STRETCHLEARN · WORKBOOK

Programmatic SEO — Workbook

This workbook turns the course into a build kit for a real programmatic page set. Work each section against one specific project idea, defining your keyword pattern, sourcing and scoring data, designing the URL and template, deduplicating, and planning the indexing rollout, filling the worksheets with that project's actual numbers and entities. The templates are editable planners for a keyword-pattern validator, a data-source and field map, a page-quality scorecard, and an indexation tracker. Leave every total, score, and calculated cell blank for you to fill, an empty cell is always better than a guessed one.

What Programmatic SEO Is and When It Works

Pick a project, write its keyword pattern, validate demand and intent, and prove each page would add value.

Worksheet: Keyword Pattern Definition Worksheet

Write your project as one bracketed sentence, then break it into its head term and modifier list and estimate the page count.

Project idea in one line

Bracketed keyword pattern (e.g. [city] coworking spaces, [tool] alternatives)

Head term / fixed intent

Modifier dimension(s) and where the values come from

Estimated number of modifier values (the raw page-count ceiling)

Two-modifier matrix? If so, canonical ordering rule (e.g. alphabetical)

Three competitor URLs that already rank with this pattern

Exercise: Demand and Intent Validation Drill

Before building anything, sample 20 to 50 real combinations of your pattern in a keyword tool and read the live SERP for several of them.

• Across your sample, what is the spread of monthly search volume, and what is the rough aggregate (sum) across the full modifier list?

• For 5 to 10 sample queries, what page type actually ranks (comparison, list, calculator, location guide), and does it match what your template would produce?

• How strong are the page-one results, are they thin, generic, or non-specialized enough that a well-built

page could win?

• Can you source real data for most modifier values, or only for a famous few?

Checklist: Go / No-Go Quality Gate Checklist

[ ] The project can be written as a single bracketed keyword sentence
[ ] A representative sample shows meaningful aggregate demand across the tail
[ ] The live SERP serves a page type the template can credibly produce
[ ] Page-one competition is weak enough to be winnable at your authority level
[ ] Real, unique data exists (or can be built) for most modifier values
[ ] Any single random page would leave a real searcher glad they clicked

Sourcing and Structuring the Data

Find and combine the data, clean and deduplicate it, set a quality threshold, and design the content schema.

Worksheet: Data Source and Enrichment Map

List every data source feeding the set, how you are allowed to use it, and the join or derived field that makes the pages unique.

Source name and type (open data / first-party / licensed / computed)

Access method (API / dataset download / internal export / scrape)

Licence or terms-of-service note (republish allowed? attribution required?)

Fields this source contributes

Join key to combine with other sources (city / product / currency)

Derived or first-party field added for uniqueness

Last-updated date and refresh cadence

Exercise: Data Cleaning and Dedup Drill

Run your raw dataset through a cleaning pass so one row equals one trustworthy page, and remove entity duplicates before they become pages.

• Which fields have inconsistent formats (casing, dates, units, currency) that must be standardized?

• Which entities appear under variant names (St. vs Saint, USA vs United States) that must be normalized to one canonical form?

• What is your unique key per row, and which duplicate or merged rows did it expose?

• What range or sanity checks will flag impossible numbers and typos before they ship at scale?

Worksheet: Content Schema Worksheet

Define the fields every page will display, mapping each on-page element to a data field, a type, and required/optional status.

On-page element (title / intro / key-facts table / FAQ / related links / CTA)

Data field(s) that fill it

Data type (text / number / list / URL / boolean)

Required or optional

Template behaviour when the field is empty (hide section / fallback / disqualify row)

Part of the value layer? (definition / computed metric / FAQ answer)

How this field legitimately varies between entities

Checklist: Data Quality Threshold Checklist

[ ] A minimum set of required fields is defined before a row qualifies for a page
[ ] Under-populated rows are held back, not published as thin stubs
[ ] Entity-level duplicates removed via a single canonical key/slug per entity
[ ] Provenance (source, licence, last-updated) stored for every field
[ ] Range/typo checks catch outliers before generation
[ ] Excluded rows are logged so data-gathering can be targeted to unlock them

URL Structure, Templates, and Avoiding Duplication

Lock a scalable URL pattern, build a value-adding edge-case-safe template, and detect and resolve near-duplicate pages.

Worksheet: URL Structure Plan

Design the permanent URL pattern, the slug rules, and how hubs link the set together.
URL pattern mirroring the keyword pattern (e.g. /coworking/[city-slug])

Slug rule (lowercase, hyphenated, accent-folded; source field)

Canonical slug tied to the deduped entity key (one URL per entity)

Matrix ordering rule and how the reverse variant is handled (skip / 301 to master)

Parent path that groups the set (e.g. /compare/, /locations/)

Hub/index pages planned and which child pages they link to

Max crawl depth from homepage (target about three clicks)

Exercise: Template Edge-Case Stress Test

Render your template against your messiest rows before bulk publishing, because any bug ships on every page that hits that case.

