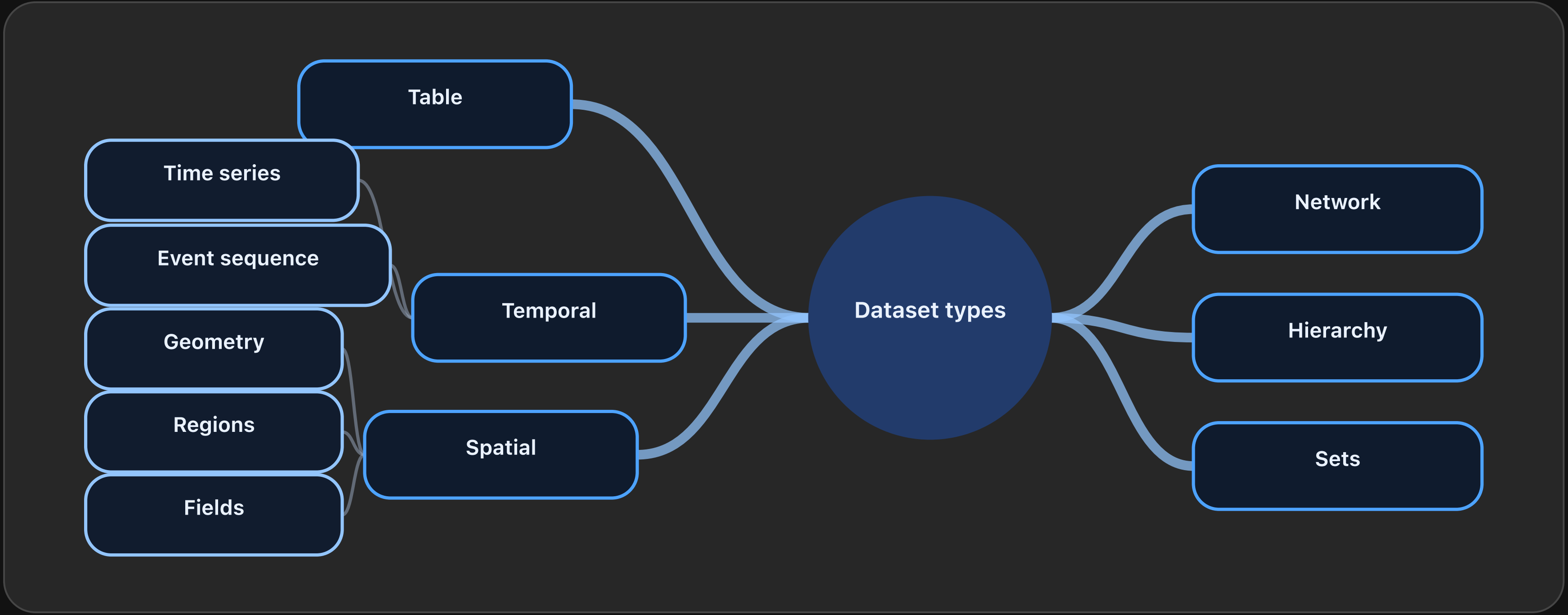
Dataset type(s) + attribute types + required transforms

- Dataset type(s): table, time series, spatial, hierarchy, network, field, sets
- “Items” vs “relationships” vs “positions”
- Variable list with attribute types + units
- Required transformations: cleaning, aggregation, binning, derived measures

Start With an Inventory (Before Any Charts)

Question	What you write down
What are the **items**?	rows / records (students, sessions, transactions)
What are the **variables**?	columns (program, score, week, minutes)
Are there **relationships**?	links (prerequisite, collaboration, referral)
Are there **positions**?	time order, coordinates, grid cells

Dataset Types (Visualization Lens)



Dataset Type: Table (Items × Attributes)

Example: student records

- Items: students or section-week records
- Typical transforms: group-by, summarize, sort, filter
- Typical views: bar chart (compare), dot plot (rank), histogram/box plot (distribution)

Dataset Type: Time Series (Ordered by Time)

Common mistakes: missing weeks, irregular sampling, mixing time zones.

