

Attribute, list or object?

Attio gives you three ways to store the same information. Pick wrong and you find out months later, when the data won't answer the question you're asking.

THE TEST – FOUR QUESTIONS, IN ORDER

1. Does this information exist on its own, or is it just a property of one record?

A property of one record, and one only → **ATTRIBUTE**

2. Will I need to hang several other pieces of information off it?

No → **ATTRIBUTE** · Yes → keep going

3. Once the operation is over, could I delete this without losing anything that matters?

Yes → **LIST**

4. Do I need statistics on the thing itself, or on how people interacted with it? Will it happen again and again? Do I need to read the relation from both sides?

Yes → **CUSTOM OBJECT**

ATTRIBUTE

Use it when the information is intrinsically tied to one object and one only, and you won't want to attach anything else to it.

- **Timestamps** — the date an event happened, stored on the object that date means something for.
- **Booleans** — do not call, do not email, has an account, has set a password. Use as many as you can: they filter cleanly.
- **External IDs** — a record ID from another platform.

WATCH OUT – THE MULTI-SELECT

The moment you reach for one, stop. Take "interested in a product": one person can want several variants of one product, then variants of another. It all lands in one field, you can't tell *when* each interest was expressed, and three months later you can't sort any of it. **A multi-select that grows is a list waiting to happen.**

LIST

Use it when the need is operational. The test: **if I could delete this list once the operation is done and it wouldn't matter, it's a list.**

- **An event** — signed up on, attending, how many guests, which city, badge printed.
- **Product interest** — one list per product: when we talked about it, when interest was expressed, budget, a comment if they didn't follow up. This is where the failing multi-select goes.
- **A task to run through** — generate the document, send it, get it signed.

It keeps operational noise out of your core database — and keeps you inside Attio, not in a disconnected Google Sheet.

LIST MOVERS

Don't add records by hand. Build a workflow — a **list mover** — that adds a record the moment a criterion is met (a purchase over €100, say), with a list status attribute carrying a default value. From there the list is a work queue. Then plug an AI assistant onto it: let it check what needs doing and ping the right person, instead of someone opening the list every morning.

WATCH OUT – LIST ATTRIBUTES DON'T SURFACE

You cannot see their values from an object view. So every time you create an attribute inside a list, ask: **does this belong on the object, or on the list?** On the list only if it is purely operational, scoped to that one task, and never needed elsewhere.

One exception worth making a habit: the **comment column belongs on the list**. With several tasks running in parallel, comments piled on the record become unreadable.

CUSTOM OBJECT

Limited in Attio — use them sparingly. You need one when you must relate records to this thing *and* build views on it, when you need statistics on the thing itself, when it happens again and again, and when you want to read the relation from both sides.

Webinars. Name, objective, start and end time, attendees. Many-to-many with people, so you can filter "who attended this webinar" from any view and "which webinars did this person attend" from their record — and pull the webinar's own stats in via API.

Calls. Two objects. **Call** : transcript, summary, recording URL, duration, start time, related to the person and to the workspace member who made it. **Call list** : number of people, type, the criteria used to build it, when it started and stopped being used — many-to-many with people.

The default setup — a note on the person after each call — is fine to read and useless to count: no table can tell you how long a prospect spends on the phone on average. A list wouldn't work either: calls keep coming, it is never "done".

Notes are readable, not countable. The moment you want to count it, it's an object.