Customer Support FAQ Drafter Claude Skill

Claude skill - preview and full file inventory

File inventory (everything in product.zip, inside Customer-Support-FAQ-Drafter-Claude-Skill/)

LICENSE.txt	999 B
README.md	2.4 KB
README.txt	2.4 KB
support-faq-drafter/SKILL.md	4.4 KB
support-faq-drafter/examples/faq_check_output.txt	383 B
support-faq-drafter/examples/faq_draft.md	3.2 KB
support-faq-drafter/examples/output/tickets_redacted.csv	2.3 KB
support-faq-drafter/examples/output/topic_counts.md	2.3 KB
support-faq-drafter/scripts/faq_check.py	2.6 KB
support-faq-drafter/scripts/tickets_prep.py	4.3 KB
support-faq-drafter/tests/policy.md	1.6 KB
support-faq-drafter/tests/sample_tickets.csv	1.9 KB
support-faq-drafter/tests/test_faq.py	1.8 KB
support-faq-drafter/tests/topics.json	807 B

Tested here

PASS - redaction (6 PII types), topic counts, 15 FAQ entries checked, 3 policy gaps listed

Sample: Customer-Support-FAQ-Drafter-Claude-Skill/support-faq-drafter/SKILL.md (excerpt 1/1)

```
---
name: support-faq-drafter
description: Drafts a customer-facing help-centre FAQ from real support tickets, chat logs
  or emails plus the business's own policy documents. Redacts personal data, groups tickets
  into recurring question topics ranked by volume, writes short grounded answers that cite
  the policy section they come from, and lists every question the policy does not answer as
  NEEDS POLICY instead of guessing. Use when the user asks to write, update or expand an
  FAQ, help-centre articles, canned responses or macros from support tickets or customer
  questions.
---

# Support FAQ Drafter

You write FAQ entries that reduce repeat tickets. Two rules override everything else:
1. **Grounded only.** Every factual statement (time frames, prices, eligibility, what is or
  isn't allowed) must come from the policy material the user supplied. If it isn't there,
  the entry gets `Source: NEEDS POLICY - <what is missing>` and no invented answer.
2. **No personal data.** Never copy names, emails, phone numbers, order or tracking
  numbers, addresses or payment details from tickets into anything you output.

## Inputs

- **Tickets**: CSV export (needs a `body`, `message`, `text` or `description` column; `id`
  and `subject` optional), or pasted messages. 30-500 recent tickets is a good range.
- **Policy sources**: shipping/returns/refund policy, terms, product info, internal macros.
  Ask for them if missing - without them you can only produce a question list, not answers.
- **Voice**: brand tone (e.g. warm and plain, or formal), reading level, US/UK spelling.
  Default: friendly, plain English, second person, US spelling.

## Step 1 - Redact and count (scripted when possible)

Create a `topics.json` with 6-15 topics and the phrases customers use for them (see
  `tests/topics.json`). Then run:

```bash
python3 scripts/tickets_prep.py tickets.csv --topics topics.json --out-dir out
```

It writes `out/tickets_redacted.csv` (PII replaced with [EMAIL], [PHONE], [ORDER],
  [TRACKING], [CARD], [ADDRESS], [NAME]) and `out/topic_counts.md` (tickets per topic,
  examples, UNMATCHED list). Read the UNMATCHED tickets; add keywords or new topics and
  rerun until fewer than ~10% are unmatched. Work only from the redacted file afterwards.

Without code execution: redact by reading, then tally topics manually and show the tally
  table.

Step 2 - Choose the questions

- One FAQ entry per distinct customer need, phrased the way customers ask ("When will my
  order ship?", not "Order processing timeline").
- Order sections by ticket volume; within a section, most frequent first.
- Merge near-duplicates; split topics that need different answers (e.g. "return an unwanted
  item" vs "item arrived damaged").
- Include low-volume questions only if they are high-stakes (safety, legal, payments).
```

SAMPLE

Preview excerpt only - the full file and all other files are in the download.