IMEN891K / IDSC501 / AXGS-501 · DATA ENGINEERING · FALL 2026

Review Session

Sangwoo Mo · POSTECH
Industrial & Management Engineering

Outline: Review Session

01	**Collection and Exploration**	Lec 2–3 · records, labels and patterns
02	**Data Preparation**	Lec 4–5 · changes to values and representation
03	**SQL**	Lec 6–8 · read, match and summarize rows
04	**NoSQL**	Lec 9–10 · models, documents and updates

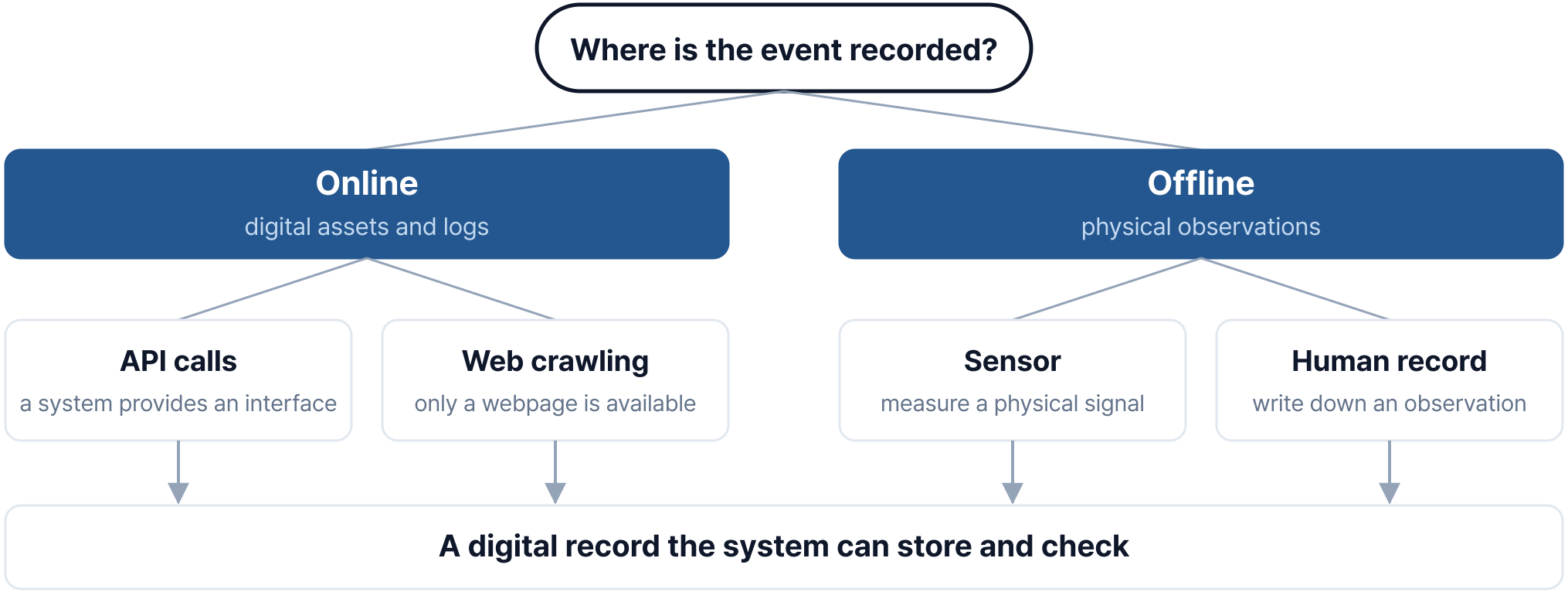
01 · Collection and Exploration

01	**Collection and Exploration**	Lec 2–3 · records, labels and patterns
02	**Data Preparation**	Lec 4–5 · changes to values and representation
03	**SQL**	Lec 6–8 · read, match and summarize rows
04	**NoSQL**	Lec 9–10 · models, documents and updates

Outline: Data Collection



Online and Offline Data

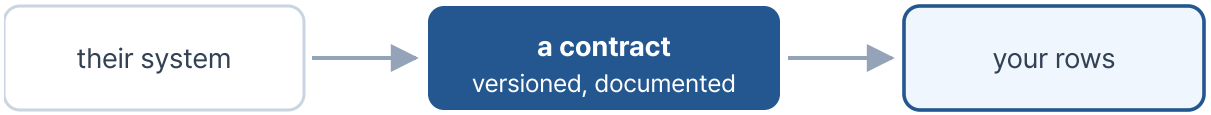


Locate the observation source before choosing the collection method

API or Crawler

Prefer an explicit interface; page extraction also depends on the page layout

API



An explicit interface can be versioned and checked

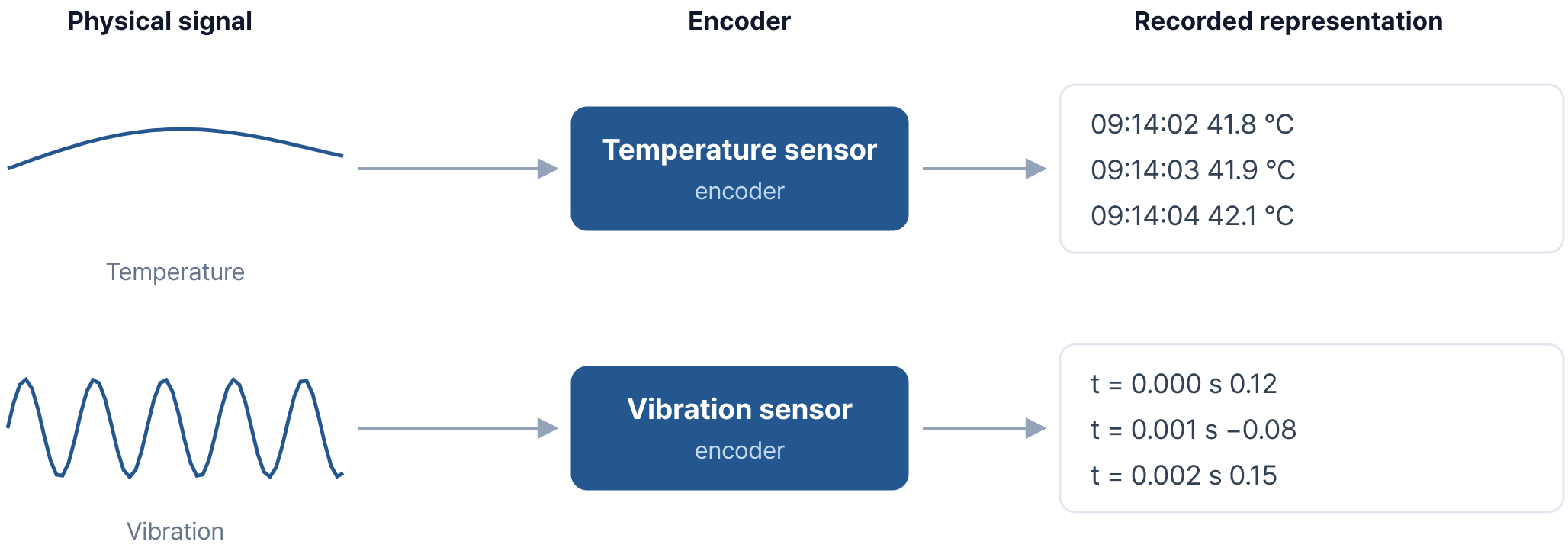
CRAWLER



A redesign breaks extraction without an HTTP error

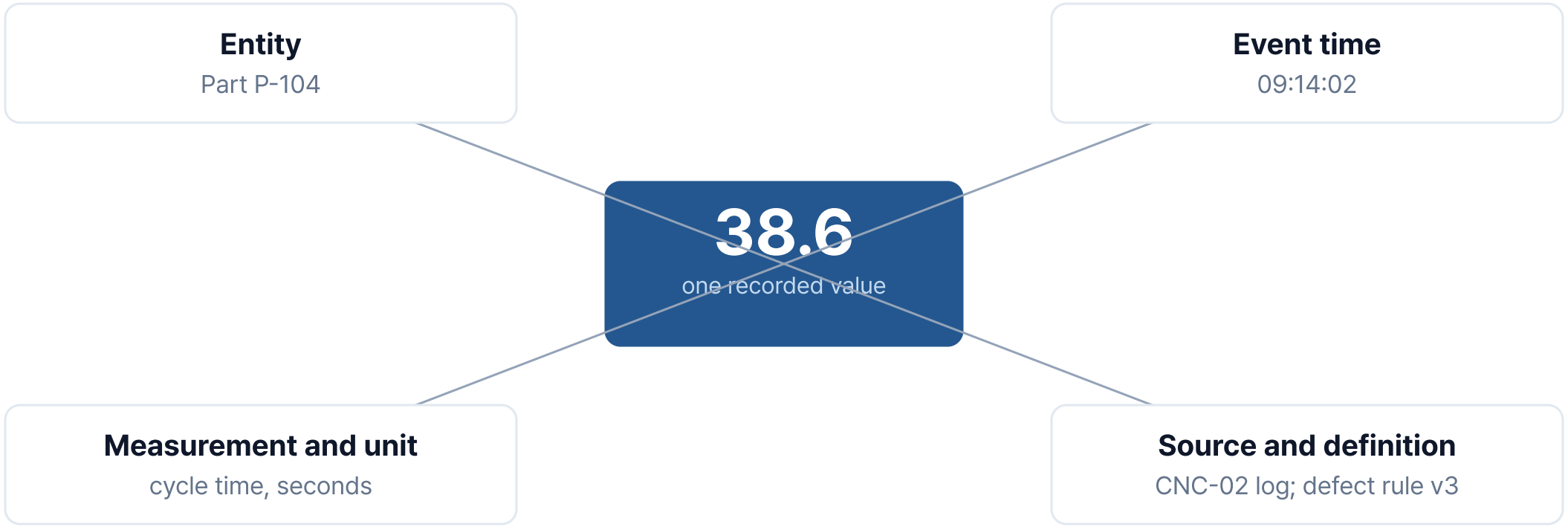
An explicit interface can be checked; page extraction also depends on layout

Sensors as Encoders



Measurement turns a physical event into a recorded observation

Record Meaning and Context

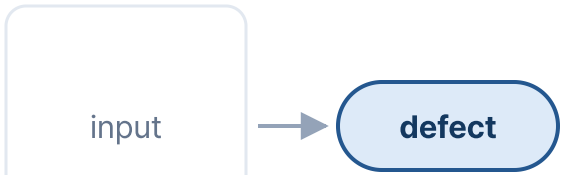


A value becomes usable when its meaning and context are recorded

Generating Training Examples

Annotation

add a label to an existing input

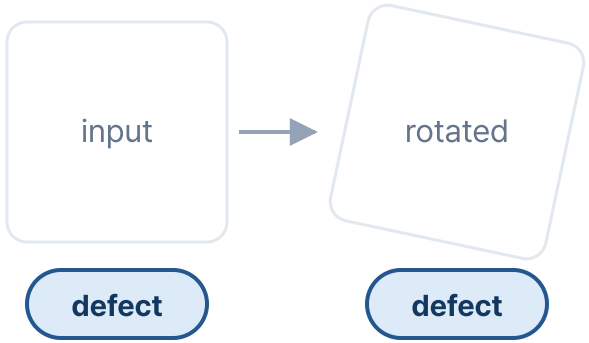


Check

Definition and label quality

Augmentation

change an input, keep its label

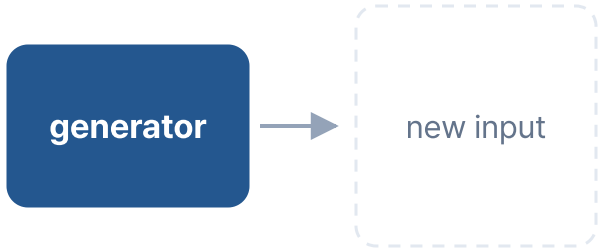


Check

Whether the label still applies

Synthesis

generate a new input



Check

Realism, coverage and usefulness

Generated inputs and candidate labels still require validation

Sources of Labels

A label can come from deliberate annotation or from an observed outcome

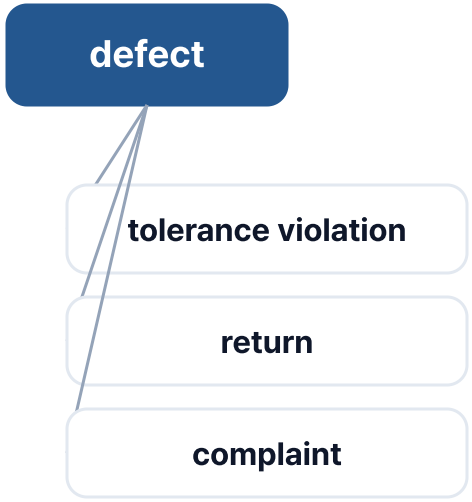
Source	Example	Main collection problem
Human annotation	An inspector judges a part	Make the guideline reproducible
Natural label	A delivered shipment has an arrival time	Record and link the outcome

Model-generated labels remain candidates until checked

Both sources need a definition and a quality check

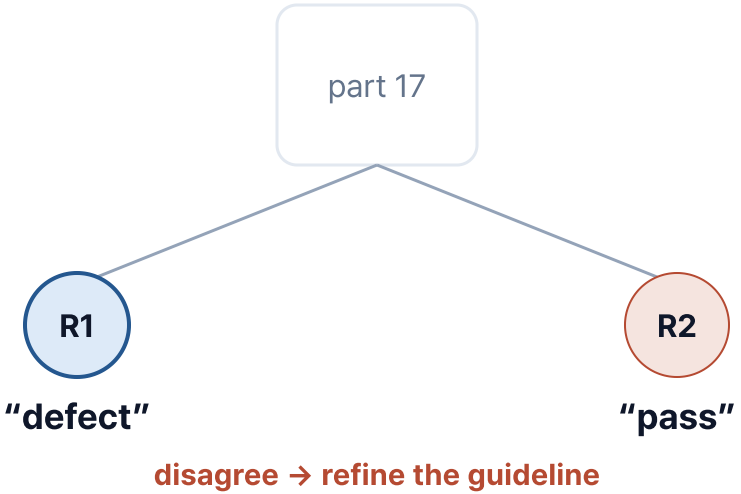
Checking Label Quality

Definition



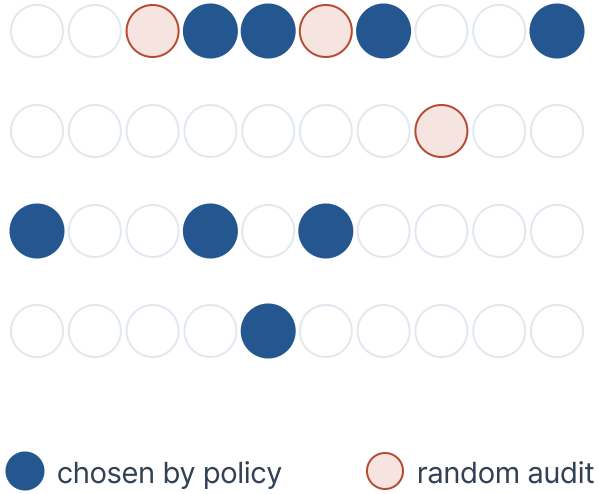
Which event is “defect”?

Agreement



Same guideline, same answer?

Independent audit

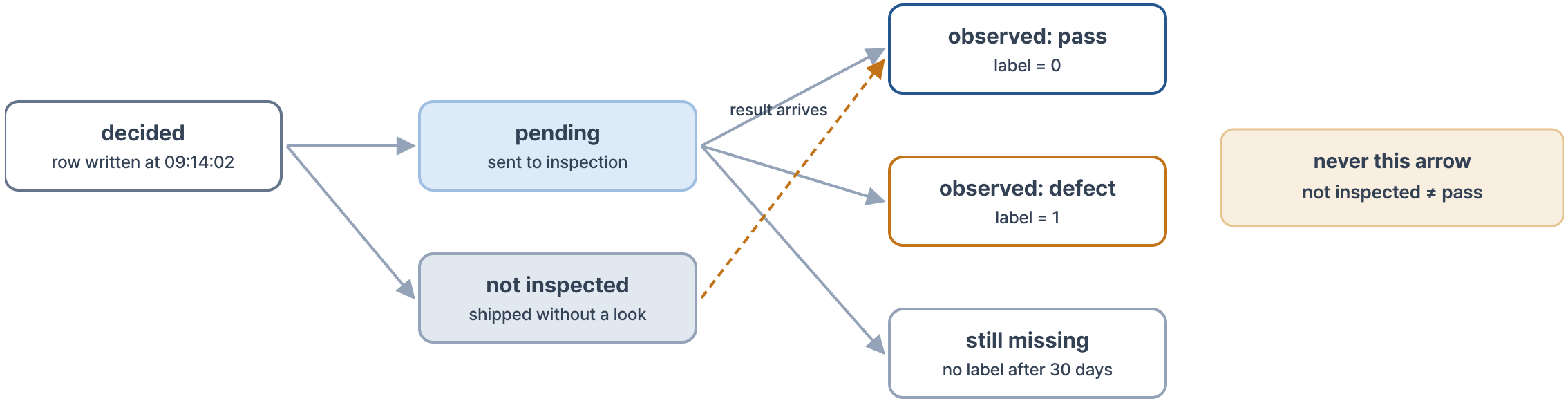


What does the selection miss?

A label needs a clear definition, reproducible judgment and independent checks

Missing Outcomes

Keep pending, unobserved and negative outcomes distinct



A model trained on "label or 0" learns that skipped parts are good

Not inspected is different from an observed pass

Selected Outcomes

The inspection policy controls which outcomes become labels

100 produced parts

20 selected for inspection

Labels will be observed

→

80 shipped without inspection

Outcomes unknown

They are not recorded as pass

→

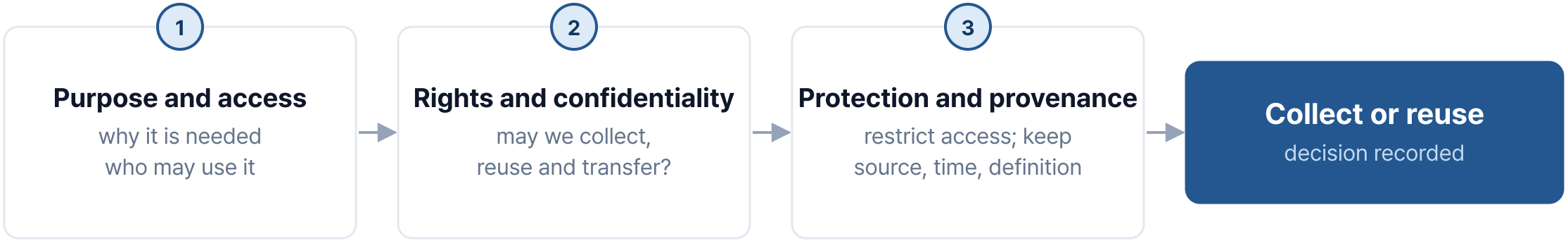
Random audit of 5

25 labels in total

75 outcomes remain unobserved

Audit outside the model-selected sample to observe what it missed

Collection Boundaries



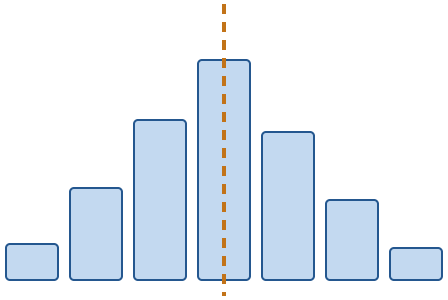
Each gate is a recorded decision before any row is collected

Collect only what the purpose needs, with permission and protection

Outline: Data Exploration

01

Descriptive Analysis



Reduce the data to a few numbers

02

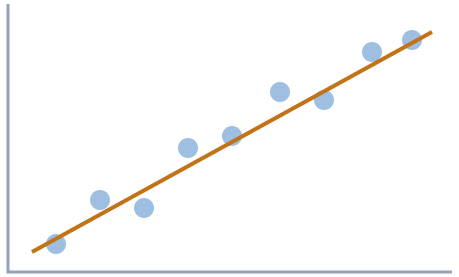
Visual Exploration



Find patterns by plotting the data

03

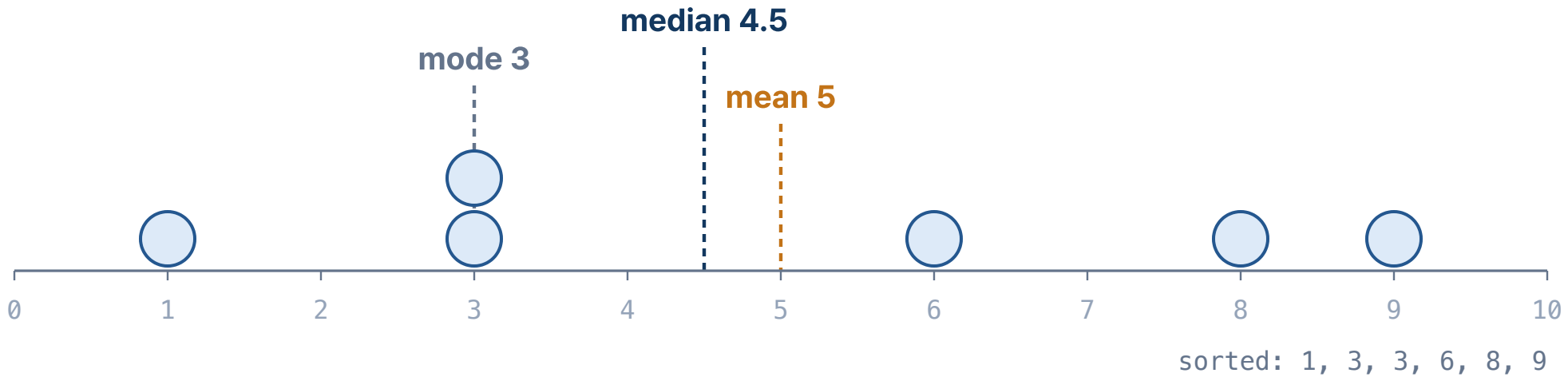
Statistical Exploration



Read relationships between variables

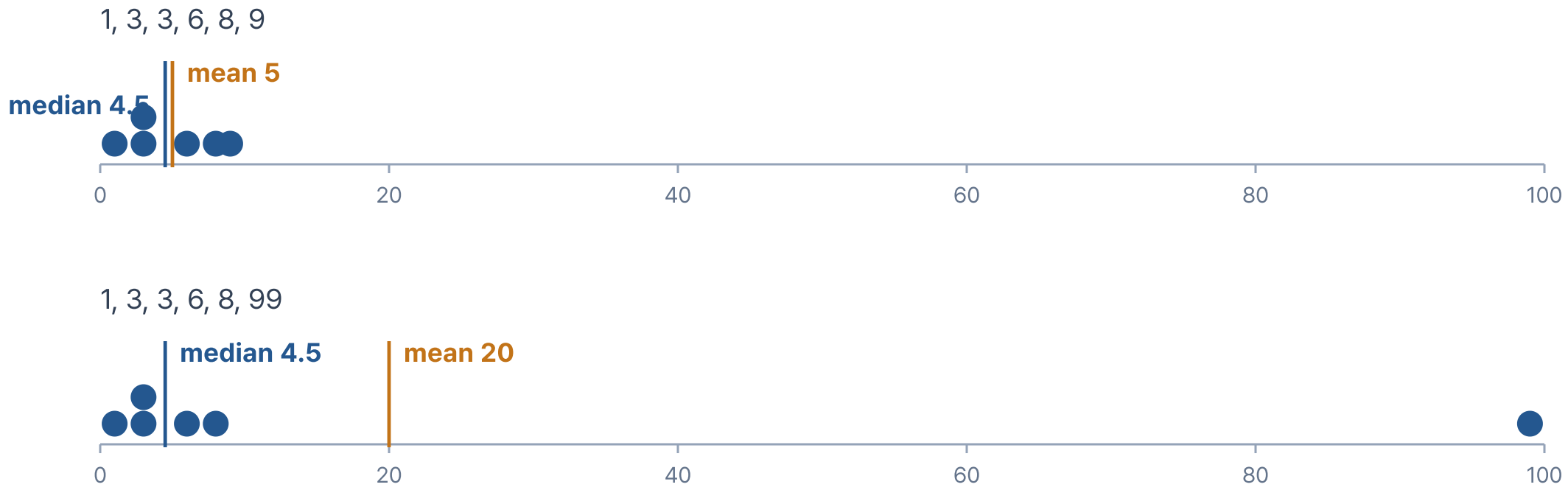
Recap: Mean, Median, Mode

Three answers to “the typical value”



From 9, 3, 1, 8, 3, 6: mean **5**, median **4.5**, mode **3**

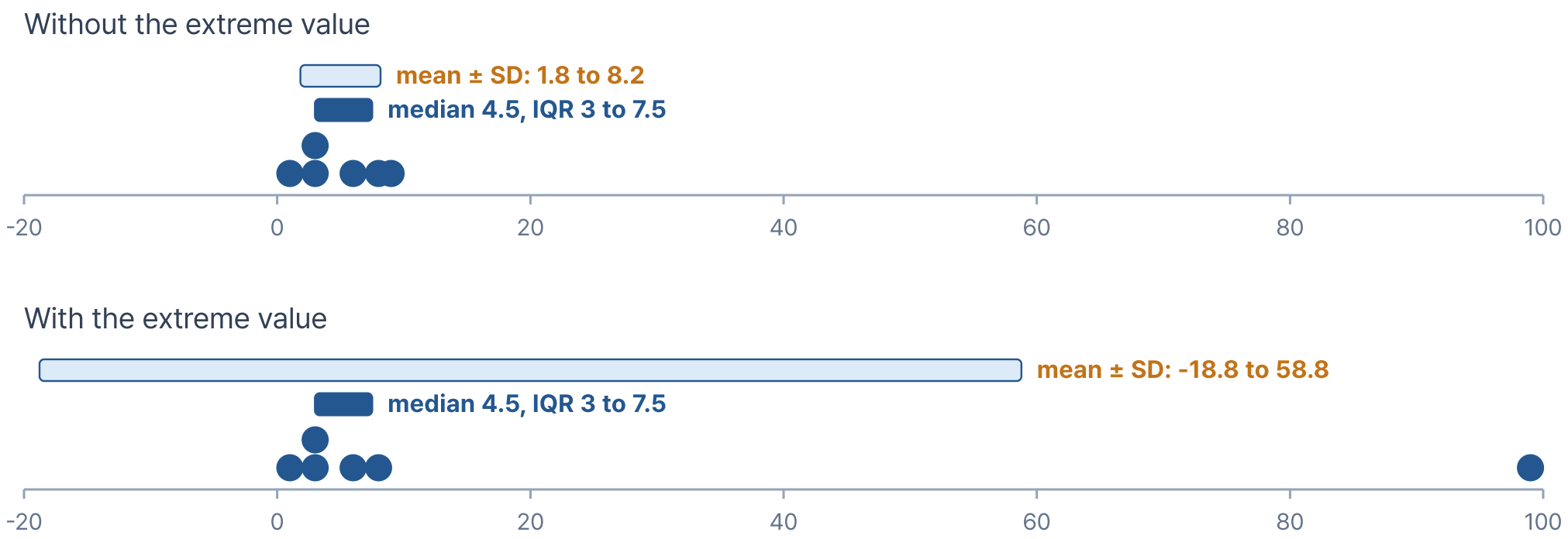
Extreme Values and the Median



The mean uses every value; the median uses the middle positions

An extreme value pulls the mean; the median is less sensitive

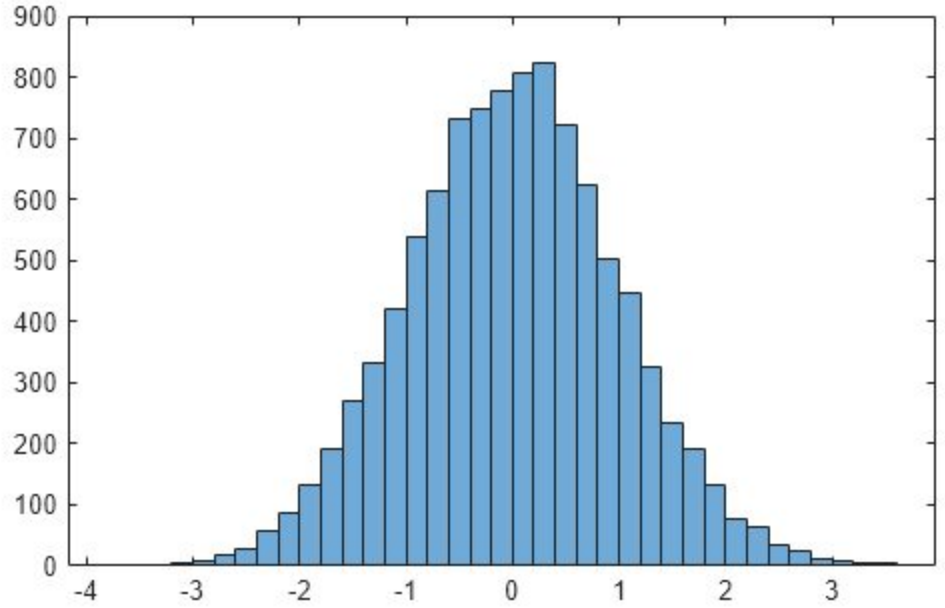
Variability and Robust Summaries



Report spread as well as center, and inspect the tails

Histogram as Empirical Distribution

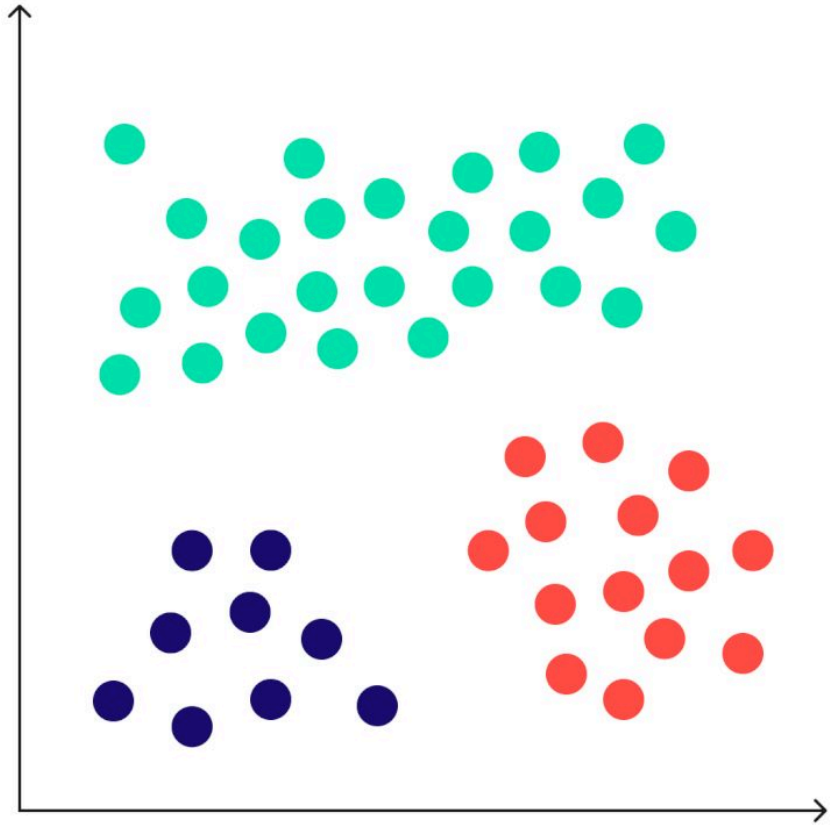
Bins summarize how many observations fall in each interval



Bin counts show distribution shape while hiding within-bin detail

Clustering with k-means

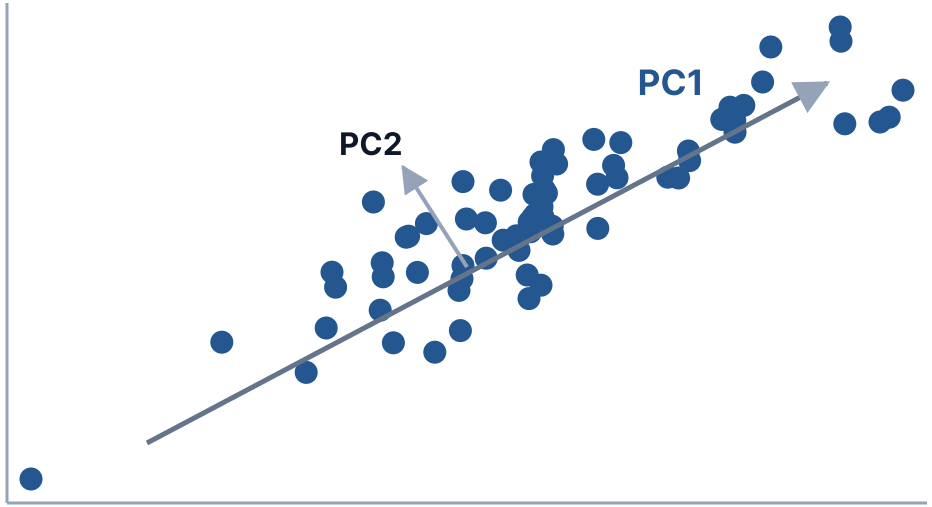
It always returns groups: the same points after asking k-means for three



Choosing k asks for that many clusters; it does not establish the true group count

PCA and t-SNE

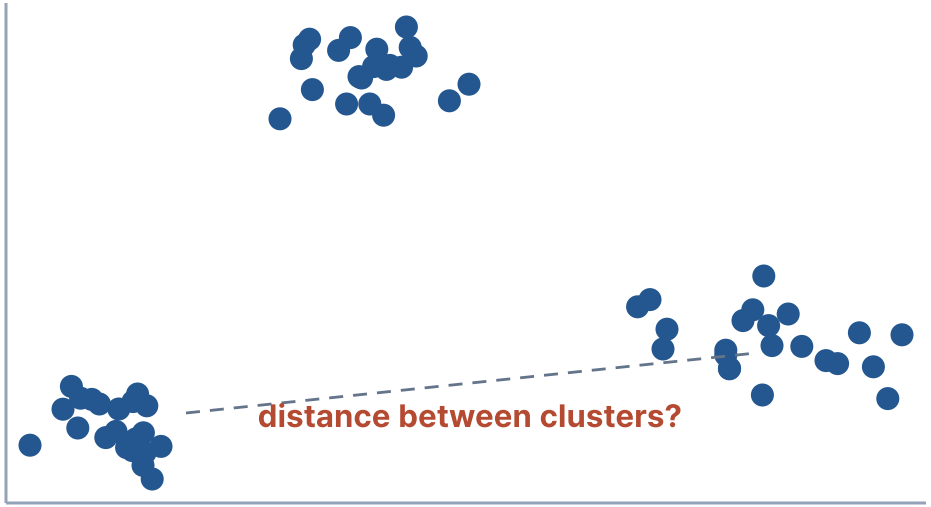
PCA



Shows: directions retaining large variance

Not: the most predictive or causal directions

t-SNE



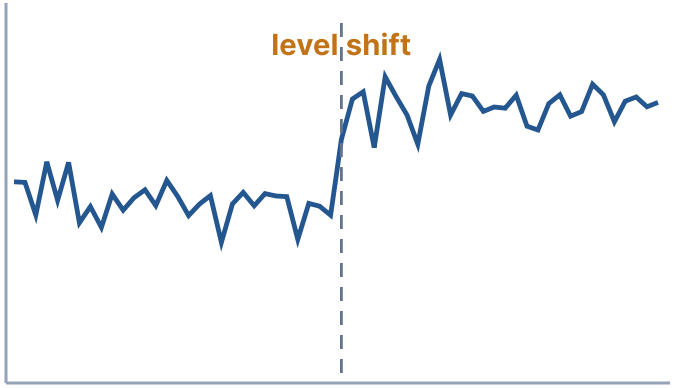
Shows: local neighborhoods

Not: reliable distances between clusters

A projection is a view of the data, with method-specific limits

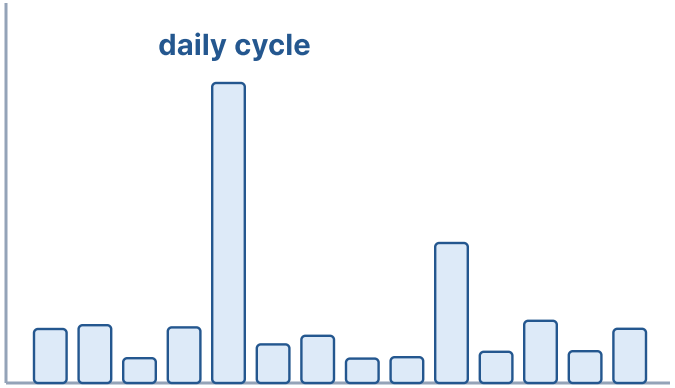
Time Series Views

Time domain



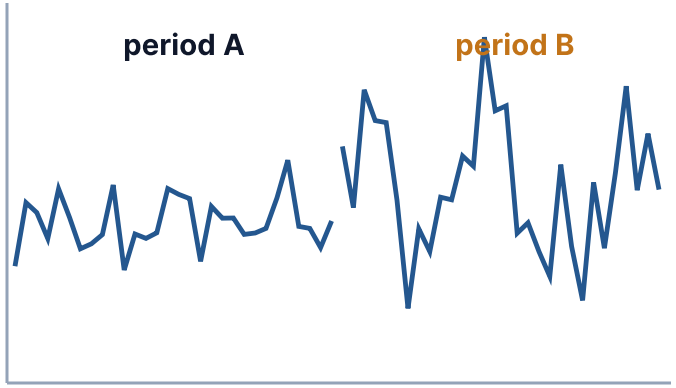
When did the level, trend or variation change?

Frequency domain



Which repeating cycles contribute?

Across time periods



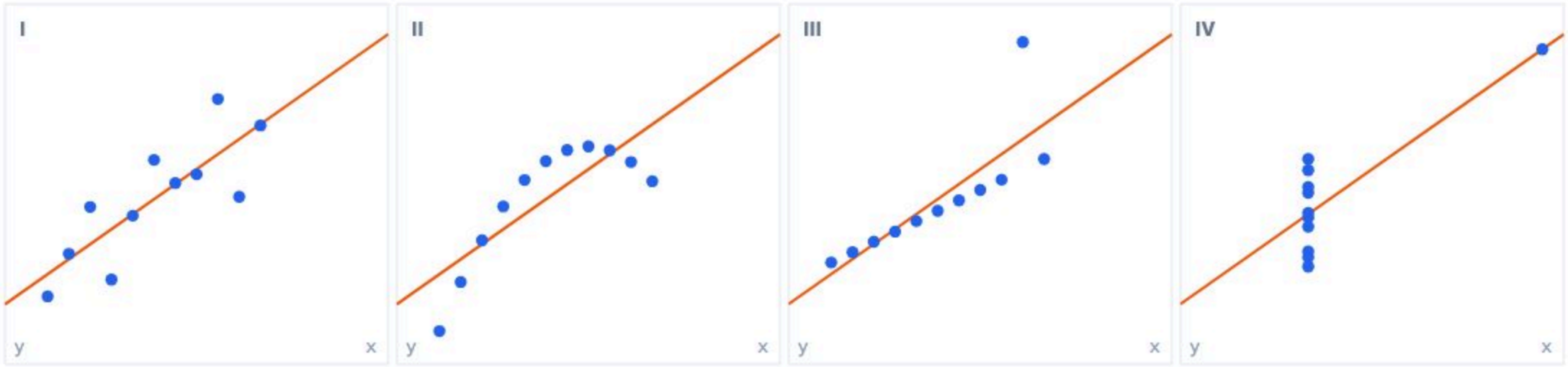
Is the process stable enough to compare?

Time plots compare values over time; scatter plots compare two variables

Preserve time order and check whether the process changes

Anscombe’s Quartet

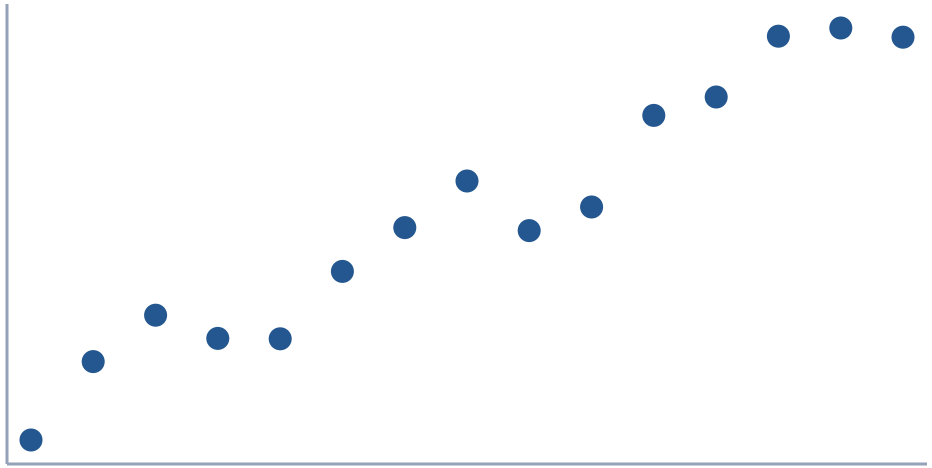
Similar summary statistics, different scatter plots



The same summaries can hide different relationships

Pearson and Spearman Correlation

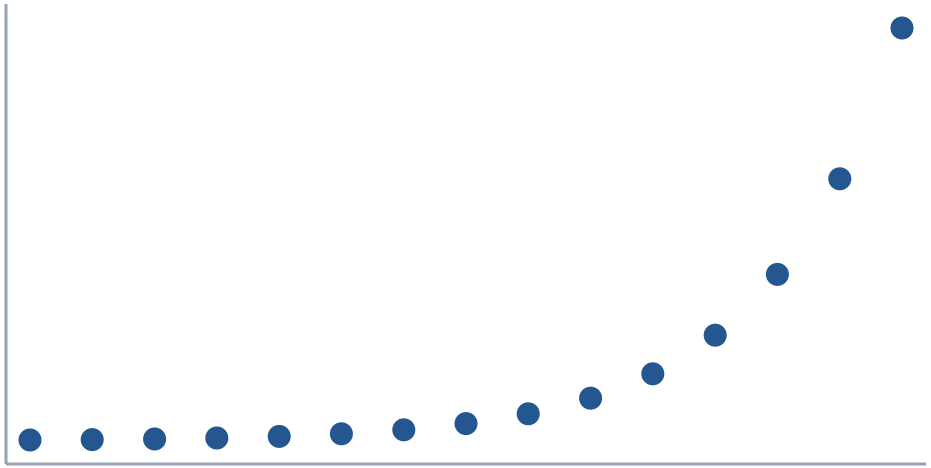
Linear relation



Pearson r = 0.97 **Spearman ρ = 0.96**

Pearson: linear association; a nonlinear relation can score lower

Monotone curve



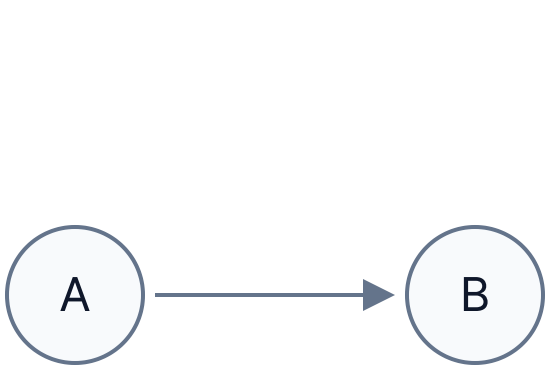
Pearson r = 0.80 **Spearman ρ = 1.00**

Spearman: rank association; neither one shows causation

Choose a relationship summary, then check the scatter plot

Causal Graphs

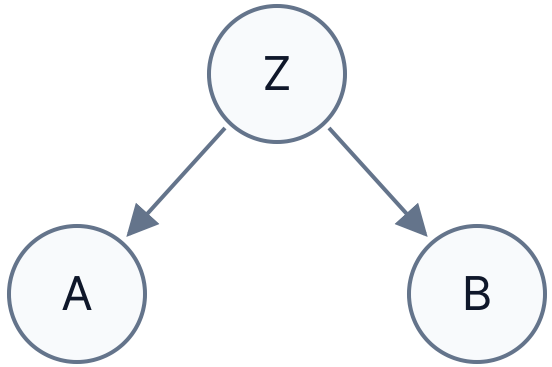
The same correlation can reflect either causal direction or a shared cause



A causes B



B causes A



Z causes both

Three structures give one correlation; **the correlation alone cannot tell them apart**

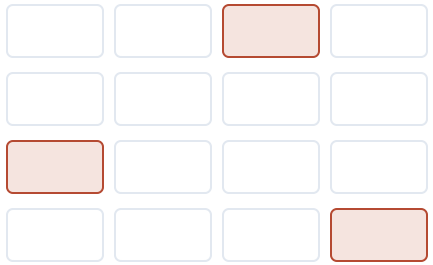
02 · Data Preparation

01	Collection and Exploration	Lec 2–3 · records, labels and patterns
02	**Data Preparation**	Lec 4–5 · changes to values and representation
03	SQL	Lec 6–8 · read, match and summarize rows
04	NoSQL	Lec 9–10 · models, documents and updates

Outline: Four Stages of Preprocessing

01

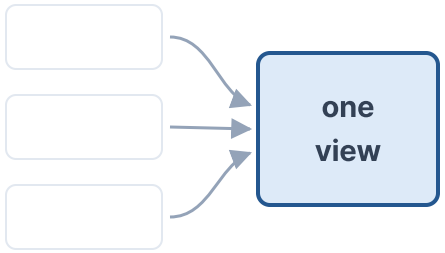
Cleaning



Wrong values in one table

02

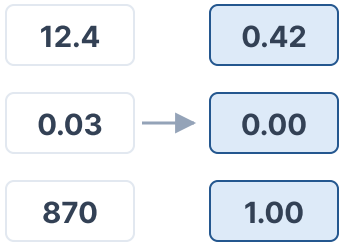
Integration



Tables that disagree

03

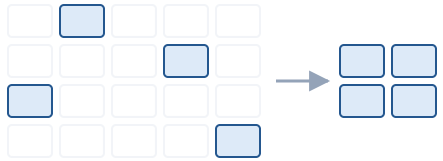
Transformation



Values a model can read

04

Reduction

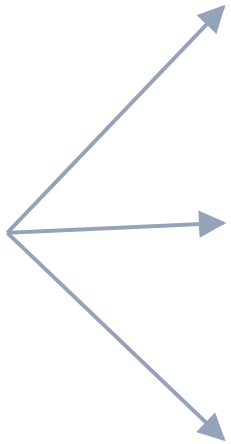


Less data, same signal

Data Imputation

Imputation estimates a missing value using observed data.

machine	temp	pressure	target
M1	71	2.1	20
M2	74	2.3	24
M3	73	2.2	?



Keep missing or remove the row

Does absence mean something? Which rows are lost?

Fill from a column summary

Is a typical value a useful estimate here?

Predict from related observations

Do temp and pressure carry information?

An imputed value is an estimate, not a new observation

Observed and Filled Values

Use other observed columns to find comparable rows

Observed measurements

part	helper	target
P1	10	20
P2	12	24
P3	11	NULL

Estimate for P3

Neighbors in helper: P1 and P2

(20 + 24) / 2 = 22

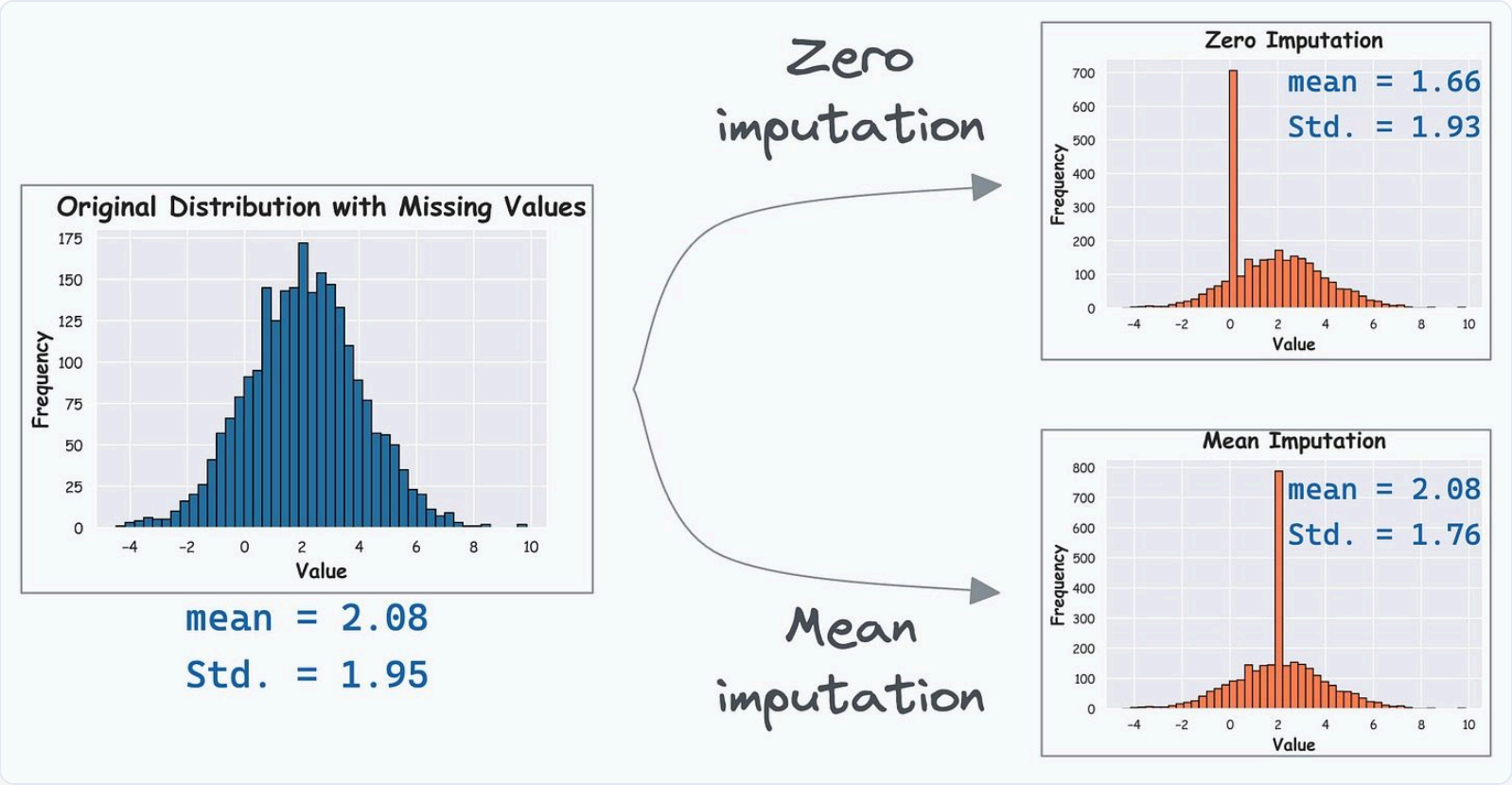
Stored with provenance

part	target	estimated
P3	22	Yes

The filled number is an estimate; keep its origin distinguishable

Caveat: Mean Imputation

Every hole gets the same value, and the spread of the column collapses



Mean stays 2.08; standard deviation falls to 1.76 and a spike appears

Caveat: Why is it missing?

Holes independent of the value

age	25	26	29	30	30	31	44	46	48	51	52	54
IQ	×	121	91	×	110	×	118	93	×	×	116	×

Holes concentrated in the younger half

age	25	26	29	30	30	31	44	46	48	51	52	54
IQ	×	×	×	×	×	×	118	93	116	141	97	104

Below, no young IQ was ever observed. **No imputer fixes this.** The question is not which method to use but why those rows are missing, and that answer lives in how the data was collected

Noise vs Outliers

Smooth measurement noise; investigate outliers before changing them

Noise: every value is a little wrong

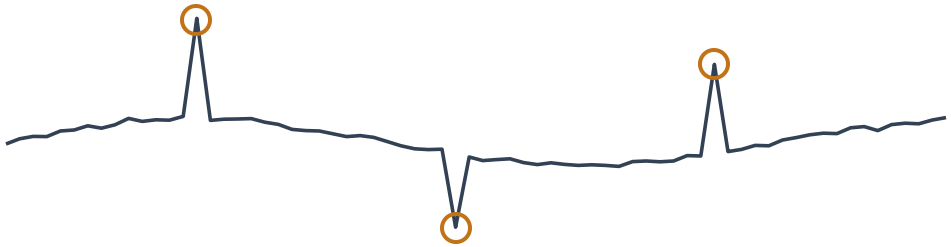


smooth

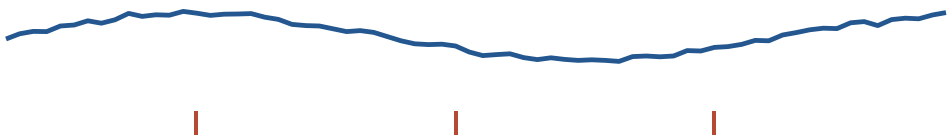


all 1,200 rows survive

Outliers: a few values sit far away



if confirmed: remove

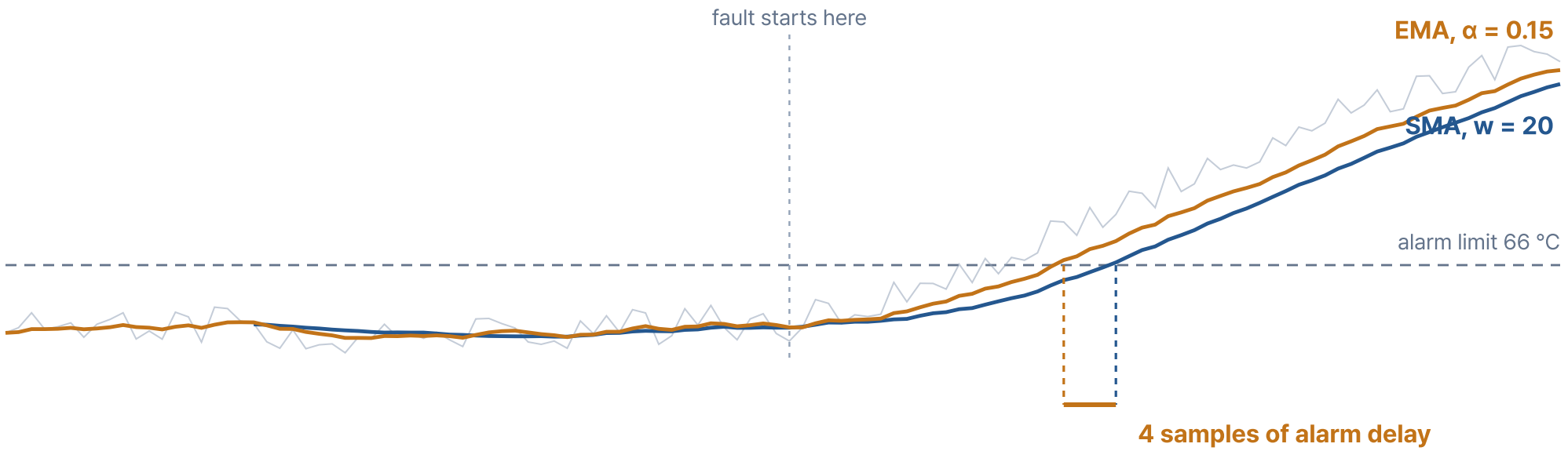


three rows are gone

Smoothing changes values; removal drops rows

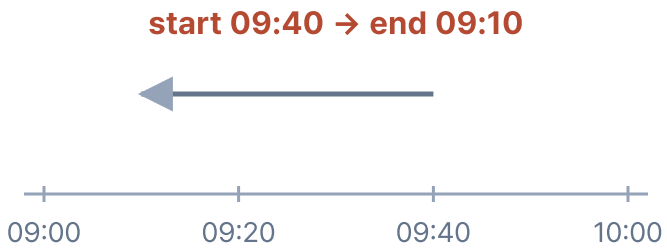
Moving Averages for Streams

Simple (SMA) and exponential (EMA) moving averages use only current and past samples



Both use past samples; here EMA crosses four samples before SMA

Consistency Checks



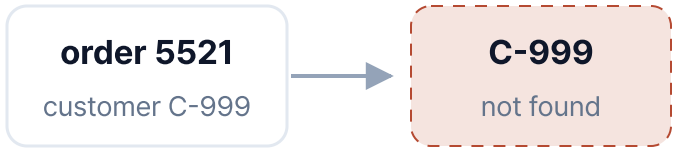
A time interval cannot end before it starts

needs investigation

cycle time (s)
38.6
41.2
0.64 min

A value must use the declared unit and type

conflicts with its definition



A reference must identify an existing entity

incomplete or wrong link

A failed check identifies a problem, not its correct replacement

One Customer, Three Systems



Different source IDs and spellings do not establish a shared identity

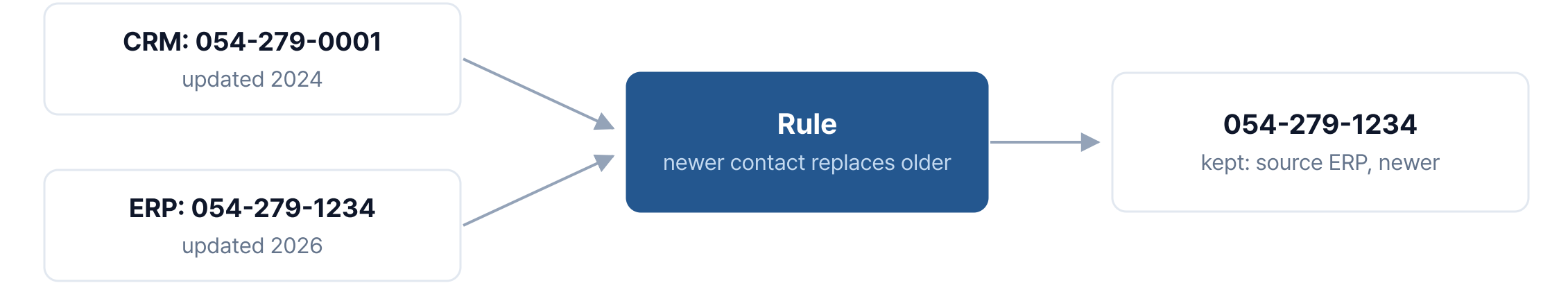
Three Questions of Integration

Schema mapping → **Entity matching** → **Value fusion**

Schema mapping	Entity matching	Value fusion
Which columns mean the same?	Which rows are the same customer?	Which value survives a conflict?
cust_name → customer	C-10482 → V0000734	054-279-0001 → 054-279-1234
Renaming columns enough?	Renaming columns enough?	Renaming columns enough?
Sometimes, if units and definitions match	**No: different IDs need evidence**	**No: choose a source or rule**

Matching columns and matching entities solve different problems

Resolving Conflicting Values



Trust a source for a field
Finance for recognized revenue

Compare event time and meaning
a newer current number

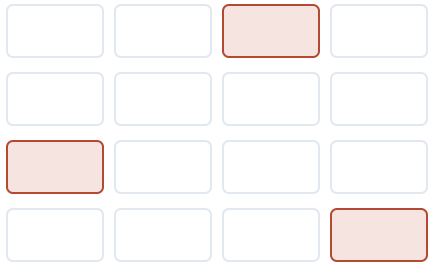
Keep the evidence
source IDs and the reason

Entity matching and value reconciliation are separate decisions

Recap: Four Stages of Preprocessing

01

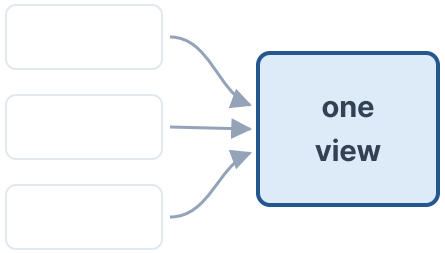
Cleaning



Wrong values in one table

02

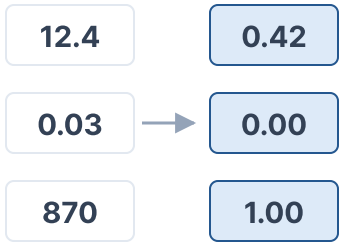
Integration



Tables that disagree

03

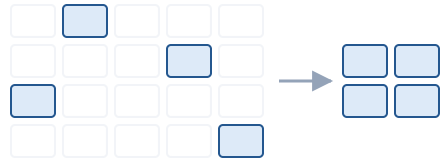
Transformation



Values a model can read

04

Reduction

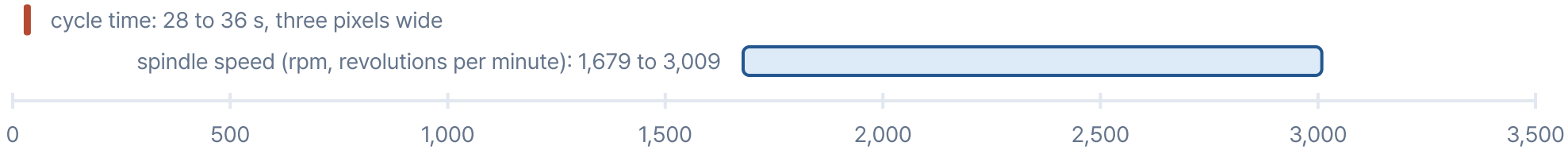


Less data, same signal

Why Scale at All

Distance-based models add up raw units, so the widest column decides

two columns from one machine, on one raw axis



how far apart two rows can be, column by column

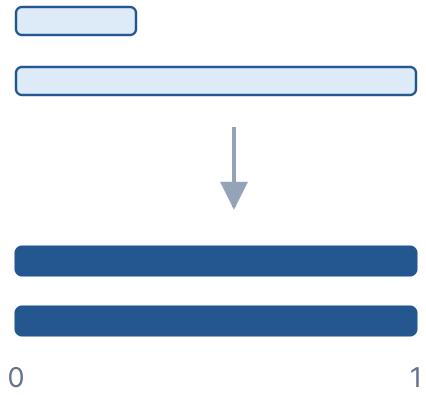


In raw Euclidean distance, the rpm difference **dominates** the cycle-time difference

