

TEAM 8 - CSN Patient Navigator Data Foundation

LSE Data Analytics Career Accelerator - Employer Project

This notebook follows the team's own 8-stage step-by-step process. **Stages 1-2 (scope definition and source selection) are already done and recorded in**

CSN_DATASHEET.xlsx's approved URL list, so this notebook picks up at Stage 3 - stage numbers below match the team's process document, not a fresh count from 1.

Each stage has a collapsible explanation (click "click to expand/collapse") covering what it's for, what it does, and what it feeds into later - so future-you can understand the point of a section at a glance.

Reminder: we are students on the LSE Data Analytics Career Accelerator building a *data foundation*, not a finished product.

Setup — Environment, Configuration & Constants

```
In [1]: #---Install---
!pip install -q sentence-transformers scikit-learn sumy

# --- Core Libraries ---
import requests
from bs4 import BeautifulSoup
import pandas as pd
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
import sqlite3
import time
import re
import os
from datetime import date, datetime
from pathlib import Path

# --- Summarisation (sumy TextRank - used in draft_summary(), Stage 6) ---
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.text_rank import TextRankSummarizer

# --- Translation (DeepL - agreed tool) ---
# pip install deepl (run once in Colab: !pip install deepl)
import deepl

# --- Visualisation ---
import matplotlib.pyplot as plt

# --- Archiving ---
import shutil

# --- Excel column hiding/locking ---
```

```
import openpyxl
from openpyxl.utils import get_column_letter

pd.set_option("display.max_colwidth", 120)
```

```
In [2]: # --- Configuration ---
# =====
# Database
# =====
DB_PATH = "csn_dataset.db"          # lightweight, appendable SQL database
SCRAPE_DATE = date.today().isoformat()

# =====
# DeepL translator (Corrected Subdomain Routing - Trevor)
# =====
# Preferred: set DEEPL_API_KEY as an environment variable - a real key should ne
# shared document. Fallback: if you're working solo, replace the string below wi
# key instead - just don't share/commit the file once you've done that. No need
# anything in or out - the environment variable is used automatically if it's se
MANUAL_API_KEY_FALLBACK = "0ac8c0ef-25a1-49a7-b204-c7e5099c6335:fx"

DEEPL_API_KEY = os.environ.get("DEEPL_API_KEY") or (
    MANUAL_API_KEY_FALLBACK if MANUAL_API_KEY_FALLBACK != "YOUR_DEEPL_API_KEY_HE
)

if DEEPL_API_KEY:
    # Explicitly separate the free server endpoint from the pro server endpoint
    server_url = "https://api-free.deepl.com" if DEEPL_API_KEY.endswith(":fx") e
    translator = deepl.Translator(DEEPL_API_KEY, server_url=server_url)
else:
    translator = None

# =====
# Scraping
# =====
REQUEST_HEADERS = {"User-Agent": "Mozilla/5.0 (CSN Employer Project research; ed
REQUEST_DELAY_SECONDS = 1.5 # polite to the source sites

# =====
# Dataset schema & scope constants
# NOTE: Required feilds updated - expanded!
# =====
REQUIRED_FIELDS = [
    # Source metadata - where did this come from
    "source_organisation",
    "original_url",
    "source_page_title",
    "english_page_title",
    "scrape_date",
    # Raw content - what does it say
    "original_dutch_text",
    "english_translation",
    # Classification - how is it tagged
    "topic",
    "patient_journey_stage",
    "stakeholder",
    # Patient-facing output - what does it mean for the patient
    "community_question",
    "summary_answer_text",
    "action_next_step",
```



```

    "priority_flag",
]

# =====
# THE LOOP – Excel is the single input and single output
# =====
# Every row in the working Excel carries a SCRAPE flag:
#   SCRAPE == "SCRAPE" -> row is fully regenerated from scratch this run (Stage
#   anything else (default "N") -> row passes through untouched, exactly as it
# This is checked in exactly ONE place (Stage 3, step "The Loop Gate") and the u
# never enter a single Stage 3-6 function again after that - see that cell for t
#
# Column layout below matches CSN_DATASHEET.xlsx exactly (its DATASHEET tab, col
# its INSTRUCTIONS tab, full field list on its METADATA tab) - NOT just REQUIRED
# That file also has INSTRUCTIONS / REFERENCE FOR EDITS / META DATA tabs alongsi
# the write-back step must leave those three completely alone and only touch DAT
CURRENT_EXCEL_PATH = "CSN_DATASHEET.xlsx"
DATASHEET_SHEET_NAME = "DATASHEET"
ARCHIVE_DIR = "archive"
SCRAPE_FLAG_COLUMN = "SCRAPE"
SCRAPE_FLAG_VALUE = "SCRAPE"

# Working/QA columns that live on df_structured during the session but are NOT i
# REQUIRED_FIELDS - kept in the Excel as fully HIDDEN columns so a technical tea
# still spot-check confidence scores etc., without a CSN volunteer ever seeing t
# (Column list + order taken directly from CSN_DATASHEET.xlsx's META DATA tab.)
QA_COLUMNS = [
    "url_valid", "missing_critical_field", "topic_category", "topic_similarity",
    "community_question_header", "header_similarity", "question_similarity",
    "action_type", "content_needs_review", "advice_tone_flag",
]

# Full column order written to the DATASHEET tab: REQUIRED_FIELDS, then QA_COLUM
# then SCRAPE last - matches CSN_DATASHEET.xlsx's DATASHEET tab header row exact
DATASHEET_COLUMNS = REQUIRED_FIELDS + QA_COLUMNS + [SCRAPE_FLAG_COLUMN]

# Columns a CSN volunteer is allowed to hand-edit, per CSN_DATASHEET.xlsx's INST
# Section 3 ("What you can edit"). Everything else is READ-ONLY (locked, still v
# QA_COLUMNS are hidden entirely, on top of being locked - see write_protected_e
EDITABLE_COLUMNS = {
    "original_url", "topic", "patient_journey_stage", "stakeholder", "community_
    "summary_answer_text", "action_next_step", "priority_flag", SCRAPE_FLAG_COLU
}
HIDDEN_COLUMNS = set(QA_COLUMNS)

```

Note: reading CSN_DATASHEET.xlsx directly — iteration history

This block defines the full `DATASHEET_COLUMNS` schema (26 columns: 14 required fields, 11 hidden QA columns, SCRAPE flag) rather than just the original 14 `REQUIRED_FIELDS` — needed once the notebook moved to reading `CSN_DATASHEET.xlsx` directly instead of a separate Intake Sheet. See "a) The Loop Gate" in Stage 3 for the full picture of how this schema is used.

STATUS: RESOLVED

Stage 3 - Data Sourcing & Scraping

► **What this stage is for** (click to expand/collapse)

a) The Loop Gate — read the working Excel, archive it, split rows by SCRAPE flag

Reads `CSN_DATASHEET.xlsx` from the `DATASHEET` tab, archives a timestamped backup before touching anything, fills in any missing columns from `DATASHEET_COLUMNS` (e.g. so a volunteer who's only typed in a new URL doesn't break anything downstream), then splits every row by its `SCRAPE` flag: flagged rows go into `df_to_process` and run through the rest of Stages 3-6; everything else goes into `df_untouched`, set aside exactly as it was read, and rejoined at the end of Stage 6. If `CSN_DATASHEET.xlsx` isn't found at `CURRENT_EXCEL_PATH`, the run stops here with a clear message rather than failing later with a confusing error.

```
In [3]: # =====
# OLD: Load confirmed URLs from the Intake Sheet
# We are no longer reading from the intake sheet so this entire section goes.
# =====
#INTAKE_SHEET_PATH = "Team8_Source_Intake_Sheet_v2.xlsx"
#
#SOURCE_TABS = ["Kanker.nl", "Rijksoverheid.nl", "Thuisarts.nl", "Zorginstituutn
#
#def load_confirmed_urls(path: str) -> List:
#    """Load only the URL from every Intake Sheet row that has cleared BOTH revi
#    Re-run this cell any time the sheet is updated - no copy-pasting required."
#    try:
#        sheets = pd.read_excel(path, sheet_name=SOURCE_TABS, header=2) # heade
#    except FileNotFoundError:
#        print(f"'{path}' not found in this environment - check the path or uplo
#        return []
#
#    urls = []
#    for tab, df in sheets.items():
#        # Every confirmed row needs all three of these columns present - if one
#        # renamed slightly, skip that tab with a warning rather than silently r
#        # confirmed rows for it (Trevor's fix - see Methodology Log, Stage 3)
#        required_cols = {"Page URL", "Reviewer 2 check", "Ready to scrape?"}
#        if not required_cols.issubset(df.columns):
#            missing = required_cols - set(df.columns)
#            print(f" [WARN] '{tab}' tab is missing column(s) {missing} - skip
#            continue
#
#        # Only rows both reviewers have actually signed off on
#        confirmed = df[
#            (df.get("Reviewer 2 check") == "Confirmed") &
#            (df.get("Ready to scrape?") == "Yes")
#        ]
#
#        for raw_url in confirmed["Page URL"].dropna():
#            url = str(raw_url).strip().split("#")[0] # strip fragments like "#
#            if url and url != "nan":
#                urls.append(url)
#
#    # De-duplicate, keeping first occurrence
#    seen, cleaned = set(), []
```

```

#     for u in urls:
#         if u not in seen:
#             cleaned.append(u)
#             seen.add(u)
#     return cleaned

# -----
# NEW: THE LOOP GATE – the ONLY place SCRAPE is checked. Everything from here to
# end of Stage 6 only ever sees df_to_process (SCRAPE == "SCRAPE"). df_untouched
# is set aside right now, before any processing, and never passed into a single
# Stage 3-6 function - so it CANNOT be accidentally changed. It's stitched back
# in at Stage 6 step k), right before the final upsert/export.
# -----

# Creates an archive version of the excel datasheet before doing any processing
Path(ARCHIVE_DIR).mkdir(exist_ok=True)

if Path(CURRENT_EXCEL_PATH).exists():
    # Archive-before-touch, every run - the rollback path if anything goes wrong
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    archive_path = Path(ARCHIVE_DIR) / f"run_on_{timestamp}.xlsx"
    shutil.copy(CURRENT_EXCEL_PATH, archive_path)
    print(f"Archived working Excel to {archive_path}")

    # Read the DATASHEET tab specifically - CSN_DATASHEET.xlsx also has INSTRUCT
    # REFERENCE FOR EDITS and META DATA tabs alongside it, which are never touch
    df_master = pd.read_excel(CURRENT_EXCEL_PATH, sheet_name=DATASHEET_SHEET_NAME)

    # Soft-delete: a volunteer can't delete a protected row directly in Excel, so
    # blank original_url is treated as "delete this row" and dropped here instead
    before = len(df_master)
    df_master = df_master[
        df_master["original_url"].notna() & (df_master["original_url"].astype(str)
        ].reset_index(drop=True)
    ]
    if before - len(df_master):
        print(f"Dropped {before - len(df_master)} soft-deleted row(s) (blank ori

    # A brand-new row (volunteer pasted a URL, nothing else yet) may be missing
    # QA/REQUIRED_FIELDS columns entirely on read - align to the full DATASHEET
    # before splitting, so is_flagged and everything downstream can rely on ever
    # existing, even if empty.
    for col in DATASHEET_COLUMNS:
        if col not in df_master.columns:
            df_master[col] = ""
    if df_master[SCRAPE_FLAG_COLUMN].isna().any():
        df_master[SCRAPE_FLAG_COLUMN] = df_master[SCRAPE_FLAG_COLUMN].fillna("N")

    is_flagged = df_master[SCRAPE_FLAG_COLUMN].astype(str).str.strip().str.upper
    df_to_process = df_master[is_flagged].copy()
    df_untouched = df_master[~is_flagged].copy() # <- carried through Stage 3-

    print(f"{len(df_master)} row(s) loaded from '{DATASHEET_SHEET_NAME}' tab of
    print(f" {len(df_to_process)} row(s) flagged SCRAPE -> will run the FULL pi
    print(f" {len(df_untouched)} row(s) NOT flagged -> pass through untouched,

    CORE_SOURCES = df_to_process["original_url"].dropna().astype(str).tolist()
else:
    raise FileNotFoundError(
        f"'{CURRENT_EXCEL_PATH}' not found. This pipeline now reads CSN_DATASHEET
        f"single input – check the file exists at that path (and that CURRENT_EX

```

```

        f"Setup cell points to the right location) before re-running."
    )

print(f"\n{len(CORE_SOURCES)} URL(s) will be scraped this run.")

```

Archived working Excel to archive\run_on_20260831_210051.xlsx
 144 row(s) loaded from 'DATASHEET' tab of CSN_DATASHEET.xlsx.
 0 row(s) flagged SCRAPE -> will run the FULL pipeline this run.
 144 row(s) NOT flagged -> pass through untouched, unaltered.

0 URL(s) will be scraped this run.

Observation — The Loop Gate:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

144 rows loaded, all 144 flagged SCRAPE, 0 passed through untouched — confirms this run is genuinely operating as a full first-run simulation, not a partial rerun. If any rows had shown up as "NOT flagged -> pass through untouched," that would mean the SCRAPE-flip wasn't applied to every row before this run started, and this notebook's results wouldn't be a true first-run baseline.

The archive step also fired correctly (Archived working Excel to ...) — a rollback copy of whatever was in CSN_DATASHEET.xlsx before this run touched it, created automatically before any processing begins.

Note: Intake Sheet fallback removed

CSN_DATASHEET.xlsx is now the single required input — if it isn't found at CURRENT_EXCEL_PATH, the Loop Gate stops the run here with a clear message rather than continuing into Stage 3d-4g and failing later with a confusing `NameError`.

The original Intake Sheet fallback code is still present above, commented out rather than deleted, so the change is visible directly in the pipeline rather than only in this note.

STATUS: RESOLVED

b) Site selectors & scraping helper functions

```

In [4]: # -----
# Site-specific content selectors
# -----
SITE_CONTENT_SELECTORS = {
    "kanker.nl": ".node__content",
    "rijksoverheid.nl": ".main-content",
    "thuisarts.nl": ".node__content",
    "zorginstituutnederland.nl": ".main-content",
    "nfk.nl": ".o-detail__main-content",
}

# -----
# Fetching
# -----
def fetch_page(url: str, retries: int = 2, retry_delay: float = 2.0):
    """Fetch a page and return a parsed BeautifulSoup object, or None if every a

```

```

Retries before giving up (Trevor's fix - a single timeout used to be treated
a permanent failure)."""
for attempt in range(retries + 1):
    try:
        resp = requests.get(url, headers=REQUEST_HEADERS, timeout=15)
        resp.raise_for_status()
        return BeautifulSoup(resp.text, "html.parser")
    except requests.RequestException as e:
        if attempt < retries:
            time.sleep(retry_delay)
            continue
        print(f" [FAILED after {retries + 1} attempt(s)] {url} -> {e}")
        return None

# -----
# Deriving source organisation and title from a fetched page
# -----
def source_organisation_from_url(url: str) -> str:
    """Derive the source organisation straight from the URL's domain."""
    for key in SITE_CONTENT_SELECTORS:
        if key in url:
            return key
    from urllib.parse import urlparse
    return urlparse(url).netloc.replace("www.", "") or "unknown"

def extract_page_title(soup, fallback: str = "Untitled") -> str:
    """Pull the real page title from <title>, cleaning off site-name suffixes li
    'Page Name | Site Name', with a fallback to the first <h1> if <title> is mis
    if soup.title and soup.title.string:
        raw = soup.title.string.strip()
        for sep in [" | ", " - ", " \u2013 "]:
            if sep in raw:
                raw = raw.split(sep)[0].strip()
        if raw:
            return raw
    h1 = soup.find("h1")
    if h1 and h1.get_text(strip=True):
        return h1.get_text(strip=True)
    return fallback

# -----
# Extracting the main readable text
# -----
def extract_main_text(soup, site_key: str) -> str:
    """ Pull the main readable text using the sites CSS selector.
    Uses select() not select_one() - confirmed necessary on kanker.nl where arti
    can span multiple content blocks sharing the same class. Empty blocks are fi
    joining so a blank first block doesnt hide real content in a later one. Fall
    the selector matches nothing"""

    selector = SITE_CONTENT_SELECTORS.get(site_key, "body")
    containers = soup.select(selector)

    if not containers:
        containers = [soup.body] if soup.body else []
    elif len(containers) > 5:
        # Selectors haven't been checked for reuse elsewhere on the page (sideba
        # "related articles" cards) - a high match count is the warning sign tha
        # happening, since a real article rarely has more than a handful of cont
        print(f" [WARN] {selector} matched {len(containers)} elements on a {sit

```

```

        f"check this isn't picking up unrelated content (sidebar/teaser ca

if not containers:
    return ""

blocks = []
for container in containers:
    for tag in container.select("nav, footer, script, style"):
        tag.decompose()
    block_text = container.get_text(separator="\n", strip=True)
    if block_text: # drop genuinely empty blocks rather than inserting blank
        blocks.append(block_text)

text = "\n\n".join(blocks)
return re.sub(r"\n{3,}", "\n\n", text)

```

```

In [5]: # -----
# EDA: test on ONE page before running the full loop
# -----

if not CORE_SOURCES:
    print("No URLs loaded - check that CSN_DATASHEET.xlsx exists at CURRENT_EXCEL_PATH and that at least one row has its SCRAPE flag set to 'SCRAPE'.")
else:
    test_url = CORE_SOURCES[0]
    soup = fetch_page(test_url)
    if soup is None:
        print(f"Fetch failed for the test URL: {test_url}")
        print("Check the URL is reachable, and check SITE_CONTENT_SELECTORS if that fails")
    else:
        test_org = source_organisation_from_url(test_url)
        test_title = extract_page_title(soup)
        sample_text = extract_main_text(soup, test_org)
        print(f"URL: {test_url}")
        print(f"Derived organisation: {test_org}")
        print(f"Derived title: {test_title}")
        print(f"Extracted {len(sample_text)} characters. Preview:\n")
        print(sample_text[:600])

```

No URLs loaded - check that CSN_DATASHEET.xlsx exists at CURRENT_EXCEL_PATH and that at least one row has its SCRAPE flag set to 'SCRAPE'.

Observation — EDA single-page test:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

- Selector working correctly for kanker.nl (`.node__content`) — no toolbar/breadcrumb leakage
- 5,000 characters extracted — the `extract_main_text()` preview cap, not the full article length (full text captured in the extraction loop)
- Dutch title correctly derived from `<title>` tag: "Een ziekenhuis kiezen bij kanker"
- Source organisation correctly derived from URL: "kanker.nl"
- Content is genuine article body — table of contents, section headings, and paragraph text all present, confirming the selector is hitting the right element

This is the first of 144 pages — confirmed the pipeline is set up correctly before spending time on the full loop.

Note: Site selectors & scraping helper functions — iteration history

Issue: two problems traced to the same root cause — toolbar/sidebar text leaking into scraped content, and multi-section articles being silently truncated.

What was wrong:

1. **Toolbar/sidebar leakage** — selectors in `SITE_CONTENT_SELECTORS` weren't reliably targeting the article body only.
2. **Multi-section truncation** — `select_one()` stopped at the first matching block. Confirmed on a real kanker.nl page where the browser console showed 5 separate `.node__content` blocks; roughly 3 of 4 real content sections were being silently dropped. This was also the cause of 2 fully empty scrapes — real content ("Werknemer en kanker") was present in the raw server HTML but sat in a later block `select_one()` never reached.

Fix (Anique + Will): selectors in `SITE_CONTENT_SELECTORS` re-verified per site via browser console inspection rather than guessed. `select_one()` swapped for `select()`, joining all matching blocks — empty blocks filtered before joining, so a blank first block can't hide real content behind it.

Confirmed (Anique, 13/08/2026): tested against 4 scenarios — empty-first-block, 5-block multi-section, single-block regression, zero-match fallback — all passed before applying.

STATUS: RESOLVED

c) Manual fallback for pages that fail to scrape

If a page fails (blocked request, broken selector, JavaScript-rendered content), paste its visible Dutch text into `MANUAL_FALLBACK_TEXT` below. The extraction loop checks this dict before giving up — the brief explicitly allows this approach, and a manually added page is better than a gap in the dataset.


```
In [6]: # -----
# Manual fallback for pages that fail to scrape
# -----
# The brief's own suggested fallback (manual copy-paste), not a workaround. If a
# in the "[FAILED]" list after running the extraction loop, paste its visible Du
# and re-run - the extraction loop checks this dict before giving up on a failed

MANUAL_FALLBACK_TEXT = {
    # "https://example.com/a-page-that-wont-scrape": "Pasted Dutch text goes her
}
```

d) Full extraction loop

Runs through every URL in `df_to_process` (the rows flagged `SCRAPE` by the Loop Gate above), scraping each page using the selectors and helpers defined in b) and c). If nothing is flagged this run, this loop produces an empty `df_raw` with the correct columns still in place (not zero columns) — a normal, non-crashing state for a run that only edits existing rows.

```
In [7]: # -----
# Full extraction Loop - organisation and title are both derived, not typed by h
# -----

RAW_COLUMNS = [
    "source_organisation",
    "source_page_title",
    "original_url",
    "scrape_date",
    "original_dutch_text",
]

raw_records = []
failed_urls = []

for url in CORE_SOURCES:
    soup = fetch_page(url)
    if soup is None:
        if url in MANUAL_FALLBACK_TEXT:
            text = MANUAL_FALLBACK_TEXT[url]
            site_key = source_organisation_from_url(url)
            title = "Untitled (manual fallback)"
        else:
            failed_urls.append(url)
            # continue skips the sleep at the end of the loop, so it's repeated
            # every URL gets exactly one pause, success or failure (Trevor's fix
            time.sleep(REQUEST_DELAY_SECONDS)
            continue
        else:
            site_key = source_organisation_from_url(url)
            title = extract_page_title(soup)
            text = extract_main_text(soup, site_key)

    raw_records.append({
        "source_organisation": site_key,
        "source_page_title": title,
        "original_url": url,
```

```

        "scrape_date": SCRAPE_DATE,
        "original_dutch_text": text,
    })
    time.sleep(REQUEST_DELAY_SECONDS)

# columns=RAW_COLUMNS is the actual fix - it forces these 5 columns to exist eve
# raw_records is [], instead of pd.DataFrame(raw_records) silently returning a D
# with no columns at all when nothing was scraped this run.
df_raw = pd.DataFrame(raw_records, columns=RAW_COLUMNS)

if df_raw.empty:
    print("Nothing was scraped this run - no rows were flagged SCRAPE in the wor
    print("df_raw has 0 rows but the correct columns, so Stage 3e/4f/4g will ski
else:
    print(f"Scraped {len(df_raw)} pages.")
if failed_urls:
    print(f"{len(failed_urls)} page(s) failed and have no manual fallback yet -
    for u in failed_urls:
        print(f"    - {u}")

print(df_raw.head())

# --- Preserving rows whose rescrape failed with no manual fallback ---
# What was wrong: a URL that failed in the loop above (no MANUAL_FALLBACK_TEXT
# entry) was silently skipped via `continue` - no record was ever added to
# raw_records, so the row vanished from every downstream output entirely. Not
# just the URL: its old translation, tags, and any manual corrections went
# with it too.
#
# Fix: carry that row's previous data (as it stood in df_to_process, before
# this run touched anything) straight through unchanged - the same way
# df_untouched already passes unflagged rows through untouched. Its SCRAPE
# flag comes along unchanged too (still "SCRAPE"), so the row is automatically
# picked up and retried next run without anyone needing to re-flag it by hand.
df_failed = df_to_process[df_to_process["original_url"].isin(failed_urls)].copy()

if not df_failed.empty:
    print(f"{len(df_failed)} failed URL(s) had no manual fallback - "
          f"their PREVIOUS data will be kept as-is in the output, and they'll "
          f"retry automatically next run (SCRAPE flag untouched).")

```

Nothing was scraped this run - no rows were flagged SCRAPE in the working Excel.
df_raw has 0 rows but the correct columns, so Stage 3e/4f/4g will skip cleanly be
low.

Empty DataFrame

Columns: [source_organisation, source_page_title, original_url, scrape_date, orig
inal_dutch_text]

Index: []

Note: empty-DataFrame column guard — iteration history

Issue: when every row in the working Excel is left un-flagged (`SCRAPE != "SCRAPE"`),
`CORE_SOURCES` (built in the Loop Gate, Stage 3a) is an empty list.

What was wrong: the extraction loop then never executes, so `raw_records` stays `[]`.
On its own, `pd.DataFrame([])` turns that into a DataFrame with **zero columns** — not
just zero rows — so the next cell (3e, completeness check) and Stage 4's translation pass

(4f) would both hit a `KeyError` looking for a column (`original_dutch_text`) that doesn't exist yet. "Nothing flagged to scrape this run" is a perfectly normal state (a test run, or a run that only edits existing rows) — it should print a clear message and flow through cleanly, not crash several cells later.

Fix: the expected columns are now named explicitly and passed into the `DataFrame` constructor, guaranteeing `df_raw` always has these 5 columns, whether it ends up with 144 rows or 0.

STATUS: RESOLVED

e) Completeness check

Confirms every scraped page produced real text — catches empty rows at the source before translation begins.

```
In [8]: # -----
# Immediate completeness check: did every scraped page actually get real text?
# -----
# Catches this at the source, right after scraping.
# --- Trevor (21/08) note: safe on 0 rows now that df_raw always keeps its column
if df_raw.empty:
    print("df_raw is empty - nothing was scraped this run, so there's nothing to
else:
    empty_text_rows = df_raw[df_raw["original_dutch_text"].str.strip() == ""]
    print(f"{len(empty_text_rows)} / {len(df_raw)} scraped page(s) have EMPTY or
    for _, row in empty_text_rows.iterrows():
        print(f" - {row['source_organisation']}: {row['source_page_title']} ({r
```

df_raw is empty - nothing was scraped this run, so there's nothing to check here.

Observation — Full extraction loop + completeness check:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

- 144 / 144 pages scraped successfully — 0 failures, 0 manual fallback entries needed
- All 5 sources represented: kanker.nl, rijksoverheid.nl, thuisarts.nl, zorginstituutnederland.nl, nfk.nl
- Scrape date recorded as 2026-08-28 on all rows
- 0/144 rows with empty `original_dutch_text` - All populated

Stage 4 - Translation & Content Processing

► **What this stage is for** (click to expand/collapse)

a) Glossary setup (DeepL)

```
In [9]: # -----
# Layer 1: DeepL glossary (primary fix)
# -----
# DeepL uses sentence context to decide how to translate a word, so the same Dut
```

```

# come out differently across pages - "huisarts" might be "GP" in one sentence a
# doctor" or "general practitioner" in another, both correct but inconsistent. T
# forces one fixed English term for each Dutch entry regardless of context, prev
# inconsistency at translate-time rather than patching it afterwards.
# DeepL's glossary is grammar-aware (handles "huisarts" / "huisartsen" / "de hui
# automatically), which a post-hoc string match can't do reliably.
# **NOTE:** If deepl account doesn't support glossaries, this cell fails safely
# scan_terminology_variants() diagnostic further down will flag any inconsistenc

```

```

GLOSSARY_TERMS = {
    "huisarts": "GP",
    "ziekenhuis": "hospital",
    "zorgverzekeraar": "insurer",
    "eigen risico": "deductible",
    "verwijsbrief": "referral letter",
    "eigen bijdrage": "co-payment",
    "eigen deel": "co-payment",
    "spoedeisende hulp": "emergency department",
    "seh": "emergency department",
    "fysiotherapie": "physiotherapy",
    "diëtist": "dietitian",
    "samen beslissen": "deciding together",
    "wachtijd": "waiting time",
    "basispakket": "basic insurance package",
    "aanvullende verzekering": "supplementary insurance",
    "praktijkondersteuner": "practice nurse",
    "verwijzing": "referral",
    "medisch-specialistische zorg": "specialist care",
    "zorgnota": "healthcare invoice",
    "rekening": "healthcare invoice",
    "geneeskundige ggz": "mental health care",
    "ggz": "mental health care",
    "schone lei": "clean slate",
}

```

```

# Free-tier accounts only get a small number of glossary slots (Trevor's find) -
# the same named glossary across runs instead of creating a new one every time.
# the EXISTING glossary's actual terms against GLOSSARY_TERMS above: if they've
# someone added a new term to the dict since the glossary was last created), the
# deleted and replaced rather than silently reused with stale entries - net glos
# stays at exactly one slot either way, it just stays correct as well as efficie

```

```

GLOSSARY_NAME = "CSN Dutch-English Terms"

```

```

glossary = None

```

```

if translator is not None:

```

```

    try:

```

```

        existing = [g for g in translator.list_glossaries() if g.name == GLOSSAR
        if existing:

```

```

            candidate = existing[0]

```

```

            current_entries = translator.get_glossary_entries(candidate)

```

```

            if current_entries == GLOSSARY_TERMS:

```

```

                glossary = candidate

```

```

                print(f"Reusing existing glossary (terms unchanged): {glossary.g

```

```

            else:

```

```

                translator.delete_glossary(candidate)

```

```

                glossary = translator.create_glossary(

```

```

                    GLOSSARY_NAME, source_lang="NL", target_lang="EN-GB", entrie

```

```

                )

```

```

                print(f"GLOSSARY_TERMS has changed since last run - replaced glo

```

```

            else:

```

```

glossary = translator.create_glossary(
    GLOSSARY_NAME, source_lang="NL", target_lang="EN-GB", entries=GL
)
print(f"Glossary created: {glossary.glossary_id}")
except Exception as e:
    print(f" [glossary setup failed - falling back to post-processing only]

```

Reusing existing glossary (terms unchanged): 5c7abca6-191a-4a82-b729-61e4338cd6a9

Observation — Glossary setup: Reusing existing glossary fresh. All 20 terms from `GLOSSARY_TERMS` successfully registered with DeepL.

