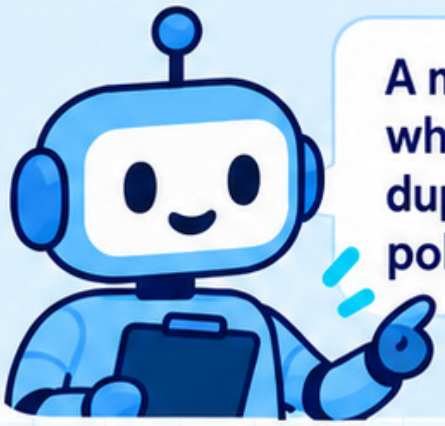
IS THE DATA FIT FOR THIS JOB?

Checking data quality.



A model cannot tell you whether a missing customer, duplicated event or outdated policy was intentional.

Today: create a small data contract, test it, and decide what happens when a check fails.

Illustrative flow

Raw data (from systems)

- Duplicated event
- Missing timestamp
- Outdated policy

Data quality checks

- Check for duplicates
- Check for required fields
- Check applicable policy

Checked data

Passed declared checks

Model

These items are stopped and handled

- Duplicated event
- Missing timestamp
- Outdated policy

A DATA CONTRACT MAKES ASSUMPTIONS EXPLICIT

For our purchase events, define the expected columns, types, identifiers and meaning.

Fictional purchase event

event_id:	event identifier
customer_id:	customer identifier
event_time:	when it happened

Define contract

Data contract for purchase events

Field	Type	Meaning
event_id	string	identifies one event
customer_id	string	links it to a customer
event_time	timestamp	records when it happened

Example: event_id identifies one event; customer_id links it to a customer; event_time records when it happened.

A contract turns “the data looks strange” into a specific check that someone can investigate and own.

CHECK COLUMNS AND TYPES

Expected: event_id is text, *amount* is numeric, event_time is a timestamp.

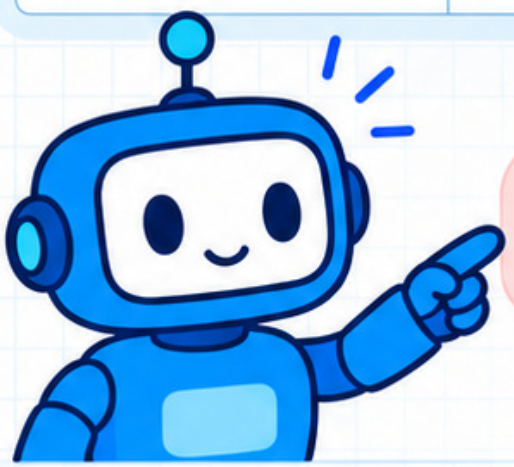
Illustrative schema mismatch

Expected Schema

column	required type
event_id	text
amount	numeric
event_time	timestamp

Received Schema

column	required type	received value
event_id	text	unchanged
total	numeric	**twenty**
event_time	timestamp	unchanged



Column renamed
amount → total

Invalid numeric value
expected a number, got “twenty”

A **batch renames** *amount* to total or sends “twenty” as a value.

Detect the mismatch before training or serving consumes it.
Convert data only through an explicit, tested rule; silent conversion can hide upstream changes.

VALIDATE VALUES AGAINST THEIR MEANING

Toy rule: purchase amounts must be nonnegative.

PURCHASE-EVENT CONTRACT
(Toy example)

Purchases

Allowed amount range:

amounts must be nonnegative.

-5 ✗ → An amount of -5 fails this purchase-event contract.

REFUND-LEDGER CONTRACT
(Toy example)

Refunds

Allowed amount range:

Negative amounts may be valid.

-5 → It might be valid in a separate refund ledger.

Define allowed categories and ranges for the particular dataset. A rule borrowed from the wrong business object can reject correct data.

MISSING IS DIFFERENT FROM ZERO

Toy example

amount
NULL

A missing amount means we do not have the value.

Application response

Decide how the application responds when the value is missing.

Toy example

amount
0

An amount of zero is a recorded value.

Application response

Decide how the application responds when the value is zero.

Decide which fields may be missing, what absence means, and how the application responds. Do not fill every blank with zero simply to make a model or validation step accept the row.

DEFINE WHAT A DUPLICATE MEANS

Two identical event_id values can indicate a repeated delivery.

