Tracking creator and affiliate payouts without a spreadsheet argument

How to know what each creator actually earned and what they actually cost, without a monthly reconciliation argument.

GUIDE · 2026-11-03 · 8 MIN READ

Most brands with a creator programme cannot answer a simple question: what did this person earn, and what did they cost us.

They can name the codes. They can find the DMs. What they cannot do is put a number next to a name without an hour of spreadsheet work, and so nobody asks, and the programme runs on impressions and goodwill until a creator disputes a payout and the relationship gets expensive.

Why this breaks in the same way everywhere

The information needed to pay somebody correctly is split across systems that do not talk:

- **The code lives in the shop platform.** Redemptions, revenue, sometimes the commission rate.
- **The agreement lives in a DM or an email.** Often a screenshot. Rarely the same terms as the last one.
- **The payment lives in the bank.** Not linked to the code, not linked to the agreement.
- **The content lives on the platform.** No record of which post drove which sale.

Four systems, one question, no common key. So the reconciliation happens in a spreadsheet maintained by whoever cares most, and it stops the week that person is busy.

The fix is a key, not a tool

Before any automation, decide the one identifier that everything hangs off. In most programmes that is the discount code, because it is the only thing that appears in the shop platform, in the creator's content and in the customer's order.

Then build one table, in whatever you already have, with these columns:

- Creator name, and the handle you actually reach them at
- Code, and the commission rate agreed for that code
- Agreement date, and where the agreement is written down
- Redemptions this period, and net revenue after returns
- Commission owed, and commission paid, with a date

That is the whole model. Nine columns. If a creator is not in the table, they are not in the programme, regardless of what was said in a DM.

The three rules that stop the arguments

One rate per code, written down. When a creator has three codes at three rates, nobody can compute the total correctly. If a rate changes, the old code retires and a new one starts.

Pay on net revenue, after returns. Gross revenue commissions mean you pay on orders that were refunded. The creator will not like discovering that later, and you will not enjoy clawing it back.

State the payment cycle and stick to it. Monthly, on a fixed date, with a statement. A creator who can predict when they get paid stops asking, and the asking is what consumes the hours.

What to automate, in order

1. **Pull redemptions weekly** from the shop platform into the table. This is the only genuinely repetitive step, and the only one worth wiring first.
2. **Compute commission** from redemptions times the rate on that row. A formula, not a person.
3. **Generate the statement** as the same one-page format every month, per creator.
4. **Record the payment** against the row so "paid" is a fact rather than a memory.

Do not automate the negotiation or the agreement. Those are judgement, and a template that pretends otherwise produces worse agreements.

How to tell it is working

- You can answer "what did this creator earn" in under a minute, for any creator.
- A creator dispute is resolved by opening one row, not by searching four systems.
- The statement goes out on the same date every month without anyone remembering to send it.
- You know your cost per creator, so you can pause the ones that do not pay back.

> *The spreadsheet is not the problem. The absence of a shared key is the problem, and no tool fixes that for you.*

Where this usually stops

Brands build the table, run it for two months, then a campaign happens and the weekly pull gets skipped, and the programme drifts back to memory.

The defence is the same as any recurring process: make the weekly step take under ten minutes, and record whether it ran. Six weeks of "not run" is visible long before the numbers go wrong.