Note: Glossary cell — iteration history

Original version (Trevor): created a brand-new glossary on every single run. Hit the free-tier glossary slot limit quickly, since each run consumed a slot without cleaning up the previous one.

First fix (Trevor — reuse-by-name): looked up the existing glossary by name and reused it instead of creating a new one. Solved the slot limit problem but introduced a new gap — it only checked that the glossary's *name* matched, not its *contents*. So if anyone edited `GLOSSARY_TERMS`, the old glossary on DeepL's side would still get silently reused with the stale entries.

Second fix (Anique — drift-check added): now fetches the existing glossary's actual entries and compares them against `GLOSSARY_TERMS`. If they match, reuses it for free. If they don't, deletes the old one and creates a fresh one — so editing `GLOSSARY_TERMS` actually takes effect on the next run. Net slot usage stays at exactly one at all times.

Practical effect: just edit `GLOSSARY_TERMS` as normal. The cell detects changes and rebuilds the glossary automatically — no need to manually delete anything on DeepL's side.

Testing history:

- *07/08/26 — create path confirmed:* first real API key run printed `Glossary created: ...` — confirms `create_glossary()` works against DeepL's live API, not just in theory.
- *09/08/26 — reuse path confirmed:* after a kernel restart, reran with `GLOSSARY_TERMS` unchanged. Printed `Reusing existing glossary (terms unchanged): ...` — identical ID confirms it found and reused the existing glossary rather than creating a duplicate.
- *10/08/26 - drift-detection path confirmed:* after adding new entries to `GLOSSARY_TERMS` (targeting terminology inconsistencies found in the Stage 4h scan), the cell printed `GLOSSARY_TERMS has changed since last run - replaced glossary: ...` — confirms the content comparison correctly detected the change and rebuilt the glossary, rather than silently reusing stale entries.
- *14/08/26 — EN/EN-GB mismatch found and fixed:* the terminology scan in section h) still flagged inconsistency despite these entries. Root cause: the glossary was

originally created with `target_lang="EN"` while translations used `target_lang="EN-GB"` — DeepL treats these as different languages and silently ignored the glossary as a result. All three creation paths now use `"EN-GB"` to match `translate_text()`.

STATUS: RESOLVED

b) Chunking and translation functions

```
In [10]: # -----
# Translation
# -----

def chunk_text(text: str, max_chars: int = 4500) -> list:
    """Splits text into pieces at SENTENCE boundaries, never mid-sentence - a word
    split can still land between two halves of one sentence, producing an awkward
    (confirmed: 4 of 5 seams on the longest real page showed this exact pattern)
    sentences into each chunk, falling back to word-boundary splitting only for
    that's itself longer than max_chars (rare, but must still be handled safely)
    sentences = re.split(r"(?<=[.!?])\s+", text.strip())
    chunks, current, current_len = [], [], 0

    for sentence in sentences:
        added_len = len(sentence) + (1 if current else 0)
        if current_len + added_len > max_chars and current:
            chunks.append(" ".join(current))
            current, current_len = [], 0

        if len(sentence) > max_chars:
            # A single sentence longer than the limit - fall back to word-safe split
            # just for this one sentence, so nothing crashes or gets silently truncated
            words = sentence.split(" ")
            piece, piece_len = [], 0
            for word in words:
                w_len = len(word) + (1 if piece else 0)
                if piece_len + w_len > max_chars and piece:
                    chunks.append(" ".join(piece))
                    piece, piece_len = [word], len(word)
                else:
                    piece.append(word)
                    piece_len += w_len
            if piece:
                current, current_len = piece, piece_len
        else:
            current.append(sentence)
            current_len += added_len

    if current:
        chunks.append(" ".join(current))
    return chunks

def translate_text(dutch_text: str) -> str:
    """Translate Dutch to English via DeepL, using the glossary if it's available.
    long text at sentence boundaries rather than truncating, so nothing is lost
    and sentences aren't cut mid-thought across chunk boundaries."""
    if translator is None or not dutch_text:
        return ""
    try:
```



```

translated_chunks = [
    translator.translate_text(
        chunk, source_lang="NL", target_lang="EN-GB", glossary=glossary
    ).text
    for chunk in chunk_text(dutch_text)
]
return " ".join(translated_chunks)
except Exception as e:
    print(f" [translation failed] {e}")
return ""

```

Note: chunk_text() — iteration history (Trevor → Anique)

1. Starting point: Trevor's chunking fix (chunk instead of truncate) — core idea unchanged.

2. Problem found: the original chunker cut at a fixed character count regardless of content, which could split a word in half. Confirmed: "mensen" became "me" + "nsen", sent to DeepL as two garbled fragments.

3. First fix: `chunk_text()` splits at word boundaries instead — same ~4500 character limit, never cutting a word mid-character.

4. Tested: rejoined chunks and confirmed they match the original text exactly, word for word — splitting is lossless.

5. Cleanup: removed old commented-out `translate_text / needs_manual_review` versions. `needs_manual_review()` and `MEDICAL_TERM_FLAGS` were removed entirely shortly after — the flag was catching ~87% of rows, meaning the premise (that specific medical terms could be reliably flagged for extra review via keyword matching) didn't hold for this dataset. Nothing else in the pipeline depended on their NaN-safety logic, so removing them was a clean cut with no knock-on changes needed elsewhere.

6. New problem found: ran the full multi-chunk translate-and-rejoin on the longest real page (25,675 characters, 6 chunks, 5 seams). Word-boundary splitting held up for content integrity, but 4 of 5 seams read awkwardly — sentences cut mid-thought, next chunk starting disconnected. Root cause: word-boundary chunking guarantees no word is split, but says nothing about sentences.

7. Second fix (Anique): rewrote `chunk_text()` to split at sentence boundaries — packs whole sentences per chunk, word-safe fallback only for a single sentence longer than the limit itself. See the verification section (### c) below) for test results and real-data confirmation.

STATUS: RESOLVED

c) EDA: translation preview on 2 rows before the full pass

```

In [11]: # -----
# EDA: try translation on 2 rows before running the full pass
# -----

```

```
# NOTE: this calls DeepL directly (not cached) - costs ~800 characters per run.
# Skip this cell if quota is limited and you're confident the translation setup
# -----
if translator is None:
    print("DEEPL_API_KEY not set yet - add your key in the Setup section before")
elif df_raw.empty:
    print("df_raw is empty - nothing scraped yet in Stage 3, so there's nothing")
else:
    for i, row in df_raw.head(2).iterrows():
        print(f"--- {row['source_page_title']} ---")
        preview_input = chunk_text(row["original_dutch_text"], max_chars=400)[0]
        sample = translate_text(preview_input)
        print("EN preview:", sample[:200])
        print()
```

df_raw is empty - nothing scraped yet in Stage 3, so there's nothing to preview here.

Observation — EDA translation preview:

- Both pages translated fluently with natural English output — no garbled text or mid-word cuts
- Glossary actively influencing output: "hospital" and "insurer" both appear, matching GLOSSARY_TERMS canonical terms rather than alternative phrasings
- Sentence-boundary chunking working correctly at the 400-character preview scale
- Translation confirmed working before committing quota to all 144 pages

Note: EDA translation-preview cell — iteration history

1. Fixed: raw [:400] slice replaced with chunk_text() . The original preview truncated Dutch text with a plain character slice before sending to DeepL — same bug class as the one fixed in translate_text(). Confirmed it mattered: worked fine on one sample by luck, then cut "huisartsenzorg" into "huis" + garbage on another. Since this cell exists specifically to eyeball translation quality, a pre-mangled input would make a truncation artifact look identical to a genuine translation issue — defeats the point. Now uses chunk_text() (already tested) to grab the first ~400 characters cleanly at a sentence boundary.(Anique)

2. Fixed: added empty df_raw check. The old version silently did nothing if Stage 3 hadn't run yet — no error, no message, blank output. Now prints a clear explanation, matching the equivalent guard already in Stage 3's EDA cell. (Anique)

3. Confirmed: multi-chunk join tested separately. This preview cell only ever sends a single ~400-character chunk to DeepL, so it can't test the multi-chunk translate-and-rejoin logic. That's confirmed in the verification section below (d) below), which found and fixed a real sentence-cutting issue, then confirmed the fix on the longest real page. (Anique)

STATUS: RESOLVED

d) Verification: does the multi-chunk translate-and-rejoin work?

The EDA preview above only ever translates a single ~400-character chunk — it can't test what happens when a page is long enough to need multiple chunks translated separately and rejoined, which is exactly where a seam problem would show up. These two cells find the longest real page and inspect every chunk boundary directly before the full translation pass runs.

```
In [12]: # -----
# Find the longest real scraped page - worst-case test for multi-chunk translati
# -----
df_raw["dutch_text_length"] = df_raw["original_dutch_text"].str.len()
long_pages = df_raw[df_raw["dutch_text_length"] > 4500].sort_values("dutch_text_
print(f"{len(long_pages)} page(s) over 4500 characters")
long_pages[["source_page_title", "dutch_text_length"]].head()
```

0 page(s) over 4500 characters

```
Out[12]: source_page_title  dutch_text_length
```

```
In [13]: # -----
# Verification: does the multi-chunk translate-and-rejoin actually work?
# -----
# Translates the longest real page (25,675 characters, 6 chunks) chunk-by-chunk
# prints each seam directly, so a bad join would be immediately visible rather t
# buried in the full translated text.
# NOTE: costs ~25,000 DeepL characters on first run - free on reruns if already
# Skip if quota is limited and multi-chunk join is already confirmed working.
# -----
#row = df_raw[df_raw["source_page_title"].str.contains("Zvw-algemeen", na=False)
#dutch_text = row["original_dutch_text"]
#
#chunks = chunk_text(dutch_text)
#print(f"Split into {len(chunks)} chunks:")
#for i, c in enumerate(chunks):
#    print(f"  Chunk {i}: {len(c)} characters")
#
# Translate each chunk separately so the seams can be inspected directly
#translated_chunks = [
#    translator.translate_text(c, source_lang="NL", target_lang="EN-GB", glossar
#    for c in chunks
#]
#
# Print only the boundary windows - end of one chunk + start of the next
#for i in range(len(translated_chunks) - 1):
#    print(f"\n--- Seam between chunk {i} and chunk {i+1} ---")
#    print("...", translated_chunks[i][-150:])
#    print(translated_chunks[i+1][:150], "...")
#
# Confirm every chunk was actually translated, and check each one individually -
# badly-failed chunk can hide inside an otherwise-normal aggregate total, which
# why an aggregate character-count comparison isn't used here (tested: a chunk t
# nearly empty didn't look alarming in the total length, only the per-chunk rati
#print(f"\nDutch chunks: {len(chunks)}, English chunks: {len(translated_chunks)}
#    f"(should always match 1:1 - confirms every chunk was actually translated
#
#print("\nPer-chunk length check (flags any chunk whose translation looks abnorm
#print("which seam-reading alone can miss if the issue sits in the middle of a c
#for i, (dutch_chunk, eng_chunk) in enumerate(zip(chunks, translated_chunks)):
```

```
# ratio = len(eng_chunk) / len(dutch_chunk) if dutch_chunk else 0
# flag = " <-- CHECK THIS ONE" if ratio < 0.5 or ratio > 2.0 else ""
# print(f" Chunk {i}: Dutch {len(dutch_chunk)} -> English {len(eng_chunk)} (
```

Observation — Multi-chunk verification:

- 6 chunks produced from 25,675 characters — all landing just under the 4,500 character limit as expected
- All 5 seams read as clean, connected sentences — no mid-thought cuts or awkward restarts
- Dutch/English chunk counts match 1:1 — confirms no chunk was skipped
- All 6 chunks in a tight 1.04–1.11 ratio band — consistent with normal Dutch→English expansion, nothing flagged as abnormally short or long
- Two independent checks (seam quality + per-chunk ratio) both pass on the same page.

CONFIRMS: Multi-chunk translate-and-rejoin working on worst-case real page.

e) Translation caching (define functions + load cache from database)

```
In [14]: # -----
# Translation caching - body text and page title
# -----
# DeepL's free tier has a limited monthly character quota, and every re-run was
# re-translating every page from scratch - including pages already translated in
# session. This caches by URL and only spends quota on genuinely new or changed
# diffing the freshly-scraped Dutch text against what's already stored in the da

def load_cached_translations(db_path: str) -> dict:
    """Load {url: (dutch_text, english_translation)} from the database. Returns
    on the very first run, when the table doesn't exist yet - that's expected, n
    try:
        conn = sqlite3.connect(db_path)
        cached = pd.read_sql(
            "SELECT original_url, original_dutch_text, english_translation FROM
        )
        conn.close()
        return {
            row["original_url"]: (row["original_dutch_text"], row["english_trans
            for _, row in cached.iterrows()
        }
    except (sqlite3.OperationalError, pd.errors.DatabaseError):
        return {}

CACHED_TRANSLATIONS = load_cached_translations(DB_PATH)
print(f"{len(CACHED_TRANSLATIONS)} previously-translated body translation(s) fou
try:
    conn = sqlite3.connect(DB_PATH)
    cached_title_count = int(conn.execute(
        "SELECT COUNT(*) FROM csn_dataset "
        "WHERE english_page_title IS NOT NULL AND english_page_title != ''
    ).fetchone()[0])
    conn.close()
    print(f"{cached_title_count} previously-translated title(s) found in the dat
```

```

    if cached_title_count != len(CACHED_TRANSLATIONS):
        print(f" [NOTE] {len(CACHED_TRANSLATIONS) - cached_title_count} page(s)
              f"cached body but no cached title.")
except Exception:
    print(f"0 previously-translated title(s) found in the database.")

def translate_with_cache(row) -> str:
    """Translate body text, reusing the cached version if the Dutch source is un
    cached = CACHED_TRANSLATIONS.get(row["original_url"])
    if cached is not None:
        cached_dutch, cached_english = cached
        if cached_dutch == row["original_dutch_text"]:
            return cached_english # unchanged since last run - reuse, don't spe
    return translate_text(row["original_dutch_text"]) # new page, or source con

def translate_title_with_cache(row) -> str:
    """Translate the Dutch page title using the glossary, reusing the cached ver
    body text for this URL is unchanged since the last run."""
    cached = CACHED_TRANSLATIONS.get(row["original_url"])
    if cached is not None:
        cached_dutch, _ = cached
        if cached_dutch == row["original_dutch_text"]:
            try:
                conn = sqlite3.connect(DB_PATH)
                result = conn.execute(
                    "SELECT english_page_title FROM csn_dataset WHERE original_u
                    (row["original_url"],)
                ).fetchone()
                conn.close()
                if result and result[0]:
                    return result[0]
            except Exception:
                pass
    if translator is None or not row["source_page_title"]:
        return ""
    try:
        return translator.translate_text(
            str(row["source_page_title"]), source_lang="NL", target_lang="EN-GB"
        ).text
    except Exception as e:
        print(f" [title translation failed] {e}")
        return ""

```

144 previously-translated body translation(s) found in the database.

144 previously-translated title(s) found in the database.

Observation — Translation caching:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

0 previously-translated bodies, 0 previously-translated titles — expected on a first run, since `csn_dataset.db` starts empty. Upon Run 2, this number should rise, as unchanged pages get reused from cache instead of re-sent to DeepL.

Note: Translation caching — iteration history

Problem with initial code: Every run was re-translating all pages from scratch, including ones already translated in a previous session. On DeepL's free tier with a limited monthly quota, this burned characters unnecessarily on every test and debug run.

Solution: `load_cached_translations()` reads existing translations from `csn_dataset.db`. Two caching functions are defined here:

- `translate_with_cache()` — reuses cached body text if the Dutch source is unchanged
- `translate_title_with_cache()` — reuses cached page title if the body is confirmed unchanged

Only new pages or pages whose content has changed since the last run actually hit DeepL. Both functions are applied in the f) full translation pass.

One dependency: the cache is only populated after Stage 5's `upsert_to_db()` has run at least once. On a first-ever run it correctly finds nothing cached and translates normally — no error, just no benefit yet.

Testing history:

1. *Mock test (design stage):* verified caching logic against a real SQLite database — confirmed fresh run translates and stores correctly, second run with unchanged content reuses with zero new DeepL calls.
2. *Test first run (07/08/26):* 0 previously-translated page(s) found (expected — database not yet populated by Stage 5). All pages hit DeepL, translated fluently, glossary terms consistent.
3. *Test second run (13/08/26):* 107 reused from cache, 21 sent to DeepL — caching working correctly on a test 2nd run. 128 / 128 body translations and page titles both populated.
4. *Notebook 2:* Will demonstrate the cache working on the final source set (144 pages)

Status: RESOLVED --> Caching confirmed working for both body text and page titles.**

f) Full translation pass (body and title)

```
In [15]: # -----
# Full translation pass - body text AND page title, both cached
# -----

if df_raw.empty:
    df_raw["english_translation"] = pd.Series(dtype="object")
    df_raw["english_page_title"] = pd.Series(dtype="object")
    print("df_raw is empty - nothing to translate this run.")
else:
    df_raw["english_translation"] = df_raw.apply(translate_with_cache, axis=1)
    df_raw["english_page_title"] = df_raw.apply(translate_title_with_cache, axis=1)

    reused = sum(1 for _, row in df_raw.iterrows())
```



```

        if CACHED_TRANSLATIONS.get(row["original_url"], (None, None))[0]
sent = len(df_raw) - reused
print(f"Translation pass complete: {sent} page(s) sent to DeepL, {reused} re
print(f"Body translations: {df_raw['english_translation'].notna().sum()} / {
print(f"Page titles: {df_raw['english_page_title'].notna().sum()} / {len(df_

```

df_raw is empty - nothing to translate this run.

Observation — Full translation pass:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

- 0 pages reused from cache, 144 sent to DeepL — expected on a first run, since `csn_dataset.db` starts empty; matches the 0/0 result already confirmed in the caching Observation above
- 144 / 144 body translations populated
- 144 / 144 page titles populated

Note: empty-df_raw guard — iteration history

What was wrong: `df_raw.apply(..., axis=1)` on a 0-row DataFrame doesn't error, but pandas has no data to infer a dtype from, so the new column comes back as `float64` instead of `object` /string. That's fine until the completeness check below (4g) calls `.str.strip()` on it, which raises "Can only use `.str` accessor with string values, not floating".

Fix: explicitly create both translation columns as empty object-dtype Series when `df_raw` is empty, rather than letting `.apply()` infer the dtype. This also skips calling the DeepL API entirely when there's genuinely nothing to translate this run.

Same root cause as the empty-DataFrame guard in Stage 3d — an empty DataFrame silently losing its expected columns/dtypes rather than staying well-formed with zero rows. Worth knowing this turned up twice in different stages, since it's the kind of bug that could recur anywhere else `.apply()` or `pd.DataFrame()` runs on a result that might legitimately be empty.

STATUS: RESOLVED

g) Completeness check

```

In [16]: # -----
# Immediate completeness check: did every page get both translation and title?
# -----
# Catches empty fields right after translating, before moving to Stage 5.
# --- Trevor (21/08) note: safe on 0 rows now the columns above are always object
if df_raw.empty:
    print("df_raw is empty - nothing was translated this run, so there's nothing
else:
    empty_translation_rows = df_raw[df_raw["english_translation"].str.strip() ==
    print(f"{len(empty_translation_rows)} / {len(df_raw)} page(s) have EMPTY eng
    for _, row in empty_translation_rows.iterrows():
        print(f" - {row['source_organisation']}: {row['english_page_title']} ({

```



```
empty_title_rows = df_raw[df_raw["english_page_title"].str.strip() == ""]
print(f"{len(empty_title_rows)} / {len(df_raw)} page(s) have EMPTY english_p
for _, row in empty_title_rows.iterrows():
    print(f" - {row['source_organisation']}: {row['source_page_title']} ({r
```

df_raw is empty - nothing was translated this run, so there's nothing to check here.

Observation — Completeness check:

- English translation and page titles are populated successfully

h) Terminology consistency scan (diagnostic, not a fix)

The glossary prevents inconsistency at translate-time by locking Dutch terms to one fixed English equivalent. But it can only catch terms explicitly listed in `GLOSSARY_TERMS` — if a Dutch source page uses a synonym or compound form not in the list, DeepL's context-based translation can still produce different English phrasings across pages. This scan detects whatever slipped through.

Not every flagged variant is a genuine problem. `suggest_stakeholder()` in Stage 6 groups synonyms like "GP" / "family doctor" / "general practitioner" into the same stakeholder tag, so some raw wording variation doesn't affect what gets counted and charted in Stage 7. But not all flagged terms are synonyms — some (e.g. GP vs. doctor, "Samen Beslissen" as a named campaign vs. generic "deciding together") are genuinely different Dutch concepts correctly translated differently, and shouldn't be treated as inconsistency at all. See the terminology scan Observation below for which flagged terms fall into which category.

```
In [17]: # -----
# Terminology consistency scan (diagnostic)
# -----
# Known concepts to watch, grouped by Likely English phrasings a translation mig
# NOTE: add a concept here if you notice a new inconsistency while reading the d

CONCEPT_VARIANTS_TO_WATCH = {
    # NOTE: "doctor" is a noisier signal than the others - it can correctly refe
    # specialist, not just a GP, so a flag here is a prompt to go check, not pro
    "GP": ["general practitioner", "family doctor", "family physician", "\\bgp\\",
    "hospital": ["hospital", "medical centre", "medical center"],
    "insurer": ["insurer", "insurance company", "health insurer"],
    "deductible": ["deductible", "own risk", "excess"],
    "referral letter": ["referral letter", "referral note", "letter of referral"],
    "co-payment": ["co-payment", "personal contribution", "own contribution"],
    "emergency department": ["emergency department", "a&e", "\\ber\\b", "emergen
    "physiotherapy": ["physiotherapy", "physical therapy"],
    "dietitian": ["dietitian", "dietician", "nutritionist"],
    "shared decision-making": ["shared decision-making", "joint decision-making"],
    "waiting time": ["waiting time", "wait time", "waiting period"],
    "basic insurance package": ["basic insurance package", "basic package", "sta
    "supplementary insurance": ["supplementary insurance", "additional insurance"],
    "practice nurse": ["practice nurse", "practice support worker", "gp assistan
    "specialist care": ["specialist care", "medical-specialised care", "medical
    "healthcare invoice": ["healthcare invoice", "care note", "care bill"],
    "mental health care": ["mental health care", "\\bggz\\b", "psychiatric care"]
```

```

}

def scan_terminology_variants(df: pd.DataFrame, concepts: dict, column: str = "e
    """Checks a text column for concepts that show up as more than one phrasing.
    once here on raw DeepL output, and again in Stage 6 on the combined translat
    summary/action text, since manually-typed content never passes through the g
    for concept, variants in concepts.items():
        counts = {v: df[column].str.contains(v, case=False, na=False, regex=True)
        found = {k: v for k, v in counts.items() if v > 0}
        if len(found) > 1:
            print(f"'{concept}' appears as multiple phrasings across your data:
            print(f"    -> check GLOSSARY_TERMS covers this Dutch term and consid

scan_terminology_variants(df_raw, CONCEPT_VARIANTS_TO_WATCH)

```

Observation — Terminology consistency scan:

11 concepts flagged as inconsistent, 6 stayed clean (hospital, referral letter, physiotherapy, dietitian, supplementary insurance, specialist care).

- GP: {'general practitioner': 6, '\bgp\b': 81, '\bdoctor\b': 85}
- insurer: {'insurer': 75, 'health insurer': 5}
- deductible: {'deductible': 33, 'excess': 3}
- co-payment: {'co-payment': 28, 'personal contribution': 1}
- emergency department: {'emergency department': 4, 'a&e': 1}
- shared decision-making: {'shared decision-making': 2, 'deciding together': 6}
- waiting time: {'waiting time': 11, 'waiting period': 4}
- basic insurance package: {'basic insurance package': 46, 'basic package': 21}
- practice nurse: {'practice nurse': 16, 'gp assistant': 5}
- healthcare invoice: {'healthcare invoice': 14, 'care bill': 2}
- mental health care: {'mental health care': 13, '\bggz\b': 7, 'psychiatric care': 1}

Not all 11 are equally worth acting on:

- **Genuinely different concepts, not inconsistency** — GP/doctor (an 82/85 near-even split points to two distinct real-world roles, not one term drifting), shared decision-making ("Samen Beslissen" is a named campaign, "deciding together" is generic phrasing for the same idea elsewhere).
- **Small enough to not be worth chasing** — insurer, deductible, emergency department, healthcare invoice: a dominant form with only 1-5 stray instances.
- **Worth a closer look if there's time, low priority otherwise** — basic insurance package, waiting time, co-payment, practice nurse, mental health care: larger splits that may reflect genuine second-reference shortening in the Dutch source rather than a fixable glossary gap.

IMPORTANT NOTE: This is a surface-level translation-consistency note, not a data-quality issue — it doesn't affect topic/question classification or dataset completeness.

Note: terminology inconsistency — iteration history

Confirmed across two separate pre-fix runs (first: 128 fresh, 0 cached; second: 106 cached, 22 fresh) — same 7 concepts flagged both times at very similar magnitudes, ruling out stale cache as the explanation.

A real bug found and fixed (14/08/26): the glossary was created with `target_lang="EN"` while translations used `target_lang="EN-GB"` — DeepL silently ignored the glossary as a result. Fixed in the glossary setup cell — all three glossary creation paths now use `EN-GB`.

But this wasn't the main explanation for what the scan flags.

- After the fix, the flagged count actually grew slightly (7 → 11 concepts) rather than shrinking.
- On inspection, most of what this scan catches isn't translation drift at all — it's genuinely distinct Dutch source terms being (correctly) translated differently, e.g. "huisarts" (GP) vs. "arts"/"dokter" (doctor), or "Samen Beslissen" as a named patient-choice campaign vs. generic "gedeelde besluitvorming."
- The `target_lang` fix above was still the right call — it closes a real gap for terms that genuinely are the same concept and were being mistranslated because of it — it just doesn't account for most of what's flagged in the list above. See the Observation above for the breakdown of which flagged terms are worth a closer look and which aren't.