• On the row with the least data, does every empty optional field cleanly hide its section and heading, with no placeholder left behind?

• Do grammar and pluralization adapt to the data (1 result vs 5 results, singular vs plural)?

• Does the entity with an unusually long or special-character name break the layout or slug?

• Does each page output a unique title tag, meta description, H1, and appropriate Schema.org markup generated from the row's data?

Exercise: Near-Duplicate Detection Drill

Crawl your own generated set with Screaming Frog's near-duplicate detection and decide a fix for each over-similar cluster.

• Which page clusters exceed a high content-similarity threshold, and which fields actually differ between them?

• Is the difference meaningful to a user, or are the pages effectively the same entity?

• For each cluster, is the right move to differentiate (add real data), consolidate (merge + 301), or canonicalize (point variants at a master)?

• After re-crawling, did similarity drop and is the intended canonical now selected?

Checklist: Template and Duplication Checklist

[ ] URL pattern is clean, consistent, and one-URL-per-entity (chosen as if unchangeable)
[ ] Every page has unique title tag, meta description, and H1 assembled from data
[ ] Value layer (tables, FAQ, computed metrics) renders from fields, not static text
[ ] Data-driven related internal links render on every page with descriptive anchors
[ ] Optional-field and grammar/pluralization edge cases handled at the template level
[ ] Self-referencing canonicals on distinct pages; symmetric/parameter variants canonicalized to a master
[ ] Near-duplicate clusters resolved by adding data, consolidating, or raising the data threshold

Indexing, Internal Linking, and Maintenance

Get the set discovered with segmented sitemaps and internal links, monitor indexation, and prune and refresh over time.

Worksheet: Sitemap and Indexation Worksheet

Plan segmented sitemaps and track what share of each page-type group actually gets indexed.

Page-type segment (e.g. comparisons / locations / glossary)

Separate sitemap file name for this segment

URL count in segment (must stay within 50,000 per file)

Sitemap contains only canonical, indexable, HTTP 200 URLs? (Y/N)

Submitted in Search Console and referenced in robots.txt? (Y/N)

Indexed count from the Indexing report (fill from GSC)

Indexation rate and main blocker if low (thin / duplicate / orphan / depth)

Exercise: Internal Linking and Orphan Drill

Connect the set so authority and crawlers reach the long tail, then verify in a crawl that nothing is orphaned.

• Does every segment have a hub/index page linking to all its children, and is that hub linked from a prominent place (nav, footer, directory)?

• How does the template choose its data-driven related links (same category, proximity, shared attributes, price tier), and is the count per page reasonable?

• Does a crawl confirm every generated URL has at least one internal link in, with no orphans?

• Which existing editorial pages can link into the programmatic set to pass established authority?

Exercise: Pruning and Refresh Planning Drill

Use Search Console performance by path to decide what to scale, improve, or cut, and define how the data stays fresh.

• Which page-path segments earn impressions and clicks, and which never index or never rank?

• For a persistent non-performing segment, is the fix to enrich the data/template, consolidate, or prune (noindex/remove)?

• What is the refresh cadence per data source so prices, stats, and listings stay current, with honest lastmod dates?

• How are new entities promoted into pages, and how are dead entities retired (noindex/301) to avoid stale or soft-404 pages?

Checklist: Launch and Maintenance Checklist

[ ] Sitemaps segmented by page type so indexation is measurable per group
[ ] Rollout phased: a high-quality batch shipped and confirmed indexing before scaling
[ ] Crawl budget protected (redirect chains fixed, junk URLs blocked/noindexed)
[ ] No orphan pages: every URL reachable via hub and related internal links within ~3 clicks
[ ] Indexation and performance tracked per segment in Search Console on a schedule
[ ] Dead-weight pages that never index or rank are consolidated or pruned
[ ] Data refresh process live, with new entities added and dead entities retired

Your Action Plan

1. Write the project as one bracketed keyword pattern and list its head term and modifier values
2. Validate demand on 20 to 50 sample combinations and confirm the SERP intent matches the page you would build
3. Map data sources and licences, then combine and enrich them so each page carries unique, useful data
4. Clean and normalize the dataset, dedupe entities to one canonical key, and set a minimum-data quality threshold
5. Design the content schema: required and optional fields, empty-field behaviour, and the value layer
6. Lock a permanent URL pattern with one slug per entity and a matrix-ordering rule for two-modifier sets

7. Build the template with unique per-page SEO fields, data-driven value layer, and edge-case handling, then stress-test it
8. Crawl the generated set for near-duplicates and resolve clusters by adding data, consolidating, or canonicalizing
9. Generate segmented XML sitemaps, roll out in batches, and submit and monitor indexation in Search Console
10. Wire hub and data-driven internal links so no page is orphaned, then monitor, prune, and refresh the set over time