EcomLabs

STEP-BY-STEP GUIDE

GA4 and Shopify: how to reconcile your sales

The checklist to reconcile GA4 with Shopify, how to check the purchase event with DebugView, Shopify's Pixel Helper and Tag Assistant, and a Node script that matches your orders to your purchases by transaction_id to list the missing and duplicated ones.

FOR	Shopify stores that measure sales with GA4
TESTED WITH	Node 22.22.3 · node:test · Windows 11 · illustrative sample CSVs
UPDATED	November 11, 2026
DOCS	help.shopify.com/en/manual/reports-and-analytics/discrepancies
ARTICLE	ecomlabs.dev/g/tracking-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you'll compare
Step 01	Reconciliation checklist
Step 02	Check the purchase event in DebugView
Step 03	Pixel Helper and Tag Assistant
Step 04	Export and match order by order
Step 05	Read the result

BEFORE YOU START

What you'll compare

Shopify orders against GA4 purchases (the `purchase` event), order by order, for closed days. Don't compare sessions or conversion rates: each tool defines them its own way, and Shopify changed how it measures sessions between September 21 and 23, 2026. You need the Shopify admin, Editor access to the GA4 property and Node 22 (we tested with 22.22.3). Nothing in this guide sends data to Google or Shopify: the script works on two CSV files on your computer.

STEP 01

Reconciliation checklist

- [ ] GA4's time zone (Admin > *Property details*) matches your store's (Shopify: *Settings > General*). Changing it in GA4 only affects data going forward.
- [ ] GA4's currency matches your store's; otherwise GA4 converts at the previous day's exchange rate.
- [ ] The store isn't in private mode: Shopify notes GA4 can't track events while it is.
- [ ] Only one tag sends `purchase` to each property. Check *Settings > Customer events* (app pixels and custom pixels) and the theme code. Never gtag.js and GTM at the same time.
- [ ] `transaction_id` is the order ID: unique, dynamic, never empty or hard-coded, with no customer data.
- [ ] You know what basis `value` uses. Google recommends the sum of `price × quantity` without shipping or tax; Shopify's GTM tutorial uses `checkout.totalPrice.amount`, which includes taxes.
- [ ] No purchase tracking is left in additional scripts or script tags on the Order status page: Shopify switched them off in 2025 and 2026.
- [ ] If you run a cookie banner, you know which regions it's active in and whether your pixel waits for consent.
- [ ] You compare days at least 48 hours old: GA4 takes 24 to 48 hours to process, and numbers change in the meantime.
- [ ] You rerun this guide's match after every change to tags, apps or the theme.

STEP 02

Check the purchase event in DebugView

1. Turn on debug mode with Tag Assistant (`tagassistant.google.com`) or GTM preview mode. With gtag.js in the theme, `gtag('config', 'G-XXXXXXXXXX', { 'debug_mode': true });` works too; to turn it off, remove the parameter, because `false` doesn't disable it.
2. In GA4, open Admin > *Data display* > *DebugView* and accept the cookie banner: DebugView doesn't show events from users who haven't consented to Analytics cookies.
3. Place a test order with Shopify's test payment gateway or Shopify Payments test mode. Test orders don't show up in your reports.

4. Click the `purchase` event to see its parameters: `transaction_id` with your order number or ID, `value`, `currency` and `items`.
5. Reload the Thank you page. No second `purchase` with a different ID should appear.
6. If DebugView doesn't list your device, open *Reports > Realtime*: the event count by event name card covers the last 30 minutes.

> **CAUTION**
> GA4 keeps only the first `purchase` with a given `transaction_id` from the same user and ignores the rest, even if other values change. To repeat a test, place another test order: each one gets its own ID.

STEP 03

Pixel Helper and Tag Assistant

Shopify Pixel Helper, for app pixels (like the Google & YouTube channel's) and custom pixels:

1. In the admin, go to *Settings > Customer events*, open the pixel's more-options menu, choose the test option and click *Test*.
2. If it says "Pixel is awaiting consent", accept on the banner or click *Give consent to continue test*.
3. Place a test order and look for `checkout_completed`: a green dot means the event was received and its callback ran; a red dot means the callback failed. Expand the dialog to see the data.

The Pixel Helper checks one pixel at a time, clears its log when you change pages, and doesn't work on headless stores or in private mode.

Tag Assistant, for a Google tag or GTM pasted into the theme: at `tagassistant.google.com`, click *Add domain*, enter your store URL starting with `https://` and click *Connect*. Under the *Google tags found* header, your tag ID should appear with a green or blue status icon; compare the tags it finds with the ones in *Customer events*. If the `_dbg` parameter breaks the page, deselect *Include debug signal in the URL*.

> **CAUTION**
> Per Shopify's GTM tutorial, Tag Assistant's *Troubleshoot tag* feature isn't compatible with custom pixels and doesn't detect Google tags in a custom pixel or loaded there by GTM. For those, use the Pixel Helper and DebugView.

STEP 04

Export and match order by order

Shopify: in *Orders*, filter the dates, click *Export* and choose CSV. **GA4:** in *Explore*, create a blank exploration, import the *Transaction ID* dimension and the *Transactions* metric, drag them to rows and values, in that order, and export as CSV with *Export data*. If the thresholding notice appears, widen the date range. Save `reconcile.mjs` next to both files; it has no dependencies:

```
// reconcile.mjs: compares a Shopify orders export with a GA4 export of transaction IDs.
// Usage: node reconcile.mjs orders_export.csv ga4-transactions.csv
import { readFileSync } from "node:fs";
import { pathToFileURL } from "node:url";
```

```
// CSV with quoted fields: commas, "" and line breaks inside quotes.
export function parseCsv(text) {
  const rows = [];
  let row = [], field = "", quoted = false;
  for (let i = 0; i < text.length; i++) {
    const c = text[i];
    if (quoted) {
      if (c === '"' && text[i + 1] === '"') { field += '"'; i++; }
      else if (c === '"') quoted = false;
      else field += c;
    } else if (c === '"') quoted = true;
    else if (c === ",") { row.push(field); field = ""; }
    else if (c === "\n") { row.push(field.replace(/\r$/, "")); rows.push(row); row = []; field = ""; }
    else field += c;
  }
  if (field || row.length) { row.push(field.replace(/\r$/, "")); rows.push(row); }
  return rows;
}

const clean = (id) => String(id ?? "").trim().replace(/^#/, "");

export function reconcile(ordersCsv, ga4Csv) {
  // Shopify: one row per line item, so keep the first row of each order (Name).
  const [head, ...rows] = parseCsv(ordersCsv.replace(/^\uFEFF/, ""));
  const col = Object.fromEntries(["Name", "Id", "Source", "Created at", "Payment Method"].map((n) => [n,
head.indexOf(n)]));
  const absent = Object.keys(col).filter((n) => col[n] < 0);
  if (absent.length) throw new Error(`orders export without column(s): ${absent.join(", ")}`);
  const orders = new Map();
  const byId = new Map();
  for (const r of rows) {
    const name = r[col.Name];
    if (!name || orders.has(name)) continue;
    const order = { name, source: r[col.Source], created: r[col["Created at"]], payment: r[col["Payment
Method"]], hits: [] };
    orders.set(name, order);
    // GA4 may hold the order number (1001 or #1001) or the order ID: index both.
    for (const key of [clean(name), clean(r[col.Id])]) if (key) byId.set(key, order);
  }

  // GA4: skip blank lines and # comments before the header; then Transaction ID, Transactions.
  const ga = parseCsv(ga4Csv.replace(/^\uFEFF/, "")).filter((r) => r.join("").trim());
  const header = ga.findIndex((r) => !r[0].startsWith("#"));
  const unknown = [];
  for (const [id, count] of header < 0 ? [] : ga.slice(header + 1)) {
    const n = Number(count);
    if (!Number.isFinite(n)) throw new Error(`GA4 row "${id},${count}": the second column must be
Transactions`);
    const order = byId.get(clean(id));
    if (order) order.hits.push({ id, n });
    else unknown.push({ id, n });
```

```
  }

  const all = [...orders.values()];
  return {
    orders: all.length,
    missing: all.filter((o) => o.hits.length === 0),
    duplicated: all.filter((o) => o.hits.reduce((s, h) => s + h.n, 0) > 1),
    unknown,
  };
}

if (import.meta.url === pathToFileURL(process.argv[1]).href) {
  const [ordersFile, gaFile] = process.argv.slice(2);
  const r = reconcile(readFileSync(ordersFile, "utf8"), readFileSync(gaFile, "utf8"));
  console.log(`Shopify orders: ${r.orders} · missing in GA4: ${r.missing.length} · duplicated:
${r.duplicated.length} · unknown IDs: ${r.unknown.length}`);
  for (const o of r.missing) console.log(`MISSING     ${o.name}  source=${o.source}  payment=${o.payment}
created=${o.created}`);
  for (const o of r.duplicated) console.log(`DUPLICATED  ${o.name}  ${o.hits.map((h) =>
`${JSON.stringify(h.id)}×${h.n}`).join(" + ")}`);
  for (const u of r.unknown) console.log(`UNKNOWN     ${JSON.stringify(u.id)}×${u.n}`);
}
```

Run it with `node reconcile.mjs orders_export.csv ga4-transactions.csv`, using your own file names. If a column is missing from the Shopify export, or the second GA4 column isn't a number, it stops with an error that says so.

STEP 05

Read the result

The first line sums up how many orders are missing, how many are duplicated and how many GA4 IDs match no order. Every line after it starts with `MISSING`, `DUPLICATED` or `UNKNOWN`; missing orders come with `source`, `payment` and `created`. The article shows a full run on ten illustrative orders.

LINE	WHAT IT MEANS	WHAT YOU DO
`MISSING` with `source` pos or draft order	A sale outside the online store	Nothing: GA4 was never going to see it
`MISSING` web with an offsite payment	The shopper didn't return to the Thank you page	Count them separately
`MISSING` web near midnight	Different time zones	Match the time zones
`MISSING` web, the rest	Declined cookies or ad blockers	That's your baseline
`DUPLICATED`	Two tags or two ID formats	Keep a single tag
`UNKNOWN` empty or `(not set)`	A purchase without `transaction_id`	Fix the tag
`UNKNOWN` with another value	Another ID format or another day	Check it in DebugView