Toy example

event_id	customer_id
E1	C1
E1	C1

Same event ID / repeated delivery

Two purchases by the same customer can be legitimate.

Toy example

event_id	customer_id
E2	C1
E3	C1

Different event IDs / same customer

Choose the key before removing duplicates.
For our events, event_id must be unique.

If repeated IDs contain conflicting amounts, investigate the conflict instead of arbitrarily keeping one row.

CHECK RELATIONSHIPS BETWEEN RECORDS

Purchases (events)

purchase_id	customer_id	amount
P001	C42	$25
P002	C17	$40
P003	C08	$15

Customers

customer_id	name
C17	Avery
C08	Blake
C99	Casey

C42 is missing

Inner Join
(purchases.customer_id = customers.customer_id)

Joined Results (inner join)

purchase_id	customer_id	amount	name
P002	C17	$40	Avery
P003	C08	$15	Blake

P001 is lost
(dropped by inner join)

Is the customer feed late, the key malformed, or the relationship invalid?

A purchase references customer C42, but C42 is absent from the customer table.

Check required keys and joins. A silent inner join can remove these purchases, changing counts without producing a visible application error.

CHECK TIMESTAMPS AND AVAILABILITY

Keep event time separate from arrival or availability time.

Example: Late delivery (valid event)

Event time
(in the real world)

Event happened on 31 July.

Data arrives later

Arrives on 2 August.

Arrival time
(in our system)

We receive it on 2 August.

A 31 July event arriving on 2 August is **late**, not automatically corrupt.

Not the same as an invalid timestamp

Invalid timestamp
e.g. unparseable, impossible future timestamp, or wrong timezone.

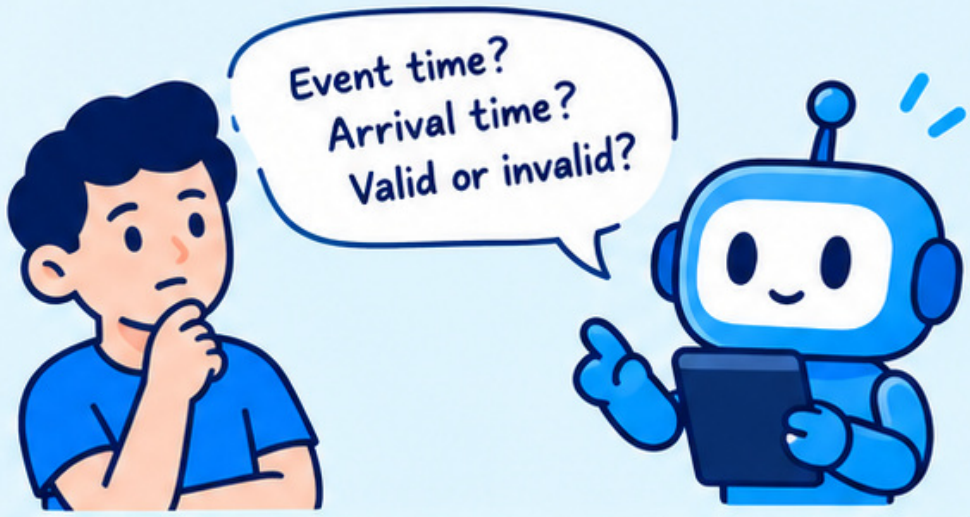
This is invalid, not just late.

Valid but late
e.g. 31 July event arriving on 2 August.

This is late, not corrupt.

What to check

- Check parsing.
- Check timezone.
- Check impossible future timestamps and delivery delay.
- Use the prediction-time availability rules from Day 7 when reconstructing features for an earlier decision.



CHECK THE BATCH AS WELL AS EACH ROW

Every row can pass while half the expected customers are missing.

Toy rows: individually valid

row_id	customer_id	amount	status
1	C001	25.00	valid
2	C002	17.50	valid
3	C003	42.00	valid
4	C004	19.00	valid
5	C005	36.00	valid

All shown rows pass validation.

