Performance Reporting mit Claude erstellen

Technische Schritt-für-Schritt-Anleitung — Sheet, Apps Script, Claude Skill, PDF-Export. Alle Beispiele verwenden fiktive Daten (Beispiel-Blog „Contentspur“).

1. Ausgangssituation

Ziel dieser Anleitung ist ein wiederverwendbares Reporting-System für einen B2B-Blog: Traffic- und SEO-Daten werden automatisiert in einem Google Sheet gesammelt, ein Claude Skill wertet sie strukturiert aus und leitet konkrete Handlungsempfehlungen ab — bis hin zu einer fertigen PDF-Präsentation. Die Beispielwerte in diesem Dokument (Blog „Contentspur“, fiktive Kennzahlen) dienen ausschließlich der Veranschaulichung und stammen aus keiner realen Quelle.

Benötigte Datenquellen:

- Google Analytics 4 (GA4) — Traffic, Engagement, Channel-Zuordnung
- Google Search Console — Suchanfragen, Klicks, Impressionen, Position
- LinkedIn-Analytics (manuell, da keine offene API) — Reichweite, Follower
- Redaktionskalender — welche Artikel wurden im Zeitraum veröffentlicht

2. Das Google Sheet vorbereiten

Lege ein zentrales Google Sheet an, z. B. „Blog Reportings“. Jede Datenquelle bekommt einen eigenen Tab. Entscheidend ist eine einheitliche Spaltenlogik über alle Tabs hinweg — sonst lassen sich die Daten später nicht sauber zusammenführen.

Tab	Zweck	Update
Blog-GA4-Wochenübersicht	Aktive Nutzer, Sitzungen, Top-Artikel	wöchentlich, automatisiert
Blog-GA4-Erweitert	Sitzungsdauer, Channel-Mix, Scroll-Events	wöchentlich, automatisiert
Search-Console-Wochenübersicht	Top-Suchanfragen, Klicks, CTR, Position	wöchentlich, automatisiert
LinkedIn-Wochenübersicht	Reichweite, Engagement, Follower	manuell
Übersicht Artikel	Redaktionskalender, Metadaten	manuell

Pflicht-Spalten in jedem Tab:

- KW (Kalenderwoche, ISO-Format, z. B. „KW27“)
- Zeitraum (Mo–So), z. B. „29.06.–05.07.2026“
- Kennzahlen-Spalten je nach Datenquelle

> Wichtig: Bestehende Zeilen der jeweiligen Kalenderwoche sollten vor jedem Update-Lauf gelöscht werden, um Duplikate zu vermeiden (siehe Apps-Script-Beispiel in Kapitel 3).

3. Explorative Datenanalyse in Google Apps Script anlegen

Statt Daten manuell zu exportieren, ruft ein Google-Apps-Script-Projekt die GA4- und Search-Console-APIs automatisiert ab und schreibt die Werte direkt in die passenden Tabs.

Schritt 3.1 — Apps-Script-Projekt anlegen

- Im Google Sheet: Erweiterungen → Apps Script öffnen
- GA4 Data API und Search Console API in den Google Cloud APIs aktivieren (über die verknüpfte GCP-Projekt-ID)
- Property-ID der GA4-Property notieren (Verwaltung → Property-Details)

Schritt 3.2 — Beispiel-Skript für einen GA4-Wochenabruf

```
function runWeeklyGA4Update() {
const PROPERTY_ID = "properties/DEINE_PROPERTY_ID";
const sheet = SpreadsheetApp.getActive()
.getSheetByName("Blog-GA4-Wochenübersicht");

const heute = new Date();
const ende = new Date(heute);
ende.setDate(ende.getDate() - 2); // GA4-Processing-Delay-Puffer
const start = new Date(ende);
start.setDate(start.getDate() - 6);

const request = {
dateRanges: [{ startDate: formatDate(start), endDate: formatDate(ende) }],
dimensions: [{ name: "pagePath" }],
metrics: [{ name: "activeUsers" }, { name: "sessions" }],
dimensionFilter: {
filter: {
fieldName: "pagePath",
stringFilter: { matchType: "CONTAINS", value: "/blog/" }
}
}
};

const response = AnalyticsData.Properties.runReport(request, PROPERTY_ID);
loescheAlteZeilenFuerKW(sheet, aktuelleKW());
schreibeZeilen(sheet, response, aktuelleKW());
}

function setupWeeklyTrigger() {
ScriptApp.newTrigger("runWeeklyGA4Update")
.timeBased()
.onWeekDay(ScriptApp.WeekDay.MONDAY)
.atHour(7)
.create();
}
```

> Die Funktion loescheAlteZeilenFuerKW() verhindert Duplikate: Vor jedem Schreibvorgang werden alle Zeilen der aktuellen Kalenderwoche im Ziel-Tab entfernt.