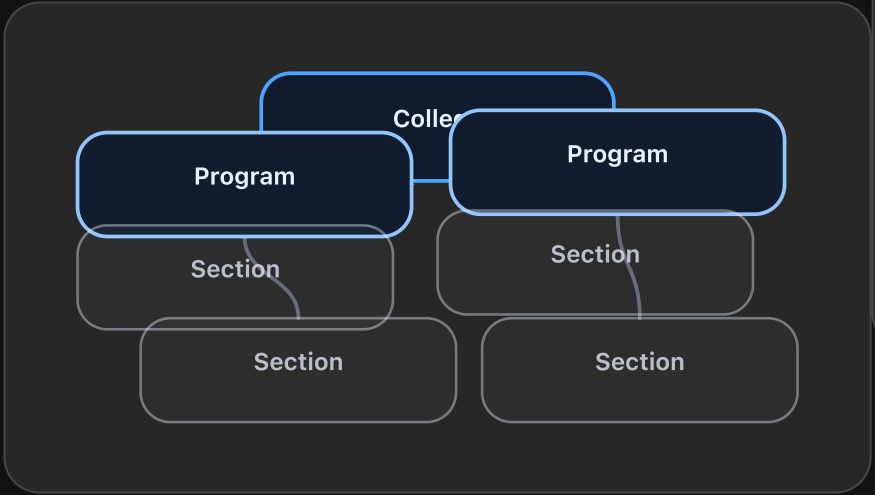
- Decide the time unit (day/week/month) and make it explicit
- Consider smoothing carefully (rolling mean can hide spikes)
- Baselines matter: compare to last term or target performance

Spatial Data: Geometry vs Regions vs Fields

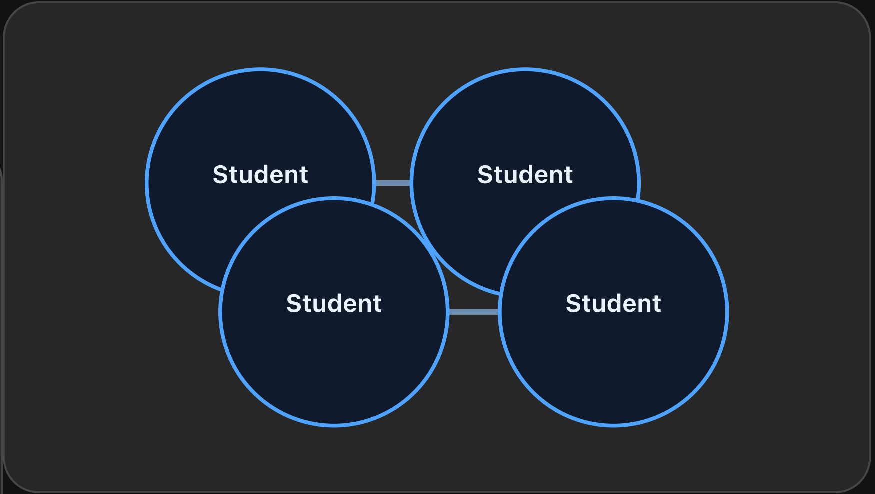
- **Geometry:** points/lines (GPS pings, routes) → proximity, clusters
- **Regions:** polygons (cities/barangays) → compare areas, choropleths (careful with population)
- **Fields:** values everywhere (density/temperature) → heatmaps, contours, binning choices

Hierarchy vs Network (Know the Difference)

Hierarchy (parent → child)

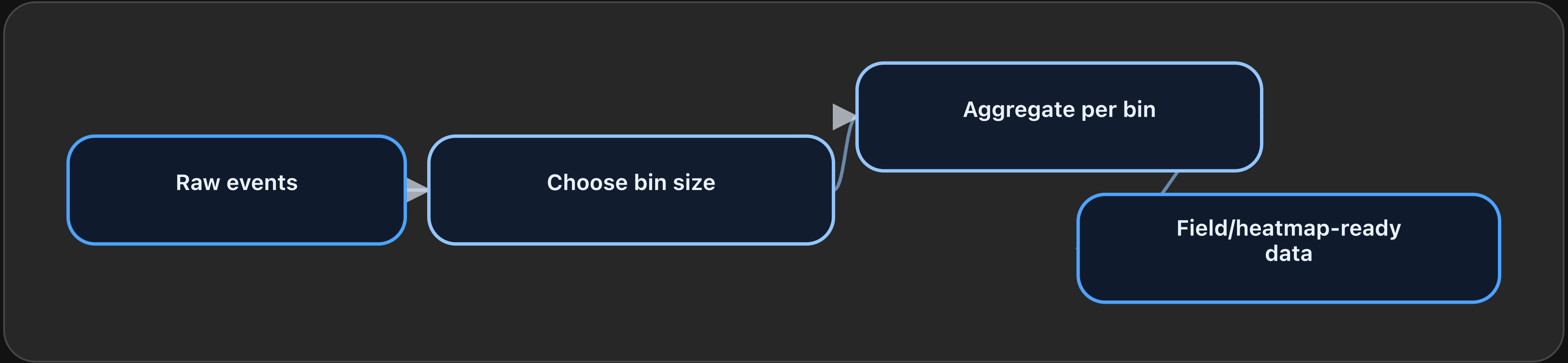


Network (links between peers)



Fields & Density: Why Binning Is a Design Decision

- Raw events → bins (grid cells, time windows) → aggregated values
- Bigger bins: smoother but can hide local patterns
- Smaller bins: detailed but noisier; may exaggerate randomness