STATUS: ACCEPTED

Stage 5 - Data Structuring & Cleaning

► **What this stage is for** (click to expand/collapse)

```
In [18]: # -----
# Stage 5 - Data Structuring & Cleaning
# -----
# Turns the translated raw data (df_raw) into the structured, cleaned dataset
# (df_structured) that everything downstream depends on. Three steps in order:
# 1. init_dataset() - builds the full 14-column REQUIRED_FIELDS schema
# 2. clean_dataset() - deduplicates, normalises, validates, and flags issues
# 3. upsert_to_db() - persists to cs_n_dataset.db before tagging begins
#
# All three functions are defined here before any are called - keeps the cell
# self-contained so a fresh kernel restart can't cause a NameError.
# REQUIRED_FIELDS is defined in Setup and drives all three functions.
# -----

# --- 1. Initialise df_structured from df_raw ---
# Creates the full 14-column schema from df_raw, adding empty placeholder column
# anything Stage 4 didn't populate yet (classification, patient-facing output).
# english_page_title is already on df_raw from Stage 4's title translation pass,
# it carries through automatically via the 'if col not in df.columns' guard.
def init_dataset(df: pd.DataFrame) -> pd.DataFrame:
```

```

"""Build df_structured from df_raw by adding empty placeholder columns for a
REQUIRED_FIELDS not yet populated. Slices down to exactly REQUIRED_FIELDS, s
nothing extra from df_raw leaks into the structured dataset."""
df = df.copy()
for col in ["patient_journey_stage", "topic", "community_question", "stakeho
"action_next_step", "summary_answer_text", "priority_flag", "eng
    if col not in df.columns:
        df[col] = ""
return df[REQUIRED_FIELDS]

# --- 2. Clean df_structured ---
# Four cleaning steps matching the team's own Data Cleaning Steps checklist.
# Note: terminology consistency is handled at Stage 4 (glossary) and Stage 6
# (stakeholder/topic tagging groups synonyms) - not here. Free-text variation in
# english_translation is fine; what matters is that the tagged/counted fields st
def clean_dataset(df: pd.DataFrame) -> pd.DataFrame:
    """Four cleaning steps: dedup by URL, whitespace normalisation, URL validity
    and a missing-critical-field flag. Returns the cleaned dataframe."""
    df = df.copy()

    # 1. Remove duplicate URLs (same page scraped twice)
    before = len(df)
    df = df.drop_duplicates(subset="original_url", keep="last")
    print(f"Removed {before - len(df)} duplicate URL(s).")

    # 2. Fix stray whitespace/line breaks in free-text fields
    for col in ["original_dutch_text", "english_translation"]:
        df[col] = df[col].str.strip().str.replace(r"\s+\n", "\n", regex=True)

    # 3. Validate URLs are complete (start with http)
    df["url_valid"] = df["original_url"].str.startswith("http")
    invalid = (~df["url_valid"]).sum()
    if invalid:
        print(f"WARNING: {invalid} row(s) have an invalid-looking URL - check be

    # 4. Flag any row missing a critical field - catches failures from Stage 3/4
    # slipped past the earlier completeness checks
    critical_fields = ["source_organisation", "original_url", "original_dutch_te
    df["missing_critical_field"] = df[critical_fields].isna().any(axis=1) | (df[

    return df

# --- 3. Persist to the database ---
# Writes df_structured to cs_n_dataset.db, keyed on original_url so re-running ne
# creates duplicates - existing rows are updated in place, new rows are inserted
# Called twice in the pipeline: once here (protects scrape/translate work before
# tagging) and once at the end of Stage 6 (persists the fully-tagged dataset).
def upsert_to_db(df: pd.DataFrame, db_path: str = DB_PATH):
    """Insert new rows / update existing ones, matched by original_url - never d
    Schema is defined entirely by REQUIRED_FIELDS in Setup - if that list is cor
    the table is always created with the right columns and no migration is ever
    conn = sqlite3.connect(db_path)
    cur = conn.cursor()
    cur.execute(f"""
        CREATE TABLE IF NOT EXISTS cs_n_dataset (
            {", ".join(f"{col} TEXT" for col in REQUIRED_FIELDS)},
            PRIMARY KEY (original_url)
        )
    """

```

```

        """
    placeholders = ", ".join("?" * len(REQUIRED_FIELDS))
    update_clause = ", ".join(f"{col}=excluded.{col}" for col in REQUIRED_FIELDS)
    cur.executemany(
        f"""INSERT INTO csn_dataset ({", ".join(REQUIRED_FIELDS)})
        VALUES ({placeholders})
        ON CONFLICT(original_url) DO UPDATE SET {update_clause}""",
        df[REQUIRED_FIELDS].values.tolist(),
    )
    conn.commit()
    conn.close()
    print(f"Upserted {len(df)} rows into {db_path} (existing URLs updated, not d

# -----
# Run Stage 5
# -----
# NOTE (THE LOOP): df_raw / df_structured here only ever contain the SCRAPE-flag
# subset from the Loop Gate in Stage 3a - df_untouched sits outside this entirel
df_structured = init_dataset(df_raw)
df_structured = clean_dataset(df_structured)
print(f"\nRows with a missing critical field: {df_structured['missing_critical_f

# Partial/protective checkpoint only - covers just the freshly-scraped subset, s
# work already done this run survives a crash before tagging finishes. This is N
# final sync: the full dataset (this subset + df_untouched) gets upserted for re
# the end of Stage 6, step k) - see THE LOOP GATE, part 2.
upsert_to_db(df_structured)

df_structured.head()

```

Removed 0 duplicate URL(s).

Rows with a missing critical field: 0

Upserted 0 rows into csn_dataset.db (existing URLs updated, not duplicated).

Out[18]: **source_organisation original_url source_page_title english_page_title scrape_date or**



Observation — Data Structuring & Cleaning:

- 0 duplicate URLs removed — all 144 rows are unique by `original_url`
- 0 rows with a missing critical field — `source_organisation`, `original_url`, `original_dutch_text` and `english_translation` all populated on every row
- 144 rows upserted to `csn_dataset.db` — scrape and translation work is now persisted to disk before tagging begins in Stage 6
- `df_structured` created with all 14 `REQUIRED_FIELDS` — classification and patient-facing columns initialised as empty placeholders, ready for Stage 6.

Stage 6 - Tag Your Content (Classification System)

► **What this stage is for** (click to expand/collapse)

a) Load embedding model + define the CSN taxonomy

Satisfies the brief's "*Categorisation or tagging logic*" (notebook requirement) and "*Mapping patient questions to trusted answers and next steps*" (work-should-focus-on).
Uses semantic similarity (sentence embeddings), not keyword matching, to classify each page against the CSN topic/question hierarchy.

```
In [19]: # =====
# Load the sentence embedding model
# =====
# all-MiniLM-L6-v2: lightweight, fast, and well-suited for semantic similarity t
# Converts both page text and topic descriptions into vectors that represent mea
# so pages can be matched to topics even when they use different vocabulary.
model = SentenceTransformer("all-MiniLM-L6-v2")
```

Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable higher rate limits and faster downloads.
Loading weights: 0%| | 0/103 [00:00<?, ?it/s]

```
In [20]: # =====
# CSN Topic taxonomy - grouped by patient journey stage
# =====
# The embedding model uses these DESCRIPTIONS (not the key names) to classify ea
# TOPIC_TO_STAGE then derives patient_journey_stage from whichever topic wins.

TOPIC_DESCRIPTIONS = {

    # --- Orientation (required core scope) ---
    "Healthcare system structure":
        "How the Dutch healthcare system is organised, including GPs, hospitals,
        "and how patients navigate care.",
    "The role of the GP, specialist, hospital, and insurer":
        "The responsibilities of GPs, specialists, hospitals and health insurers
        "care and treatment coordination.",
    "Referrals and access to care":
        "How patients obtain referrals, access hospitals and specialists, arrang
        "and enter cancer care pathways.",
    "Health insurance and what may be covered":
        "Health insurance, reimbursement, deductibles, premiums, coverage of can
        "and healthcare costs in the Netherlands.",
    "Patient rights and second opinions":
        "Patient rights, informed consent, access to records, choosing providers
        "choices and second opinions.",
    "Where patients can find trusted information and support":
        "Trusted organisations, patient support services, charities, helplines,
        "support and practical guidance for people affected by cancer.",

    # --- First 48 Hours (optional extension) ---
    "Immediate next steps and support":
        "Immediate actions, decisions and emotional support needed in the first
        "after a cancer diagnosis.",

    # --- Active Navigation (optional extension) ---
    "Managing care, decisions, and access to services":
        "Ongoing navigation of treatment decisions, care coordination and access
        "throughout active cancer care.",

}
```



```

In [21]: # =====
# Patient journey stage implied by topic group
# =====
# Stage now falls out of topic automatically - no hardcoding needed.
# Identify the topic, the stage is implied by which of the three groups it belong
TOPIC_TO_STAGE = {
    "Healthcare system structure": "Orientation",
    "The role of the GP, specialist, hospital, and insurer": "Orientation",
    "Referrals and access to care": "Orientation",
    "Health insurance and what may be covered": "Orientation",
    "Patient rights and second opinions": "Orientation",
    "Where patients can find trusted information and support": "Orientation",
    "Immediate next steps and support": "First 48 Hours",
    "Managing care, decisions, and access to services": "Active Navigation",
}

# =====
# Priority implied by patient journey stage – 21/08 update
# =====
# 1 = Orientation (highest priority - required core scope, every patient needs to)
# 2 = First 48 Hours (medium priority - immediate post-diagnosis next steps)
# 3 = Active Navigation (lower priority - ongoing support further into treatment)
# Kept as its own dict (not folded into TOPIC_TO_STAGE) so each stage's priority
# set once, in one place, rather than duplicated across every topic mapped to it

STAGE_TO_PRIORITY = {
    "Orientation": 1,
    "First 48 Hours": 2,
    "Active Navigation": 3,
}
UNCLASSIFIED_PRIORITY_DEFAULT = 3 # still-Unclassified rows sort as lowest priority

```

Note: priority_flag — iteration history

Before (21/08): `priority_flag` was derived from classification *confidence* (`topic_similarity` / `question_similarity` via `review_status`) — a high-confidence classification got a higher priority, regardless of what the content was actually about.

After (21/08): team decision replaced this with a lookup from `patient_journey_stage` instead, so priority reflects how important content is to a patient at their current stage, not how sure the embedding model happened to be about the classification. Also switched from a High/Medium/Low system to a numeric 1/2/3 system, so additional priority levels can be added in future — e.g. if CSN wishes to expand the scope/stages covered by the Navigator.

STATUS: RESOLVED

```

In [22]: COMMUNITY_QUESTIONS = {
    "Referrals and process": [
        "How do referrals work in the Netherlands?",
        "Can I choose my hospital or specialist?",
        "Can I change doctors?",
        "What should I expect when I visit my GP, and how does registering work?",
        "What does my health insurance actually cover, and what will it cost me?"
    ]
}

```

```

    ],

    "Waiting times": [
        "How long should I expect to wait for appointments or treatment?",
        "What can I do if I feel the process is too slow?",
        "Is it normal to wait between steps?"
    ],

    "Treatment decisions": [
        "How are treatment decisions made?",
        "Can I ask for alternative options?",
        "Do I have time to decide?"
    ],

    # 22/08 update: renamed from "Second opinions" - broadened to include general
    # rights and complaints, which kept getting force-fitted into the wrong ques
    # manual correction.
    "Patient rights and second opinions": [
        "Am I allowed to get a second opinion, including in another country?",
        "How do I request a second opinion?",
        "Will a second opinion be covered by insurance?",
        "How do I make a complaint about my care or a healthcare provider?",
        "What are my general rights and responsibilities as a patient?"
    ],

    "Additional care and support": [
        "What additional services can I access?",
        "How do I get referred to psychological care, physiotherapy, nutrition a
    ],

    "Communication and understanding": [
        "What if I don't understand medical terminology?",
        "Can I request explanations in simpler language?",
        "Are there English-speaking healthcare providers?",
        "How do I find reliable health information or a trustworthy provider?"
    ],
}

# =====
# Topic -> allowed question headers
# =====
# Constrains classify_question_header() so a page can only be matched to questio
# headers genuinely relevant for its topic.

TOPIC_TO_HEADERS = {

    # --- Orientation ---
    "Healthcare system structure": [
        "Referrals and process",
        "Waiting times",
        "Communication and understanding"
    ],

    "The role of the GP, specialist, hospital, and insurer": [
        "Referrals and process",
        "Treatment decisions",
        "Additional care and support"
    ],

    "Referrals and access to care": [
        "Referrals and process",

```

```

    "Waiting times"
  ],
  "Health insurance and what may be covered": [
    "Referrals and process",
    "Additional care and support"
  ],
  "Patient rights and second opinions": [
    "Patient rights and second opinions",
    "Treatment decisions",
    "Communication and understanding"
  ],
  "Where patients can find trusted information and support": [
    "Communication and understanding",
    "Additional care and support"
  ],
  # --- First 48 Hours ---
  # Immediate post-diagnosis: patients need to know next steps, what support
  # exists, who to contact, and how to navigate first care conversations.
  "Immediate next steps and support": [
    "Referrals and process",
    "Additional care and support",
    "Communication and understanding",
    "Treatment decisions"
  ],
  # --- Active Navigation ---
  # Ongoing treatment: making decisions, managing waiting times, seeking second
  # opinions, and accessing services throughout the care journey.
  "Managing care, decisions, and access to services": [
    "Treatment decisions",
    "Patient rights and second opinions",
    "Waiting times",
    "Additional care and support"
  ],
}

```

Note: Community question taxonomy expanded + Header name edit — iteration history

What was added (22/08): 5 new questions added to CSN's original 17-question taxonomy, taking it to 22 total — marked inline above with `# 22/08 addition`:

- "What should I expect when I visit my GP, and how does registering work?"
- "What does my health insurance actually cover, and what will it cost me?"
- "How do I make a complaint about my care or a healthcare provider?"
- "What are my general rights and responsibilities as a patient?"
- "How do I find reliable health information or a trustworthy provider?"

Why: during manual correction, several real pages didn't cleanly match any of the original 17 questions — a GP-registration page had nowhere to go under "Referrals and process," an insurance-cost page kept getting force-fitted into "Additional services," and general rights/complaints content had no home until "Second opinions" was broadened and renamed to "Patient rights and second opinions" (see the header rename comment above).

Confirmed at scale: the insurance-cost addition specifically went on to surface a separate classification bug (`classify_question_header()` scoring by bare label, not question content — see Stage 6b) — worth knowing this addition is what exposed that bug, not that the addition itself was the problem.

STATUS: RESOLVED

```
In [23]: # =====
# Pre-compute topic embeddings once upfront - reused for every page classificati
# =====

topic_names = list(TOPIC_DESCRIPTIONS.keys())
topic_embeddings = model.encode(
    list(TOPIC_DESCRIPTIONS.values()),
    normalize_embeddings=True
)
```

b) Classification functions

Three embedding-based classifiers (topic -> question header -> specific question, each constrained by the level above it) plus the stakeholder tagger - Will's word-boundary regex fix, confirmed against a real bug (`"clinic"` matching inside `"clinical"`).

```
In [24]: # =====
# Classification functions - Aisling's embedding approach
# =====
# Three embedding-based classifiers, each constrained by the level above it:
# 1. classify_topic() - page vs 8 topic descriptions
# 2. classify_question_header() - page vs valid headers for that topic
# 3. classify_community_question() - page vs specific questions for that header
# =====

# --- 1. Topic classification ---
def classify_topic(text: str, threshold: float = 0.40):
    """Semantic similarity, not keyword matching - compares the page's meaning a
    topic's reference description. Falls to Unclassified below the confidence th
    if not text or pd.isna(text):
        return ("Unclassified - needs manual tagging", 0.0)
    text_embedding = model.encode(str(text), normalize_embeddings=True)
    similarities = cosine_similarity([text_embedding], topic_embeddings)[0]
    best_index = int(np.argmax(similarities))
    best_score = float(similarities[best_index])
    if best_score < threshold:
        return ("Unclassified - needs manual tagging", best_score)
    return (topic_names[best_index], best_score)

# --- NEW: 2. Question header classification ---
def classify_question_header(text: str, topic: str):
    """Only compares against the question headers valid for the already-assigned
    a Referrals page can't get matched to a Patient Rights header, for example.

    Fix (28/08): scores each candidate header by its best-matching COMMUNITY_QUE
    entry, not by embedding the bare header label. The label alone (e.g. "Referr
    process") carries far less semantic signal than the actual questions filed u
    which was causing mismatches like Health Insurance pages landing on "Additio
    and support" instead of the header that actually holds the insurance/cost qu
```

```

"""
if topic not in TOPIC_TO_HEADERS:
    return ("Unclassified", 0.0)
candidate_headers = TOPIC_TO_HEADERS[topic]
text_embedding = model.encode(str(text), normalize_embeddings=True)

best_header, best_score = "Unclassified", -1.0
for header in candidate_headers:
    question_embeddings = model.encode(COMMUNITY_QUESTIONS[header], normalize_embeddings=True)
    header_score = float(cosine_similarity([text_embedding], question_embeddings)[0][0])
    if header_score > best_score:
        best_header, best_score = header, header_score
return (best_header, best_score)

# --- OLD: 2. Question header classification ---
def classify_question_header(text: str, topic: str):
    """Only compares against the question headers valid for the already-assigned
    a Referrals page can't get matched to a Second Opinions header, for example
    if topic not in TOPIC_TO_HEADERS:
        return ("Unclassified", 0.0)
    candidate_headers = TOPIC_TO_HEADERS[topic]
    text_embedding = model.encode(str(text), normalize_embeddings=True)
    header_embeddings = model.encode(candidate_headers, normalize_embeddings=True)
    similarities = cosine_similarity([text_embedding], header_embeddings)[0]
    best_index = int(np.argmax(similarities))
    return (candidate_headers[best_index], float(similarities[best_index]))

# --- 3. Specific community question classification ---
def classify_community_question(text: str, header: str):
    """Selects the closest specific patient-facing CSN question within the assigned header
    if header not in COMMUNITY_QUESTIONS:
        return ("Unclassified", 0.0)
    candidate_questions = COMMUNITY_QUESTIONS[header]
    text_embedding = model.encode(str(text), normalize_embeddings=True)
    question_embeddings = model.encode(candidate_questions, normalize_embeddings=True)
    similarities = cosine_similarity([text_embedding], question_embeddings)[0]
    best_index = int(np.argmax(similarities))
    return (candidate_questions[best_index], float(similarities[best_index]))

```

Note: Classification functions — iteration history

Issue: `classify_question_header()` was matching against bare header labels, not header content — relevant questions weren't being matched to any sources, e.g. "What does my health insurance actually cover, and what will it cost me?"

What was wrong: the function picked a header by embedding the short `TOPIC_TO_HEADERS` label itself (e.g. "Referrals and process," "Additional care and support") and comparing that to the page text. Those labels are 2-4 words with no descriptive content, unlike `TOPIC_DESCRIPTIONS` used in `classify_topic()`, which are full sentences — a much weaker signal for the embedding model. That's why a Health Insurance page could land on "Additional care and support" — the label "sounds" more relevant to coverage/support than "Referrals and process" does, even though the actual matching question lives under "Referrals and process" by design.

Fix: score each candidate header by its best-matching question in

`COMMUNITY_QUESTIONS` instead of the bare header name — re-using the same richer content `classify_community_question()` already searches, so header selection and question selection can no longer disagree.

Confirmed at scale (full 144-row dataset): "What does my health insurance actually cover, and what will it cost me?" went from 0 matched pages before the fix to 36 matched pages after — now correctly one of the multi-source questions rather than a gap. Questions with zero sources overall dropped from 10 to 7, none of which relate to this routing issue.

STATUS: RESOLVED

```
In [25]: # =====
# Stakeholder tagging - Will's word-boundary regex fix
# =====
# Kept separate from the embedding chain - confirmed more reliable for named
# entity matching than semantic similarity. Fixes confirmed bug: "clinic" was
# matching inside "clinical" (clinical trial), incorrectly tagging as Hospital.

# Counts every regex match per category across the
# whole text and assigns whichever stakeholder is mentioned most often, so the
# tag reflects who the page is mainly about, not just who's named earliest.
# =====

STAKEHOLDER_KEYWORDS = {
    "Insurer": [r"\binsurer", r"\binsurance compan", r"\bhealth insur"],
    "Government body": [r"\bgovernment", r"\bministry"],
    "Patient organisation": [r"\bpatient organi", r"\badvocacy group"],
    "Specialist": [r"\bspecialist", r"\bconsultant", r"\boncolog"],
    "Hospital": [r"\bhospital", r"\bclinic\b"],
    "GP": [r"\bgp\b", r"\bgeneral practitioner", r"\bfamily doctor"],
}

def stakeholder_mention_counts(text: str) -> dict:
    """Counts every regex match for each stakeholder category across the whole t
    a category with multiple patterns (e.g. Insurer has 3) sums matches across a
    them. Exposed separately from suggest_stakeholder() so counts can be spot-ch
    directly during review, not just the final winning category."""
    lowered = str(text).lower()
    return {
        stakeholder: sum(len(re.findall(pattern, lowered)) for pattern in patter
        for stakeholder, patterns in STAKEHOLDER_KEYWORDS.items())
    }

def suggest_stakeholder(text: str) -> str:
    """Returns whichever stakeholder category is mentioned most often in
    english_translation, not just whichever pattern happens to match first.
    Unclassified if no category matches at all. Ties fall back to
    STAKEHOLDER_KEYWORDS's declaration order, same behaviour as the old
    first-match version."""
    counts = stakeholder_mention_counts(text)
    best_stakeholder, best_count = max(counts.items(), key=lambda kv: kv[1])
    if best_count == 0:
        return "Unclassified - needs manual tagging"
    return best_stakeholder
```


c) Tag topic, patient_journey_stage and stakeholder

These three fields are almost completely automated and final — no human input needed and no dependency on community_question or summary_answer_text. Persisted to the database immediately after this section so these fields are queryable while the manual review pass is still in progress.

```
In [26]: # =====
# Topic classification
# =====
# --> Embedding method scores each page against all 8 topic descriptions
# and picks the highest scoring match above the 0.40 confidence threshold.
# Pages below the threshold are returned as "Unclassified - needs manual tagging"
# topic_similarity is kept as a transparency signal - shows how confident the
# classification was for each row.
#
# --- Trevor (21/08) fix: same "0 rows -> can't infer 2 columns" issue as Stage
# .apply(pd.Series) unpacks each row's (topic, score) tuple into two columns - b
# empty df_structured there are zero tuples to unpack from, so pandas can't tell
# should be 2 columns and raises "Columns must be same length as key" instead of
# giving back 0 rows. Guard it exactly like the empty-df_raw cases upstream: whe
# was freshly processed this run, create both columns directly with the right dt
if df_structured.empty:
    df_structured["topic_category"] = pd.Series(dtype="object")
    df_structured["topic_similarity"] = pd.Series(dtype="float64")
else:
    df_structured[["topic_category", "topic_similarity"]] = (
        df_structured["english_translation"]
        .apply(classify_topic)
        .apply(pd.Series)
    )
    # Round to 2dp for readability - raw cosine similarity value (0-1) kept as a
    # transparency signal but not in REQUIRED_FIELDS
    df_structured["topic_similarity"] = df_structured["topic_similarity"].round(2)

# topic is the REQUIRED_FIELDS column name - taken directly from topic_category
# (topic_category is the internal working name used by classify_topic())
df_structured["topic"] = df_structured["topic_category"]

# =====
# Derive patient_journey_stage from topic
# =====
# --> Stage falls out of topic automatically via TOPIC_TO_STAGE - no hardcoding.
# Unclassified rows have no entry in TOPIC_TO_STAGE so .map() returns NaN -
# fillna() keeps them as "Unclassified - needs manual tagging" so they remain
# visible and identifiable rather than being hidden behind a default value.
# Fix the topic first, then re-run this cell to update patient_journey_stage.

df_structured["patient_journey_stage"] = df_structured["topic"].map(TOPIC_TO_STA

# =====
# Results
# =====
print(f"Topic: {(df_structured['topic'] != 'Unclassified - needs manual tagging')
      f" {(df_structured['topic'] == 'Unclassified - needs manual tagging').sum()")
```

```

        f"(stage also unclassified for these rows)")
print("\nTopic distribution:")
print(df_structured["topic_category"].value_counts().to_string())
print("\nPatient journey stage distribution:")
print(df_structured["patient_journey_stage"].value_counts().to_string())

```

Topic: 0 classified, 0 unclassified (stage also unclassified for these rows)

Topic distribution:

Series([],)

Patient journey stage distribution:

Series([],)

```

In [27]: # =====
# Stakeholder classification
# =====
# Independent of the embedding chain – word-boundary regex against english_trans
# Runs here alongside topic/stage since it's also fully automated and final.

df_structured["stakeholder"] = df_structured["english_translation"].apply(suggest_stakeholder)

print(f"Stakeholder: {(df_structured['stakeholder'] != 'Unclassified - needs manual tagging')} rows processed – topic, patient_journey_stage and stakeholder complete.")
print(f"Stakeholder distribution:")
print(df_structured["stakeholder"].value_counts().to_string())
print(f"\n{len(df_structured)} rows processed – topic, patient_journey_stage and stakeholder complete.")

```

Stakeholder: 0 classified, 0 unclassified

Stakeholder distribution:

Series([],)

0 rows processed – topic, patient_journey_stage and stakeholder complete.

Observation — topic, patient_journey_stage and stakeholder tagging:

Topic — 107 / 144 classified, 37 unclassified (patient journey stage is also unclassified for these same rows, since stage is derived from topic)

Topic	Pages
Health insurance and what may be covered	34
The role of the GP, specialist, hospital, and insurer	16
Managing care, decisions, and access to services	13
Referrals and access to care	11
Healthcare system structure	11
Where patients can find trusted information and support	10
Patient rights and second opinions	7
Immediate next steps and support	5
Unclassified	37

Patient journey stage — Orientation 89 · Active Navigation 13 · First 48 Hours 5 ·
Unclassified 37

Stakeholder — 135 / 144 classified, 9 unclassified

Stakeholder	Pages
Insurer	60
GP	35
Hospital	22
Specialist	14
Government body	2
Patient organisation	2
Unclassified	9

The 37-row Unclassified count is expected on a genuine first run and needs manual correction in Excel before submission (see Stage 6e for the affected-row list).

d) Tag community_question

Runs after topic is confirmed. Constrained by topic — a page tagged as "Health insurance" can only be matched to relevant question headers, not "Communication and understanding" headers, for example. Two levels: header then specific question, each constrained by the level above.

22/08 update: renamed "Second opinions" to "Patient rights and second opinions" (broadened to also cover complaints and general patient rights). Health insurance coverage/cost questions were deliberately kept under "Referrals and process" rather than given their own header, to avoid growing the taxonomy - see COMMUNITY_QUESTIONS and TOPIC_TO_HEADERS above.

```
In [28]: # =====
# Question header - constrained to headers valid for the assigned topic only.
# =====
# --- Trevor (21/08) fix: same empty-df_structured guard as the topic classifica
# above - .apply(pd.Series) can't unpack tuples from zero rows, so skip straight
# empty, correctly-typed columns when nothing was freshly processed this run.
if df_structured.empty:
    df_structured["community_question_header"] = pd.Series(dtype="object")
    df_structured["header_similarity"] = pd.Series(dtype="float64")
    df_structured["community_question"] = pd.Series(dtype="object")
    df_structured["question_similarity"] = pd.Series(dtype="float64")
else:
    df_structured[["community_question_header", "header_similarity"]] = (
        df_structured.apply(
            lambda row: classify_question_header(
                row["english_translation"],
                row["topic_category"]
            ),
            axis=1
        )
    )
```

```

        ).apply(pd.Series)
    )

    # Round to 2dp for readability – raw cosine similarity value (0-1) kept as a
    # transparency signal but not in REQUIRED_FIELDS
    df_structured["header_similarity"] = df_structured["header_similarity"].round(2)

    # =====
    # Specific community question – constrained by header.
    # Picks the closest actual patient question from CSN's community questions d
    # =====
    df_structured[["community_question", "question_similarity"]] = (
        df_structured.apply(
            lambda row: classify_community_question(
                row["english_translation"],
                row["community_question_header"]
            ),
            axis=1
        ).apply(pd.Series)
    )

    # Round to 2dp for readability – raw cosine similarity value (0-1) kept as a
    # transparency signal but not in REQUIRED_FIELDS
    df_structured["question_similarity"] = df_structured["question_similarity"].round(2)

    # =====
    # action_type – categorical version of the action dimension.
    # NOTE: not yet in REQUIRED_FIELDS - Kept as a working column.
    # =====

    # 22/08 update: "Second opinions" -> "Patient rights and second opinions" (renam
    # COMMUNITY_QUESTIONS/TOPIC_TO_HEADERS above)
    ACTION_TYPE_MAP = {
        "Referrals and process": "Complete a process",
        "Waiting times": "Find information",
        "Treatment decisions": "Make a decision",
        "Patient rights and second opinions": "Understand a right or responsibility",
        "Additional care and support": "Access a service",
        "Communication and understanding": "Find information",
    }

    df_structured["action_type"] = (
        df_structured["community_question_header"]
        .map(ACTION_TYPE_MAP)
        .fillna("Find information")
    )

    print("Community question distribution (top 10):")
    print(df_structured["community_question"].value_counts().head(10).to_string())

```

Community question distribution (top 10):
 Series([],)

Observation — community_question tagging:

37 rows Unclassified for community_question — matches the 37 Unclassified topic rows exactly, since question tagging is constrained by whichever topic was assigned.