Toy coverage example

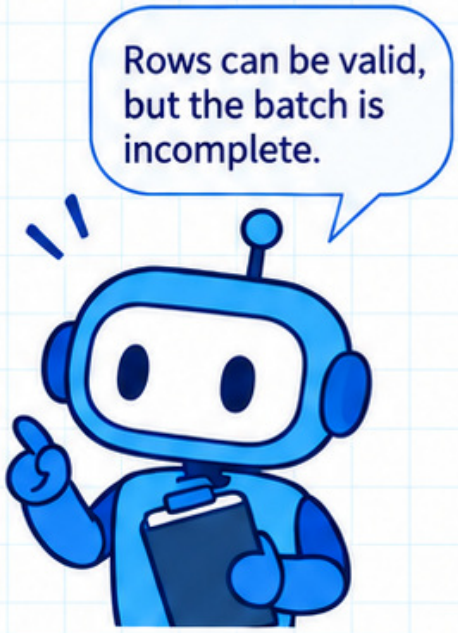
Two equal source partitions were expected.

One source partition is absent,
so only half the expected customers arrived.

Batch level checks

- Check batch size
- Check source coverage
- Check update freshness
- Check missing-value rates

ALERT
Source B is missing from this batch.
Investigate source delivery and coverage.



Compare with the appropriate day and population. A weekend, new market or upstream outage can change volume for very different reasons. A useful alert includes that context.

AI SYSTEM DESIGN • DAY 8

10/18

DISTRIBUTION CHANGE IS NOT AUTOMATICALLY BAD DATA

A sale increases the number of large purchases.

Before promotion
Illustrative distributions

During promotion
Illustrative distributions

The data distribution changed, but the values may be valid.

Invalid data violates a defined contract.

Invalid purchase value

-5

This value fails our toy nonnegative purchase rule and does not meet the defined contract.

Drift describes change in a distribution. Invalid data violates a defined contract. Investigate drift and measure its effect on the model before deciding whether to reject data, retrain or take no action.

@learn.machinelearning

CHECK ONE SMALL BATCH

Toy contract: unique event IDs, known customers, nonnegative amounts, and required timestamps.

Toy batch

Event ID	Result	Reason
E1	Valid	E1: valid row.
E1	Invalid	E1 again: duplicate ID.
E2	Invalid	E2: amount = -5.
E3	Invalid	E3: missing timestamp.
E4	Invalid	E4: unknown customer.

These are four different findings. Preserve the reason for each failure so the source owner can repair it.

DEFINE THE RESPONSE TO A FAILURE

Missing required column: stop this batch.

Stop this batch

Reason:
Missing required column (e.g. user_id).

Owner:
Data Engineering

Malformed row: quarantine it with the reason.

Quarantine row

Reason:
Malformed row (e.g. invalid date).

Owner:
Data Engineering

Unusual distribution: alert and investigate.

Alert and investigate

Reason:
Unusual distribution (e.g. spike in a category).

Owner:
Data Science

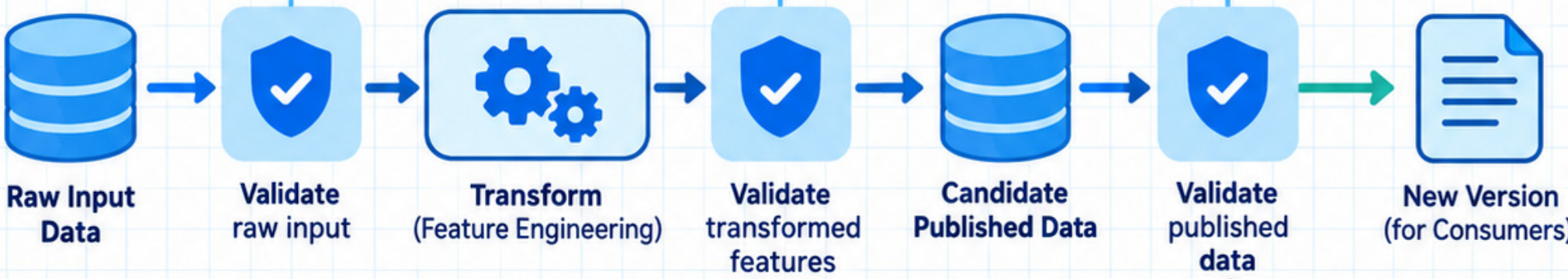
These are example policies. Choose severity by product impact, and track rejected volume. Dropping bad rows silently can leave a biased or incomplete dataset that still looks clean.

PUT CHECKS AT USEFUL BOUNDARIES

Check raw input before transformation.

Check transformed features before training or serving.

Check published data before consumers switch versions.



USE SQL WHERE THE DATA LIVES

SQL can find duplicate keys, missing relationships and invalid values.

Find issues early where the data lives.

Database constraints can enforce required fields, uniqueness and some value rules at write time.

A CHECK on a numeric range does not automatically forbid NULL. Use the appropriate nullability rule too. Batch coverage and freshness need additional checks.

NOT NULL

Ensures a value is always present.

UNIQUE

Enforces uniqueness under the declared key and NULL rules.

CHECK

Enforces a value rule, such as a numeric range.

A CHECK on a numeric range does not automatically forbid NULL.
Use the appropriate nullability rule too.

Batch coverage and freshness need additional checks.

USE PANDERA FOR DATAFRAME CHECKS

Pandera lets Python code describe expected dataframe columns and validation rules.

Use it to check types, missing values, uniqueness and custom conditions in a processing step.

DataFrame (example data)

event_id	amount	purchase_date
E1	10	2024-01-01
E2	20	2024-01-02
E3	15	2024-01-03
E2	-5	2024-01-04

A pandas DataFrame with purchase data (toy example).

Declared schema and checks (with Pandera)

event_id: string type, unique

amount: integer type, greater than or equal to 0

purchase_date: date type, no missing values

Custom check: amount must be greater than or equal to 0

Validation report (illustrative summary)

BATCH SUMMARY

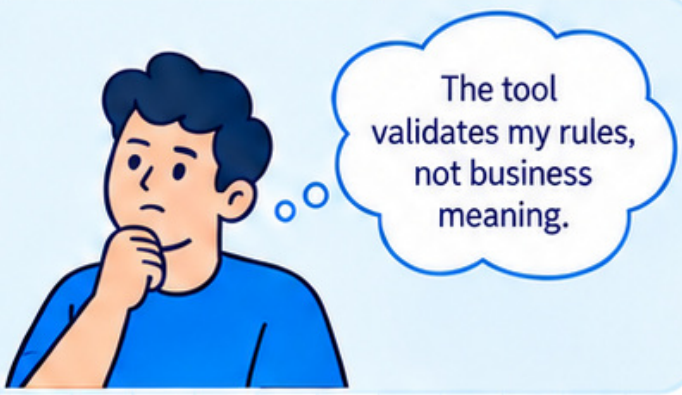
4 rows examined

FAIL

Found 2 validation errors:

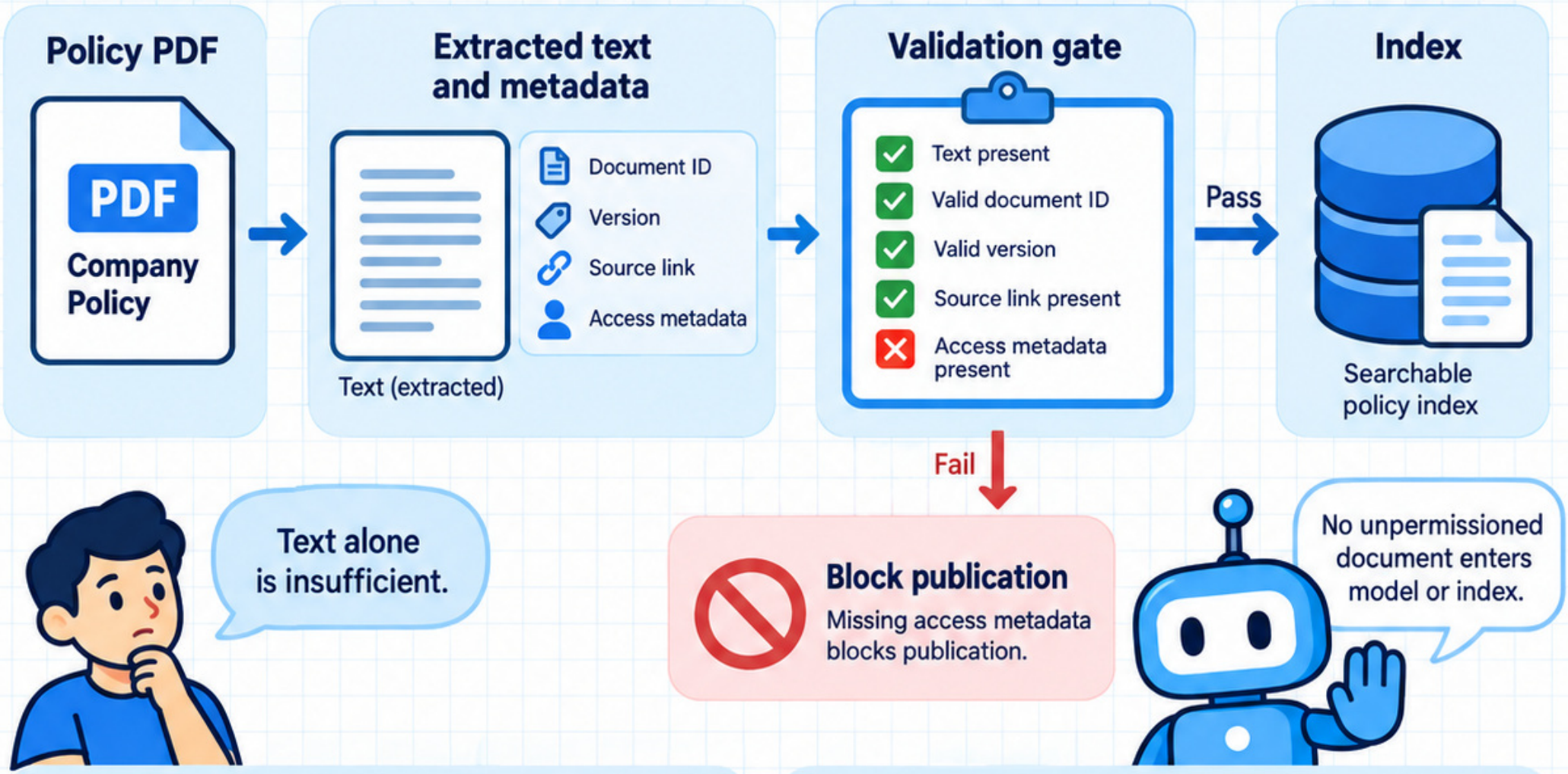
- Duplicate event_id: E2
- Negative amount: -5

The tool checks the rules you write. It does not discover the correct purchase definition, freshness target or business response for you.



GENAI DOCUMENTS NEED A CONTRACT TOO

For a policy assistant, inspect extracted text, document IDs, versions, source links and access metadata.



An empty extraction should not become a searchable policy. A replaced version should not silently remain current.

This differs from validating purchase rows, but the principle is the same: check the data against its intended use.

YOUR TURN: CHOOSE CHECKS AND RESPONSES

A batch contains a duplicated event ID, an unknown customer and a sudden rise in valid purchase amounts.

A policy document has text but no access metadata.

For each case, name the check, explain the risk and choose a response.
Distinguish confirmed contract failures from a change that needs investigation.

Issue 1: Duplicated event ID

A batch contains a duplicated event ID.

Check:

Risk:

Response:

Issue 2: Unknown customer

A batch contains an unknown customer.

Check:

Risk:

Response:

Issue 3: Sudden rise in valid purchase amounts

A batch shows a sudden rise in valid purchase amounts.

Check:

Risk:

Response:

Issue 4: Policy text with no access metadata

A policy document has text but no access metadata.

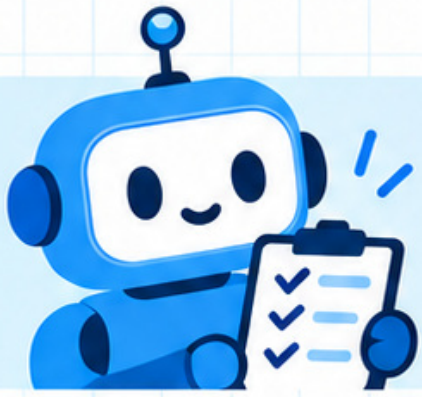
Check:

Risk:

Response:

VALID DATA IS THE BEGINNING

Define the contract. Test rows and whole batches. Preserve failure reasons. Publish versions deliberately.



Data can pass every schema check and still be unsuitable for a prediction.

Input Data

Rows and batches arrive for validation.

Validation Gate

- Schema checks
- Row tests
- Batch tests
- Preserve failure reasons
- Publish versions deliberately

If the data is valid, it passes the gate.

Next check: prediction-time availability

Prediction time t

Available by t

Arrives after t

Historical features must use only information available by t.

DAY 9: Data leakage

Next, we check whether training accidentally uses information that would be unavailable when the real prediction is made.