Sets & Membership Data

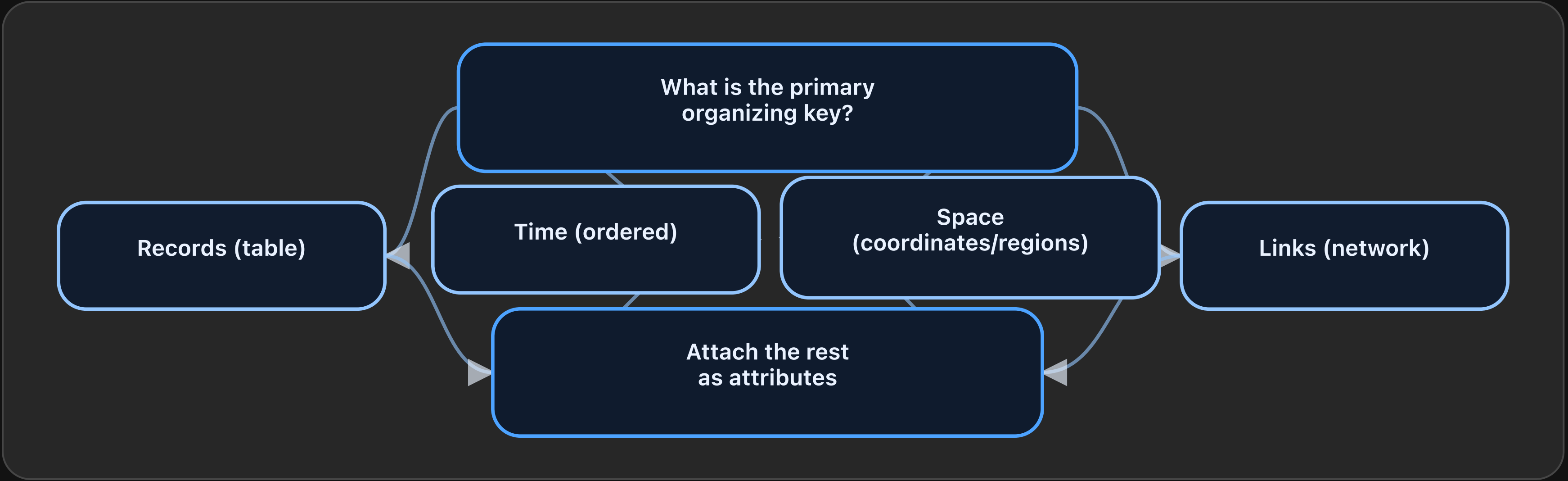
When items belong to multiple groups (e.g., students in orgs + electives).

- Dataset structure: items + membership lists
- Typical tasks: overlap, exclusive groups, coverage
- Warning: Venn diagrams don't scale; consider tables or UpSet-style views

Hybrid Datasets (Most Real Problems)

Many datasets are **table + time + category** (and sometimes spatial).

- Choose a primary structure (often a table of records)
- Decide whether time/space are axes or attributes; keep stable IDs (student_id, section_id)



Attribute Types (Semantics of Variables)

Categorical
different kinds

Examples: program, device_type

Channels: color hue, shape, grouping

Ordinal
ranked kinds

Examples: Likert 1–5, grade bands

Channels: position, ordered color

Quantitative
magnitude

Examples: score, minutes, count

Channels: position, length, size

Temporal
time

Examples: week, timestamp

Channels: position (x), ordering

Measurement Scales (What Math Is Valid?)

Scale	Example	You can do…	Don't…
Nominal	program	count, mode	average it
Ordinal	rank, Likert	median, order	assume equal gaps
Interval	°C	differences	claim “twice as hot”
Ratio	counts, ₱	ratios, % change	ignore units