Choosing a Numeric Transformation

Min-max / standardize

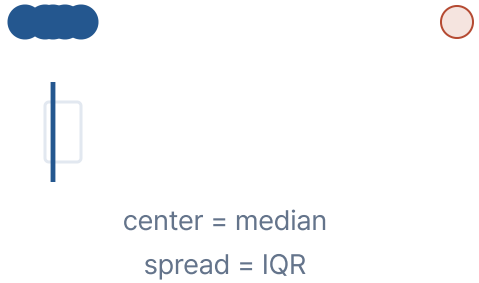
comparable scales



fit on training values

Robust scaling

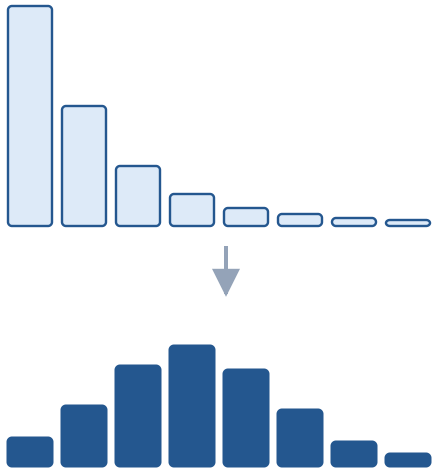
median and IQR



extreme values ignored

Log transform

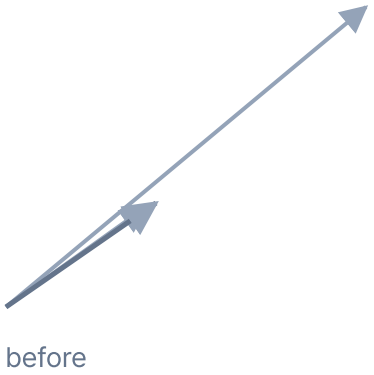
compress a long tail



positive values only

Unit-norm

direction, not size

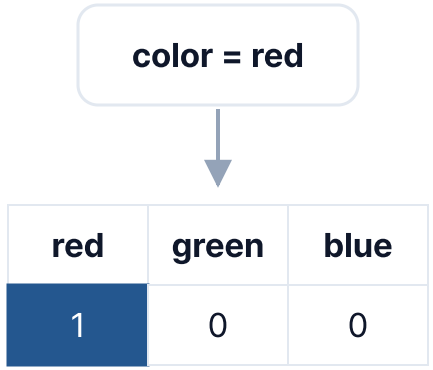


size may carry meaning

Choose a transformation for the comparison the model needs

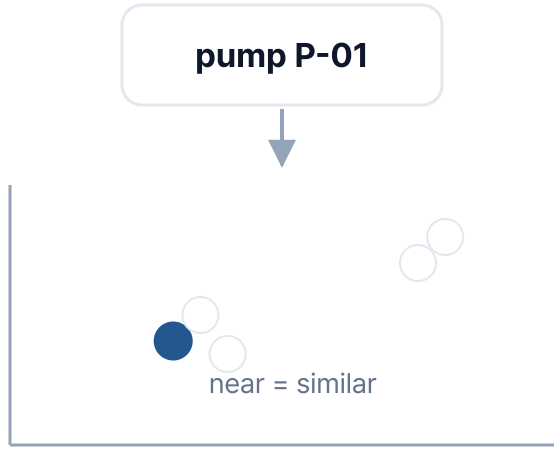
Encoding, Embeddings and Binning

Category encoding



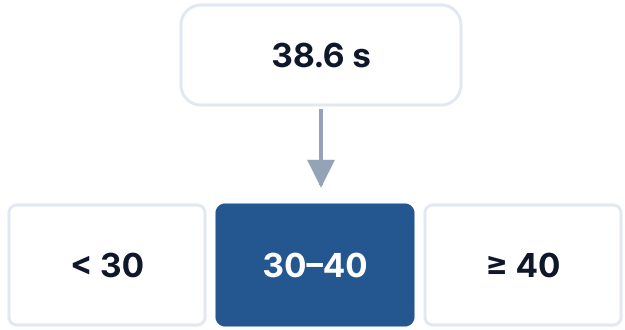
Preserve: membership and any intended order

Embedding



Preserve: what similarity it has learned

Binning



Preserve: thresholds; exact values are lost

Encoding changes representation; binning also loses within-bin detail

Nominal and Ordinal Categories

A numeric code can introduce order and spacing into the model

Nominal: no known order

Color	Arbitrary code
red	1
green	2
blue	3

Ordinal: a known order

Size	Ordered code
small	1
medium	2
large	3

Color codes invent an order; size codes express an existing order

One-Hot Encoding

size	small	medium	large
small	1	0	0
medium	0	1	0
large	0	0	1

Each column answers one question:
“is it this category?”

small < medium < large
not expressed by these columns

Ordered levels need ordinal encoding

One-hot represents membership; an ordinal encoding must preserve the intended order

Constructing Features

Written by a person
combine measured fields with domain knowledge

Learned from the data
extract a compact representation

Borrowed from a trained model
reuse an embedding; check its fit

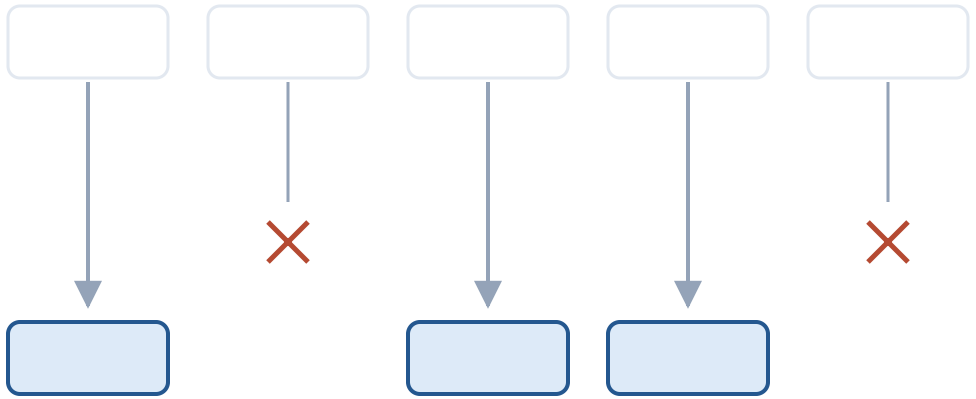


A useful feature represents information available when the decision is made

Extraction or Selection

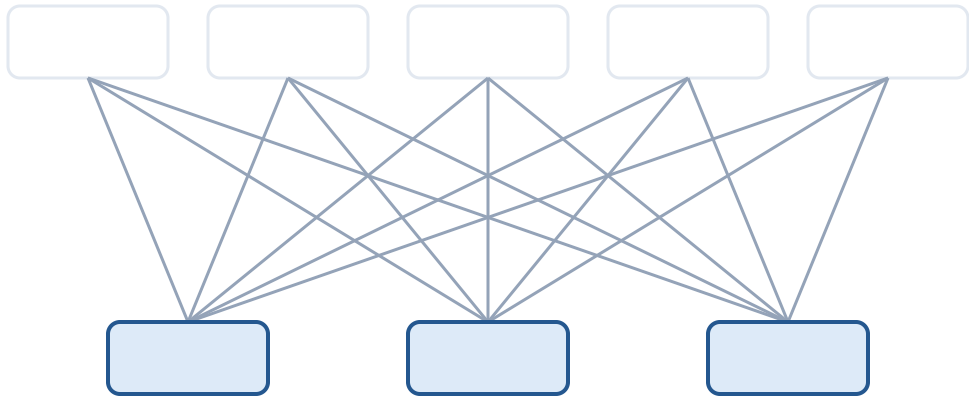
Keep some original columns, or mix all of them into new ones

SELECTION keep some of the original columns



5 columns in, 3 of the same columns out

EXTRACTION make new columns from all of them



5 columns in, 3 new columns out

Selection keeps original columns; extraction creates new combinations

Sampling vs Curation

Sampling

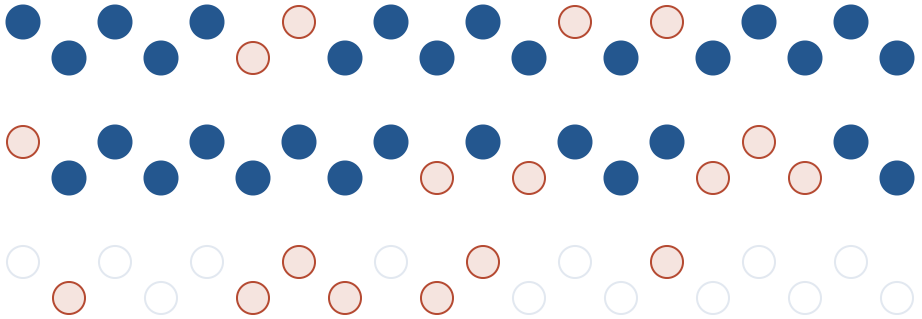
criterion: a sampling design



Inspect: inclusion probabilities and group weights

Curation

criterion: quality and coverage rules



Inspect: which groups and examples were removed

Curation example: low-quality rows dropped, and group 3 left out entirely

Both select rows; their criteria and resulting biases can differ

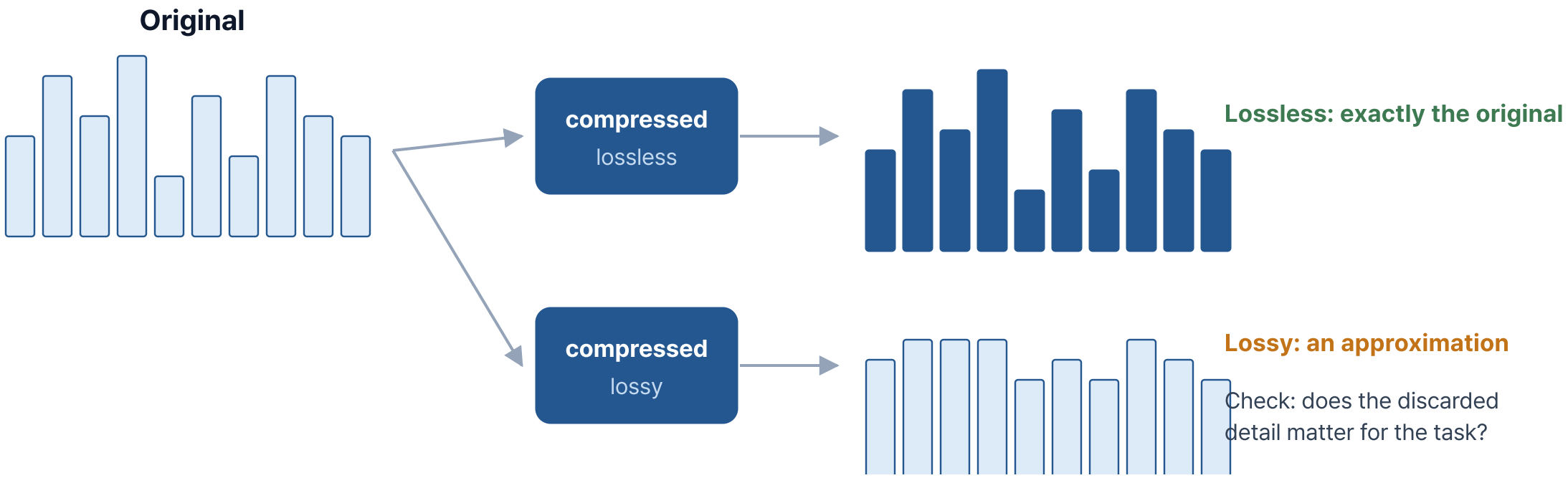
Sampling Design

The selection rule determines which parts of the population survive

Method	How rows are chosen	Failure to check
Simple random	Draw rows at random	Small groups may be missed
Systematic	Every kth record	A periodic list can align with k
Stratified	Randomly sample at least one per group	Keep the intended group proportions
Cluster	Select whole groups	One factory may not represent the rest

Check the population structure before choosing the sampling rule

Lossless or Lossy



A smaller file is useful only if the retained information still serves the task

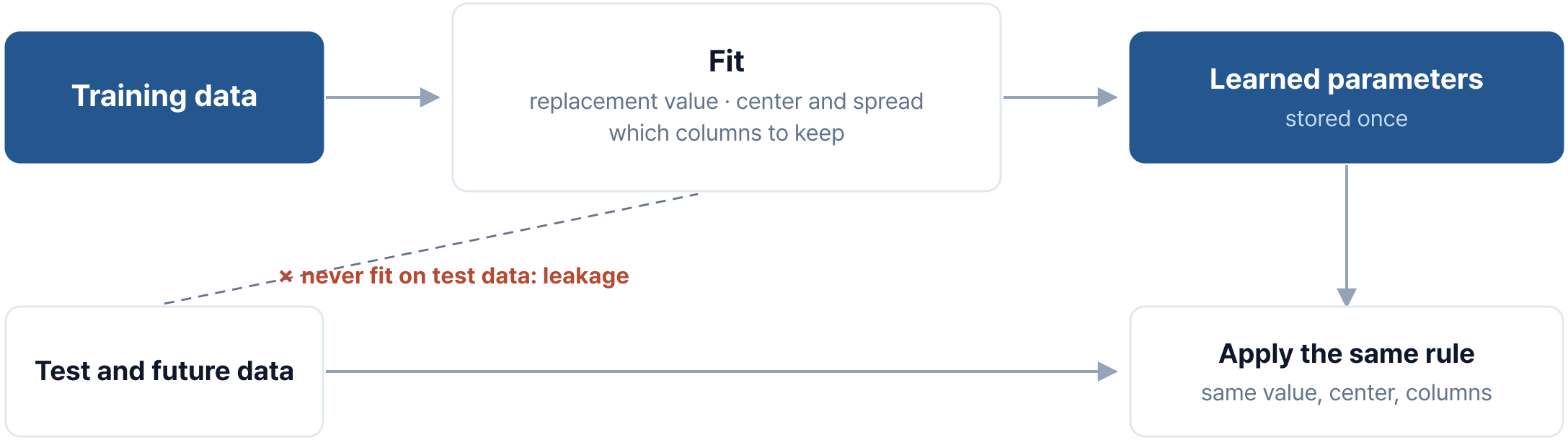
The Evaluation Must Match Deployment

Define what will be new at prediction time before splitting the records

Deployment question	Useful split	What a random row split can hide
Predict future events	Earlier train; later test	Future information reaches training
Predict new machines	Hold out whole machines	Same-machine patterns occur on both sides
Predict new customers	Hold out whole customers	Repeated identity makes the test easier

A held-out row can still share an entity or future information with training

Fitting Preprocessing Steps



Fit on training data, then apply the learned transformation consistently

03 · SQL

01	Collection and Exploration	Lec 2–3 · records, labels and patterns
02	Data Preparation	Lec 4–5 · changes to values and representation
03	**SQL**	Lec 6–8 · read, match and summarize rows
04	NoSQL	Lec 9–10 · models, documents and updates

Outline: SQL I

01	**Read a Table**	find rows and choose columns
02	**Refine a Result**	remove repeats, sort and keep top rows
03	**Handle Missing Values**	choose how absent prices affect the result
04	**Write Safely**	keep constraints and related changes together

Schema and Rows

A schema defines the columns; rows supply their values

Schema: the definition

```
Product
  pname          TEXT, primary key
  price          REAL
  category       TEXT
  manufacturer   TEXT
```

Rows: the stored values

pname	price	category
Gizmo	19.99	Gadgets
PowerGizmo	29.99	Gadgets
SingleTouch	149.99	Photography
MultiTouch	203.99	Household

One definition, many product rows

Keys and References

cname	stockprice	country
GizmoWorks	25.0	USA
Canon	65.0	Japan
Hitachi	15.0	Japan

pname	manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks
SingleTouch	Canon
MultiTouch	Hitachi

A primary key identifies a row; a foreign key requires an existing referenced key

Company facts are stored once; product rows refer to them by key

SELECT, FROM and WHERE

The question names one input table, a row condition and output columns

FROM

Read **Product**

→

WHERE

Keep **Gadgets**

→

SELECT

Show **name and price**

```sql
SELECT pname, price
FROM Product
WHERE category = 'Gadgets';
```

Read the table, keep matching rows, then choose columns

Refining a Query Result

Rows

product	price
bagel	1.50
banana	0.50
bagel	1.50
banana	1.00

Unique values

product	price
bagel	1.50
banana	0.50
banana	1.00

does each occurrence matter?

Ordered by price

product	price
bagel	1.50
banana	1.00
banana	0.50

the ranking the question needs

First 2 rows

product	price
bagel	1.50
banana	1.00

stable order before keeping

Rows, distinct values and a ranked list answer different questions

Missing Values in a Table

part	defects
P-1	0
P-2	NULL
P-3	3

recorded 0: known, equals zero

NULL: absent, unknown, not applicable



part	defects
P-1	0
P-2 ?	

Decide explicitly whether missing records belong

Missing is a state of knowledge, not an ordinary numeric value

Safe Changes to Related Facts

Keys and references

id	company
C1	Canon
C1	Hitachi

duplicate key rejected

reject duplicate IDs and broken links

A deliberate target set

item	status
journal	A
paper	D
postcard	A

preview: WHERE status = A

check which rows will change

A transaction

stock −1 for item 7

order 88 status = paid

commit both, or roll back both

all related changes, or none

Preserve constraints and change related facts as one unit

Outline: SQL II

01	**Match Rows**	return one row per matching pair
02	**Keep Unmatched Rows**	keep the left row when no match exists
03	**Test for Matches**	keep each outer row if a match exists
04	**Stack Results**	append compatible result rows

Matching a Product to Its Company

The shared company name lets one result contain product and country

Product

pname	manufacturer
Gizmo	GizmoWorks

Match the keys

```
manufacturer = cname
```

Company

cname	country
GizmoWorks	USA

Gizmo and USA meet through the same company key

JOIN on Related Keys

```
SELECT P.pname,P.price,C.country
FROM Product P JOIN Company C
  ON P.manufacturer=C.cname;
```

↑ match a product to its maker

pname	price	country
Gizmo	19.99	USA
PowerGizmo	29.99	USA
SingleTouch	149.99	Japan
MultiTouch	203.99	Japan

Each product matches its company; the result has four product rows

Pairs, Required Records and Match Existence



Choose the output entity before deciding how records are combined

Matching and Stacking Results

Match records: side by side

product	cid
Gizmo	C1
SingleTouch	C2

+

cid	country
C1	USA
C2	Japan

product	cid	country
Gizmo	C1	USA
SingleTouch	C2	Japan

keys must refer to the same kind of entity

Stack records: one under another

city
Boston
Austin

+

city
Austin
Dallas

repeated row: keep or drop?

columns must have compatible meanings and types

Matching connects related facts; stacking combines compatible lists

Outline: SQL III

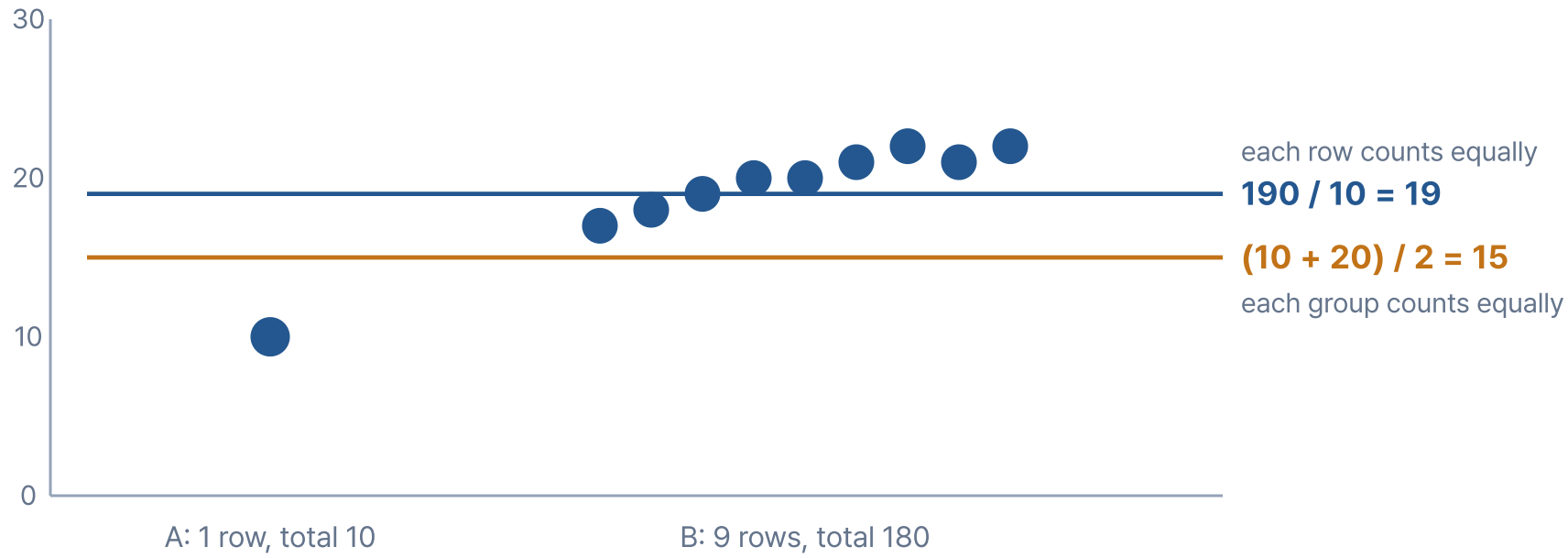
01	**Calculate Measures**	calculate a quantity from detail rows
02	**Summarize Groups**	choose the key and keep one row per group
03	**Reuse Summaries**	use a computed result as another input
04	**Keep Detail Rows**	add group context without merging rows

Aggregate Functions

Function	Question
COUNT(*)	How many purchase rows?
SUM(quantity)	How many units?
SUM(price*quantity)	How much revenue?
AVG(price)	What is the average recorded unit price?
MIN(price), MAX(price)	What are the smallest and largest prices?

Choose the expression that measures the quantity named in the question

Quantities and Weights



The denominator defines which units receive equal weight

Analytical Result Shapes

Detail input

product	price
bagel	1.0
bagel	1.5
banana	0.5
banana	1.0

GROUP BY

product	mean
bagel	1.25
banana	0.75

Grouping turns detail records into one summary row per group

Grouping and Average

One row per product group

```
SELECT product, AVG(price) AS mean
FROM Purchase
GROUP BY product;
```

Result

product	mean
bagel	1.25
banana	0.75

The grouping key defines each result row; AVG defines its measure

A Query Produces a Table

Purchase rows

product	price
bagel	1.0
bagel	1.5
banana	0.5
banana	1.0

Query 1: mean by product

product	mean
bagel	1.25
banana	0.75

Query 2: keep mean ≥ 1.0

product	mean
bagel	1.25

The output of one query is a table, so it can be the input of the next

A query returns a table that another query can use as input

Detail, Summary and Group Context

Detail

product	price
bagel	1.0
bagel	1.5
banana	0.5
banana	1.0

one purchase · 4 rows

Grouped summary

product	mean
bagel	1.25
banana	0.75

one product group · 2 rows

Detail with group context

product	price	mean
bagel	1.0	1.25
bagel	1.5	1.25
banana	0.5	0.75
banana	1.0	0.75

purchase + its product average · 4 rows

Group information can summarize rows or accompany the original detail rows

04 · NoSQL

01	Collection and Exploration	Lec 2–3 · records, labels and patterns
02	Data Preparation	Lec 4–5 · changes to values and representation
03	SQL	Lec 6–8 · read, match and summarize rows
04	**NoSQL**	Lec 9–10 · models, documents and updates

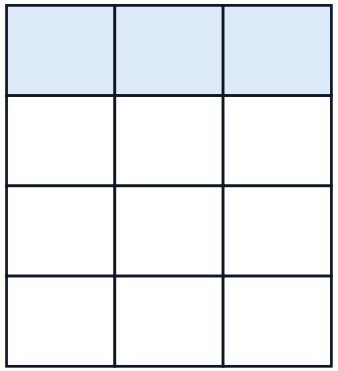
Outline: NoSQL I

01	**Four Families of NoSQL**	choose a model by its access pattern
02	**JSON and Semi-Structured Data**	read named fields and ordered arrays
03	**Between Trees and Tables**	convert nesting and repeated values
04	**Why NoSQL**	check flexibility, scale and guarantees

Four NoSQL Families

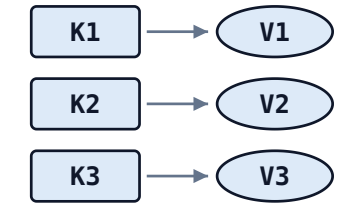
The four NoSQL families

SQL

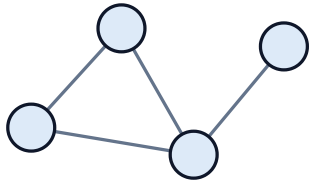


Relational

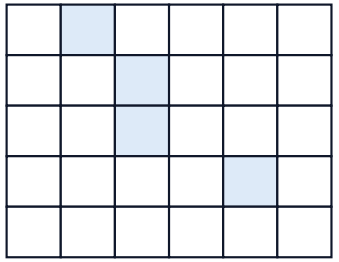
NoSQL



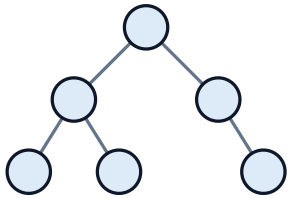
Key-value



Graph



Wide-column



Document

NoSQL is not one thing: four alternative **storage models**

Key-Value Records

Request

```
get("user:1")
```



Stored pair

```
key: "user:1"
value: {
  name: "John",
  age: 28
}
```



Answer

```
{ name: "John", age:
28 }
```

A key retrieves its associated value

Sparse Cells Under a Row Key

In a Bigtable-style model, rows can hold different column qualifiers

user:1

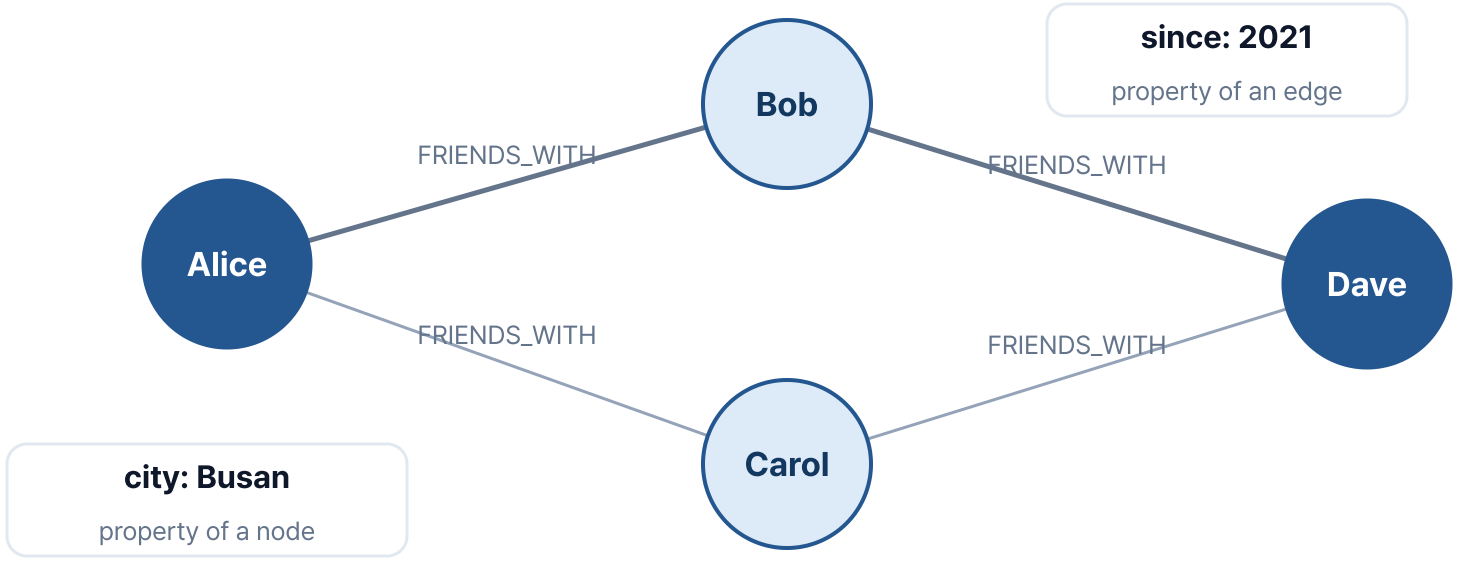
Family: activity	Value
page:home	09:01
page:product	09:03
purchase:1	Laptop

user:2

Family: activity	Value
page:home	09:02
purchase:2	TV

Same column family, different populated cells

Property Graphs



Node: an entity

Edge: a relationship

Properties: attributes

Friends of friends:
follow edges, Alice → Bob → Dave

Graph models make relationships and paths explicit

Evolving Document Shapes

One collection

```
{"name":"John", "age":28}
{"name":"Sue", "phone":"555-0134"}
```

Different optional fields

John has an age field.

Sue has a phone field.

Both documents can belong to the same collection.

Optional fields may differ; shared fields still need consistent meanings

Semi-Structured Data

Relational: schema declared once

id	name	city
1	John	Boston
2	Sue	Austin

order	cust_id
101	1

related facts linked by keys

Document: structure travels with values

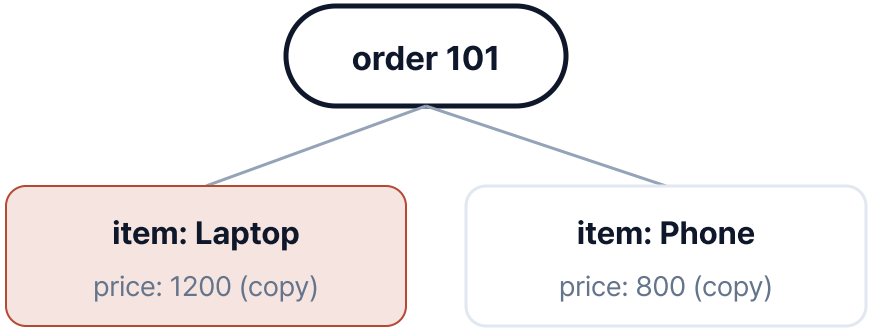
```
{ "id": 1, "name": "John",
"address": { "city": "Boston" },
"orders": [
{ "order": 101, "item": "Laptop" },
{ "order": 104, "item": "Phone" } ] }
```

nested objects and arrays; field names inside

Semi-structured data carries structure with the values

Trees and Tables

Nested document



read together in one fetch

copies need coordinated updates

Related tables

order	pid
101	P1
101	P2

pid	price
P1	1200
P2	800

each fact stored once

queries reconnect them by key

Choose nesting for the access pattern and account for repeated facts

Caveat: Stale Copies

Two documents cache the current catalog price, but only one copy receives the update

John's Document

```
{"name": "John",
 "Orders": [
   {"date": 2004,
    "product": {"prodName": "Gadget",
                "currentCatalogPrice": 12.99}}
 ]}
```

Sue's Document

```
{"name": "Sue",
 "Orders": [
   {"date": 2002,
    "product": {"prodName": "Gadget",
                "currentCatalogPrice": 9.99}}
 ]}
```

Remedy: reference the current price or maintain every cached copy

One product, **two prices**: copies need an owner who updates them all

Partitioning and Replication

Splitting records and copying records solve different problems

Partition: different records

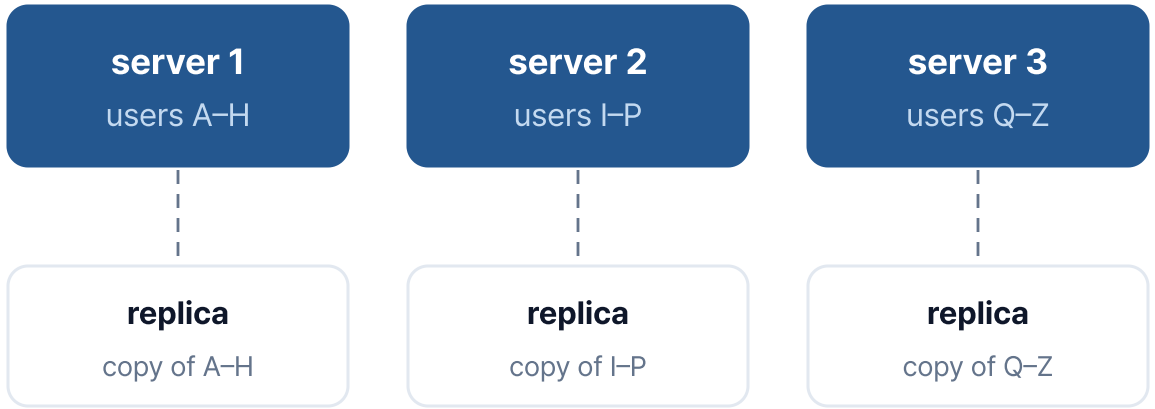
```
Server A: users 1–100
Server B: users 101–200
```

Replicate: copies of records

```
Leader: user 1, value 2
Follower: user 1, value 1
          update pending
```

Partitions divide the data; replicas copy it and must coordinate updates

Scaling and Operation Guarantees



partitions split rows; replicas copy them

a replica can lag behind

Capacity

bigger machine, partitions, or both

Reading updates

can a read hit a lagging replica?

Atomic changes

one record or several related ones

Failures

which reads and writes stay available

Choose guarantees for the operation, not from the database label alone

Outline: NoSQL II

01	**Documents and Queries**	read fields, nesting and paths
02	**Finding Documents**	choose documents and output fields
03	**Summarizing Documents**	transform documents through ordered stages
04	**Updating Documents**	select write targets and verify changes

Collection and Document

Relational terms

Table	Row	Column
Product	one product	price

Document terms

Collection	Document	Field
inventory	one item	qty

A collection holds documents; each document has named fields

Fields, Objects and Arrays

One inventory document

```
{
  item: "paper",
  qty: 100,
  size: {h: 8.5, w: 11, uom: "in"},
  tags: ["red", "blank", "plain"],
  instock: [
    {loc: "A", qty: 60},
    {loc: "B", qty: 15}
  ]
}
```

Levels inside it

Field	Value
qty	number
size	nested object
tags	array of strings
instock	array of objects

A field can hold one value, a nested object or an array

Finding Documents

One inventory record

```
{item:"paper", qty:100, status:"D"}
```

Two choices

Choice	Purpose
Filter: status D	Keep matching documents
Fields: item and qty	Choose what the result shows

Choose documents and output fields before summarizing them

Documents and Nested Entries

One item document

```
item: "paper"
instock: [
  {loc:"A", qty:60},
  {loc:"B", qty:15}
]
```

Choose the relevant level

Question	Unit to inspect
Does one location satisfy both conditions?	The same stock entry
How much stock is recorded by location?	Each location entry, not each item document

Keep related conditions and measures at the same record level

A Pipeline Changes Its Input

Input documents

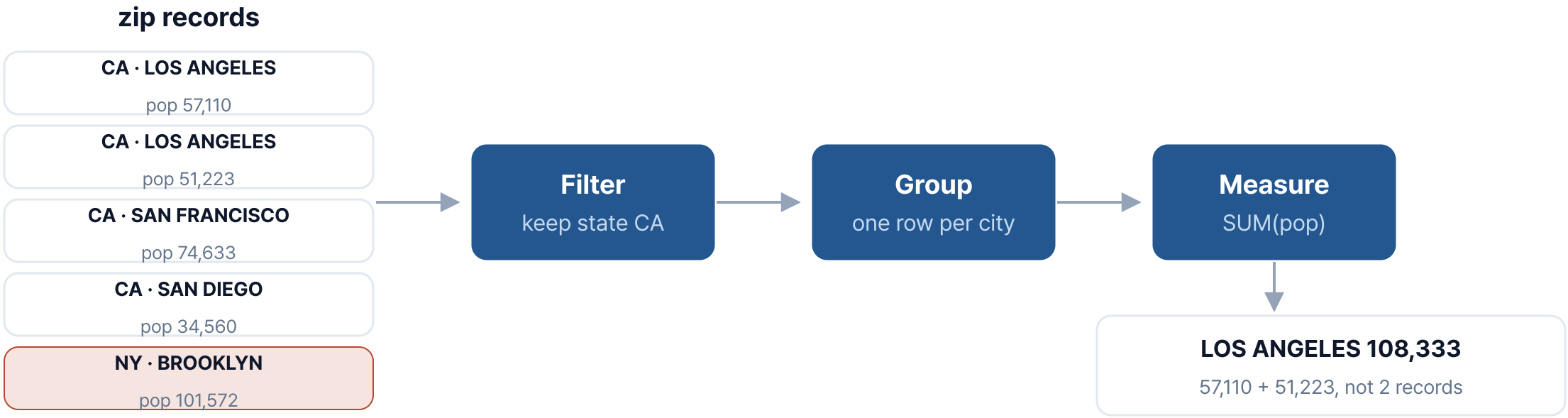
```
{state:"CA", pop:10}
{state:"CA", pop:20}
{state:"NY", pop:40}
```

Stages and outputs

Stage	Output
Keep pop ≥ 20	CA:20 · NY:40
Group by state; sum pop	CA total:20 · NY total:40

Each stage passes its output documents to the next stage

Filter, Group and Measure



Choose the filter, grouping key and measure independently

Safe Document Updates

Preview the selection

item	status
journal	A
notebook	A
paper	D
planner	D
postcard	A

status A: three items

Change the intended field

item	status
journal	Review
notebook	Review
paper	D
planner	D
postcard	Review

set status to Review

Verify

item	status
journal	Review
notebook	Review
paper	D
planner	D
postcard	Review

three changed; paper and planner stay D

Atomic for one document; a multi-document change is not automatically atomic

Preview the targets, apply the change, and verify its scope

Takeaway: Keep the Meaning Through Each Step

- Define what a record measures and which outcomes were observed
- Check what preparation changes, estimates or removes
- Choose a representation and a result that answer the question