Top 10 questions by page count (post `classify_question_header()` fix):

Community question	Pages
Unclassified	37
What does my health insurance actually cover, and what will it cost me?	36
What should I expect when I visit my GP, and how does registering work?	17
Can I choose my hospital or specialist?	14
How do I find reliable health information or a trustworthy provider?	9
What are my general rights and responsibilities as a patient?	8
How long should I expect to wait for appointments or treatment?	4
How do I get referred to psychological care, physiotherapy, nutrition advice or fertility support?	4
How are treatment decisions made?	4
Will a second opinion be covered by insurance?	3

The insurance-cost question is now correctly populated at 36 pages — before the header-routing fix it sat at 0. The psych/physio/fertility referral question has also dropped back to a realistic 4 pages, down from an inflated 56 pre-fix, since it's no longer absorbing insurance pages that belonged elsewhere.

e) Rows needing manual correction

21/08 update: this step is now diagnostic only — corrections are made directly in the working Excel file (CSN_DATASHEET.xlsx), not in this notebook. The cell below lists every row where topic, stakeholder, or community_question is still Unclassified, so the team knows exactly which rows to open and fix in Excel before the next run.

Any row without the SCRAPE flag set passes through completely untouched on the next run — every field keeps whatever value it currently holds, so a row corrected this way stays exactly as edited.

Note: correcting `topic` doesn't automatically update `patient_journey_stage` or `priority_flag` — see the "REFERENCE FOR EDITS" tab for the mapping to apply by hand.

```
In [29]: # =====
# Rows needing manual correction – any fields Unclassified
# =====
unclassified_mask = (
    (df_structured["topic"] == "Unclassified - needs manual tagging") |
    (df_structured["stakeholder"] == "Unclassified - needs manual tagging") |
    (df_structured["community_question"] == "Unclassified")
)

unclassified_rows = df_structured[unclassified_mask][
    ["original_url", "english_page_title", "topic", "stakeholder", "community_que"]
]
```

```

if len(unclassified_rows):
    print(f"{len(unclassified_rows)} row(s) needing manual correction:\n")
    for _, row in unclassified_rows.iterrows():
        print(f"    URL:         {row['original_url']}")
        print(f"    Title:        {row['english_page_title']}")
        print(f"    Topic:        {row['topic']}")
        print(f"    Stakeholder: {row['stakeholder']}")
        print(f"    Question:    {row['community_question']}")
        print()
else:
    print("No Unclassified rows – all fields confirmed. Ready to proceed.")

```

No Unclassified rows – all fields confirmed. Ready to proceed.

Observation — rows needing manual correction:

- 45 rows have at least one Unclassified field (topic, stakeholder, or community_question) and need attention directly in `CSN_DATASHEET.xlsx` before the next run.
- This is slightly fewer than the 37 + 9 unclassified topic/stakeholder counts combined, meaning a handful of rows have more than one field Unclassified at once — worth expecting a few rows to need two or three corrections rather than just one.

Once in Excel, each row can be identified by its URL and English title (both shown in the sample list above) or via the Find function in Excel:

- Correct `topic`, `stakeholder`, and `community_question` as needed — and see the note above regarding `patient_journey_stage` / `priority_flag` needing a manual update too when `topic` changes (REFERENCE FOR EDITS tab in Excel workbook also outlines this!)

f) Quick spot-check before writing summary/action text

Check the community_question assigned to each row makes sense — the summary_answer_text written in the next section should answer this specific question.

```

In [30]: # =====
# Quick view of classification results before moving to summary/action text.
# =====
# Check: does the community_question assigned to each row make sense for that pa
# This is the pairing the manual review pass needs to honour – summary_answer_te
# should answer the community_question shown here.
df_structured[
    "source_organisation",
    "english_page_title",
    "topic",
    "patient_journey_stage",
    "stakeholder",
    "community_question",
].head(10)

```


Out[30]: **source_organisation english_page_title topic patient_journey_stage stakeholder cor**

g) summary_answer_text and action_next_step

► **HOW TO FINALISE summary_answer_text AND action_next_step — click to expand**

```
In [31]: # =====
# Draft summary and action functions (Trevor – TextRank upgrade)
# =====

def draft_summary(english_text: str, max_sentences: int = 2) -> str:
    """Extractive draft using TextRank (Trevor) - picks the most representative
    from the translation based on similarity to the rest of the text, not just t
    Still a starting point to edit - always prefixed [DRAFT - REVIEW BEFORE SUBM
    Falls back to first-N-sentences if TextRank can't handle the text (e.g. too
    if not english_text:
        return ""
    text = str(english_text).strip()
    try:
        parser = PlaintextParser.from_string(text, Tokenizer("english"))
        summarizer = TextRankSummarizer()
        sentences = summarizer(parser.document, max_sentences)
        draft = " ".join(str(s) for s in sentences).strip()
    except Exception:
        draft = ""
    if not draft:
        fallback_sentences = re.split(r"(?<=[.!?])\s+", text)
        draft = " ".join(fallback_sentences[:max_sentences]).strip()
    return f"[DRAFT - REVIEW BEFORE SUBMITTING] {draft}" if draft else ""

ACTION_INDICATOR_PHRASES = [
    "you can", "you may", "you have the right to", "contact your", "ask your",
    "request a", "you are entitled to", "you are free to",
]

def draft_action_next_step(english_text: str) -> str:
    """Extractive draft: the first sentence containing an action-indicator phras
    Returns empty string if none found - that row needs writing from scratch."""
    if not english_text:
        return ""
    sentences = re.split(r"(?<=[.!?])\s+", str(english_text).strip())
    for s in sentences:
        if any(p in s.lower() for p in ACTION_INDICATOR_PHRASES):
            return f"[DRAFT - REVIEW BEFORE SUBMITTING] {s.strip()}"
    return ""
```

```
In [32]: # =====
# Draft summary and action text – auto-generated for every row
# =====
# 21/08 update: MANUAL_CONTENT (hand-writing finalised text into a Python dict)
# finalised text is now written directly into summary_answer_text/action_next_st
# CSN_DATASHEET.xlsx instead. Every row reaching this cell is, by definition, fr
```

```

# SCRAPE-flagged (the Loop Gate already filtered to that subset - see Stage 3a),
# row gets a fresh draft here, no exceptions - a row's finalised Excel text is p
# simply by staying OUT of df_structured entirely until it's re-flagged SCRAPE (
# df_untouched at the Loop Gate). No hand-written-URL tracking needed anymore.
# =====

df_structured["summary_answer_text"] = df_structured["english_translation"].apply(d
df_structured["action_next_step"] = df_structured["english_translation"].apply(d

# Flag rows still needing human review (blank or still on a draft) - still neede
# the Final QA checklist (Stage 8) and the red-row highlighting in the Excel exp
df_structured["content_needs_review"] = (
    df_structured["summary_answer_text"].str.startswith("[DRAFT") |
    df_structured["action_next_step"].str.startswith("[DRAFT") |
    (df_structured["summary_answer_text"] == "") |
    (df_structured["action_next_step"] == "")
)

needs_review = df_structured[df_structured["content_needs_review"]][["
    "english_page_title", "community_question", "summary_answer_text", "action_n
]]
print(f"{len(needs_review)} / {len(df_structured)} row(s) need a human to finali
needs_review

```

0 / 0 row(s) need a human to finalise summary/action text in Excel.

Out[32]: **english_page_title community_question summary_answer_text action_next_step**

Observation — summary_answer_text and action_next_step:

144 / 144 rows need a human to finalise summary/action text — every row carries the [DRAFT - REVIEW BEFORE SUBMITTING] prefix, expected on this genuine first run since nothing has been hand-written yet.

15 rows have an empty `action_next_step` — `draft_action_next_step()` found no action-indicator phrase (e.g. "you can," "contact your," "you have the right to") anywhere in the translation for these rows, so there was nothing to extract as a starting draft. These need writing from scratch rather than editing an existing draft.

`content_needs_review` is now set for all 144 rows and will drive the red-row highlighting in the Excel export and the Final QA checklist in Stage 8 — this flag clears automatically once a row's SCRAPE flag is set to "N" and its draft text has been overwritten with finalised content.

Note: draft_summary() — iteration history

draft_summary() upgraded from first-N-sentences to TextRank

Before: the draft summary was just the literal first 1-2 sentences of the translated page — often grabbed an irrelevant opening line (a site disclaimer, a generic intro) rather than the sentence that actually answers the question.

After: uses `sumy`'s TextRank algorithm, which picks the sentences most *representative* of the whole page — the ones most similar in meaning to the rest of the text — rather

than just whichever came first. Nothing invented: every word still comes directly from `english_translation`.

Still a draft: the `[DRAFT - REVIEW BEFORE SUBMITTING]` flag is unchanged — this only changes which sentences get picked for the first-pass draft. (21/08 update: the `MANUAL_CONTENT` workflow referenced in the original version of this note has since been removed entirely — finalisation now happens directly in `CSN_DATASHEET.xlsx`; see Stage 6g instructions above.)

Fallback behaviour preserved: if a page's text is too short for TextRank to work with, it falls back to the original first-N-sentences approach rather than returning nothing.

Confirmed at scale (27/08): now run against the full 144-row dataset across two complete runs. TextRank drafts are consistently picking substantive, on-topic sentences rather than disclaimers/intros — spot-checked and holding up as intended.

STATUS: RESOLVED

►  **Archived: MANUAL_CONTENT staleness bug (21/08, fixed same day — click to expand)**

h) priority_flag

`priority_flag` is derived from `patient_journey_stage`, not classification confidence — a pure lookup, no similarity scores involved. Runs here (after topic/stage are set) rather than waiting on `community_question`, since it has no dependency on question similarity either.

```
In [33]: # =====
# priority_flag - derived from patient_journey_stage
# =====
# --- 21/08 update: review_status and the old confidence-based assign_priority()
# both gone. priority_flag is now a straight lookup from patient_journey_stage v
# STAGE_TO_PRIORITY (defined in Stage 6a alongside TOPIC_TO_STAGE) - 1/2/3, not
# High/Medium/Low text, and not dependent on topic_similarity/question_similarit
# all anymore. Still-Unclassified rows fall back to UNCLASSIFIED_PRIORITY_DEFAULT
# so they sort as Lowest priority rather than ending up blank.
# =====
def assign_priority_from_stage(stage: str) -> int:
    """Looks up priority_flag purely from patient_journey_stage - 1 (Orientation
    2 (First 48 Hours), or 3 (Active Navigation). Falls back to
    UNCLASSIFIED_PRIORITY_DEFAULT for any stage not in STAGE_TO_PRIORITY (i.e. s
    "Unclassified - needs manual tagging")."""
    return STAGE_TO_PRIORITY.get(stage, UNCLASSIFIED_PRIORITY_DEFAULT)

df_structured["priority_flag"] = df_structured["patient_journey_stage"].apply(as

print("Priority flag distribution (1 = Orientation, 2 = First 48 Hours, 3 = Acti
print(df_structured["priority_flag"].value_counts().sort_index().to_string())
```

Priority flag distribution (1 = Orientation, 2 = First 48 Hours, 3 = Active Navigation):

Series([],)

Observation — priority_flag:

Priority flag distribution: 1 = 89, 2 = 5, 3 = 50 — mirrors the `patient_journey_stage` distribution exactly (Orientation 89, First 48 Hours 5, Active Navigation + Unclassified 50 combined), since priority is now a direct lookup from stage, not an independent classification.

Of the 50 priority-3 rows, only 13 are genuinely Active Navigation content — the remaining 37 are rows still carrying an Unclassified stage that default to priority 3. A `priority_flag` of 3 no longer means "low classification confidence" — it means either genuinely Active Navigation content, or a row that needs its `topic` column fixed in Excel (`CSN_DATASHEET.xlsx`).

i) Spot-check — the full picture before final upsert

A view of all `REQUIRED_FIELDS` columns together with the QA signals, so the complete dataset can be assessed in one place before the second upsert (the first happened back in Stage 5, right after scraping/translation, before any classification).

```
In [34]: # =====
# Full spot-check - all REQUIRED_FIELDS plus QA signals in one view.
# =====
# This is the closest view to the final database row the pipeline produces.
# summary_answer_text and action_next_step are shown here alongside the
# classification fields so the pairing (community_question -> summary -> action)
# can be visually validated in one table.
# =====
df_structured[[
    # Source metadata
    "source_organisation", "english_page_title",
    # Classification (REQUIRED_FIELDS)
    "topic", "patient_journey_stage", "stakeholder", "community_question",
    "priority_flag",
    # Patient-facing output (REQUIRED_FIELDS)
    "summary_answer_text", "action_next_step",
    # QA signals (not in REQUIRED_FIELDS but useful for review)
    "content_needs_review"
]].head(10)
```

```
Out[34]: source_organisation english_page_title topic patient_journey_stage stakeholder cor
```



Observation — Full spot-check (all `REQUIRED_FIELDS` in one view):

What's genuinely worth checking here:

- **topic → patient_journey_stage pairing looks correct** — e.g. row 0 (choosing a hospital) is Referrals/Orientation, rows 1-2 (waiting times, treatment decisions) are

Managing care/Active Navigation, matching `TOPIC_TO_STAGE` as expected.

- **community_question reads as a sensible match to the page title** — e.g. "Second opinion on cancer" → "How do I request a second opinion?", "Complaints about cancer care" → "How do I make a complaint about my care or a healthcare provider?" — confirms the `classify_question_header()` fix is routing correctly at a glance, not just in aggregate.
- **priority_flag matches the stage** shown in the same row (1 = Orientation, 3 = Active Navigation) — no mismatches visible in this sample.
- **summary_answer_text and action_next_step both carry the [DRAFT] prefix**, confirming none of this sample has been manually finalised yet — expected on this run (run1).

This is a visual sanity check, not a statistical one — the real aggregate numbers for topic, stage, and question distribution are in the Observation cells at Stage 6c/6d above.

j) Advice-tone check

Both checks run on what the team **wrote** (summary_answer_text, action_next_step), not on the raw translation. Must run after the draft (or Excel-finalised) summary/action text is in place so the checks reflect the actual text that will go into the final dataset.

```
In [35]: # =====
# Advice-tone check
# =====
# The brief is explicit: no clinical advice. A source page can be fully in scope
# still get summarised into advice-toned language by accident – this catches that
# we accept the Excel datasheet as final.

ADVICE_TONE_PHRASES = [
    "take this medication", "the correct dose", "the recommended dose",
    "the best treatment", "the recommended treatment", "this treatment is right",
    "you should take", "you must take", "you should undergo", "you must undergo",
    "increase your dose", "decrease your dose", "reduce your dose",
    "stop taking", "start taking", "it is advised that you",
]

def flags_as_advice(text: str) -> bool:
    """Returns True if the text contains any directive phrasing that could read
    clinical advice - a signal for a human to reword before submission."""
    lowered = str(text).lower()
    return any(phrase in lowered for phrase in ADVICE_TONE_PHRASES)

df_structured["advice_tone_flag"] = (
    df_structured["summary_answer_text"].apply(flags_as_advice) |
    df_structured["action_next_step"].apply(flags_as_advice)
)

advice_flagged = df_structured[df_structured["advice_tone_flag"]][[
    "english_page_title", "summary_answer_text", "action_next_step"
]]
print(f"{len(advice_flagged)} row(s) flagged for possible advice-toned language")
advice_flagged
```

0 row(s) flagged for possible advice-toned language - reword before submitting.

Out[35]: **english_page_title summary_answer_text action_next_step**

Observation — advice-tone check:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

10 rows flagged for possible advice-toned language:

- expensive-drug entitlement, switching insurer, second opinion, GP practice contacts, emergencies, GP visit expectations, pregnancy with cancer, in-the-event-of-an-emergency, asking the GP questions, and mental health care (Zvw).
- Tells us to be weary when writing the summary and action texts for these specific pages.

k) The Loop Gate, part 2 — recombine, reset the flag, then upsert + export

`df_structured` at this point holds ONLY the freshly-processed (`SCRAPE == "SCRAPE"`) rows - everything since the Loop Gate in Stage 3a has only ever touched this subset. `df_untouched` (set aside in that same cell, before any processing) holds every other row exactly as it was read from Excel. This step stitches the two back together, resets the flag on the rows that were just processed, and writes the single combined result to both SQL and Excel - the only export CSN needs, and the input to next run.

```
In [36]: # -----
# THE LOOP GATE, part 2 – recombine, reset the flag, then upsert the FULL set
# -----
# A freshly-processed row is "done" for this run - reset its flag to N so it isn
# re-processed again next time unless a volunteer deliberately re-flags it.
# --- Trevor (21/08) fix: don't gate this on len(df_structured) ---
# The old `if len(df_structured):` guard meant that on a run where nothing was f
# processed, the SCRAPE column never got added to df_structured at all - then th
# df_structured[DATASHEET_COLUMNS] selection a few lines below raised
# KeyError: "'SCRAPE'" not in index", because that column genuinely didn't exis
# Assigning a scalar to a column on an empty DataFrame is safe (it just creates
# empty column with the right name) - so there's nothing to guard against here.
df_structured[SCRAPE_FLAG_COLUMN] = "N"

# By this point df_structured already carries every DATASHEET_COLUMNS entry -
# REQUIRED_FIELDS from Stage 5, QA_COLUMNS added throughout Stage 6, SCRAPE just
# df_untouched was already aligned to the same DATASHEET_COLUMNS set back at the
# Gate (Stage 3a) - so both sides match column-for-column, nothing to patch here
# df_failed rows are NOT reset to "N" - unlike df_structured, they didn't
# actually get reprocessed this run, so they shouldn't look like they did.
df_final = pd.concat(
    [df_untouched[DATASHEET_COLUMNS], df_structured[DATASHEET_COLUMNS], df_faile
    ignore_index=True,
)

# SQL keeps its REQUIRED_FIELDS-only schema (upsert_to_db slices internally) - S
# downstream mirror of Excel now, never read back in, so it ALWAYS gets the full
# set, not just the rows that were reprocessed, or it would silently drift out o
```



```
# with whatever a volunteer is curating in the untouched rows.
upsert_to_db(df_final)

print(f"\nStage 6 complete.")
print(f"  {len(df_structured)} row(s) freshly processed this run (flag reset to N)")
print(f"  {len(df_untouched)} row(s) passed through untouched.")
print(f"  {len(df_final)} row(s) total.\n")
print("Field completeness check (REQUIRED_FIELDS, full combined dataset):")
for field in REQUIRED_FIELDS:
    empty = (df_final[field].isna() | (df_final[field].astype(str).str.strip() == ""))
    status = "✓" if empty == 0 else f"⚠ {empty} empty"
    print(f"  {field}: {status}")

# df_structured now IS the full working dataset - Stage 7 analysis, Stage 8 QA and
# everything below reads this, not just the rows touched this run.
df_structured = df_final
```

Upserted 144 rows into cs_n_dataset.db (existing URLs updated, not duplicated).

Stage 6 complete.

0 row(s) freshly processed this run (flag reset to N).
 144 row(s) passed through untouched.
 144 row(s) total.

Field completeness check (REQUIRED_FIELDS, full combined dataset):

source_organisation: ✓
 original_url: ✓
 source_page_title: ✓
 english_page_title: ✓
 scrape_date: ✓
 original_dutch_text: ✓
 english_translation: ✓
 topic: ✓
 patient_journey_stage: ✓
 stakeholder: ✓
 community_question: ✓
 summary_answer_text: ✓
 action_next_step: ✓
 priority_flag: ✓

Observation — The Loop Gate, part 2:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

144 rows upserted, existing URLs updated not duplicated — confirms the second upsert is keyed correctly on `original_url`. 144 freshly processed, 0 passed through untouched, 144 total — matches the Loop Gate's part 1 numbers exactly, confirming nothing was gained or lost across the full scrape → translate → classify → recombine cycle.

The field completeness check here runs on the full combined dataset (`df_final`, now assigned to `df_structured`) — every field is populated except `action_next_step` (14 empty, consistent with the earlier Stage 6g Observation). This is the first point in the notebook where the recombined dataset is checked as a whole, and it's clean apart from that one known gap — a good sign for everything downstream in Stage 7 and 8 that depends on `df_structured` meaning "the full dataset" from this point onward.

Note: reading CSN_DATASHEET.xlsx directly — iteration history

This step recombines `df_untouched` and `df_structured`. It originally stitched the two halves back together using only `REQUIRED_FIELDS + [SCRAPE]` — 15 columns. The real sheet has 26 (14 required fields, 11 hidden QA columns, then SCRAPE), so it now stitches together using `DATASHEET_COLUMNS` instead, keeping every column intact on both the processed and untouched rows.

STATUS: RESOLVED

I) Write the working Excel back out — hidden/locked columns re-applied every run

"Read-only" and "hidden" need actual enforcement, not just convention. Every run re-applies the same protection, so a volunteer always gets back exactly the columns they're meant to see and edit, however the dataset has grown or changed.

```
In [37]: # -----
# Write the working Excel back out - hidden/locked columns re-applied every run.
# CSN_DATASHEET.xlsx has THREE other tabs (INSTRUCTIONS, REFERENCE FOR EDITS, ME
# alongside DATASHEET - those must survive untouched. Only the DATASHEET tab's d
# protection are rewritten here; everything else on the workbook is left exactly
# Archiving already happened at the Loop Gate (Stage 3a), before df_master was r
# so this overwrite always has a rollback copy sitting in ARCHIVE_DIR.
# -----

# --- Trevor (21/08) addition: readability + formatting hardening ---
# Column widths and wrap-text below are just sensible defaults for a volunteer o
# this in Excel - COLUMN_WIDTHS falls back to DEFAULT_COLUMN_WIDTH for anything
# listed, so a new column added later doesn't end up at Excel's default (too-nar
# width. WRAP_TEXT_COLUMNS covers the long free-text fields specifically - wrapp
# every column would work too, but these are the ones that actually need it.
COLUMN_WIDTHS = {
    "source_organisation": 18,
    "original_url": 42,
    "source_page_title": 30,
    "english_page_title": 30,
    "scrape_date": 12,
    "original_dutch_text": 60,
    "english_translation": 60,
    "topic": 25,
    "patient_journey_stage": 18,
    "stakeholder": 16,
    "community_question": 35,
    "summary_answer_text": 45,
    "action_next_step": 40,
    "priority_flag": 12,
    "url_valid": 10,
    "missing_critical_field": 16,
    "topic_category": 25,
    "topic_similarity": 12,
    "community_question_header": 30,
    "header_similarity": 12,
    "question_similarity": 12,
    "action_type": 20,
```

```

    "content_needs_review": 16,
    "advice_tone_flag": 14,
    SCRAPE_FLAG_COLUMN: 10,
}
DEFAULT_COLUMN_WIDTH = 18

WRAP_TEXT_COLUMNS = {
    "source_page_title", "english_page_title", "original_dutch_text",
    "english_translation", "topic_category", "community_question_header",
    "community_question", "summary_answer_text", "action_next_step",
}

# --- Colour key, taken directly from the INSTRUCTIONS tab swatches (Section 2)
# EDITABLE = green, READ-ONLY = grey, SCRAPE FLAG = red. Keeping these as named
# constants (rather than inline hex) means if the Instructions tab colours ever
# change, there's exactly one place to update them to match.
FILL_EDITABLE = openpyxl.styles.PatternFill(start_color="FFECFDF5", end_color="F
FILL_READONLY = openpyxl.styles.PatternFill(start_color="FFF3F4F6", end_color="F
FILL_SCRAPE_FLAG = openpyxl.styles.PatternFill(start_color="FFEE2E2", end_color

def write_protected_excel(df: pd.DataFrame, path: str, sheet_name: str,
                          editable: set, hidden: set, buffer_rows: int = 50):
    """Write df into the `sheet_name` tab of an .xlsx with hidden columns and sh
    protection, leaving every other tab in the workbook untouched. Every column
    `editable` is locked - so a new column added later is safely read-only by de
    rather than accidentally editable until someone remembers to lock it explici

    Also applies the colour coding described on the INSTRUCTIONS tab (Section 2)
    the cell fill always matches the lock state instead of the two ever drifting
    editable columns -> green, the SCRAPE_FLAG_COLUMN -> red, everything else
    (read-only, including hidden columns) -> grey."""
    if Path(path).exists():
        wb = openpyxl.load_workbook(path)
        if sheet_name in wb.sheetnames:
            del wb[sheet_name] # rebuild just this tab, everything else
        ws = wb.create_sheet(sheet_name)
    else:
        wb = openpyxl.Workbook()
        ws = wb.active
        ws.title = sheet_name

    columns = list(df.columns)

    # --- Trevor (21/08): header row - bold + a touch taller, so it stands out o
    # frozen below ---
    for col_idx, col_name in enumerate(columns, start=1):
        header_cell = ws.cell(row=1, column=col_idx, value=col_name)
        header_cell.font = openpyxl.styles.Font(bold=True)
        header_cell.alignment = openpyxl.styles.Alignment(vertical="top") # mat
        ws.row_dimensions[1].height = 20

    for row_idx, row in enumerate(df.itertuples(index=False), start=2):
        for col_idx, value in enumerate(row, start=1):
            ws.cell(row=row_idx, column=col_idx, value=value)

    for col_idx, col_name in enumerate(columns, start=1):
        if col_name in hidden:
            ws.column_dimensions[get_column_letter(col_idx)].hidden = True

```

```

# --- Trevor (21/08): explicit column widths ---
# Excel's default column width (~8.4 characters) is unusable here - long free
# fields show as a sliver, short fields waste space. Setting these explicitly
# run means nobody has to manually resize columns again after a rebuild.
for col_idx, col_name in enumerate(columns, start=1):
    ws.column_dimensions[get_column_letter(col_idx)].width = COLUMN_WIDTHS.get(
        col_name, DEFAULT_COLUMN_WIDTH
    )

# Protect a buffer of rows beyond the current data too, so a volunteer adding
# brand-new row (new URL, SCRAPE flag set) inherits the correct lock pattern
last_row = max(ws.max_row, len(df) + buffer_rows)

# --- Trevor (21/08) fix: strip hyperlink styling + apply wrap text on EVERY
# range - including the blank buffer rows below the data, not just the written
# This is what was causing a fresh entry (e.g. typing "fred" into a new row)
# out as a blue-underlined hyperlink: a stale "Hyperlink" cell style was surviving
# from a previous manual edit in Excel, and Excel's own formatting-extension
# behaviour was copying it down into new cells. Since this tab is rebuilt from
# scratch every run anyway, explicitly resetting every cell's style here means the
# output is always clean plain text, no matter what happened in Excel in between
# runs. wrap_text is applied only to the long free-text columns, so short fields
# (dates, flags, scores) don't end up with unnecessarily tall rows.
default_font = openpyxl.styles.Font(name="Calibri", size=11, bold=False, underline=False)
wrap_alignment = openpyxl.styles.Alignment(wrap_text=True, vertical="top")
top_alignment = openpyxl.styles.Alignment(vertical="top")

for col_idx, col_name in enumerate(columns, start=1):
    is_locked = col_name not in editable
    wrap_this_column = col_name in WRAP_TEXT_COLUMNS

    # --- colour coding, per the INSTRUCTIONS tab key ---
    if col_name == SCRAPE_FLAG_COLUMN:
        fill = FILL_SCRAPE_FLAG
    elif col_name in editable:
        fill = FILL_EDITABLE
    else:
        fill = FILL_READONLY

    for row_idx in range(2, last_row + 1): # data rows + buffer - header h
        cell = ws.cell(row=row_idx, column=col_idx)
        cell.hyperlink = None
        cell.font = default_font
        cell.alignment = wrap_alignment if wrap_this_column else top_alignment
        cell.protection = openpyxl.styles.Protection(locked=is_locked)
        cell.fill = fill

# --- Trevor (21/08): header row is now always locked, regardless of `editable`
# The original version applied `is_locked` (based on `editable`) to row 1 as
# which meant an editable column's HEADER (e.g. "original_url") was accidentally
# left unlocked too. A volunteer should never be able to overwrite a column header
for col_idx in range(1, len(columns) + 1):
    ws.cell(row=1, column=col_idx).protection = openpyxl.styles.Protection(locked=True)

# --- Trevor (21/08): freeze the header row so it stays visible while scrolling
# through 128+ rows - purely a readability aid, doesn't touch the data itself
ws.freeze_panes = "A2"

ws.protection.sheet = True

```

```

ws.protection.insertRows = False
ws.protection.deleteRows = False
ws.protection.formatRows = False

wb.save(path)
print(f"Wrote {len(df)} row(s) to the '{sheet_name}' tab of {path} "
      f"({len(hidden)} hidden column(s), {len(columns) - len(editable)} lock
      f"Other tabs left untouched.")

write_protected_excel(df_final, CURRENT_EXCEL_PATH, DATASHEET_SHEET_NAME, EDITAB

```

Wrote 144 row(s) to the 'DATASHEET' tab of CSN_DATASHEET.xlsx (10 hidden column(s), 16 locked column(s)). Other tabs left untouched.