Identifiers vs Measures vs Categories

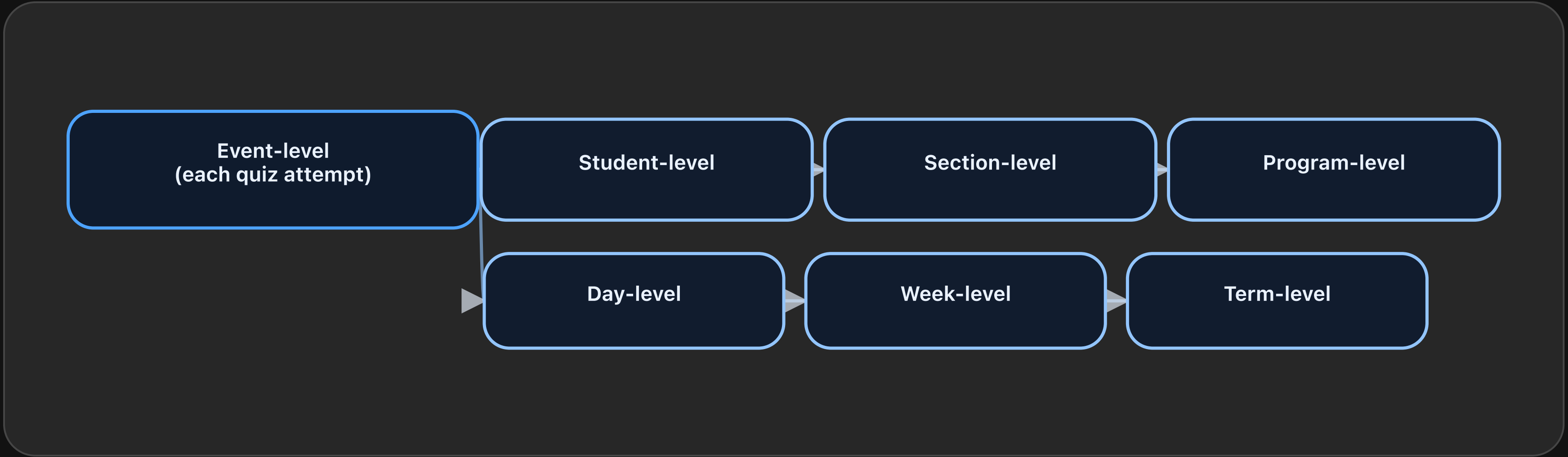
- **Identifier:** labels one item (StudentID, SectionCode) → use for joins, not charts
- **Measure:** numeric value with meaning (score, minutes, count) → plot/analyze
- **Category code:** looks numeric but is categorical (1=CS, 2=IT) → treat as categorical

Quick test: “If I average this, does the result mean anything?”

Derived Measures (Often the Real KPI)

- Rates: pass_rate = passes / enrolled
- Normalization: incidents per 1,000 students (not raw counts)
- Change: week-over-week difference or percent change
- Composite indices: only if components and weights are justified

Granularity & Aggregation (Choose With Tasks)



Aggregation hides variance; keep distributions when decisions affect individuals.

Reshaping for Visualization

Wide (one row per student)

student	quiz1	quiz2	quiz3
A	7	8	6

Long / tidy (one row per observation)

student	quiz	score
A	quiz1	7
A	quiz2	8
A	quiz3	6

Data Quality & Bias (A Fast Checklist)

- Missingness: random or systematic? (e.g., absent students)
- Outliers: errors or rare events?
- Units: consistent? (minutes vs hours; ₱ vs $)
- Denominators: use rates when group sizes differ
- Coverage: who is excluded by the data collection process?

Practice 1 (5 minutes): Abstract This Dataset

Wi-Fi session log

Columns: `timestamp`, `student_program`, `access_point`, `session_minutes`, `device_type`

- Dataset type(s)?
- Attribute type of each variable?
- One derived measure you might need (rate/ratio/change)?

PART 2 · TASKS

Task Abstraction

From domain questions → actions + targets + constraints

Task Abstraction: What You Produce

Action + Target + Constraints + Output

- Action: compare, rank, summarize, detect, locate, filter
- Target: items, groups, attributes, time ranges, links
- Constraints: “this term only”, “by program”, “top 5 sections”
- Output: “a ranked list”, “a time window”, “a set of flagged outliers”

Chart Request → Task Statement (Rewrite)

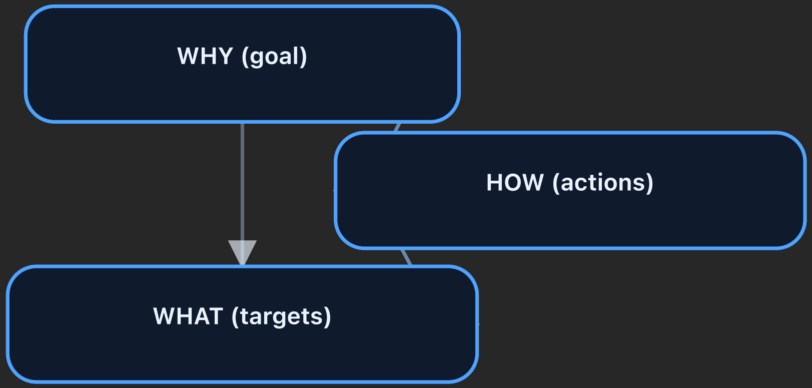
- ❌ “Make a bar chart of programs”

  ✅ “Compare programs by **pass rate** this term”
- ❌ “Use a line chart for quizzes”

  ✅ “Detect **when** quiz performance drops and **which sections** drop the most”
- ❌ “Create a dashboard with filters”

  ✅ “Enable **browsing** by program and **drill-down** to student-level details on demand”

A Strong Framework: WHY / HOW / WHAT



WHY / HOW / WHAT Vocabulary

WHY (GOAL)

Motivation

- Discover: find unknown patterns
- Present: communicate clearly
- Monitor: track known metrics
- Lookup: answer a specific question

HOW (ACTIONS)

