VINETA MARKETING

Reporting Dashboard

Example Setup

Instructions to set up a custom Braze reporting dashboard. This example includes brief/prompts, the connectors and Skills you need, and a QA checklist ready to be adapted for your use case.

What this is

This is a demo, not a live client dashboard. No real client data or numbers appear anywhere in this guide: every canvas name, ID, and figure is made up for illustration. Built for a Claude and Braze connection, but the same approach works with any AI tool and CRM platform that supports similar access.

AI tools change constantly, so the exact screens and steps here may shift as Claude and Braze evolve. The underlying approach will still hold.

The process behind a build matters as much as the build itself: it takes someone who understands your business well enough to know what's worth building and how it'll actually save your team time. These briefs get reused across different companies and projects: only the specifics change. Fill one in and run it as-is first to see how it behaves, then adapt it to your own team.

Whether you're setting up your CRM the right way from the start or helping your team move faster with what you already have, the real skill is knowing what output you need and how to get it before you reach for AI. That's the kind of work we do at Vineta.

If you want help adapting this to your program, visit vinetamarketing.com to see how we're using AI to boost full-funnel growth marketing.

We've built similar AI-assisted workflows for:

- **Campaign briefing**
- **Campaign setup**
- **QA checking**

If a repeatable process is eating your team's time, there's usually a way to build it once and let it run.

Overview

Before the detailed brief, here's the full sequence this guide walks through, start to finish. Each step depends on the one before it, so don't skip ahead.

1. **Do the one-time admin step.** Before anyone can connect Braze at all, a company admin has to turn on MCP OAuth access in Braze and grant the "Use MCP Server" permission. This isn't an engineering build. It's a permissions toggle, and it only has to happen once per organization. If you don't have Braze admin access yourself, this is the one step you'll need to hand off, everything after it you can do yourself.
2. **Connect your two connectors.** Once the admin step is done, connect Braze and your destination sheet yourself, no API key needed. This is access, not behavior. Connecting them doesn't make anything run yet.
3. **Set up the Skill.** Save the brief you're using as a Skill, so Claude follows the same metrics, rules, and guardrails every run instead of you retyping instructions each time. Part 2 and Part 2B are two separate briefs on two different cadences. If you're running both, save each as its own Skill, not one combined Skill.
4. **QA it manually, before you schedule anything.** Run the brief by hand once, as a normal chat, not on a schedule yet. Check the output against the Part 3 checklist. Don't turn on the schedule until a manual run passes clean. Catching an issue in a one-off run is a lot cheaper than catching it after it's been writing bad data for a week.
5. **Schedule the pull.** Once a manual run checks out, turn it into a Scheduled Task so it runs on its own, on whatever cadence you set, no manual trigger needed from here.
6. **Check the Artifact whenever you want.** It's a live view, not a one-off report, so you open it any time and see the latest pull, not just on the day something got flagged. Find it under Artifacts in Claude's sidebar and click to reopen it.
7. **Keep the Skill current.** Update the Skill directly whenever your metrics, rules, or guardrails change. Don't let the brief and the Skill drift apart. If you need something beyond this brief, like an add-on report or a different task entirely, build it as its own Skill instead of overloading this one.

Part 1: Skills and Knowledge You Need

Two different things live under this heading: what you need to understand before you write a brief like this, and what you should package as a reusable Skill so the agent behaves the same way every run.

KNOWLEDGE, BEFORE YOU START

- **Your platform's data model.** Know the difference between canvas-level and step-level metrics, and what each field actually measures. This brief only works because the channel key mapping (push: delivered = sent minus bounces; in-app: sent = delivered = impressions) was worked out ahead of time, not discovered mid-build.
- **Detail your conversion events, canvas by canvas.** Conversion A through D mean nothing until you've mapped them to real events in your program, and that mapping can change from one canvas to the next. Don't define it once for the whole program and assume it holds everywhere. This is the step most people skip, and it's the one that causes false reads later.
- **The difference between a rule and a guardrail.** Rules decide what gets flagged. Guardrails decide what the agent is never allowed to do, regardless of what it finds. Both need to be explicit in the brief, not assumed.

CONNECTORS AND SKILL TO SET UP

Two different things, worth keeping straight. A connector is access: it's what lets Claude touch a real tool. A Skill is behavior: it's the written instructions Claude follows so it does the task the same way every time. You need both, but they're not the same thing.