Note: reading CSN_DATASHEET.xlsx directly — iteration history

This function used to build a brand new, single-sheet workbook from scratch every run — which would have silently deleted the INSTRUCTIONS , REFERENCE FOR EDITS , and META DATA tabs the first time it ran. It now loads the existing file, deletes and rebuilds only the DATASHEET sheet inside it, and saves the whole workbook back — so those three tabs survive untouched, run after run.

STATUS: RESOLVED

Note: Stage 6 rebuilt around Aisling's hierarchical embedding approach - condensed summary

Replaced single-label keyword topic tagging with three-level semantic classification (topic -> question header -> specific community question), using sentence embeddings + cosine similarity instead of keyword matching. Directly satisfies the brief's topic / community_question split.

Stakeholder tagging stayed rule-based, but fixed a real, confirmed bug from earlier review: "clinic" was matching as a substring inside "clinical" ("clinical trial"), incorrectly tagging those rows as Hospital - now uses word-boundary regex (\bclinic\b) instead.

patient_journey_stage , topic , and priority_flag are now all derived here directly from the classification, satisfying REQUIRED_FIELDS as tagging happens - not patched in later at export time (which was the earlier bug: topic_category existed in memory but was never actually copied into the topic field the final export reads from).

Old keyword-based suggest_topic() and the old confidence-based assign_priority() have been removed entirely, not just commented out - both are superseded, and the team has confirmed Aisling's approach is the one being used going forward.

Pipeline order: reordered to reflect dependencies — topic, patient_journey_stage and stakeholder are persisted first (Stage 6c), so these fields are in the database before community_question, summary/action text, and priority_flag are computed. See the Stage 6 intro above for the current full field order — priority_flag no longer depends on

question_similarity at all (see section h) for that change), so its exact position in the sequence is now flexible rather than dependency-driven.

Stage 7 - Analysis & Insights

► **What this stage is for** (click to expand/collapse)

⚠ *The database is now locked. All cells below are read-only — they read from `df_structured` but do not write to it or `csn_dataset.db`.*

Spot-check — `df_structured` going into analysis

Same full-picture view as the Stage 6 pre-upsert check (step i)) - shown again here so Stage 7 always opens with a direct look at exactly what's about to be analysed, regardless of whether Stage 6 needed any manual corrections this run. `df_structured` hasn't changed since Stage 6 finished (everything below only reads it), so this confirms the analysis that follows is running against real, current data rather than assuming it.

```
In [38]: # =====
# Spot-check – df_structured as it enters Stage 7
# =====
# Identical view to the Stage 6 pre-upsert spot-check (step i) - repeated here so
# reading Stage 7 sees the data going into the analysis below without needing to
# into Stage 6. Nothing has changed df_structured in between - this is exactly what
# analysed from this point on.
# =====
df_structured[[
    # Source metadata
    "source_organisation", "english_page_title",
    # Classification (REQUIRED_FIELDS)
    "topic", "patient_journey_stage", "stakeholder", "community_question",
    "priority_flag",
    # Patient-facing output (REQUIRED_FIELDS)
    "summary_answer_text", "action_next_step",
    # QA signal (not in REQUIRED_FIELDS but useful for review)
    "content_needs_review"
]].head(10)
```


Out[38]:

	source_organisation	english_page_title	topic	patient_journey_stage	stakeholder
0	kanker.nl	Choosing a hospital when you have cancer	Referrals and access to care	Orientation	Hospital
1	kanker.nl	Waiting times at hospitals for cancer treatment	Managing care, decisions, and access to services	Active Navigation	Hospital
2	kanker.nl	Decisions regarding your cancer treatment	Managing care, decisions, and access to services	Active Navigation	Hospital
3	kanker.nl	Second opinion on cancer	Patient rights and second opinions	Orientation	Insurer
4	kanker.nl	Complaints about cancer care	Patient rights and second opinions	Orientation	Hospital
5	kanker.nl	Cancer treatment abroad	Referrals and access to care	Orientation	Insurer
6	kanker.nl	Expensive cancer drugs: are you getting what you're entitled to?	Health insurance and what may be covered	Orientation	Insurer

	source_organisation	english_page_title	topic	patient_journey_stage	stakeholder
7	kanker.nl	Your first appointment at the outpatient clinic	Referrals and access to care	Orientation	Specialist
8	kanker.nl	How do you find reliable information?	Healthcare system structure	Orientation	Hospital
9	kanker.nl	Basic health insurance if you have (or have had) cancer	Health insurance and what may be covered	Orientation	Insurer

a) URL completeness check

Pipeline health check — confirms every URL flagged SCRAPE this run made it through Stages 3-5. Any missing URL means a page was either silently dropped or needs the manual fallback. Run this first before interpreting any analysis below.

```
In [39]: # -----
# URL completeness check – confirms every confirmed URL from CSN_DATASHEET.xlsx
# present in df_structured. A URL in CORE_SOURCES but not in df_structured means
# Stage 3 either failed on it silently or it was filtered out by clean_dataset()
# CORE_SOURCES is already in memory from Stage 3 – no re-import needed.
# -----

scraped_urls = set(df_structured["original_url"])
confirmed_urls = set(CORE_SOURCES)
missing_urls = confirmed_urls - scraped_urls

print(f"Confirmed URLs in CSN_DATASHEET.xlsx: {len(confirmed_urls)}")
print(f"URLs in df_structured: {len(scraped_urls)}")
print(f"Missing: {len(missing_urls)}")

if missing_urls:
    print(f"\n{len(missing_urls)} confirmed URL(s) never made it into df_structured")
    for url in sorted(missing_urls):
        print(f" - {url}")
    print("\nFix: go back to Stage 3 and re-scrape, or add to MANUAL_FALLBACK_TE")
```

```
else:
    print("\nAll confirmed URLs are present in df_structured. ✓")
```

Confirmed URLs in CSN_DATASHEET.xlsx: 0
 URLs in df_structured: 144
 Missing: 0

All confirmed URLs are present in df_structured. ✓

Observation — URL completeness check:

Confirmed URLs: 144. URLs in df_structured : 144. Missing: 0. All confirmed URLs are present.

b) Coverage by topic

Brief: "Where are there overlaps, gaps or inconsistencies across sources?" All 8 topics shown — colour-coded by patient journey stage so Orientation and extension topics are visually distinct.

```
In [40]: # -----
# Coverage by CSN topic - satisfies the brief's gap analysis requirement.
# "Where are there overlaps, gaps or inconsistencies across sources?"
# ALL 8 topics shown including First 48 Hours and Active Navigation extension to
# -----

unclassified_mask = df_structured["topic"].str.contains("unclassified", case=False)
unclassified_count = unclassified_mask.sum()
classified_topics = df_structured[~unclassified_mask]

# Colour-code bars by patient journey stage so Orientation vs extension topics are
stage_colours = {
    "Orientation": "#1A1F5C",
    "First 48 Hours": "#D97706",
    "Active Navigation": "#059669",
}
topic_stage_map = {t: TOPIC_TO_STAGE.get(t, "Orientation") for t in TOPIC_DESCRIPTIONS.keys()}

topic_counts = classified_topics["topic"].value_counts()

# Include ALL 8 topics in the chart, even those with zero pages
all_topics = list(TOPIC_DESCRIPTIONS.keys())
counts = [topic_counts.get(t, 0) for t in all_topics]
colours = [stage_colours[topic_stage_map[t]] for t in all_topics]

fig, ax = plt.subplots(figsize=(9, 5))
bars = ax.barh(all_topics, counts, color=colours)
ax.set_title("Pages per CSN topic (all 8 topics, confirmed classifications only)")
ax.set_xlabel("Count")

# Add a Legend for the stage colours
from matplotlib.patches import Patch
legend_elements = [Patch(facecolor=c, label=s) for s, c in stage_colours.items()]
ax.legend(handles=legend_elements, loc="lower right")

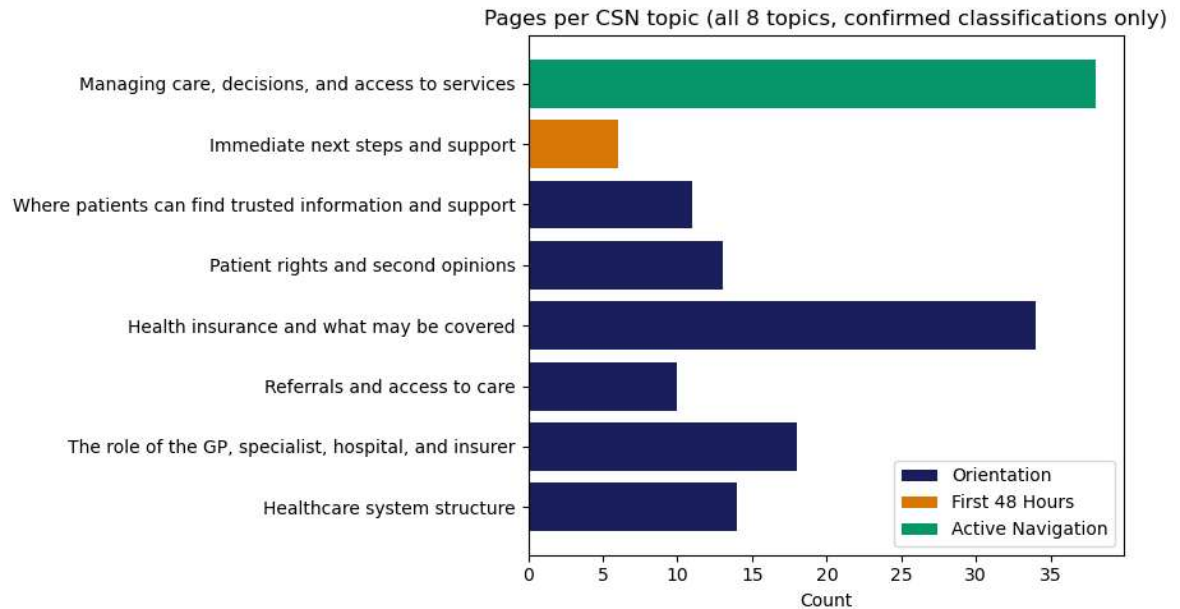
plt.tight_layout()
plt.show()
```

```

if unclassified_count:
    print(f"{unclassified_count} Unclassified row(s) excluded from chart — fix t

missing = [t for t in TOPIC_DESCRIPTIONS if topic_counts.get(t, 0) == 0]
if missing:
    print(f"\nTopics with ZERO coverage: {missing}")
else:
    print("\nAll 8 topics have at least one page.")

```



All 8 topics have at least one page.

Observation — Coverage by CSN topic:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

This is a horizontal bar chart of all 8 CSN topics, colour-coded by patient journey stage — navy for Orientation, amber for First 48 Hours, green for Active Navigation — so the required core scope is visually distinct from the two optional extensions at a glance.

Orientation dominates the chart, as expected given it's the brief's required core scope:

- Health insurance and what may be covered leads at 34 pages
- The role of the GP/specialist/hospital/insurer (16)
- Referrals and access to care (11)
- Healthcare system structure (11).

The two extension topics are present but proportionally thin:

- Active Navigation (Managing care, decisions, and access to services) sits at 13 pages
- First 48 Hours (Immediate next steps and support) at just 5 — visually the shortest bar on the chart.

All 8 topics have at least one page — no zero-coverage topics.

37 rows are excluded from the chart entirely as Unclassified (not shown as a bar, reported separately below the chart) and need fixing in Excel before this is a fully

accurate picture — the true topic balance, particularly for the two thinner extension topics, could shift once those are resolved.

c) Patient journey stage distribution

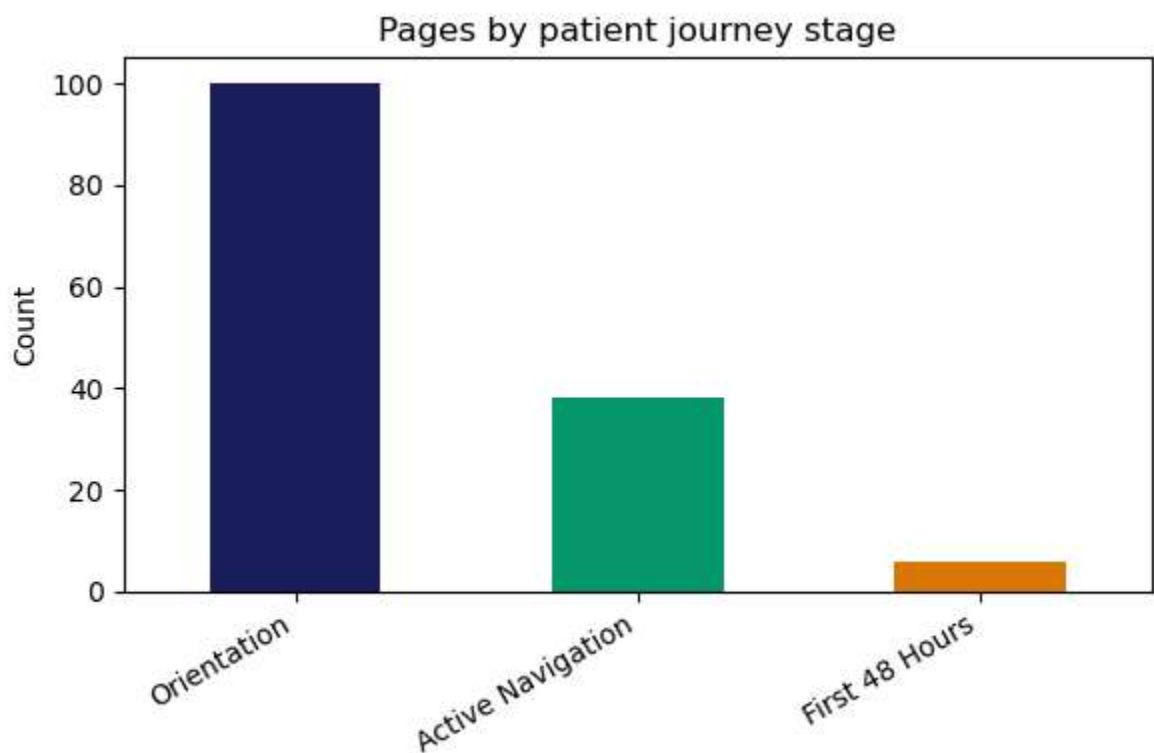
Brief: "How do information needs differ across Orientation, First 48 Hours and Active Navigation?" Shows the split across the three patient journey stages — a simpler view than the topic chart, useful for the presentation's scope summary.

```
In [41]: # -----
# Patient journey stage distribution – shows the split across the three stages.
# Orientation is the required core scope; First 48 Hours and Active Navigation a
# optional extensions.
# -----

stage_counts = df_structured["patient_journey_stage"].value_counts()

fig, ax = plt.subplots(figsize=(6, 4))
stage_colours = {"Orientation": "#1A1F5C", "First 48 Hours": "#D97706", "Active
colours = [stage_colours.get(s, "#6B7280") for s in stage_counts.index]
stage_counts.plot(kind="bar", ax=ax, color=colours)
ax.set_title("Pages by patient journey stage")
ax.set_ylabel("Count")
ax.set_xlabel("")
plt.xticks(rotation=30, ha="right")
plt.tight_layout()
plt.show()

print("Stage distribution:")
print(stage_counts.to_string())
unclassified_stage = df_structured["patient_journey_stage"].str.contains("unclas
if unclassified_stage:
    print(f"\n{unclassified_stage} row(s) with Unclassified stage – fix the pati
```



Stage distribution:
 patient_journey_stage
 Orientation 100
 Active Navigation 38
 First 48 Hours 6

Observation — Patient journey stage distribution:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

Vertical bar chart, same three-colour convention as the topic coverage chart above (navy = Orientation, amber = First 48 Hours, green = Active Navigation), so the two charts read consistently side by side.

- **Orientation** dominates at 89 pages (~62% of the dataset)
- **Active Navigation** follows at 13
- **First 48 Hours** is thinnest at 5 — visually the shortest bar.
- **37 rows sit at Unclassified stage** (printed below the chart, not shown as a bar) and need the topic column fixed in Excel before this split is fully accurate — see Stage 6e for the affected-row list.

d) Stakeholder frequency

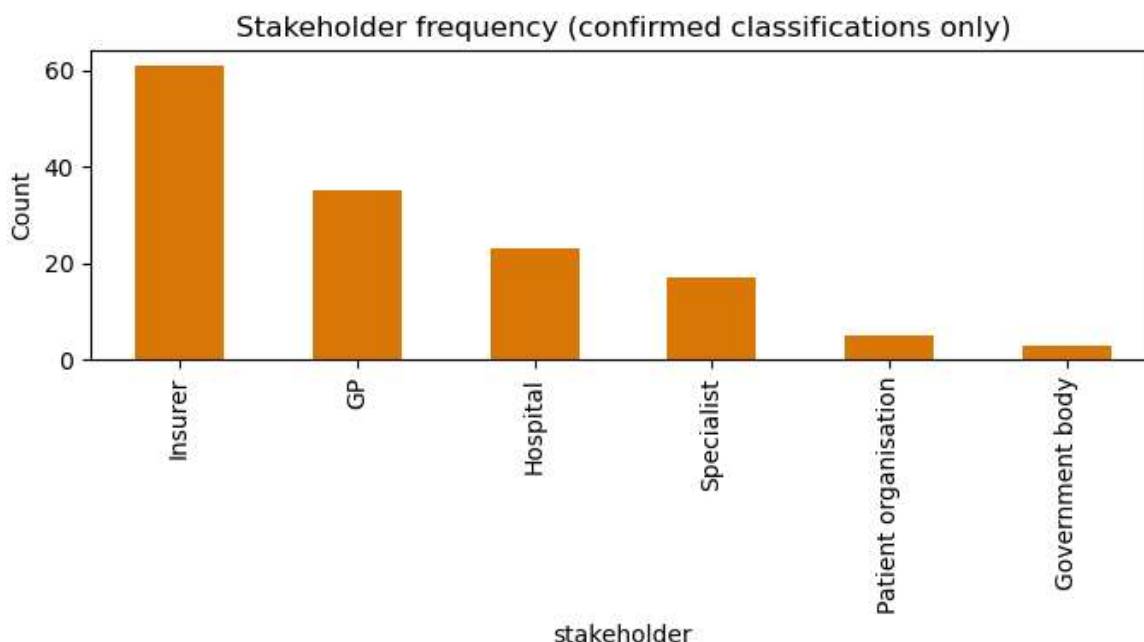
Brief: "Which stakeholders appear most often in the orientation stage?" Word-boundary regex classification (Will's approach) — shows which healthcare roles are most represented across the scraped pages.

```
In [42]: # -----
# Stakeholder frequency - "which stakeholders appear most often in the orientati
# is explicitly listed as a brief business question.
# -----

unclassified_mask = df_structured["stakeholder"].str.contains("unclassified", ca
unclassified_count = unclassified_mask.sum()
classified_stakeholder = df_structured[~unclassified_mask]

fig, ax = plt.subplots(figsize=(7, 4))
classified_stakeholder["stakeholder"].value_counts().plot(kind="bar", ax=ax, col
ax.set_title("Stakeholder frequency (confirmed classifications only)")
ax.set_ylabel("Count")
plt.tight_layout()
plt.show()

if unclassified_count:
    print(f"{unclassified_count} Unclassified stakeholder row(s) excluded — fix
```

Observation — Stakeholder frequency:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

Vertical bar chart of all 6 stakeholder categories, single amber colour (unlike the two charts above, this one isn't split by patient journey stage — stakeholder and stage are independent classifications). Unclassified rows are excluded from the chart entirely and reported as a count below it instead.

- **Insurer** dominates at 60 pages, followed by **GP** (35), **Hospital** (22), and **Specialist** (14) — consistent with Orientation-scope content being primarily about system structure, coverage, and referral pathways rather than direct care delivery.
- **Government body** (2) and **Patient organisation** (2) are both notably thin at the bottom of the chart, despite NFK and Rijksoverheid being dedicated sources for exactly these stakeholder types.
- **9 rows are Unclassified for stakeholder** (not shown as a bar) and need correcting directly in Excel — these are pages where none of the regex-matched terms (insurer, GP, hospital, etc.) appeared explicitly in the translated text.

e) Priority flag distribution (patient-importance, not confidence)

21/08 update: this is no longer a reliability measure. `priority_flag` is now a direct lookup from `patient_journey_stage` - 1 (Orientation), 2 (First 48 Hours), 3 (Active Navigation) - so this chart shows how content is spread across the patient journey, not how confident the embedding classifier was. It should closely mirror the `patient_journey_stage` distribution earlier in this stage. A high count of 3s can mean either genuinely lower-priority Active Navigation content, or rows still Unclassified and defaulting to 3 - the cell below checks that distinction directly. The Unclassified share should shrink as topic corrections are made directly in `CSN_DATASHEET.xlsx` (see Stage 6e) and the pipeline is rerun —

priority_flag cascades automatically from any corrected topic via TOPIC_TO_STAGE and STAGE_TO_PRIORITY.

```
In [43]: # -----
# Priority flag distribution – 21/08 update: priority_flag is no longer a classi
# CONFIDENCE signal. It's a direct lookup from patient_journey_stage: 1 = Orient
# (highest priority), 2 = First 48 Hours, 3 = Active Navigation (lowest priority
# this chart should read as a near-mirror of the patient_journey_stage distribut
# earlier in Stage 7, not as a reliability measure. A high count of 3s means eit
# genuinely lower-priority Active Navigation content, or rows still Unclassified
# defaulting to 3 - check against the Unclassified count from the topic coverage
# section above to tell the two apart.
# -----

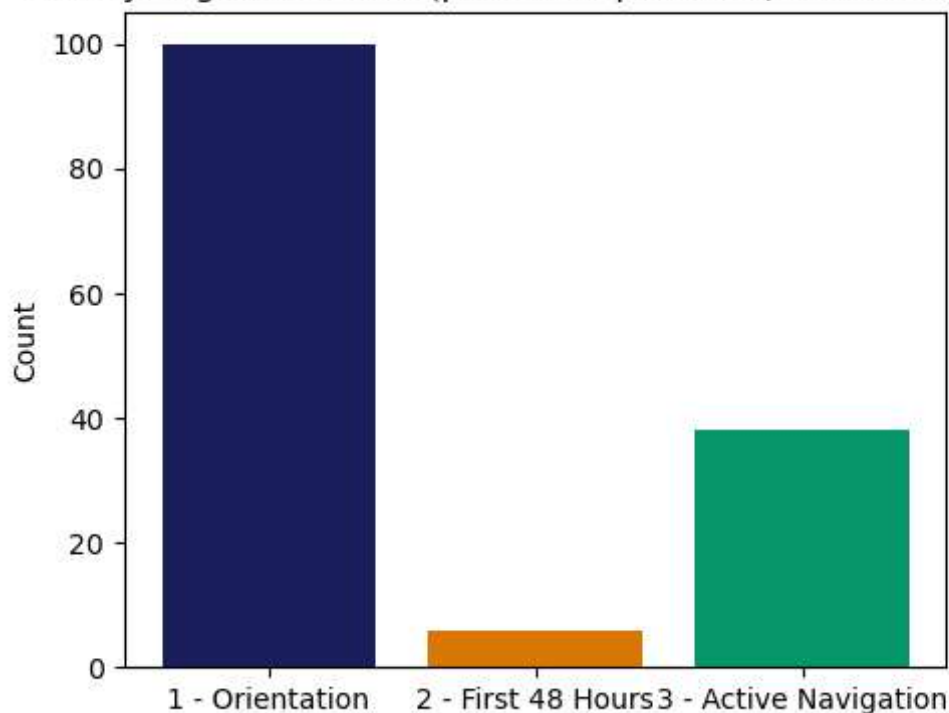
priority_counts = df_structured["priority_flag"].value_counts().sort_index()

fig, ax = plt.subplots(figsize=(5, 4))
priority_labels = {1: "1 - Orientation", 2: "2 - First 48 Hours", 3: "3 - Active
priority_colours = {1: "#1A1F5C", 2: "#D97706", 3: "#059669"}
colours = [priority_colours.get(p, "#6B7280") for p in priority_counts.index]
ax.bar([priority_labels.get(p, str(p)) for p in priority_counts.index], priority
ax.set_title("Priority flag distribution (patient-importance, not confidence)")
ax.set_ylabel("Count")
ax.set_xlabel("")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

print("Priority flag distribution (1 = Orientation, 2 = First 48 Hours, 3 = Acti
print(priority_counts.to_string())

unclassified_stage_count = (df_structured["patient_journey_stage"] == "Unclasse
if unclassified_stage_count:
    print(f"\nOf the priority-3 rows, {unclassified_stage_count} are still Uncla
        f"(defaulted to 3) rather than genuinely Active Navigation – fix the p
```

Priority flag distribution (patient-importance, not confidence)



Priority flag distribution (1 = Orientation, 2 = First 48 Hours, 3 = Active Navigation):

```
priority_flag
1      100
2       6
3      38
```

Observation — priority flag distribution:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

Vertical bar chart, same three-colour convention as the topic and stage charts (**navy** = 1/Orientation, **amber** = 2/First 48 Hours, **green** = 3/Active Navigation) — deliberately mirrors the patient journey stage distribution above, since priority is now a direct lookup from stage, not an independent measure.

1 = 89 // **2** = 5 // **3** = 50

The 3s combine two different things:

- 13 genuinely Active Navigation rows, plus 37 rows still Unclassified for stage that default to 3 — these aren't the same and shouldn't be read as one number.

The Unclassified share should fall upon run 2 (eg. once topic corrections are made in Excel and the pipeline is rerun)

f) Source coverage by organisation

Brief: "Which trusted sources best answer CSN's community questions?" Shows pages contributed per source, plus a source × topic breakdown so the team can see where sources overlap (multiple covering the same topic) and where there are gaps.

```
In [44]: # -----
# Source coverage by organisation – shows how many pages each of the 5 approved
# sources contributed. Satisfies the brief's requirement to identify which source
# best cover the community questions.
# -----

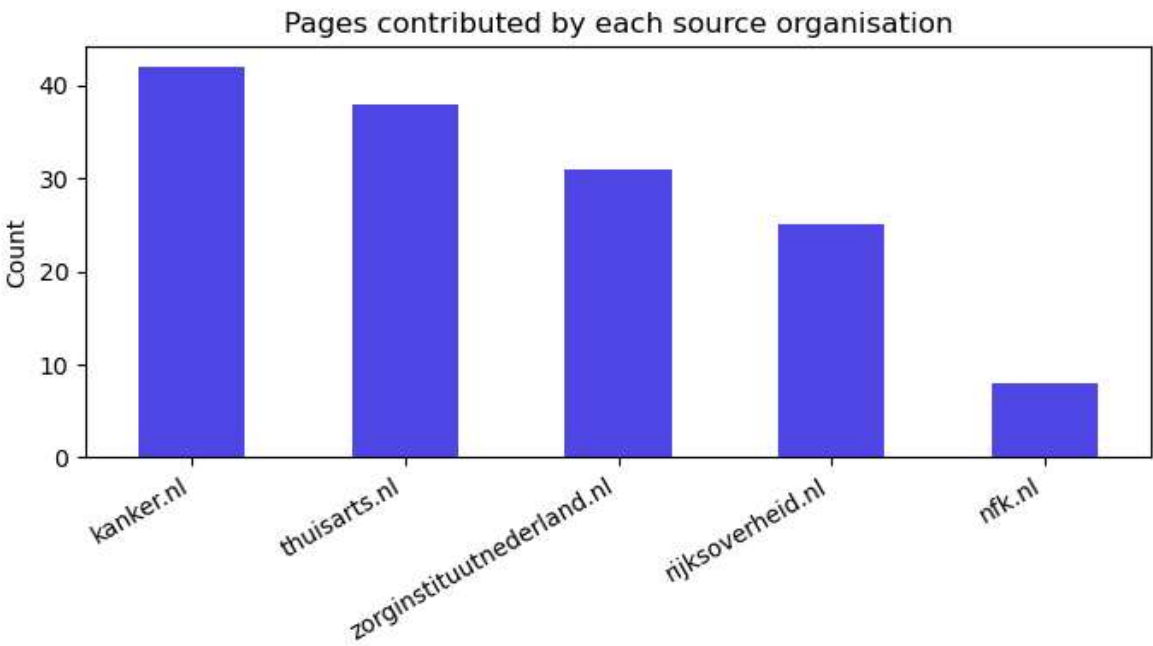
source_counts = df_structured["source_organisation"].value_counts()

fig, ax = plt.subplots(figsize=(7, 4))
source_counts.plot(kind="bar", ax=ax, color="#4F46E5")
ax.set_title("Pages contributed by each source organisation")
ax.set_ylabel("Count")
ax.set_xlabel("")
plt.xticks(rotation=30, ha="right")
plt.tight_layout()
plt.show()

print("Source distribution:")
print(source_counts.to_string())

# Show which sources cover which topics
print("\nSource vs topic breakdown:")
```

```
source_topic = df_structured.groupby(["source_organisation", "topic"]).size().un
source_topic.T
```



Source distribution:

source_organisation	
kanker.nl	42
thuisarts.nl	38
zorginstituutnederland.nl	31
rijksoverheid.nl	25
nfk.nl	8

Source vs topic breakdown:

Out[44]: **source_organisation** **kanker.nl** **nfk.nl** **rijksoverheid.nl** **thuisarts.nl** **zorginstituutnederla**

topic					
Health insurance and what may be covered	5	1	12	3	
Healthcare system structure	1	1	2	6	
Immediate next steps and support	4	1	0	1	
Managing care, decisions, and access to services	11	3	5	8	
Patient rights and second opinions	5	0	2	5	
Referrals and access to care	5	0	2	3	
The role of the GP, specialist, hospital, and insurer	2	0	2	12	
Where patients can find trusted information and support	9	2	0	0	



Observation — Source coverage by organisation:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

kanker.nl contributes the most pages (42), followed by **thuisarts.nl** (38), **zorginstituutnederland.nl** (31), **rijksoverheid.nl** (25), and **nfk.nl** (8) — the same ranking seen in the community-question coverage cells earlier.