Verbs

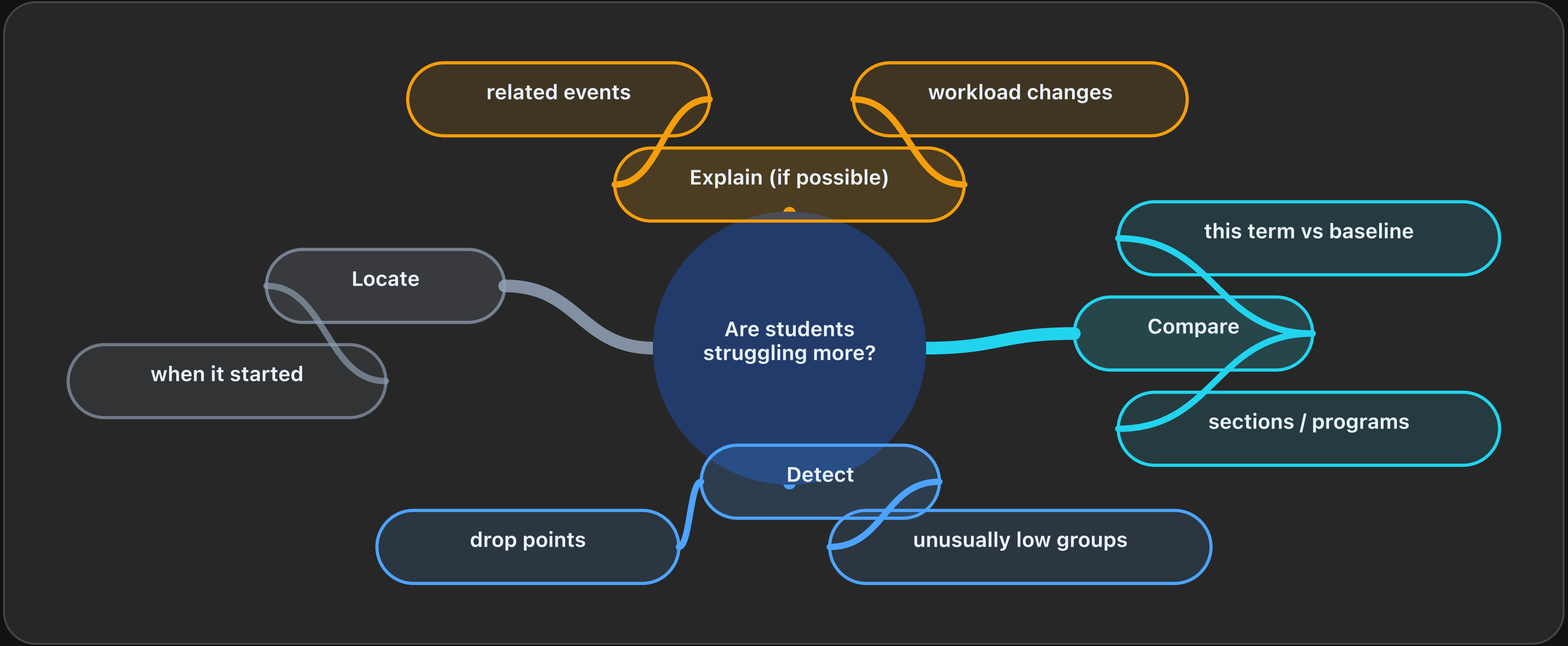
- Search: lookup · locate · browse · explore
- Query: filter · sort · group
- Compare: rank · contrast · benchmark
- Detect: outliers · change points

WHAT (TARGETS)

Objects

- Items: student, section, record
- Groups: program, cohort
- Attributes: score, pass_rate, minutes
- Ranges: week 3–6, pre/post event
- Links: prereq, collaboration, referral

Decompose the Running Example Into Subtasks



Search Tasks: Lookup → Explore

- Shneiderman: overview first → zoom/filter → details on demand

Compare Tasks: Three Common Patterns

- **Compare categories:** section A vs B (use aligned scales; sort when needed)
- **Rank:** top/bottom N (make ordering explicit; show ties)
- **Benchmark:** compare to a target (add reference lines/bands)

If comparison is the task, design for *alignment* and *readable differences*.