Connectors (the access):

- **A sheet connection**, so Claude can write to (and check for existing rows in) your destination sheet.
- **A Braze connection**, added once a company admin has turned on MCP OAuth access in Braze and granted the "Use MCP Server" permission (a one-time permissions toggle, not an engineering build). This is what gives Claude the live pull described in Part 2.

Skill (the behavior):

- One Skill per brief: the metrics, rules, and guardrails from Part 2 (or Part 2B), saved as instructions Claude reads before every run. Part 2 and Part 2B are separate briefs on separate cadences, so running both means saving two separate Skills, not one combined Skill. That's what keeps each run consistent, not the connectors. The connectors just get Claude in the door.

That's the whole list: two connectors, plus one Skill per brief you run. If you find yourself wanting more automation later, that's a good use case for building additional Skills with Claude directly.

Part 2: Braze Canvas Dashboard

This is based on a production brief, made generic: the exact structure, rules, and thresholds you'd actually hand an agent, with client-identifying names and IDs swapped for placeholders. Copy the pattern, drop in your own canvases.

THE USE CASE

A daily Braze canvas monitor that pulls canvas- and step-level performance for a fixed set of production canvases, compares each one to its own trailing 7-day baseline, and flags only the patterns that actually need a person's attention (entry drops, conversion drops, deliverability issues, weak individual steps, fatigue, and funnel stalling) instead of surfacing every metric and leaving you to find the problem yourself. It writes to a running sheet rather than a one-off report, and posts a same-day summary ranked by severity, so checking six canvases becomes a two-minute read instead of a manual pull every morning.

BEFORE YOU FILL THIS IN

Answer these first, the brief below writes itself once you have:

- **What decision should this dashboard drive?** (e.g. "which canvases need a fix this week," not just "show me numbers")
- **Which metrics matter most for this program?** Rank them. Don't pull every metric Braze offers just because it's available.
- **What do Conversion A, B, C, and D mean for your funnel?** Map each to a real event in your program (e.g. Conversion A = purchase completed, Conversion D = added to cart). Every business's funnel is different, so this mapping is yours to define, not something to copy from an example.
- **What does "working" vs "needs attention" actually look like** for a canvas in your program?
- **Any channels, segments, or canvases that need special-case handling** the rules below don't already cover?