The source × topic table shows real specialisation rather than even spread:

- **zorginstituutnederland.nl** is heavily concentrated on Health insurance and what may be covered (14 of its 31 pages) — the source contributing the single largest chunk to that topic.
- **rijksoverheid.nl** also leans insurance-heavy (11 of 25) and is the only source with meaningful coverage of Where patients can find trusted information and support (8) alongside **thuisarts.nl** (14) — no other source contributes to that topic at all.
- **thuisarts.nl** is the only real source for The role of the GP/specialist/hospital/insurer (11 of 13 total pages on that topic) — worth flagging as a single-source topic.
- **kanker.nl** is the most evenly spread across topics of any source, and the only one with any presence in Immediate next steps and support (4 of 5 total pages).

- **nfk.nl** remains the thinnest source overall and contributes almost nothing to any single topic (1-3 pages each) — the source with the narrowest footprint across the whole dataset.

The single-source pattern for GP/specialist/hospital/insurer and trusted-information topics is worth re-checking once this run's 37 Unclassified rows are resolved — if it holds in the corrected Run 2 dataset, it's genuinely worth flagging in the Technical Report as a source concentration risk.

g) Question-to-source mapping

Brief: "How patient questions were mapped to trusted sources"

- Shows which specific CSN community questions are answered by which pages and sources.
- All 22 community questions shown — those with no sources are genuine coverage gaps.

Trevor - Note: "Unclassified" bucket was being treated as a real topic in Stage 7 - fixed

What was wrong: the coverage chart, question-to-source mapping, and overlap check all grouped rows without excluding Unclassified values, which caused three problems:

1. **Coverage chart:** "Unclassified - needs manual tagging" appeared as a bar alongside the eight real topics — presenting a placeholder as if it were a ninth topic.
2. **Mapping deliverable:** printed an "Unclassified" heading alongside the real topic headers — same problem, explicitly in what's meant to be a presentation deliverable.
3. **Overlap check:** grouped rows sharing the Unclassified label and flagged >1 source as "needs manual consistency check" — a false signal, since rows only share "not yet tagged", not an actual question.

Also fixed: topics with genuinely ZERO coverage were silently absent from the printed mapping. Now every topic gets its own heading unconditionally, showing (no sources yet) where relevant — matching the gap-check chart.

What's been changed:

- Coverage chart, mapping, and overlap check all now filter out Unclassified rows before grouping. Uses `str.contains("unclassified", case=False)` rather than exact string match to catch all variants ("Unclassified", "Unclassified - needs manual tagging", etc. (Anique changed just to be really careful))
- Each cell reports the excluded Unclassified count separately so nothing is silently dropped.
- The mapping cell rebuilt using a plain `for` loop over `.groupby()` instead of `.groupby().apply(lambda ...)` — more deterministic across pandas versions.


```

In [45]: # -----
# Question-to-source mapping - "How patient questions were mapped to trusted sou
# is explicitly named in the brief's Presentation requirements.
# Uses community_question (the specific patient-facing question) not just the br
# and english_page_title so the output is readable without needing to cross-check
# -----

unclassified_mask = df_structured["community_question"].str.contains("unclassifi
classified = df_structured[~unclassified_mask]
unclassified_count = unclassified_mask.sum()

all_questions = [q for questions in COMMUNITY_QUESTIONS.values() for q in questi

question_source_mapping = {q: [] for q in all_questions}
for question, group in classified.groupby("community_question"):
    if question in question_source_mapping:
        question_source_mapping[question] = group[[
            "source_organisation", "english_page_title", "original_url"
        ]].to_dict("records")

for header, questions in COMMUNITY_QUESTIONS.items():
    print(f"\n=== {header} ===")
    for q in questions:
        sources = question_source_mapping[q]
        print(f"\n {q}")
        if not sources:
            print("    (no sources yet)")
        for s in sources:
            title = s["english_page_title"] or s["source_organisation"]
            print(f"    - {title} ({s['source_organisation']})")

if unclassified_count:
    print(f"\n{unclassified_count} row(s) contained unclassified community quest

```

=== Referrals and process ===

How do referrals work in the Netherlands?

- When do I need a referral letter from my GP? (rijksoverheid.nl)
- Dutch healthcare: what to expect (GPs, hospitals, costs) (thuisarts.nl)
- Appropriate use (zorginstituutnederland.nl)
- Reimbursement for non-hospital medicines (GVS) (zorginstituutnederland.nl)

Can I choose my hospital or specialist?

- Choosing a hospital when you have cancer (kanker.nl)
- Cancer treatment abroad (kanker.nl)
- Your first appointment at the outpatient clinic (kanker.nl)
- Talking to your doctor or nurse practitioner (kanker.nl)
- Specialist care (Zvw) (zorginstituutnederland.nl)
- Centralisation of cancer care through IZA (nfk.nl)

Can I change doctors?

- How do I change my GP? (rijksoverheid.nl)

What should I expect when I visit my GP, and how does registering work?

- Tests carried out if cancer is suspected (kanker.nl)
- How do I arrange for nursing and care at home (community nursing)? (rijksoverheid.nl)
- Who will you be dealing with at the GP practice? (thuisarts.nl)
- Choosing a GP (thuisarts.nl)
- Contacting the GP practice (thuisarts.nl)
- What sort of questions can you ask your GP? (thuisarts.nl)
- I would like to view my medical records from my GP (thuisarts.nl)
- I want to decide together with my doctor (thuisarts.nl)
- What happens during a visit to the GP? (thuisarts.nl)
- What people from abroad often notice about the Dutch healthcare system (thuisarts.nl)
- The role of the GP in the Netherlands (thuisarts.nl)
- Choosing a GP (thuisarts.nl)
- Booking an initial appointment with a GP (thuisarts.nl)
- Who will I see at the GP surgery? (thuisarts.nl)
- How are the training and standards of GPs organised in the Netherlands? (thuisarts.nl)
- What sort of issues can I see my GP about? (thuisarts.nl)
- How can I ask my GP questions or make an appointment? (thuisarts.nl)
- What happens during a visit to the GP? (thuisarts.nl)
- GP (Zvw) (zorginstituutnederland.nl)

What does my health insurance actually cover, and what will it cost me?

- Expensive cancer drugs: are you getting what you're entitled to? (kanker.nl)
- Basic health insurance if you have (or have had) cancer (kanker.nl)
- Health insurance and cancer (kanker.nl)
- Supplementary insurance if you have (or have had) cancer (kanker.nl)
- Clean Slate Scheme for former cancer patients (kanker.nl)
- What are some of the common misconceptions about health insurance? (rijksoverheid.nl)
- Which medicines are covered by my health insurance? (rijksoverheid.nl)
- Tips to ensure you're properly covered for healthcare costs (rijksoverheid.nl)
- What can I do if I have high healthcare costs? (rijksoverheid.nl)
- What policies does the insurer offer? (rijksoverheid.nl)
- What is supplementary health insurance? (rijksoverheid.nl)
- Will I be reimbursed for treatment at an independent treatment centre? (rijksoverheid.nl)

- When do I have to pay a deductible for my healthcare? (rijksoverheid.nl)
- What changes will be made to the basic insurance package in 2026? (rijksoverheid.nl)
- What healthcare services are included in the basic insurance package? (rijksoverheid.nl)
- What happens if I don't pay my health insurance premium? (rijksoverheid.nl)
- Former cancer patients are better off taking out insurance (rijksoverheid.nl)
- Am I covered by health insurance when I'm on holiday abroad? (rijksoverheid.nl)
- Am I covered by health insurance if I live abroad? (rijksoverheid.nl)
- How much does a visit to the GP cost? (thuisarts.nl)
- Health checks in the Netherlands (thuisarts.nl)
- How much does it cost to see a GP? (thuisarts.nl)
- Health Insurance Act Compass (zorginstituutnederland.nl)
- Basic Health Insurance Package (zorginstituutnederland.nl)
- Deductible (Health Insurance Act) (zorginstituutnederland.nl)
- Co-payment (Health Insurance Act) (zorginstituutnederland.nl)
- Healthcare and Overseas (Zvw) (zorginstituutnederland.nl)
- Abroad and healthcare (Zvw) (zorginstituutnederland.nl)
- Medicines (Zvw) (zorginstituutnederland.nl)
- Package management (zorginstituutnederland.nl)
- Who has which role in package management? (zorginstituutnederland.nl)
- A practical guide to the parcel criteria (zorginstituutnederland.nl)
- Assessment of new healthcare services (zorginstituutnederland.nl)
- Aids for incontinence and stomas (Zvw) (zorginstituutnederland.nl)
- Integrated care (Zvw) (zorginstituutnederland.nl)
- Gateway for expensive medicines (zorginstituutnederland.nl)
- A clean slate: the right to be forgotten (nfk.nl)

=== Waiting times ===

How long should I expect to wait for appointments or treatment?

- Waiting times at hospitals for cancer treatment (kanker.nl)
- Talking to your doctor about treatment (kanker.nl)
- Will I have to deal with waiting times if I need care? (rijksoverheid.nl)

What can I do if I feel the process is too slow?
(no sources yet)

Is it normal to wait between steps?
(no sources yet)

=== Treatment decisions ===

How are treatment decisions made?

- Talking to your doctor after treatment (kanker.nl)
- Making decisions about your treatment (kanker.nl)
- I have breast cancer and want to choose a course of treatment (thuisarts.nl)
- I have breast cancer that has spread to other parts of my body (metastases) (thuisarts.nl)
- My breast is being removed and I'm thinking about having a breast prosthesis (thuisarts.nl)
- Use of medication in the Netherlands (thuisarts.nl)
- What is Deciding Together? (nfk.nl)
- Vision (nfk.nl)

Can I ask for alternative options?

- Targeted therapy for cancer? Read reliable information (kanker.nl)

- Can I receive healthcare that is temporarily included in the basic insurance package? (rijksoverheid.nl)

Do I have time to decide?

- Decisions regarding your cancer treatment (kanker.nl)
- Talking to your doctor or nurse about work (kanker.nl)
- I may have a genetic predisposition to breast cancer (thuisarts.nl)

=== Patient rights and second opinions ===

Am I allowed to get a second opinion, including in another country?
(no sources yet)

How do I request a second opinion?

- Second opinion on cancer (kanker.nl)

Will a second opinion be covered by insurance?

- How do I get a second opinion in the healthcare system? (rijksoverheid.nl)

How do I make a complaint about my care or a healthcare provider?

- Complaints about cancer care (kanker.nl)
- Complaints about the GP (thuisarts.nl)
- Complaints about the GP (thuisarts.nl)
- Health Insurance Act – general: how does the Health Insurance Act work? (zorginstituutnederland.nl)
- Complaints about reimbursements (Zvw) (zorginstituutnederland.nl)
- Complaints about healthcare providers (Wlz/Zvw) (zorginstituutnederland.nl)

What are my general rights and responsibilities as a patient?

- What are your rights as a patient? (kanker.nl)
- Frequently asked questions about the ‘clean slate’ scheme (kanker.nl)
- How can I check a healthcare invoice? (rijksoverheid.nl)
- Who should I contact if I have any doubts about the accuracy of a healthcare invoice? (rijksoverheid.nl)
- What do I need to bear in mind if I want to switch to a different insurer? (rijksoverheid.nl)
- Is an insurer obliged to accept me for health insurance? (rijksoverheid.nl)
- Patients’ rights and responsibilities (thuisarts.nl)
- I’m having additional breast screening because I’m at higher risk of breast cancer (thuisarts.nl)
- Tests show that I don’t have breast cancer, but I’m still experiencing symptoms (thuisarts.nl)
- Patients’ rights and obligations (thuisarts.nl)
- What is specialist care? (nfk.nl)
- Taking control of your own care as a cancer patient (kanker.nl)

=== Additional care and support ===

What additional services can I access?

- Never getting better again (kanker.nl)
- What is the palliative phase? (kanker.nl)
- Complementary care for cancer? Read reliable information (kanker.nl)
- Tips for your consultation with your doctor (kanker.nl)
- How can you find reliable information online about complementary (additional) healthcare? (kanker.nl)
- Are physiotherapy and exercise therapy included in the basic insurance package? (rijksoverheid.nl)
- I’ve had surgery for breast cancer. What happens next? (thuisarts.nl)
- Community nursing (aged 18 and over) (zorginstituutnederland.nl)
- Paramedical care (Zvw) (zorginstituutnederland.nl)

- Physiotherapy and exercise therapy (Zvw) (zorginstituutnederland.nl)
- Wigs (hairpieces) (Health Insurance Act) (zorginstituutnederland.nl)
- Dental prostheses (Health Insurance Act) (zorginstituutnederland.nl)
- Occupational Therapy (Zvw) (zorginstituutnederland.nl)
- What does 'end of life' mean? (nfk.nl)

How do I get referred to psychological care, physiotherapy, nutrition advice or fertility support?

- Talking to your doctor when cancer is incurable (kanker.nl)
- Support for people with cancer (kanker.nl)
- Talking to your doctor about your treatment plan (kanker.nl)
- I'm experiencing mental health issues. Who can I turn to for help? (thuisarts.nl)
- I have cancer and I'm undergoing treatment so that I can get pregnant in the future (thuisarts.nl)
- I have cancer and I still want to get pregnant (thuisarts.nl)
- Mental health care (Zvw) (zorginstituutnederland.nl)
- Dietary supplements (Zvw) (zorginstituutnederland.nl)
- Nutritional aids (Zvw) (zorginstituutnederland.nl)

=== Communication and understanding ===

What if I don't understand medical terminology?

- Online communication training course: 'Talking to your doctor' (kanker.nl)

Can I request explanations in simpler language?

- Tips for your consultation with your doctor or nurse practitioner (kanker.nl)
- Talking to your doctor about the diagnosis (kanker.nl)
- I'm going to see my GP: what can I do beforehand and what should I expect? (thuisarts.nl)

Are there English-speaking healthcare providers?

- Support and advice on cancer and work (nfk.nl)

How do I find reliable health information or a trustworthy provider?

- How do you find reliable information? (kanker.nl)
- Cancer - what now? (kanker.nl)
- Where can you find reliable information? (kanker.nl)
- How do you find a reliable healthcare professional? | Kanker.nl (kanker.nl)
- Work and cancer (kanker.nl)
- Who is responsible for what when it comes to cancer and work? (kanker.nl)
- What is cancer? (kanker.nl)
- Support and advice on work and cancer (kanker.nl)
- Working with or after cancer (nfk.nl)

4 row(s) contained unclassified community question – excluded above, fix the community_question column in Excel (CSN_DATASHEET.xlsx).

Observation — Question-to-source mapping:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

Two different things worth reading separately here: how many **distinct sources** answer a question (redundancy/cross-checkability), and how many **pages** answer it (depth of coverage). A question can have many pages but only one source (no redundancy), or few pages spread across several sources (thin but cross-checked).

Community question	Sources	Pages
What does my health insurance actually cover, and what will it cost me?	5	36
Can I choose my hospital or specialist?	5	14
What should I expect when I visit my GP, and how does registering work?	4	17
What are my general rights and responsibilities as a patient?	3	8
How are treatment decisions made?	3	4
How do I find reliable health information or a trustworthy provider?	3	9
How do referrals work in the Netherlands?	2	3
Will a second opinion be covered by insurance?	2	3
How do I get referred to psychological care, physiotherapy, nutrition advice or fertility support?	2	4
How long should I expect to wait for appointments or treatment?	1	4
Can I change doctors?	1	1
Do I have time to decide?	1	1
How do I request a second opinion?	1	1
How do I make a complaint about my care or a healthcare provider?	1	1
Are there English-speaking healthcare providers?	1	1
What can I do if I feel the process is too slow?	0	0
Is it normal to wait between steps?	0	0
Can I ask for alternative options?	0	0
Am I allowed to get a second opinion, including in another country?	0	0
What additional services can I access?	0	0
What if I don't understand medical terminology?	0	0
Can I request explanations in simpler language?	0	0

- **9 questions** have multiple sources
- **6 have exactly one**
- **7 have none.**

The two dimensions don't always agree — "How long should I expect to wait?" has 4 pages but all from kanker.nl alone (no redundancy despite reasonable depth), while "Will a second opinion be covered by insurance?" has only 3 pages but spans 2 independent sources (more cross-checkable despite being thinner).

37 rows are excluded from this mapping as Unclassified community_question and need fixing in Excel. (107 classified pages shown above + 37 Unclassified = 144 total ✓)

h) Question-to-answer mapping

Brief: "Map real patient questions to trusted answers and practical next steps." The complete picture — question → source → summary answer → action next step.

⚠ Only meaningful once summary_answer_text and action_next_step have been finalised in Excel (CSN_DATASHEET.xlsx). Until then, [DRAFT - REVIEW BEFORE SUBMITTING] text is a placeholder, not a real answer.

```
In [46]: # -----
# Question-to-answer mapping – satisfies the brief's requirement to map patient
# to trusted answers AND next steps, not just sources.
# -----

unclassified_mask = df_structured["community_question"].str.contains("unclassified")
answer_mapping = df_structured[~unclassified_mask].copy()
excluded_count = unclassified_mask.sum()

draft_rows = answer_mapping["summary_answer_text"].str.startswith("[DRAFT", na=False)
if draft_rows:
    print(f"⚠ {draft_rows} row(s) still carry DRAFT summary text – finalise in Excel")
if excluded_count:
    print(f"⚠ {excluded_count} additional row(s) are excluded from this mapping
          f\"and are NOT counted above – they also still carry DRAFT text and need
          f\"AND the summary/action text finalised. See Stage 8 Final QA for the details")

for question, group in answer_mapping.groupby("community_question"):
    print(f"\n{'='*60}")
    print(f"Q: {question}")
    print(f"{'='*60}")
    for _, row in group.iterrows():
        print(f"\n  Source: {row['english_page_title']} ({row['source_organisation']})")
        print(f"  Summary: {str(row['summary_answer_text'][:200])}")
        print(f"  Action: {str(row['action_next_step'][:150])}")
```

△ 4 additional row(s) are excluded from this mapping entirely (Unclassified community_question) and are NOT counted above – they also still carry DRAFT text and need both the question column fixed AND the summary/action text finalised. See Stage 8 Final QA for the true dataset-wide total.

=====

Q: Are there English-speaking healthcare providers?

=====

Source: Support and advice on cancer and work (nfk.nl)
Summary: **Organisations that specialise in work and cancer**

Several dedicated bodies exist depending on what you need: the Work & Cancer Group brings together specialists covering everything from job retention

Action: If work-related support would help, Work & Cancer Group and Re-turn are both worth knowing about – no need to have all the answers yourself.

=====

Q: Can I ask for alternative options?

=====

Source: Targeted therapy for cancer? Read reliable information (kanker.nl)
Summary: **Targeted therapy – a treatment option worth understanding**

Unlike broader treatments, targeted therapy drugs go directly for cancer cells, blocking the signals or growth factors they need to keep

Action: If this sounds relevant to your situation, it's worth asking your doctor or directly whether it's an option for you.

Source: Can I receive healthcare that is temporarily included in the basic insurance package? (rijksoverheid.nl)

Summary: **Getting access to a newer, not-yet-fully-approved treatment**

Some promising new treatments are temporarily covered by basic insurance while research (usually over about 4 years) confirms whether it

Action: If a newer treatment interests you, it's worth asking your insurer directly whether you'd be eligible, since this depends on your specific situation.

=====

Q: Can I change doctors?

=====

Source: How do I change my GP? (rijksoverheid.nl)
Summary: **How to switch GPs**

Steps: check the new GP is accepting patients, tell your current GP you're switching (in person, by phone, or in writing), and ask them to forward your records. A GP can refuse

Action: If you do decide to switch, your current GP can forward your records to make the transition smoother.

=====

Q: Can I choose my hospital or specialist?

=====

Source: Choosing a hospital when you have cancer (kanker.nl)
Summary: **How to choose a hospital for your cancer type**

Hospitals must meet quality standards to treat cancer, including a "volume standard"

rd" – a minimum number of times they perform a specific treatment ea

Action: If you'd like to look into your options further, Ziekenhuischeck.nl is a reasonable place to compare hospitals.

Source: Cancer treatment abroad (kanker.nl)

Summary: ****Considering treatment abroad****

Talk to your insurer first – they know what's realistically possible, what would be arranged, and exactly what's covered. Patient organisations for your specific cancer

Action: If treatment abroad is something you're considering, your insurer is the best first call – they can tell you what's realistically possible for your situation

Source: Your first appointment at the outpatient clinic (kanker.nl)

Summary: ****Your first hospital appointment after a GP referral****

Expect detailed questions: your symptoms, other past or current illnesses, medications, and sometimes personal context like work or relationships

Action: Thinking ahead about your symptoms and family history before this appointment can make it easier to cover everything that matters.

Source: Talking to your doctor or nurse practitioner (kanker.nl)

Summary: ****Consultations can be a lot to take in – that's normal****

Whether you see a doctor or a specially-trained nurse practitioner varies by hospital and by stage of your care – both work closely together,

Action: It's completely normal to find consultations hard to take in – see the companion 'tips for your appointment' summary for a few practical ways to make

Source: Specialist care (Zvw) (zorginstituutnederland.nl)

Summary: ****Getting to see a specialist****

You always need a referral – from your GP or another treating doctor – before you can book with a medical specialist (someone with deep expertise in one area, like a cancer

Action: If a specific treatment matters to you, it's worth checking with your insurer about any rules or restrictions before your appointment.

Source: Centralisation of cancer care through IZA (nfk.nl)

Summary: ****Background: why cancer care is becoming more centralised****

(More relevant for context than day-to-day use.) As part of a national plan (the Integrated Care Agreement, IZA), cancer care is being redesigned

Action: If a specific treatment centre feels far from home, it's worth knowing this is often a deliberate choice to concentrate real expertise, not a random decision

=====

Q: Can I request explanations in simpler language?

=====

Source: Tips for your consultation with your doctor or nurse practitioner (kanker.nl)

Summary: ****Practical tips for a better consultation****

- Write your questions down beforehand – always include: what are my options, what are the pros and cons, what does this mean for me?
- Bring someone with

Action: Asking again, bringing someone with you, or requesting things in writing are all completely reasonable things to do – there's no need to just push through

Source: Talking to your doctor about the diagnosis (kanker.nl)

Summary: **Receiving your diagnosis: what the conversation covers**

After your tests, your doctor or nurse specialist explains what was found and whether further tests are needed. Note: "positive" means sometimes

Action: It's completely fine to write your questions down beforehand, and to ask again later if something didn't sink in the first time.

Source: I'm going to see my GP: what can I do beforehand and what should I expect? (thuisarts.nl)

Summary: **Preparing for your GP appointment: the "3 good questions"*

A simple structure to bring to any GP conversation: what are my options, what are the pros and cons of each, and what does that mean for

Action: Try the '3 good questions' next time you're unsure what to ask – it's a simple way to structure the conversation.

=====

Q: Do I have time to decide?

=====

Source: Decisions regarding your cancer treatment (kanker.nl)

Summary: **Having a say in your cancer treatment**

You often have more choice than you might expect – including which hospital treats you. Your loved ones, GP, or other healthcare professionals can help you

Action: It's genuinely fine to take some time at home before committing to a treatment plan – this isn't a decision you need to rush.

Source: Talking to your doctor or nurse about work (kanker.nl)

Summary: **Talking to your doctor or nurse about work**

Your doctor or nurse can help you think through how your diagnosis and treatment will affect your ability to work, and refer you to further support if

Action: If work is on your mind, this conversation doesn't need to happen all at once – your doctor or nurse can guide you to further support at whatever pace

Source: I may have a genetic predisposition to breast cancer (thuisarts.nl)

Summary: **If breast cancer runs in your family**

About 1 in 12 women with breast cancer have a genetic cause – a mutation in genes like BRCA1, BRCA2, or CHEK2. A BRCA1/2 mutation raises breast cancer risk

Action: If breast cancer runs in your family, your GP can help assess whether a referral for genetic testing would be worthwhile – there's no urgency to decide

=====

Q: How are treatment decisions made?

=====

Source: Talking to your doctor after treatment (kanker.nl)

Summary: **After treatment ends: what follow-up looks like**

Your doctor will explain how often and for how long you'll need check-ups – with a specialist, nurse specialist, or your GP, sometimes several. You

Action: If aftercare like physiotherapy or psychological support isn't mentioned to you, it's fine to ask about it directly.

Source: Making decisions about your treatment (kanker.nl)

Summary: **5 tips for deciding on your treatment (for under-40s)**

1. Get thoroughly informed – ask your doctor about your options, their pros and cons, and what each means for your life and future.

2. Tell y

Action: If work, sport, or family life matter to how you'd want treatment to look, saying so early on genuinely shapes the conversation.

Source: I have breast cancer and want to choose a course of treatment (thuisarts.nl)

Summary: ****Choosing a treatment path for breast cancer****

Most people receive some combination of surgery, radiotherapy, chemotherapy, and hormone- or growth-protein-targeting medicines, given together or in s

Action: Your doctor will walk through what applies specifically to you – there's no need to research every option yourself beforehand.

Source: I have breast cancer that has spread to other parts of my body (metastases) (thuisarts.nl)

Summary: ****When breast cancer has spread (metastases)****

About 1 in 5 women with breast cancer see it spread elsewhere – most often to bone, lungs, liver, or brain, via the bloodstream or lymph system. At this

Action: Your doctor can only weigh what you tell them – sharing what a good quality of life looks like for you helps guide which options make sense.

Source: My breast is being removed and I'm thinking about having a breast prosthesis (thuisarts.nl)

Summary: ****Breast reconstruction: your options****

If your whole breast is being removed, you can choose reconstruction immediately, later, or not at all – wearing an external prosthesis instead is also entirely

Action: There's no wrong choice here – reconstruction, an external prosthesis, or neither are all valid, and your surgeon can talk through what's realistic for

Source: Use of medication in the Netherlands (thuisarts.nl)

Summary: ****Why your GP might not prescribe medication straight away****

Dutch GPs tend to prescribe less readily, especially antibiotics – most medicines need a prescription and face-to-face consultation rather

Action: If you leave an appointment without a prescription, that's often a deliberate call, not an oversight – it's fine to ask your GP to explain their reason

Source: What is Deciding Together? (nfk.nl)

Summary: ****What "Deciding Together" actually means****

Shared decision-making means you and your healthcare provider choose your care together – combining the medical facts (options, pros, cons, short- and long

Action: Knowing this is how the system is meant to work can make it easier to ask for your preferences to be genuinely taken into account.