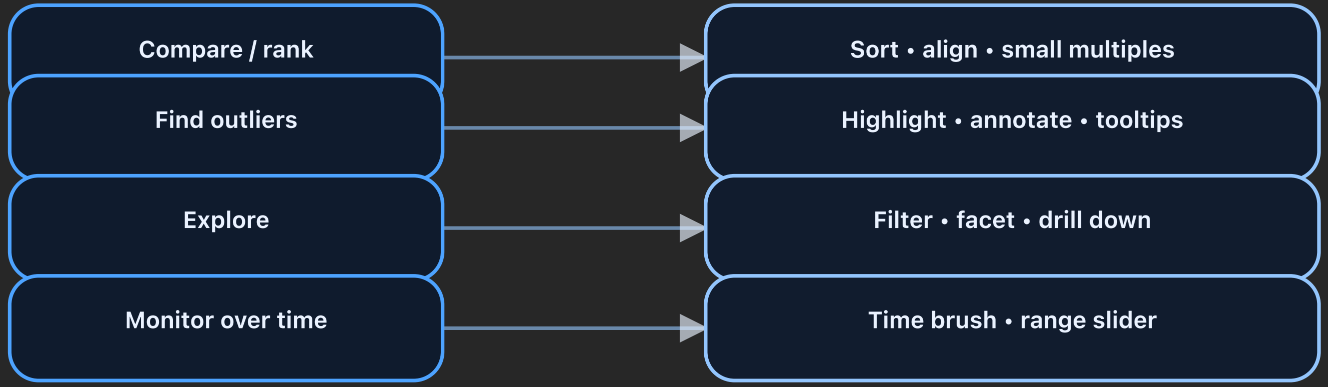
Distribution Tasks: “What’s Typical?” + “Who Is Different?”

- Ask for: center, spread, skew, outliers
- Use: histogram (shape), box plot (summary), violin (density)
- Don’t hide the distribution behind a single average when decisions affect people

Relationship Tasks: Correlate, Cluster, or Explain?

- **Correlate:** do two measures move together?
- **Cluster:** do groups form naturally (segments)?
- **Explain:** what factors predict an outcome? (needs modeling + careful claims)
- Reminder: correlation ≠ causation; check confounders and sampling bias

Tasks ↔ Interactions (Design on Purpose)



Task Quality Rubric (For Reports and Projects)

- Uses a clear **action verb** (compare/rank/detect…)
- Names an explicit **target** (items/groups/attributes/time range)
- States constraints (population/timeframe/baseline)
- Produces an **output** that someone can verify (ranked list, flagged cases, chosen window)

Practice 2 (7 minutes): Write Two Task Statements

Pick one dataset from Practice 1 and write:

- One **monitoring** task (ongoing tracking)
- One **discovery** task (exploration)

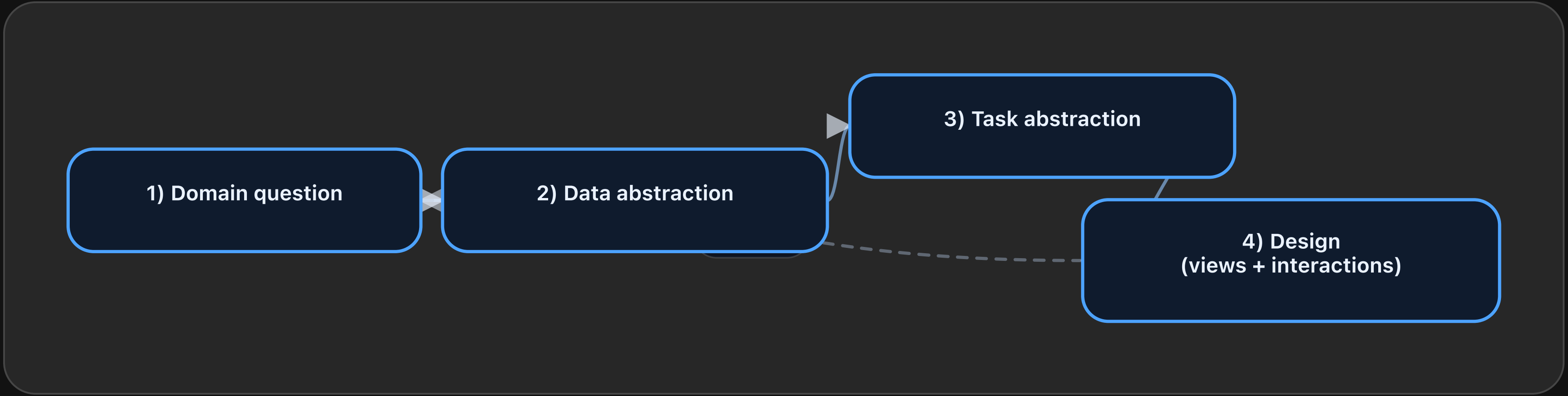
Use: **Action + Target + Constraints + Output**

PART 3 · DESIGN

Putting It Together

From abstractions → justified visualization designs

The 4-Step Abstraction Worksheet (Use This Every Time)



Channel Effectiveness

RULE OF THUMB

Most precise → least precise

- **Position (common scale)**
- **Length**
- **Angle / Area**
- **Color / Shape**

If the task is comparison, prioritize **position** and **alignment**.

DESIGN IMPLICATIONS

How this changes chart choices

- **Ranking / comparison** → dot plots, sorted bars, small multiples
- **Magnitude** → avoid area-only encodings for precision
- **Categories** → use hue for grouping, not “how much”
- **Many groups** → sort + label; reduce legend hunting

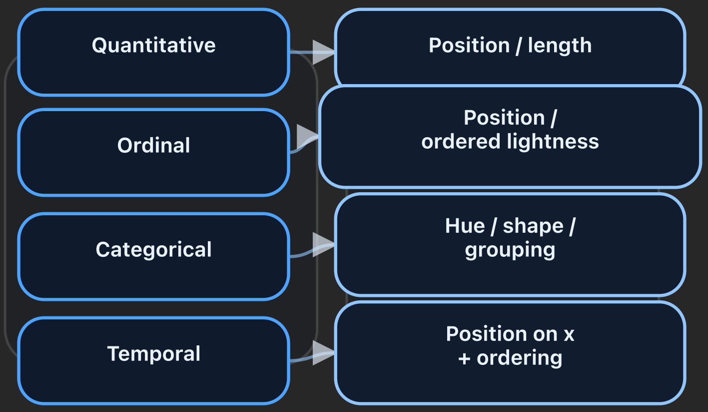
Match Encoding to Type

ABSTRACTION → SAFE CHANNELS

Attribute types constrain what encodings mean.

- Quantitative → position / length for comparison
- Ordinal → position or ordered lightness
- Categorical → hue, shape, grouping
- Temporal → position on x + ordering

If the type is wrong, the chart is wrong—even if it looks polished.



Evaluation Checklist (Before You Submit a Viz)

TASKS

- Action verb is explicit (compare/rank/detect...)
- Target + baseline are named
- Output is verifiable (top-5 list, flagged weeks...)

DATA

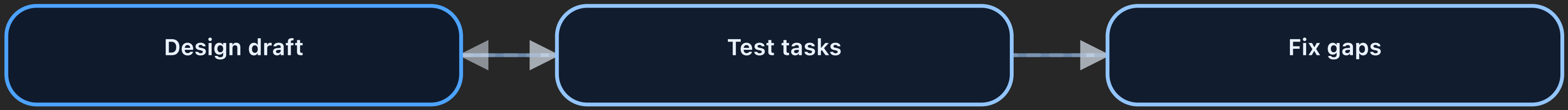
- Types + units are correct
- IDs are stable; joins are valid
- Denominators are handled (rates vs counts)

DESIGN

- Aligned scales for comparisons
- Legible labels, annotations, and legends
- Uncertainty + missingness are disclosed

ITERATION

- Test with 2–3 real task questions
- Revise at the abstraction level first
- Then adjust encodings/interactions

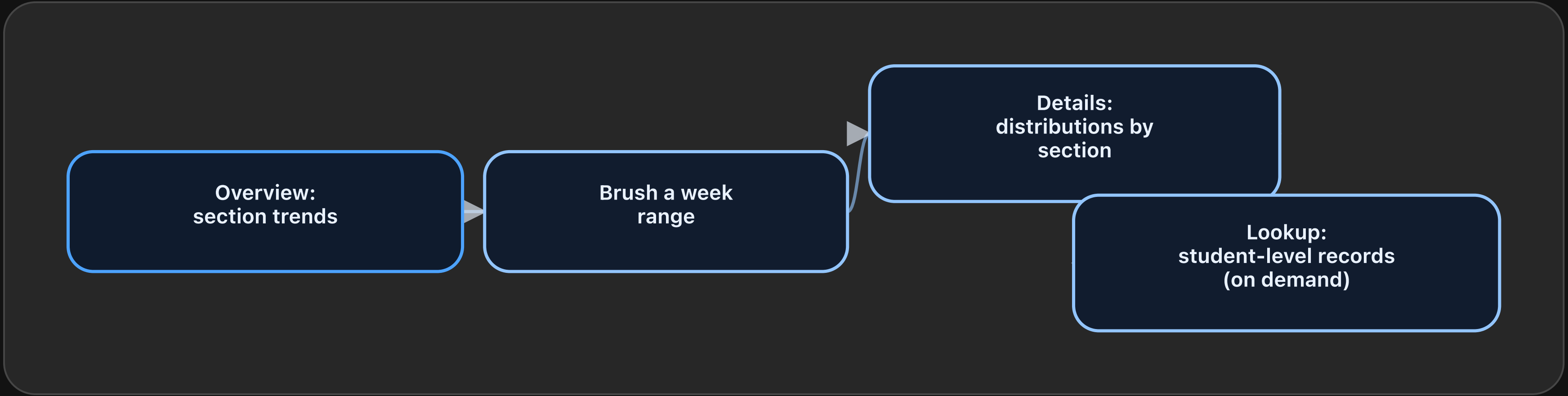


Case Study: Student Performance (A Task-Driven Design)