THE BRIEF

Save this as a Skill (for example, `braze-canvas-monitor.md`). This is what runs every day, so it belongs in a Skill, not a one-off chat message. Copy everything below in, placeholders filled in first:

```
Run the daily Braze canvas monitor for these six production Canvases:

- [CANVAS_1_NAME] ([CANVAS_1_ID])
- [CANVAS_2_NAME] ([CANVAS_2_ID])
- [CANVAS_3_NAME] ([CANVAS_3_ID])
- [CANVAS_4_NAME] ([CANVAS_4_ID])
- [CANVAS_5_NAME] ([CANVAS_5_ID])
- [CANVAS_6_NAME] ([CANVAS_6_ID])

Use production app_group_id [YOUR_APP_GROUP_ID]. Pull yesterday's
data with step-level breakdown (include_step_breakdown).

Metrics to pull:

- Canvas level: entries, Conversion A (primary), Conversion B,
  Conversion C, Conversion D, conversion_rate, revenue, plus the
  trailing 7-day baseline median for entries and conversion_rate
  and the % change vs baseline.
- Step level, per channel: sent, delivered, delivery_rate,
  unique_opens, open_rate, unique_clicks, ctr (from unique_clicks,
  not raw clicks, for all channels including push and in-app),
  baseline_median_ctr, ctr_change_pct, unsubscribes, unsub_rate,
  bounces, dismissals.
- Channel key mapping: push, delivered = sent minus bounces,
  unique_opens = total_opens, clicks = direct_opens. In-app,
  sent = delivered = impressions, clicks = PrimaryCTA +
  SecondaryCTA, dismissals = CloseButton.

Rules to evaluate (vs the trailing 7-day median):

- Entries < 70% of baseline -> "Entry drop"
- conversion_rate < 75% of baseline, with entries within +/-15%
  of baseline -> "Conversion drop, check offer/landing/final step"
- Email delivery rate < 95%, or push delivery rate < 90% ->
  "Deliverability"
- Step CTR < 50% of the same-channel baseline median -> "Weak
  step, test copy/CTA/timing"
- Email unsub rate > 0.3% -> "Fatigue, reduce email frequency /
  tighten segment" (email only, Braze doesn't track unsubscribes
  for push or in-app)
- (Conversion A + Conversion B) = 0 and (Conversion C +
  Conversion D) > 0 and entries > 100 -> "Stalling before final
  conversion": users are hitting your earlier-funnel events (C
  and D) but not completing the primary conversion (A and B).
  Check tracking on the final step first, this pattern can also be
  a tracking break, then look at friction if tracking checks out.
- Canvas has enabled = false or has_post_launch_draft = true ->
  "Status"
- Volume floor: CTR and unsub-rate rules only fire when step
  delivered >= 50. Below that, mark the step "Below volume floor"
  and skip rule evaluation, don't flag on small-sample noise.
```

```
Writing to the Google Sheet ("[Your Program Name] Canvas
Dashboard", tabs "Canvas Daily" and "Step Daily"):

- Before appending any row, check whether a row already exists
  for this exact date + canvas_name (Canvas Daily) or date +
  canvas_name + step_name + channel (Step Daily). If one exists,
  overwrite it in place instead of appending a new row, never
  create duplicates.
- The canvas status field must only ever contain Braze's own
  status value (e.g. "enabled", "disabled", "draft"). Do not
  append any fired rule name to this field. If you want fired
  rules visible in the sheet, put them in a separate note, not
  blended into status.

Output: after writing both tabs, post a chat summary of what
fired, grouped by urgency (most severe first), naming the canvas,
the rule, and the canvas's revenue for the day.

Guardrails:

- Never send, post, or modify a live canvas. The only output is
  the sheet write and the chat summary.
- Never create a duplicate row. Check for an existing date +
  canvas (or date + canvas + step + channel) row first and
  overwrite it in place.
- Keep the status field limited to Braze's own status values
  only. Put any fired rule names in a separate note field, never
  blended into status.
- Below the volume floor (step delivered < 50), skip rule
  evaluation and mark it "Below volume floor" instead of flagging
  on small-sample noise.
```

Once your placeholders are filled in, this is ready to run as-is.

Building the live view. Once the sheet is running, this is the prompt that turns it into a filterable Artifact, so you have somewhere to check anytime, not just the day something fired in the chat summary. **Run this once in a regular conversation, not as a Skill.** It sets up the Artifact. It doesn't need to run again unless you're changing the filters.

```
Build an Artifact from the "[Your Program Name] Canvas Dashboard"
sheet, showing both the Canvas Daily and Step Daily tabs.

Update the Artifact to add three filters: Date, Canvas, and Channel.

For Canvas: use canvas_name directly, it's already a column in both
the Canvas Daily and Step Daily tabs.

For Channel: use the channel column from Step Daily (email, push,
in-app).

For Date: use the date column from either tab.

Add filter controls at the top of the Artifact: a date range picker,
a canvas dropdown (multi-select, defaulting to all six canvases),
and a channel dropdown (multi-select, defaulting to all channels).
Have both the canvas-level and step-level tables respond to all
three.

Add a Flagged section above the tables that surfaces just the rows
where a rule fired, grouped by canvas and ranked by severity (most
severe first, matching the chat summary's own ranking), so the
same read is available live in the Artifact, not just the day it
posted.
```

PART 2B: EXAMPLE BRIEF — LIFECYCLE CANVAS WEEKLY REPORT

A second pattern worth having on hand. This one doesn't flag anything, it just reports weekly, useful when you want a running view across every lifecycle canvas without building a rules engine on top of it. Same idea as Part 2, different cadence: it pulls the trailing week directly from Braze and posts once, on Mondays, with a manual override for any other day.

Save this as its own Skill (for example, `lifecycle-weekly-report.md`), separate from Part 2's Skill since the two run on different schedules and shouldn't be merged into one. Copy everything below in, placeholders filled in first:

```
TRIGGER
- Runs Mondays only (account timezone), or on manual trigger
- No daily run needed, there's no state to accumulate between runs

Pull the trailing 7 days of canvas- and step-level data (ending
yesterday) directly from Braze, with step-level breakdown enabled,
for these lifecycle Canvases:

- [LIFECYCLE_CANVAS_1_NAME] ([LIFECYCLE_CANVAS_1_ID])
- [LIFECYCLE_CANVAS_2_NAME] ([LIFECYCLE_CANVAS_2_ID])
- [LIFECYCLE_CANVAS_3_NAME] ([LIFECYCLE_CANVAS_3_ID])

Metrics to pull, per step and channel: sent, unique_opens,
open_rate, unique_clicks, click_rate. Channels: email, push, sms,
however this program uses them.

Also pull, per step, the count of entrants who reached that step
and later completed the canvas's own goal event (first login, first
purchase, reactivation, whatever [YOUR_GOAL_EVENT] is for that
canvas) within its attribution window. That's the step's Conversion
number, distinct from Open Rate and Click Rate, it's the one that
actually ties a message back to the outcome you care about.

POST (Mondays only, or manual trigger)
Append one row per Canvas Step to the "[Your Program Name]
Lifecycle Canvas Weekly Report" sheet:
Report Date (= today) | Canvas | Step | Channel | Market |
Open Rate | Click Rate | Conversion | Suggested Action (blank if none)

SUGGESTED ACTIONS
For any step where Click Rate falls under half its channel's
trailing baseline, or Conversion is the lowest in its canvas's
sequence, write one plain-language line for the Suggested Action
column, something a CRM manager could act on without digging
further (test a different CTA, swap the channel, move the step
earlier or later, strengthen the incentive). Leave the column blank
for steps that are performing normally, don't manufacture a
suggestion where there isn't a real signal.

Guardrails:
- Append-only, never overwrite a prior week's row.
- Never post outside the Monday cadence unless manually triggered.
- If a canvas ID in the list is disabled or deleted, skip it and
  flag it in the summary, don't error the whole run.
- A Suggested Action is a prompt for a human to look, never an
  instruction to change the canvas itself.
```

Building the live view. Once the sheet is running, this is the prompt that turns it into a filterable Artifact. **Run this once in a regular conversation, not as a Skill.**

```
Build an Artifact from the "[Your Program Name] Lifecycle Canvas
Weekly Report" sheet.

Update the Artifact to add three filters: Date, Market, and Channel.

For Market: it isn't a separate column in the sheet yet, extract
it from Canvas Step. These names follow the pattern
LC_{MARKET}_EMAIL_..., where {MARKET} is a 2-letter country code
(DE, FR, ES, IT, NL, US, GB, etc.) right after "LC_". Pull that
code out as the Market for each row.

For Channel: use the Channel column directly (email, push, sms).

For Date: use Report Date from the "[Your Program Name] Lifecycle
Canvas Weekly Report" sheet.

Add filter controls at the top of the Artifact: a date range picker,
a market dropdown (multi-select, defaulting to all markets), and a
channel dropdown (multi-select, defaulting to all channels). Have
the canvas step table respond to all three.

Add a Suggested Actions section below the table that surfaces just
the rows with a non-blank Suggested Action, grouped by canvas, so
it reads as a short action list rather than something you have to
dig for in the table.
```

Part 3: QA Checklist

Once this is running on a schedule, the risk isn't that it breaks loudly. It's that it quietly says something wrong, and you don't catch it for a week. This is what to check before you trust what it's telling you.

- **Verify against the actual sheet, not the chat summary.** A written recap can say a fix landed when the sheet hasn't actually refreshed yet. Always check the source, not the summary of it.
- **Confirm your Conversion A-D mapping still matches what the canvas tracks.** A rule reading zero on both sides of a comparison isn't automatically broken. It can also mean nobody's reached that step yet. Check the canvas itself before treating a flag as an error.
- **Confirm per-channel normalization holds before trusting a cross-channel number.** Push, email, and in-app each define "sent" and "delivered" differently. If that mapping drifts, every rule built on top of it drifts with it.
- **Spot-check one canvas by hand against native reporting.** Pick any canvas, pull it manually, compare it to what the dashboard flagged. If they disagree, the mapping or a rule threshold needs a second look, not the canvas.
- **Re-verify after any rule change.** Don't assume a new threshold is live until you've checked the next day's output against the source, not the chat recap of it.