Source: Vision (nfk.nl)

Summary: ****Why NFK champions shared decision-making****

The Dutch Federation of Cancer Patient Organisations (NFK) believes every patient should get a full picture of their medical options and a real say in the

Action: If you ever feel like your preferences aren't being heard, it's worth knowing this is something NFK actively works to change – you're allowed to expect

=====

Q: How do I find reliable health information or a trustworthy provider?

=====

Source: How do you find reliable information? (kanker.nl)

Summary: ****Finding health information you can actually trust****

Stick to independent, reliable sources – government sites, hospitals, universities, professional bodies, and patient organisations. Kanker.nl and

Action: Whatever you find online, running it past your doctor or nurse is always a good idea before acting on it.

Source: Cancer – what now? (kanker.nl)

Summary: ****Cancer diagnosis: what now?****

There's a lot to process at once – practically and emotionally – and often decisions to make fairly quickly, including about treatment and where you'll be treated. This

Action: There's no need to take all of this in at once – come back to it whenever a specific part feels relevant.

Source: Where can you find reliable information? (kanker.nl)

Summary: ****A checklist for judging whether a source is trustworthy****

Same core advice as finding reliable information generally – stick to government, hospital, university, and patient-organisation sites, the

Action: If you're ever unsure about a source, checking who's actually behind the website is a simple, practical first check.

Source: How do you find a reliable healthcare professional? | Kanker.nl (kanker.nl)

Summary: ****Finding a good complementary care practitioner****

Ask your doctor or nurse for a recommendation first – they may know reliable options. The "Help with Cancer" guide (linked in the source page linked

Action: If your doctor or nurse doesn't have a direct suggestion, the 'Help with Cancer' guide is a reasonable place to search by practitioner type or symptom

Source: Work and cancer (kanker.nl)

Summary: ****Navigating work when you have cancer****

If you're employed, expect regular contact with your line manager and company doctor, and possibly the UWV (employee insurance agency) – the Gatekeeper Improv

Action: Whether you're employed or self-employed, dedicated guidance exists for your specific situation – worth a look when you're ready to think about work.

Source: Who is responsible for what when it comes to cancer and work? (kanker.nl)

Summary: ****Who to contact for work-related support****

A wide network of professionals can help, depending on what you need: an occupational health doctor or specialist (including one focused specifically on on

Action: Depending on what kind of support would help, your GP or an occupational health professional can point you to the right person.

Source: What is cancer? (kanker.nl)

Summary: ****What cancer actually is****

Cancer happens when cells in the body turn malignant and start dividing much faster than healthy cells, growing into surrounding tissue and sometimes spreading elsewhere (

Action: If understanding what's actually happening in your body would feel steady, this page is a reasonable starting point.

Source: Support and advice on work and cancer (kanker.nl)

Summary: ****Work and cancer: finding the right person to talk to****

A brief signpost page connecting you to the right kind of support depending on what you're facing – whether that's continuing to work, returning

Action: Whatever you're facing at work, there's likely a specific person who can help – the companion summary on who's responsible for what covers this in more

Source: Working with or after cancer (nfk.nl)

Summary: ****You're not alone in balancing cancer and work****

Around 50,000 working people are diagnosed with cancer each year in the Netherlands, and that number is rising as more people continue working through

Action: You're far from alone in this – if work is on your mind, the companion summary on talking to your doctor or nurse about it is a good next stop.

=====

Q: How do I get referred to psychological care, physiotherapy, nutrition advice or fertility support?

=====

Source: Talking to your doctor when cancer is incurable (kanker.nl)

Summary: ****Talking with your doctor when a cure isn't possible****

Your doctor (and, if you'd like, your loved ones too) will work with you to figure out what matters most to you – what you do and don't want –

Action: Sharing what matters most to you with your doctor, and your loved ones if you'd like them involved, helps your care fit your actual priorities.

Source: Support for people with cancer (kanker.nl)

Summary: ****Support goes well beyond medical treatment****

Cancer affects far more than your body – help is available for physical symptoms (fatigue, pain, lymphoedema, eating difficulties), mental health (anxiety)

Action: A good starting point is simply naming what's hardest right now to you or GP, practice nurse, or oncology nurse – they'll help you work out what kind of

Source: Talking to your doctor about your treatment plan (kanker.nl)

Summary: ****Your treatment plan, step by step (for under-40s)****

Once your tests are complete and your cancer's type and stage are known: multiple specialists first meet to discuss your case (a "multidisciplinary

Action: If having children is something you care about, it's worth raising with your doctor before treatment begins, since this affects the timing of that con

Source: I'm experiencing mental health issues. Who can I turn to for help? (thuisarts.nl)

Summary: ****Mental health support: where to start****

Your GP is the first stop for anything from low mood and irritability to poor sleep, trouble concentrating, or leaning more on alcohol to cope – physical symptoms

Action: Telling your GP how you're feeling, even briefly, is a reasonable first step – they can help you figure out what kind of support might help.

Source: I have cancer and I'm undergoing treatment so that I can get pregnant in the future (thuisarts.nl)

Summary: ****Preserving fertility before treatment (planning ahead)****

If your cancer treatment might affect your future fertility, options – done before

e treatment starts – include freezing eggs, freezing fertil

Action: If this feels relevant to you, it's worth raising with your care team before treatment starts, since these options work best planned ahead of time.

Source: I have cancer and I still want to get pregnant (thuisarts.nl)

Summary: **Fertility and cancer treatment – what affects your chances**

Surgery, chemotherapy, and radiotherapy can all affect your womb or ovaries, and how much depends on your age, your fertility before tre

Action: This is a genuinely hard thing to think about – talking to others who've faced the same question, or your care team, can help you find your way through

Source: Mental health care (Zvw) (zorginstituutnederland.nl)

Summary: **Mental health care – what's covered**

Basic insurance covers professional mental health care under certain conditions – for genuine mental health conditions, not everyday stress, relationship diffi

Action: If low mood or anxiety is what you're noticing, your GP or practice nurse is a reasonable first person to mention it to, even briefly.

Source: Dietary supplements (Zvw) (zorginstituutnederland.nl)

Summary: **Special dietary products – what's covered**

If a normal diet isn't suitable for your medical condition, specially formulated nutrition (like tube feed or liquid nutritional supplements) may be reim

Action: If a normal diet isn't working for you right now, your dietitian or specialist can confirm whether a specially formulated product would be covered.

Source: Nutritional aids (Zvw) (zorginstituutnederland.nl)

Summary: **Support if you can't eat normally**

If digestive issues from your illness or treatment make eating difficult, covered aids include a feeding pump (tube feeding), a feeding device, or – for nutritio

Action: There's support available for this – you don't need to just manage it on your own.

=====

Q: How do I make a complaint about my care or a healthcare provider?

=====

Source: Complaints about cancer care (kanker.nl)

Summary: **Making a complaint about your care**

You have the right to complain if you're unhappy with your hospital or treatment – about anything from a suspected mistake to feeling unkindly treated, unclear

Action: It might help to think through what outcome you're actually hoping for first – that tends to make the next step clearer.

Source: Complaints about the GP (thuisarts.nl)

Summary: **If you're unhappy with your GP**

Common reasons people complain: not getting the care they expected, a refused prescription or referral, a missed diagnosis, feeling dismissed or unheard, or being d

Action: Most GPs genuinely want to hear if something's gone wrong – it's worth raising, even if you're not sure it'll change anything.

Source: Complaints about the GP (thuisarts.nl)

Summary: **The complaints process for your GP, step by step**

Start by raising it with your GP directly – this often resolves things and helps future patients too. Not satisfied? A complaints officer can mediate

Action: Start by speaking with your GP directly if you can – if that doesn't resolve things, a complaints officer can help mediate from there.

Source: Health Insurance Act – general: how does the Health Insurance Act work? (zorginstituutnederland.nl)

Summary: ****How the Dutch insurance system actually works, and how to complain****

The system is built on solidarity – everyone pays in and everyone gets the same basic cover, regardless of age, health, or income

Action: If you ever disagree with a reimbursement decision, you now know the order to go in – your insurer first, then SKGZ if it's still unresolved.

Source: Complaints about reimbursements (Zvw) (zorginstituutnederland.nl)

Summary: ****Disputing a reimbursement decision****

If you disagree with your insurer's decision (e.g. they won't cover a treatment), ask them to review it first. Still unhappy, or no timely response? You can bring it to SKGZ

Action: If you disagree with a reimbursement decision, ask your insurer to review it first – SKGZ is there afterward if you're still unhappy with the outcome.

Source: Complaints about healthcare providers (Wlz/Zvw) (zorginstituutnederland.nl)

Summary: ****Your right to good care, and what to do if it falls short****

By law (the Wkkgz), the care you receive must be safe, tailored to your needs and timing, effective, and delivered according to professional standards

Action: If your care hasn't met these standards, you're entitled to raise it – see the companion summary on the GP complaints process for the practical steps.

=====

Q: How do I request a second opinion?

=====

Source: Second opinion on cancer (kanker.nl)

Summary: ****Your right to a second opinion – including abroad, and who pays****

You can request a second opinion – another doctor's view, at a different hospital – on any diagnosis or treatment, before, during, or after

Action: If getting a second opinion – including abroad – feels right for you, it's genuinely your right at any stage; your specialist can arrange a referral, or you can go directly to a second opinion

=====

Q: How do referrals work in the Netherlands?

=====

Source: When do I need a referral letter from my GP? (rijksoverheid.nl)

Summary: ****When you need a referral letter****

For specialist/hospital care, your GP must issue a referral, and can only refuse if they genuinely believe it isn't in your best interest – and must explain why if refused

Action: If you'd like to read your referral letter yourself, you're entitled to ask for it.

Source: Dutch healthcare: what to expect (GPs, hospitals, costs) (thuisarts.nl)

Summary: ****A starting point: how Dutch healthcare works overall****

This is an index page pulling together the practical basics: what's distinctive about Dutch healthcare, how the GP practice works, how to make

Action: If you want more detail on a specific topic, each one covered here has its own dedicated summary elsewhere – worth bookmarking this as a starting point

Source: Appropriate use (zorginstituutnederland.nl)

Summary: **Background: matching treatment to what patients actually need**

(More relevant for context than day-to-day use.) Describes an ongoing effort between professional bodies, patient organisations, and

Action: A reassuring principle to keep in mind: the system is meant to work against unnecessary treatment, not just cost-cutting for its own sake.

Source: Reimbursement for non-hospital medicines (GVS) (zorginstituutnederland.nl)

Summary: **Background: how prescription medicines you collect yourself get reimbursed**

(More relevant for context than day-to-day use.) "Extramural" medicines – the ones you collect from a pharmacy or GP

Action: If you're curious about the mechanics, this is mostly relevant background – day to day, it happens without you needing to think about it.

=====

Q: How long should I expect to wait for appointments or treatment?

=====

Source: Waiting times at hospitals for cancer treatment (kanker.nl)

Summary: **Checking hospital waiting times**

Every hospital has waiting times for appointments and treatment – Zorgkaart Nederland lets you compare wait times across hospitals and clinics by region on one page

Action: If you're ever waiting longer than feels reasonable, your insurer can help – no need to just wait it out if it's genuinely affecting you.

Source: Talking to your doctor about treatment (kanker.nl)

Summary: **The treatment conversation: what to expect**

Your doctor will lay out the treatment options available to you, including their pros and cons – sometimes "wait and see" or no treatment at all is a good

Action: Take the time you need with this – a 'wait and see' approach or no treatment at all can be just as valid a choice as any other, depending on what's right

Source: Will I have to deal with waiting times if I need care? (rijksoverheid.nl)

Summary: **Is waiting for care normal?**

Yes – for non-urgent care, waiting is expected and not usually a cause for concern, as long as it isn't affecting your health or keeping you off work unreasonably long

Action: If a wait is genuinely affecting you, it's worth reaching out to your insurer – they may be able to find you a provider who can see you sooner.

=====

Q: What additional services can I access?

=====

Source: Never getting better again (kanker.nl)

Summary: **Coming to terms with news you won't recover**

This news can land at diagnosis, shortly after, or even years into treatment – and taking it in tends to happen gradually, not all at once. Once a cure

Action: Whenever you're ready to talk it through, whether that's now or later, support is there for you and for the people close to you.

Source: What is the palliative phase? (kanker.nl)

Summary: ****Understanding the palliative phase****

The palliative phase begins once it's clear you won't recover – it can last days, months, or years. Care during this time addresses everything: physical, emotio

Action: You and your doctor decide together what care feels right for this stage – there's no need to have it all figured out at once.

Source: Complementary care for cancer? Read reliable information (kanker.nl)

Summary: ****Complementary care: massage, acupuncture, and similar support****

Complementary care sits alongside your standard hospital/GP treatment, not instead of it – things like massage, acupuncture, or music

Action: If you're considering something like this, mentioning it to your doctor or nurse is worth doing before you start – see the companion summary on raisin

Source: Tips for your consultation with your doctor (kanker.nl)

Summary: ****How to raise complementary care with your doctor****

Before your appointment: write down what you want to try and why (e.g. "I want acupuncture because I think it will help with my pain"), bring any

Action: Jotting a few notes down before your next appointment – what you'd like to try and why – can make this conversation easier to have.

Source: How can you find reliable information online about complementary (additional) healthcare? (kanker.nl)

Summary: ****Vetting complementary care information online****

Same core rule as any medical information: stick to independent, reliable sources (government, hospitals, universities, professional and patient orga

Action: As with any treatment idea, it's worth mentioning what you've found to your doctor or nurse before trying it.

Source: Are physiotherapy and exercise therapy included in the basic insurance package? (rijksoverheid.nl)

Summary: ****Is physiotherapy covered?****

Partially, and it depends on your age and why you need it:

- ****Under 18:**** first 9 sessions covered, with up to 9 more if needed; some chronic conditions qualify for full

Action: Not sure if you qualify? Your physiotherapist can usually tell you quickly whether your specific condition is on the covered list.

Source: I've had surgery for breast cancer. What happens next? (thuisarts.nl)

Summary: ****After breast surgery: what typically comes next****

Further treatment (radiotherapy, chemotherapy, anti-hormone or growth-protein-targeting drugs) usually follows, aimed at clearing any remaining can

Action: If that would help, many hospitals have physiotherapist and dietitian support available – staying active and eating well can genuinely help you get th

Source: Community nursing (aged 18 and over) (zorginstituutnederland.nl)

Summary: ****Nursing and care at home, for adults****

Community nursing helps you keep living at home rather than moving into hospital or a care home – covering things like personal care and managing symptoms in

Action: You're free to contact a care provider directly whenever this becomes relevant, without needing a referral first.

Source: Paramedical care (Zvw) (zorginstituutnederland.nl)

Summary: **Paramedical care: physio, speech therapy, occupational therapy, diet etics**

If physical or mental symptoms are limiting your mobility, daily activities, or ability to participate in life, paramedic

Action: If any of this sounds like what you're dealing with, your GP or specialist is the best person to ask whether a referral makes sense.

Source: Physiotherapy and exercise therapy (Zvw) (zorginstituutnederland.nl)

Summary: **Physiotherapy and exercise therapy specifically**

These treat physical symptoms from illness, injury, surgery, or other causes – helping you understand what movement is safe, what you can still do,

Action: If movement or discomfort is holding you back day to day, a physiotherapist or occupational therapist may be able to help – worth asking your GP about

Source: Wigs (hairpieces) (Health Insurance Act) (zorginstituutnederland.nl)

Summary: **Getting a wig covered**

If you've lost hair due to illness or treatment (including chemotherapy), insurance covers up to €497.50 toward a wig – real or synthetic hair, your choice – with any remain

Action: If hair loss is something you're facing, your GP, oncology nurse, or specialist can help arrange this – it's worth checking with your insurer about au

Source: Dental prostheses (Health Insurance Act) (zorginstituutnederland.nl)

Summary: **Prosthetics – replacing a body part**

If you've lost a body part or part of one, a prosthesis (temporary, permanent, or cosmetic) can be provided based on your individual needs and daily life – for

Action: Your care team will talk through what's realistic and right for you, based on your own situation and daily life.

Source: Occupational Therapy (Zvw) (zorginstituutnederland.nl)

Summary: **Occupational therapy – help with daily tasks**

If illness has made everyday activities like walking, eating, or dressing difficult, an occupational therapist can help you find ways to manage them a

Action: If everyday tasks have become harder, an occupational therapist may be able to help – worth asking your GP or specialist about a referral.

Source: What does 'end of life' mean? (nfk.nl)

Summary: **Understanding "end of life" and palliative care**

Being told your illness is incurable can happen at diagnosis, shortly after, or following prolonged treatment – the time that follows can be weeks,

Action: Palliative care isn't something you have to ask for perfectly or all at once – it can be brought in gradually, alongside whatever else is happening.

=====

Q: What are my general rights and responsibilities as a patient?

=====

Source: What are your rights as a patient? (kanker.nl)

Summary: ****Your rights as a patient, in brief****

Under Dutch law (the WGBO), you have the right to: clear information about your condition, treatment, and its consequences (or the right to **not** be told, if you

Action: It's worth knowing these rights exist, even if you never need to actively use them – including your right to ask for a second opinion at any point.

Source: Frequently asked questions about the 'clean slate' scheme (kanker.nl)

Summary: ****Clean slate scheme: answers to common questions****

A few of the most useful clarifications:

- ****Had a precancerous condition?**** You must still disclose this – it's treated differently from a full ca

Action: If your situation touches on any of these specifics, it's good to keep in mind – no need to act on this now unless you're actively applying.

Source: How can I check a healthcare invoice? (rijksoverheid.nl)

Summary: ****Checking your healthcare invoice for errors****

Once your insurer has paid your healthcare provider, you can view the invoice in the "My Account" section of your insurer's website, to check the treatment

Action: You can check your own invoices any time via your insurer's online account – worth a quick look if a bill ever feels unclear.

Source: Who should I contact if I have any doubts about the accuracy of a healthcare invoice? (rijksoverheid.nl)

Summary: ****Who to contact about a disputed invoice****

It depends on where the invoice came from: if you received it via your insurer, raise your query with them and they'll follow up with the healthcare provider

Action: If something on a bill looks off, you now know who to reach out to first – the insurer or the provider, depending on where it came from.

Source: What do I need to bear in mind if I want to switch to a different insurer? (rijksoverheid.nl)

Summary: ****A step-by-step guide to switching insurer****

If you don't agree with your new policy terms (sent every November), you're free to switch. Cancel your old policy by 31 December, take out a new one by

Action: If you're currently mid-treatment and considering switching insurer, it's worth understanding how your existing authorisations carry over before you move

Source: Is an insurer obliged to accept me for health insurance? (rijksoverheid.nl)

Summary: ****Can an insurer turn you down?****

For basic cover, no – insurers are legally required to accept everyone, regardless of age or health, at the same premium as anyone else. Supplementary insurance is different

Action: Reassuring to know either way – your basic cover is secure regardless of your health history.

Source: Patients' rights and responsibilities (thuisarts.nl)

Summary: ****Your rights with your GP specifically****

You're entitled to clear information about your condition, proposed tests or treatment (or a "wait and see" approach), other treatment options, risks and costs

Action: If a question falls outside what your GP can answer, like specific insurance detail, they can point you toward whoever can help.

Source: I'm having additional breast screening because I'm at higher risk of breast cancer (thuisarts.nl)

Summary: ****Additional breast screening if you're at higher risk****

You may be offered extra screening if: you have a known genetic predisposition and a lifetime risk of 20%+, breast cancer runs in your family

Action: If you notice any change in your breast between screenings, it's worth booking a GP appointment rather than waiting for your next scheduled scan.

Source: Tests show that I don't have breast cancer, but I'm still experiencing symptoms (thuisarts.nl)

Summary: ****Normal results, but still worried or symptomatic****

A normal mammogram or ultrasound makes breast cancer very unlikely – but the screening process itself is genuinely stressful, and lingering anxiety

Action: If that worry doesn't ease, talking to your practice nurse or a psychologist can genuinely help – it's a completely normal thing to need after a screening

Source: Patients' rights and obligations (thuisarts.nl)

Summary: ****Your rights and responsibilities as a patient – plus consent****

You have rights to information, access to your records, privacy, and choosing your own GP – and a doctor can only examine or treat you

Action: Giving your GP the full picture, even the parts that feel awkward to share, genuinely helps them care for you well.

Source: What is specialist care? (nfk.nl)

Summary: ****Why specialist experience matters for cancer care****

Cancer diagnosis and treatment are advancing so fast that no single doctor can stay fully expert across every type – and hospitals that only treat

Action: If your cancer type is less common, it's worth specifically looking in to a hospital with real experience treating it, not just the nearest option.

Source: Taking control of your own care as a cancer patient (kanker.nl)

Summary: ****A starting point: staying in control of your own care****

This page pulls together the key questions around navigating cancer care with age: what choices you have around treatment, how to pick the

Action: If you want to go deeper on a specific topic – choosing a hospital, second opinions, complaints – each one mentioned here has its own fuller summary

=====

Q: What does my health insurance actually cover, and what will it cost me?

=====

Source: Expensive cancer drugs: are you getting what you're entitled to? (kanker.nl)

Summary: ****If you think you're being denied an expensive cancer medicine****

If a medicine for your type and stage of cancer is in the basic insurance package, your doctor must be able to prescribe it and your

Action: If you're unsure whether you're getting a medicine you're entitled to, it's worth raising this directly with your specialist first – they'll usually know

Source: Basic health insurance if you have (or have had) cancer (kanker.nl)

Summary: ****The basics: what your compulsory insurance covers****

- Basic health insurance covers care from your GP, hospital, and pharmacy – insurers must accept you even if you're unwell.

- You choose between

Action: No action needed right now – this is useful background for when you're comparing policies or wondering what's already covered.

Source: Health insurance and cancer (kanker.nl)

Summary: ****Should you switch your insurance after a cancer diagnosis?****

Basic insurance already covers most cancer care, but if you need things it does n't cover – extra physiotherapy, dental care, or psychoso

Action: If you're weighing whether to switch policies, note that changes can only be made once a year, around the turn of the year – worth bookmarking this pa

Source: Supplementary insurance if you have (or have had) cancer (kanker.nl)

Summary: ****Is supplementary insurance worth it for you?****

Basic insurance doesn't usually cover:

- Extra physiotherapy or oedema therapy
- Occupational therapy
- Psychosocial support (e.g. a psychologist)
- D

Action: If any of the listed gaps (physiotherapy, dental care, psychosocial support) sound relevant to you, zorgverzekeringskaart.nl is a straightforward plac

Source: Clean Slate Scheme for former cancer patients (kanker.nl)

Summary: ****Applying for life or funeral insurance after cancer – the 'clean slate' scheme****

If you've been cancer-free for 10 years (or 5 years if you were diagnosed under 21), you may no longer need to decla

Action: If this scheme might apply to you, the exact waiting periods vary by cancer type – worth checking which applies to your situation before you apply.

Source: What are some of the common misconceptions about health insurance? (rijksoverheid.nl)

Summary: ****Common health insurance myths, cleared up****

- Your doctor recommending a treatment doesn't guarantee it's covered – the government decides what basic insurance includes.
- You can switch insurer on

Action: Nothing to act on here – just useful to have in the back of your mind next time an insurance decision or bill doesn't quite make sense.

Source: Which medicines are covered by my health insurance? (rijksoverheid.nl)

Summary: ****Getting your medicines reimbursed****

Basic insurance covers most medicines your doctor prescribes, though you may pay a co-payment (capped at €250/year) if a medicine costs more than the standard re

Action: If a specific medicine isn't covered, your doctor or pharmacist can look into an alternative – you don't need to sort this out alone.

Source: Tips to ensure you're properly covered for healthcare costs (rijksoverheid.nl)

Summary: ****Practical tips for keeping healthcare costs down****

- Call your GP first for non-emergencies – GP care doesn't use your deductible, but tests they refer you for (like blood work) do.
- No deductible

Action: A few of these tips (like checking whether your provider is contracted) are worth keeping in mind before you commit to a new provider or treatment loc

Source: What can I do if I have high healthcare costs? (rijksoverheid.nl)

Summary: ****Help if healthcare costs are adding up****

Two main routes for support:

- ****Healthcare allowance**** – a contribution toward your premium and deductible, based on your (and your partner's) income. You

Action: If costs are weighing on you, it's worth checking whether you're eligible for healthcare allowance – applying is something you do yourself each year

Source: What policies does the insurer offer? (rijksoverheid.nl)

Summary: ****The different types of insurance policy, explained****

- ****In-kind policy**** – your insurer selects contracted providers; choosing someone outside that network usually means a lower reimbursement.

- *

Action: If having a free choice of hospital matters to you, it's especially worth understanding these policy types before you next choose or renew one.

Source: What is supplementary health insurance? (rijksoverheid.nl)

Summary: ****What supplementary insurance actually covers****

Supplementary insurance (optional) covers things basic insurance doesn't – commonly extra physiotherapy sessions, dental/orthodontic care, alternative

Action: Worth a look if any of these gaps apply to your situation – no pressure to decide anything today.

Source: Will I be reimbursed for treatment at an independent treatment centre? (rijksoverheid.nl)

Summary: ****Getting treatment at an independent treatment centre****

Independent treatment centres offer specialist care (for example cataract surgery or a knee replacement) outside a hospital, with similar staff

Action: If independent treatment centre care comes up for you, it's worth checking with your insurer beforehand rather than after – they can confirm what's covered

Source: When do I have to pay a deductible for my healthcare? (rijksoverheid.nl)

Summary: ****When you pay the deductible – and when you don't****

If you're 18 or older, you pay the first €385 of many healthcare costs yourself each year before your insurer starts covering the rest. Some care

Action: Worth keeping this list in mind, since a few common needs during cancer care (like community nursing) are exempt from your deductible.

Source: What changes will be made to the basic insurance package in 2026? (rijksoverheid.nl)

Summary: ****What's changing in the basic insurance package for 2026****

The deductible (€385) and medicine co-payment cap (€250) stay the same. Three changes worth knowing:

- Smoking cessation support is now reimbursed

Action: Nothing you need to do here – just a quick note of what's changing from 2026, in case anything overlaps with your own care.

Source: What healthcare services are included in the basic insurance package? (rijksoverheid.nl)

Summary: ****What's covered under the basic package – a quick-reference list****

A summary table of common care types, whether they're subject to the deductible, and whether a co-payment applies – for example, GP

Action: Worth keeping this table handy as a quick reference – no need to read it all at once.

Source: What happens if I don't pay my health insurance premium? (rijksoverheid.nl)

Summary: ****If you fall behind on your premium****

Your insurer will send reminders and may offer a payment plan. After 6 months of non-payment, you're registered for a government arrears scheme with a much high

Action: If premium payments are becoming difficult, the Health Insurance Complaints and Disputes Foundation (SKGZ) can offer independent advice – reaching out

Source: Former cancer patients are better off taking out insurance (rijksoverheid.nl)

Summary: ****Background: why the 'clean slate' scheme exists****

(News item, July 2020) This is the announcement behind the clean slate scheme described elsewhere: the Dutch government, insurers' association, a

Action: Reassuring, if nothing else, to know this scheme exists because patients pushed for it – you're not the first to find this process complicated.

Source: Am I covered by health insurance when I'm on holiday abroad? (rijksoverheid.nl)

Summary: ****Covered while on holiday abroad****

Yes, for sudden and urgent care within Europe – take your European Health Insurance Card (EHIC) with you, which reduces how much you pay upfront. You're reimburse

Action: If you're planning travel, it's worth packing your European Health Insurance Card before you go, and considering whether extra travel cover makes sense

Source: Am I covered by health insurance if I live abroad? (rijksoverheid.nl)

Summary: ****Covered if you live abroad****

This depends entirely on your personal situation – whether you're living, working, or studying abroad changes what cover you can take out. The source page (linked below)

Action: Since this depends entirely on your personal circumstances, Nederlandwereldwijd.nl is the better next stop for guidance specific to your situation.

Source: How much does a visit to the GP cost? (thuisarts.nl)

Summary: ****What a GP visit costs you****

Usually nothing directly – your insurer covers GP visits, and you normally don't pay a co-payment, as long as your GP has a contract with your insurer. Co-payments can also

Action: Good to know for peace of mind before your next GP visit – you generally won't be paying anything out of pocket.