- **Data abstraction:** table of section-week records

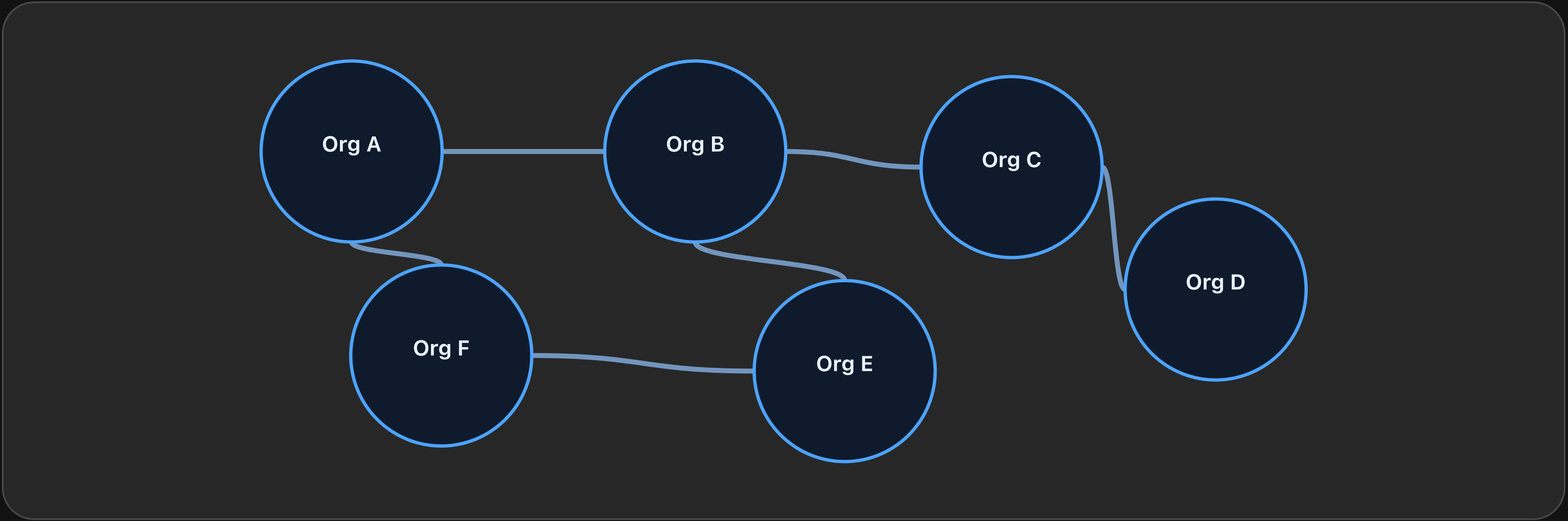
  Variables: section (cat), week (temp), avg_score (quant), pass_rate (quant), n_students (quant)

- **Key tasks:** compare sections, detect drops, locate weeks, drill down to details



Case Study: Collaboration Network (Different Data, Different Tasks)

- **Data abstraction:** network (nodes=orgs, links=collaborations), link weight=quant
- **Tasks:** find hubs, bridge orgs, communities; compare before/after an event
- **Design hint:** combine network view with a sortable table for reliable ranking



Key Takeaways

- Abstraction is the bridge from **domain** to **design**
- Data abstraction: dataset types + attribute types + transformations
- Task abstraction: goals + actions + targets (+ constraints + output)
- Good charts are **defensible** because they directly support tasks
- Avoid common failures: type mixing, raw counts without denominators, over-aggregation, vague tasks

Exit Ticket + References

Exit ticket (answer in 2–3 sentences each)

What is the dataset type and attribute types for your chosen example?

Write one task as **Action + Target + Constraints + Output**

What interaction would most help that task, and why?

References

- Munzner, *Visualization Analysis & Design*
- Brehmer & Munzner (2013), abstract task typology
- Wickham (2014), tidy data

Marc Reyes · `marc.reyes@dlsu.edu.ph`

Python Assignment (Take-Home): Abstraction → Design

GOAL

Turn a domain question into a defendable visualization workflow.

WRITE

- **1 domain question** (one sentence)
- **Data abstraction**: dataset type(s) + variable types
- **Task abstraction**: 3 tasks (Action + Target + Constraints + Output)

BUILD (PYTHON)

- **Transforms**: clean, tidy/reshape, derive measures
- **2 charts** that directly support your tasks
- **Justification**: 4–6 sentences mapping choices to tasks

Starter Code + Deliverables

```python
import pandas as pd

df = pd.read_csv("your_data.csv")

# 1) Data abstraction: fix types (example)
# df["date"] = pd.to_datetime(df["date"])

# 2) Transforms: tidy + aggregate for a task
result = (
    df.dropna()
      .groupby(["group", "time"], as_index=False)
      .agg(value=("value", "mean"), n=("value", "size"))
)
```

SUBMIT

- `abstraction.md` (data spec + task statements)
- `analysis.ipynb` (transforms + charts)
- Export charts as `.png` or `.svg`

RUBRIC (SIMPLE)

- Correct types + meaningful derived measures
- Tasks are specific and verifiable
- Charts clearly support tasks (not “favorite charts”)