Schritt 3.3 — Search Console analog einrichten

Für die Search Console gilt derselbe Aufbau, allerdings mit einem Trigger am Mittwoch statt Montag — die Search-Console-Daten haben eine typische Verzögerung von 2–3 Tagen.

```
function runWeeklyGSCUpdate() {
const SITE_URL = "https://www.deinblog.de/";
const request = {
startDate: formatDate(letzteWoche().start),
endDate: formatDate(letzteWoche().ende),
dimensions: ["query"],
rowLimit: 15
};
const response = SearchConsole.Searchanalytics.query(request, SITE_URL);
// Werte in Tab "Search-Console-Wochenübersicht" schreiben
}

function setupWeeklyTriggerGSC() {
ScriptApp.newTrigger("runWeeklyGSCUpdate")
.timeBased()
.onWeekDay(ScriptApp.WeekDay.WEDNESDAY)
.atHour(7)
.create();
}
```

Schritt 3.4 — Explorative Sichtprüfung vor dem ersten Live-Lauf

- Skript einmal manuell über „Ausführen" testen, nicht direkt über den Trigger
- Ergebnis-Zeilen im Sheet stichprobenhaft mit dem GA4-Interface abgleichen
- Erst danach den zeitgesteuerten Trigger aktivieren

4. Den Claude Skill erstellen

Ein Skill besteht aus einer SKILL.md-Datei: einem Beschreibungsblock (wann soll der Skill greifen?) und einer strukturierten Anweisung (was soll passieren?). Zwei bewährte Wege, ihn zu bauen:

Weg A — Interview-Methode

Claude wird gebeten, gezielt nachzufragen, bevor der Skill entsteht. Beispiel-Prompt:

```
"Ich möchte einen Skill erstellen, der aus meinen Blog-Reporting-Daten einen Wochenbericht
baut. Interviewe mich zu allen Details, die du dafür brauchst, eine Frage nach der anderen.
Erstelle den Skill erst, wenn alles geklärt ist."
```

Claude fragt daraufhin typischerweise: Wer liest den Bericht? Welche Tabs/Datenquellen fließen ein? Welches Ausgabeformat (Text, PDF, beides)? Welche Sonderfälle (fehlende Daten, Feiertage)?

Weg B — Aus der erledigten Aufgabe heraus

Ein Reporting wird einmal manuell mit Claude durchgesprochen. Anschließend die Bitte: „Mach aus diesem Ablauf einen wiederverwendbaren Skill." Claude destilliert daraus Struktur und Regeln.

Minimal-Struktur einer SKILL.md für Blog-Reporting

```
---
name: blog-performance-reporting
description: >
Erstellt ein Performance-Reporting für den Blog aus dem Google Sheet
"Blog Reportings" für eine Kalenderwoche oder einen Zeitraum, inkl.
Handlungsempfehlungen und PDF-Export. Nutze diesen Skill bei Anfragen
wie "erstell das Reporting für KW X" oder "mach den Monatsreport".
---

# Blog Performance Reporting

## Schritt 1 — Zeitraum klären
Frage nach Kalenderwoche(n) bzw. Zeitraum und Vergleichsperiode
(Default: Vorperiode gleicher Länge).

## Schritt 2 — Daten holen
Google-Drive-Tools nutzen, Sheet "Blog Reportings" öffnen,
relevante Tabs nach KW filtern.

## Schritt 3 — Reporting erstellen
Reihenfolge: Executive Summary, Traffic & Engagement, Channel-Mix,
Content-Performance je Artikel, SEO/Suche, Social Media,
Redaktionskalender-Check, Handlungsempfehlungen (Tabelle:
Quick-Fix / Mittelfristig / Workflow-Änderung).

Handlungsempfehlungen müssen aus den beobachteten Zahlen
abgeleitet sein, keine generischen Tipps.

## Schritt 4 — PDF-Präsentation
Nach dem Text-Report zusätzlich eine kompakte Präsentation bauen
(1 Slide pro Abschnitt) und als PDF bereitstellen.
```

Die SKILL.md wird als Datei in den Claude-Projektskills abgelegt. Danach erkennt Claude automatisch, wann der Skill greifen soll — auf Basis der description im Kopf der Datei.

5. Das fertige Reporting durchlaufen lassen

Sobald Sheet, Automatisierung und Skill stehen, reicht eine kurze Anfrage im Chat:

```
"Erstell mir das Reporting für KW 27–31."
```

Claude durchläuft daraufhin automatisch:

- Zeitraum und Vergleichsperiode bestätigen
- Relevante Tabs aus dem Sheet lesen und nach KW filtern
- Kennzahlen mit Delta-Wert zur Vergleichsperiode berechnen (%, Trendpfeil)
- Alle acht Report-Abschnitte befüllen — inkl. Handlungsempfehlungen als Tabelle
- Eine Präsentation (PPTX) bauen und zu PDF konvertieren

Beispielhafter Ausschnitt (fiktive Daten, Blog „Contentspur“)

Kennzahl	Zeitraum	Vorperiode	Delta
Aktive Nutzer	39	14	+179 % ↑
Sessions	30	12	+150 % ↑
GSC-Impressionen	592	328	+80 % ↑
LinkedIn-Reichweite	170	347	-51 % ↓

Alle Zahlen in dieser Tabelle sind frei erfunden und dienen nur der Veranschaulichung der Report-Struktur.

Handlungsempfehlungen — Format

Kategorie	Empfehlung	Aufwand
Quick-Fix	Liegen gebliebenen Entwurf fertigstellen und veröffentlichen	gering
Mittelfristig	Content-Rhythmus wiederherstellen (mind. 1 Artikel/Monat)	mittel
Workflow-Änderung	Entwurfs-Status-Tracking im Redaktionskalender ergänzen	einmalig

Checkliste vor dem ersten Live-Durchlauf

- Sheet-Tabs enthalten mind. 2–3 Wochen an Testdaten
- Apps-Script-Trigger laufen zuverlässig (Kontrolle über „Trigger"-Menü in Apps Script)
- SKILL.md ist im Claude-Projekt hinterlegt und wird bei der Testanfrage korrekt erkannt
- Erster Testlauf mit fiktiven oder alten Daten, bevor reale Wochenzahlen verwendet werden

---

Diese Anleitung ist ein Begleitdokument zum Blogartikel „Performance Reporting mit Claude erstellen“ auf Konversion Pilot. Alle Beispielwerte sind fiktiv.