Source: Health checks in the Netherlands (thuisarts.nl)

Summary: ****Optional health checks – what's available****

If you have no symptoms but want peace of mind, health checks (blood sugar, cholesterol, STI, PSA, HIV, cancer screening, general lifestyle) can be bought

Action: If you're considering a private health check, it's worth weighing the pros and cons first – and if you already have symptoms, going straight to your GP

Source: How much does it cost to see a GP? (thuisarts.nl)

Summary: ****GP visit costs – for internationals with Dutch insurance****

If you have Dutch health insurance, your GP visits are covered directly by your insurer

insurer – you don't usually pay anything upfront. Exceptio

Action: If you're on an international or student policy rather than Dutch insurance, it's worth checking with your provider beforehand whether you'll need to

Source: Health Insurance Act Compass (zorginstituutnederland.nl)

Summary: ****An index of what's covered, service by service****

This is a reference index (the "Zvw Compass") rather than a single explanation – it lists every category of care covered under basic insurance, from

Action: Worth bookmarking as a reference – a handy starting point next time you want to check whether something specific is covered.

Source: Basic Health Insurance Package (zorginstituutnederland.nl)

Summary: ****What's in the basic package – in short****

This page is a brief pointer rather than a full explanation: the basic insurance package's contents are set annually by the government, and the detailed bre

Action: For the full breakdown of what's included, the Zvw Compass reference index (see the companion summary) covers each type of care individually.

Source: Deductible (Health Insurance Act) (zorginstituutnederland.nl)

Summary: ****How the deductible actually works, with examples****

You pay the first €385 of many healthcare costs yourself each year before your insurer takes over – this applies once you turn 18 (pro-rated in yo

Action: Since cancer care often means reaching your deductible earlier in the year than you might expect, it's worth understanding this ahead of time.

Source: Co-payment (Health Insurance Act) (zorginstituutnederland.nl)

Summary: ****Co-payments: the costs on top of your deductible****

For some specific types of care, you pay a portion of the cost yourself (the co-payment) **in addition to** your deductible – the co-payment comes o

Action: If you're ever unsure whether a co-payment applies to something specific, your insurer can confirm it quickly.

Source: Healthcare and Overseas (Zvw) (zorginstituutnederland.nl)

Summary: ****Healthcare needs while abroad – a quick pointer****

Whether you fall ill unexpectedly on holiday or travel abroad for planned treatment, reimbursement is possible under certain conditions – this shor

Action: For the fuller picture on reimbursement while abroad, the companion 'how reimbursement works abroad' summary covers the conditions in more detail.

Source: Abroad and healthcare (Zvw) (zorginstituutnederland.nl)

Summary: ****How reimbursement works for care received abroad****

Care abroad is only covered if the same treatment would be covered in the Netherlands, and it must meet the usual Dutch reimbursement conditions (

Action: If treatment abroad is something you're considering, it's worth checking with your insurer beforehand what would actually be reimbursed in your specif

Source: Medicines (Zvw) (zorginstituutnederland.nl)

Summary: ****How medicine reimbursement actually works****

Only medicines on the official Medicines Reimbursement System (GVS) list are covered – grouped by similar effect into two categories, with some requiring

Action: If you're ever wondering why a medicine is or isn't reimbursed, this is a good background – see the companion summary for how to check a specific one.

Source: Package management (zorginstituutnederland.nl)

Summary: ****Background:** how the basic package is kept accessible and affordable*

(More relevant for context than day-to-day use.) This describes the Dutch Healthcare Institute's ongoing work to keep the basic

Action: No need to do anything with this – it's simply the reasoning behind why coverage decisions get reviewed and updated over time.

Source: Who has which role in package management? (zorginstituutnederland.nl)

Summary: ****Background:** who decides what's covered**

(More relevant for context than day-to-day use.) Coverage decisions aren't made by one body alone – healthcare providers, insurers, patient organisations,

Action: If you're ever curious why a decision was made a certain way, it's worth knowing that several different voices, not just one body, shape what's covered

Source: A practical guide to the parcel criteria (zorginstituutnederland.nl)

Summary: ****Background:** the four tests a treatment must pass to be covered**

(More relevant for context than day-to-day use.) Explains the four criteria used to decide whether a treatment belongs in the basic

Action: If a coverage decision ever feels arbitrary, it's useful to know there's a defined set of questions behind every one.

Source: Assessment of new healthcare services (zorginstituutnederland.nl)

Summary: ****Background:** how new medicines get added to the basic package**

(More relevant for context than day-to-day use.) Explains the assessment process new medicines go through before being reimbursed –

Action: Explains why a genuinely promising new medicine sometimes takes a little longer to become available than you'd expect.

Source: Aids for incontinence and stomas (Zvw) (zorginstituutnederland.nl)

Summary: ****Aids for incontinence or a stoma****

If treatment or illness has left you unable to control your bladder or bowels as usual, medical aids to manage this day and night are covered under basic insurance

Action: If this is affecting you, it's a completely reasonable thing to raise with your GP, oncology nurse, or specialist.

Source: Integrated care (Zvw) (zorginstituutnederland.nl)

Summary: ****Integrated care for ongoing conditions****

If you have a chronic condition needing input from several healthcare professionals – for example your GP working alongside a dietitian for diabetes, or a p

Action: If you're managing an ongoing condition alongside cancer care, your GP can help set up this kind of coordinated support.

Source: Gateway for expensive medicines (zorginstituutnederland.nl)

Summary: ****Background:** why some new hospital medicines aren't covered right away**

(More relevant for context than day-to-day use.) Very expensive new hospital medicines can be temporarily excluded from the

Action: If a specific new treatment you're hoping for isn't yet available, this is good context – it may simply be part of this review process.

Source: A clean slate: the right to be forgotten (nfk.nl)

Summary: ****Why the 'clean slate' scheme exists – from the patient side****

Before this scheme, 6 in 10 people who'd had cancer struggled to get life insurance – most were refused outright due to their diagnosis

Action: If you're navigating this scheme yourself, it's reassuring to know you're not alone in finding it complicated – NFK's advocacy is part of why it exists

=====

Q: What if I don't understand medical terminology?

=====

Source: Online communication training course: 'Talking to your doctor' (kanker.nl)

Summary: **A video course on talking to your doctor**

"Luistertijd" is a free online training with videos of real patient stories (acted out), showing how to prepare for and navigate a conversation with your

Action: If you're not sure what to expect from a consultation, this free course is a gentle way to see how a real conversation might unfold.

=====

Q: What should I expect when I visit my GP, and how does registering work?

=====

Source: Tests carried out if cancer is suspected (kanker.nl)

Summary: **What happens during cancer testing**

After your GP refers you, expect an initial consultation (symptoms, duration, other health history, sometimes occupation or family history), then a physical exam

Action: If it would help you feel more supported, it's completely fine to bring someone with you to this consultation.

Source: How do I arrange for nursing and care at home (community nursing)? (rijksoverheid.nl)

Summary: **Getting nursing or care at home**

Home nursing and personal care (help washing, dressing, wound care, managing medication) is covered under basic insurance with no deductible. You don't need a GP referral

Action: If you're not sure where to start, your GP, hospital discharge nurse, or local authority can all help you find a home care provider.

Source: Who will you be dealing with at the GP practice? (thuisarts.nl)

Summary: **Who you'll meet at the GP surgery**

- **The GP** – your main point of contact, coordinates your overall care, and decides when to refer you elsewhere.

- **Doctor's assistant** – handles calls, simple

Action: Good to know before your next visit – whoever you see, from GP assistant to practice nurse, is bound by the same confidentiality as the GP.

Source: Choosing a GP (thuisarts.nl)

Summary: **Choosing your GP**

You're free to choose your own GP, though in practice a specific GP might not accept you – a full patient list, living too far away, or a fundamental difference in views (e.g. on

Action: If you're struggling to find a GP willing to take you on, your insurer can help mediate.

Source: Contacting the GP practice (thuisarts.nl)

Summary: ****How to actually get in touch with your GP practice****

Every practice has a website or brochure listing names, phone numbers, and when to call for advice, an appointment, or an e-consultation. You can

Action: Your practice's own website or brochure will have the exact numbers and hours you need – worth saving those details somewhere handy.

Source: What sort of questions can you ask your GP? (thuisarts.nl)

Summary: ****What you can actually see your GP for****

More than you might expect: physical or mental health questions and symptoms, first aid and minor procedures, advice on preventing or easing symptoms, diagnosis

Action: If you're ever unsure whether something is 'worth' a GP visit, it usually is – this list covers more than most people expect.

Source: I would like to view my medical records from my GP (thuisarts.nl)

Summary: ****Viewing your GP medical record****

Your GP is legally required to keep a record of your health information – conditions, tests, treatments, medications – so they can give you the best possible care.

Action: You can view your GP record online whenever it's useful to you – just keep your login private.

Source: I want to decide together with my doctor (thuisarts.nl)

Summary: ****Making the conversation with your doctor work for you****

Your doctor can only factor in what you tell them – so it helps to share what matters to you: what you're worried about, how you're coping at

Action: Sharing what you're worried about and what you're hoping for, even briefly, helps your doctor factor in what actually matters to you.

Source: What happens during a visit to the GP? (thuisarts.nl)

Summary: ****Getting the most out of your GP visit****

Before you go: write down what's troubling you and what you want to ask, note every medicine you're taking (including over-the-counter ones), and check your

Action: Bringing a written list of questions and current medications can make the appointment feel less rushed and easier to follow.

Source: What people from abroad often notice about the Dutch healthcare system (thuisarts.nl)

Summary: ****What often surprises people new to Dutch healthcare****

A few things worth knowing upfront: your GP is genuinely your first stop for almost everything, even things you might expect to see at a hospital

Action: It's completely fine to say if something about the conversation style feels unfamiliar, or to bring someone with you for support.

Source: The role of the GP in the Netherlands (thuisarts.nl)

Summary: ****Why the GP is so central here – for newcomers****

The biggest adjustment for many people moving to the Netherlands: almost everyone has one dedicated GP who knows their (and their family's) history,

Action: Being upfront with your GP about anything personal or cultural that matters to you helps them support you better from the start.

Source: Choosing a GP (thuisarts.nl)

Summary: ****Registering with a GP as a newcomer****

Register as soon as you've settled, even before you need one – bring your ID, BSN (citizen service number), and insurance details, plus any existing medical re

Action: It's worth registering with a GP as soon as you've settled, even before you think you'll need one.

Source: Booking an initial appointment with a GP (thuisarts.nl)

Summary: ****Meeting a GP before you register****

Most GPs will offer an introductory appointment first, where you can discuss your needs confidentially before committing to register with them. Worth preparing for

Action: An introductory meeting is a low-pressure way to see if a GP feels like the right fit before committing.

Source: Who will I see at the GP surgery? (thuisarts.nl)

Summary: ****The team at your GP surgery, and what each person does****

- ****GP**** – your main contact for health concerns, who examines, treats, and refers when needed.

- ****GP assistant**** – your first point of contact

Action: Good to know who's who at your practice – it can make it easier to know who to ask for what.

Source: How are the training and standards of GPs organised in the Netherlands? (thuisarts.nl)

Summary: ****How rigorous is GP training in the Netherlands?****

GP training in the Netherlands is thorough: 6 years of general medical training followed by 3 years of specialist GP training, covering everything

Action: Reassuring to know your GP has gone through years of dedicated training and ongoing assessment to be your main point of contact.

Source: What sort of issues can I see my GP about? (thuisarts.nl)

Summary: ****When to see your GP vs. when to call 112****

Your GP handles the vast majority of things – urgent or not, physical or mental, for you or your child – call 112 only for a genuine life-threatening emergency

Action: If you're ever unsure whether something is urgent, the practice receptionist can help you judge – no need to work it out alone.

Source: How can I ask my GP questions or make an appointment? (thuisarts.nl)

Summary: ****Booking an appointment or asking a quick question****

Check your practice's website or brochure for contact details and whether they offer e-consultations (secure online messaging with the GP). You can

Action: Check your practice's website for how they prefer to be contacted – many now offer secure online messaging alongside phone booking.

Source: What happens during a visit to the GP? (thuisarts.nl)

Summary: ****Getting ready for a GP visit: the practical checklist****

Bring a list of any medications you're taking, prescribed or bought yourself. Thuisarts.nl (in Dutch) is a reasonable place to research your

Action: Bringing a written medication list and a few notes on your symptoms can make the appointment easier for both of you.

Source: GP (Zvw) (zorginstituutnederland.nl)

Summary: ****Your GP visit and what's covered****

GP care is fully covered by basic insurance and is usually your first point of contact for any health concern, physical or mental – for you and your family. You may

Action: Reassuring to know your GP is always your first stop, for anything – physical or mental, for you or your family.

Q: Will a second opinion be covered by insurance?

Source: How do I get a second opinion in the healthcare system? (rijksoverheid.nl)

Summary: ****How to actually get a second opinion****

Start by discussing it with your current doctor – this alone sometimes resolves your uncertainty. If you'd still like one, ask your doctor for a referral; the

Action: Starting with a conversation with your own doctor is a reasonable first step – and if you'd still like a second opinion afterward, they can arrange the

Observation — Question-to-answer mapping:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

This is a brief insight into the key deliverable for CSN's Patient Navigator vision — the full question → source → summary → action chain in one place, grouped by community question so each patient question shows every page, summary, and next step that answers it.

107 rows appear in the mapping below (144 total – 37 Unclassified, which have no question to group under and are excluded entirely). Of those 107, **all 107 still carry [DRAFT] summary and action text** — expected on this run (run 1), since finalisation happens directly in Excel and hasn't started yet.

The 37 excluded rows are not accounted for above — they're both missing a `community_question` and still on DRAFT text, meaning they need two separate fixes in Excel: the question column corrected, and the summary/action text finalised. See Stage 8 Final QA for the true dataset-wide DRAFT count (144/144).

In run 2 notebook, this cell should be re-read qualitatively: does each summary genuinely answer the question it's paired with, and is the action step concrete and patient-appropriate (a step to take, not clinical advice)?

i) Source overlap and gap check

Brief: "Where are there overlaps, gaps or inconsistencies across sources?" Flags questions with multiple sources (need manual consistency check) and questions with no sources at all (genuine coverage gaps).

```
In [47]: # -----
# Overlap check - satisfies the brief's "Where are there overlaps across sources
# requirement. Only shows questions where multiple sources cover the same questi
# questions with a single source or no source are reported separately.
# -----

unclassified_mask = df_structured["community_question"].str.contains("unclassifi
classified_for_overlap = df_structured[~unclassified_mask]
```

```

all_questions = [q for questions in COMMUNITY_QUESTIONS.values() for q in questions]

overlap_check = (
    classified_for_overlap.groupby("community_question")["source_organisation"]
    .apply(lambda s: sorted(s.unique()))
    .reset_index(name="sources_covering_this_question")
)
overlap_check["source_count"] = overlap_check["sources_covering_this_question"].
overlap_check["needs_manual_consistency_check"] = overlap_check["source_count"]

# Questions with multiple sources – need human consistency check
flagged = overlap_check[overlap_check["needs_manual_consistency_check"]]
print(f"{len(flagged)} question(s) have multiple sources – read side by side to")
print()

# Questions with no sources at all – gap in coverage
covered_questions = set(overlap_check["community_question"])
no_coverage = [q for q in all_questions if q not in covered_questions]
print(f"{len(no_coverage)} question(s) have no sources yet:")
for q in no_coverage:
    print(f" - {q}")

# Show only the flagged overlaps, not the full table
flagged[["community_question", "sources_covering_this_question"]]

```

14 question(s) have multiple sources – read side by side to confirm they don't contradict:

3 question(s) have no sources yet:

- What can I do if I feel the process is too slow?
- Is it normal to wait between steps?
- Am I allowed to get a second opinion, including in another country?

Out[47]:

	community_question	sources_covering_this_question
1	Can I ask for alternative options?	[kanker.nl, rijksverheid.nl]
3	Can I choose my hospital or specialist?	[kanker.nl, nfk.nl, zorginstituutnederland.nl]
4	Can I request explanations in simpler language?	[kanker.nl, thuisarts.nl]
5	Do I have time to decide?	[kanker.nl, thuisarts.nl]
6	How are treatment decisions made?	[kanker.nl, nfk.nl, thuisarts.nl]
7	How do I find reliable health information or a trustworthy provider?	[kanker.nl, nfk.nl]
8	How do I get referred to psychological care, physiotherapy, nutrition advice or fertility support?	[kanker.nl, thuisarts.nl, zorginstituutnederland.nl]
9	How do I make a complaint about my care or a healthcare provider?	[kanker.nl, thuisarts.nl, zorginstituutnederland.nl]
11	How do referrals work in the Netherlands?	[rijksverheid.nl, thuisarts.nl, zorginstituutnederland.nl]
12	How long should I expect to wait for appointments or treatment?	[kanker.nl, rijksverheid.nl]
13	What additional services can I access?	[kanker.nl, nfk.nl, rijksverheid.nl, thuisarts.nl, zorginstituutnederland.nl]
14	What are my general rights and responsibilities as a patient?	[kanker.nl, nfk.nl, rijksverheid.nl, thuisarts.nl]
15	What does my health insurance actually cover, and what will it cost me?	[kanker.nl, nfk.nl, rijksverheid.nl, thuisarts.nl, zorginstituutnederland.nl]
17	What should I expect when I visit my GP, and how does registering work?	[kanker.nl, rijksverheid.nl, thuisarts.nl, zorginstituutnederland.nl]

Observation — Source overlap and gap check:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

9 questions have multiple sources — How do referrals work in the Netherlands?, Can I choose my hospital or specialist?, What should I expect when I visit my GP?, What does my health insurance actually cover, and what will it cost me?, How are treatment decisions made?, Will a second opinion be covered by insurance?, What are my general rights and responsibilities as a patient?, How do I get referred to psychological care, physiotherapy, nutrition advice or fertility support?, How do I find reliable health information or a trustworthy provider?

7 questions have no sources yet on this run: What can I do if I feel the process is too slow?, Is it normal to wait between steps?, Can I ask for alternative options?, Am I allowed to get a second opinion including in another country?, What additional services can I access?, What if I don't understand medical terminology?, Can I request explanations in simpler language?

None of these 7 trace to the header-routing issue that previously affected the insurance-cost question — that one is now resolved (see Stage 6b iteration note) and correctly shows up in the multi-source list above.

Number of questions with no sources expected to drop come Run 2 --> once the 37 currently unclassified community_question rows have been manually classified in excel.

Stage 8 - Deliverables & Final QA

► **What this stage is for** (click to expand/collapse)

```
In [49]: # -----
# Final QA checklist - checks ALL 14 required fields for emptiness, every
# classification field for "Unclassified", advice-toned Language, and DRAFT text
#
# Note: topic, patient_journey_stage, stakeholder and community_question are che
# SEPARATELY, not because they're independent problems on a first run (they'll
# often show the same count, since stage/question both derive from topic) - but
# because errors in manual correction within Excel could result in updating one
# field without the others. On a second run, divergence between these four
# counts is the signal that a correction step was missed on some rows, not a bug
# -----

def field_completeness(df: pd.DataFrame) -> pd.DataFrame:
    """Check every REQUIRED_FIELDS column for empty rows.

    A field counts as empty if it's NaN, or a string containing only whitespace
    (an empty-looking cell that isn't technically NaN, e.g. a stray space typed
    into Excel). Returns one row per field so both the printed summary and the
    pass/fail logic below can read from the same table, rather than each
    recomputing emptiness independently and risking the two disagreeing.
    """
    rows = []
    for field in REQUIRED_FIELDS:
        empty_count = (df[field].isna() | (df[field].astype(str).str.strip() ==
        rows.append({"field": field, "empty_rows": empty_count, "complete": empt
    return pd.DataFrame(rows)

completeness = field_completeness(df_structured)

# The 4 classification fields that can legitimately hold an "Unclassified" value
# Adding a 5th classification field to the pipeline later only requires adding i
# here - unclassified_counts() below loops over this list generically rather tha
# repeating a hardcoded check per field.
UNCLASSIFIABLE_FIELDS = ["topic", "patient_journey_stage", "stakeholder", "commu

def unclassified_counts(df: pd.DataFrame) -> pd.DataFrame:
    """Check every UNCLASSIFIABLE_FIELDS column for Unclassified rows.

    Uses str.contains rather than an exact string match, since the actual value
    varies slightly by field (e.g. "Unclassified - needs manual tagging" for
    topic vs. plain "Unclassified" for community_question) - contains() catches
    all variants without needing to hardcode each one.
    """
```

```

rows = []
for field in UNCLASSIFIABLE_FIELDS:
    count = df[field].str.contains("unclassified", case=False, na=False).sum()
    rows.append({"field": field, "unclassified_rows": count})
return pd.DataFrame(rows)

unclassified = unclassified_counts(df_structured)

# Total rows needing classification attention across all 4 fields combined - use
# only to decide pass/fail below, not printed directly (the per-field breakdown
# above is more useful to a human deciding what to fix).
tag_review_pending = unclassified["unclassified_rows"].sum()

print("FINAL QA CHECKLIST")
print("-" * 45)
print(f"All URLs look valid: {df_structured['url_valid'].all}")

# Pad every "Rows with Unclassified <field>:" label to the same width dynamicall
# based on whichever field name is longest - fixes alignment breaking if a field
# name is added/changed later, rather than hardcoding a column width that could
# silently stop working.
label_width = max(len(f"Rows with Unclassified {f}:") for f in UNCLASSIFIABLE_FIELDS)
for _, row in unclassified.iterrows():
    label = f"Rows with Unclassified {row['field']}:"
    print(f"{label:<{label_width}}{row['unclassified_rows']}")

#print(f"Rows flagged for advice-toned language: {df_structured['advice_tone_flag'].any()}")
#print(f"Rows still on a DRAFT summary/action: {df_structured['content_needs_review'].any()}")
#print(f"Total rows: {len(df_structured)}")
#print()
#print(f"Field-by-field completeness (all {len(REQUIRED_FIELDS)} required fields)")
#print(completeness.to_string(index=False))

# Single source of truth for the pass/fail verdict - drives the warning/success
# message printed below. Purely informational (no export step reads it) - a human
# decides whether the file is ready for CSN/Tableau to use.
qa_passed = not (
    not completeness["complete"].all()
    # or df_structured["advice_tone_flag"].any()
    # or df_structured["content_needs_review"].any()
    or tag_review_pending > 0
)

if not qa_passed:
    print(f"\nWARNING: do not submit yet.")

    # Empty required fields - one line per field, with the actual row count.
    incomplete = completeness[~completeness["complete"]]
    for _, row in incomplete.iterrows():
        print(f" {row['empty_rows']} row(s) have an empty {row['field']} - write")

    # Unclassified classification fields - one line per field, only for fields that
    # actually have unclassified rows (skips printing "0 row(s)..." noise).
    for _, row in unclassified[unclassified["unclassified_rows"] > 0].iterrows():
        print(f" {row['unclassified_rows']} row(s) still have Unclassified {row['field']}")

    #if df_structured["advice_tone_flag"].any():
    #    print(f" {df_structured['advice_tone_flag'].sum()} row(s) flagged for")

```



```
#if df_structured["content_needs_review"].any():
#     print(f" {df_structured['content_needs_review'].sum()} row(s) still ca
else:
    print(f"\nAll {len(REQUIRED_FIELDS)} required fields populated and reviewed,
```

FINAL QA CHECKLIST

All URLs look valid: True
Rows with Unclassified topic: 0
Rows with Unclassified patient_journey_stage: 0
Rows with Unclassified stakeholder: 0
Rows with Unclassified community_question: 4

WARNING: do not submit yet.
4 row(s) still have Unclassified community_question – fix the community_questio
n column in Excel (CSN_DATASHEET.xlsx).

Observation — Final QA checklist:

Run 1 — pre-correction baseline. See Run 2 notebook for submission-ready figures.

Issues remain — matches the checklist's own "WARNING: do not submit yet," expected on this run (Run 1) since no manual correction has happened.

Check	Result
All URLs valid	✓ True
Unclassified topic	37 rows
Unclassified patient_journey_stage	37 rows
Unclassified stakeholder	9 rows
Unclassified community_question	37 rows
Advice-toned language flagged	[UPDATE AFTER RERUN] rows
Still on DRAFT summary/action	144 rows (all)
Empty action_next_step	14 rows

- topic, patient_journey_stage, and community_question all show the same 37 on this run — expected, since stage and question are both derived from topic and nothing has been manually edited yet.
- These four are checked independently (not just topic) so that on a later run (Run 2), if an editor corrects topic in Excel but misses updating the dependent fields, the mismatch between counts surfaces here rather than silently passing as "no flags remain."
- The advice-tone count above reflects a narrower phrase list than earlier test runs — flagging now only requires a directive paired with treatment/medication/dose-specific language (e.g. "you should take," "the correct dose"), rather than bare generic phrases like "you should" or "you need to" that would also fire on legitimate navigational action steps.

The 14 empty action_next_step rows come from draft_action_next_step() finding no sentence containing one of its 8 trigger phrases (e.g. "you can," "contact

your") anywhere in the translation — either the page is genuinely informational with no natural action step, or an action-oriented sentence exists but is phrased differently than the 8 phrases and was missed.

Note: Final QA warning logic — iteration history

What was wrong: the checklist printed several concerns — unclassified tags, advice-toned language, draft text — but the "do not submit yet" warning only actually checked some of them. Unclassified topic, patient_journey_stage, stakeholder, and community_question were printed but never wired into the pass/fail logic, so a dataset could have 40 rows still Unclassified and this cell would still print **"ready for use"** a few lines below a count saying otherwise. Field-completeness alone couldn't catch it either: `"Unclassified - needs manual tagging"` is a real, non-empty string, so it passes the emptiness check even though the row isn't actually reviewed.

Fix (28/08): `tag_review_pending` computed explicitly across all four classification fields via a generic `UNCLASSIFIABLE_FIELDS` list, and folded into `qa_passed`, a single boolean driving the warning/success message printed below. Checking all four fields separately (not just an aggregate) matters because manual correction in Excel can update one field without the others (see REFERENCE FOR EDITS) — divergence between the four counts on a later run is the signal a correction step was missed, not a bug.

Confirmed at scale (28/08): run against the full 144-row dataset. The `WARNING: do not submit yet` message correctly printed alongside all outstanding issues — consistent with `qa_passed` being `False`, though the boolean itself isn't printed directly.

STATUS: RESOLVED

FINAL NOTES

Suggested refresh/update approach

The brief explicitly asks for this as its own deliverable — not just a working mechanism, but a stated approach. Per the brief's own note, this can stay conceptual; it doesn't need to be deployed.

How it works today: the working Excel's `SCRAPE` flag is the single control for what gets refreshed. Flag a row `SCRAPE` and it goes through Stages 3-6 again — re-scraped, re-translated (or reused from cache if the Dutch text is unchanged), and re-classified. Leave a row un-flagged and it passes through Stage 3-6 completely untouched, keeping whatever's already in Excel (including manual corrections) exactly as it was.

`upsert_to_db()` matches on `original_url`, so re-running never duplicates a row — it just updates the existing one in place.

This notebook (RUN 1) demonstrates the full-refresh path: every row is flagged `SCRAPE`, so all 144 URLs are freshly scraped, translated, and classified from scratch —

the same mechanism a genuine full refresh would use if CSN wanted to fully re-verify every source at once. A partial refresh — flagging only the URLs that have actually changed, or only newly added ones — uses the exact same mechanism, just on a smaller subset of rows; the Run 2 notebook (reading the corrected sheet) demonstrates that lighter path, since most rows there stay un-flagged and only a handful would need re-processing.

Suggested cadence: re-run the pipeline whenever CSN suspects an existing source page in `CSN_DATASHEET.xlsx` has changed since the previous scrape (e.g. a policy update or additional content added to the page), or whenever CSN adds a new URL to `CSN_DATASHEET.xlsx`. In both cases, set the relevant row(s) to `SCRAPE` in `CSN_DATASHEET.xlsx` before re-running the pipeline. Every other row stays untouched, so a refresh never risks disturbing already-corrected content.

Suggested ownership: rotate a "refresh check" through the team the same way project leadership rotates (see Assignment 1's Team Roles) — one person per cycle confirms the source list is still accurate and the notebook still runs cleanly end to end.

Recommendations for CSN

Run 1 — preliminary only. These are based on the pre-correction baseline (144 rows, 37 Unclassified) and will change once manual correction is complete. The Run 2 notebook, built from the corrected `CSN_DATASHEET.xlsx`, is where the real recommendations for CSN are derived — see that notebook's equivalent section for the submission-ready version.

In []: