The ArcGIS Workbook

ON THIS PAGE

The ArcGIS Workbook

Hands-on exercise sets for all eight Compendium volumes, using free resources wherever possible, plus two capstone project specs with self-assessment rubrics.

Workbook: Volume A - Foundations

These five exercises put the ideas from Volume A into your hands. Everything here runs on a free ArcGIS public account and data from the Living Atlas or data you create yourself in Map Viewer. No organizational account, no ArcGIS Pro, no credits. Where a

concept is taught in depth in the Compendium, this workbook cross-references the chapter rather than re-teaching it; do the reading first if a step feels unfamiliar.

Work the exercises in order. Each one builds an instinct the next one leans on: how the world becomes data, how the flat map lies about the round earth, how styling choices manufacture stories, how to interrogate a stranger's data, and how to spot and repair a bad map.

Exercise 1: Choose a representation for five real-world things

Objective: Decide, and then verify against real layers, whether five real-world phenomena are best represented as vector points, lines, polygons, or as rasters.

What you need: A free ArcGIS public account, Map Viewer, and Living Atlas search. Background: Compendium Chapter 1 (How GIS Thinks) and Compendium Chapter 5 (Finding Data).

Steps:

1. On paper, before touching software, write down how you would represent each of these: a river, a city, elevation, land cover, and air temperature. For each, commit to vector or raster, and if vector, to point, line, or polygon. Also note whether the thing is a discrete object with edges or a continuous field with a value everywhere.
2. Sign in to ArcGIS Online and open a new map in Map Viewer. Use `Add > Browse layers` and switch the source to Living Atlas.
3. Search for a hydrography or rivers layer and add one. Zoom from continental scale down to a single river. Note where the representation changes character: at small scales a major river may appear as a line; zoomed far in, some datasets represent wide rivers as polygons. Ask yourself why the same real thing gets two geometries.
4. Search for a populated places or cities layer. Observe that a city is a point at one scale and, in boundary datasets, a polygon at another. Neither is wrong; each answers a different question.
5. Add a terrain or elevation layer and a land cover layer from Living Atlas. Zoom in until you can see individual cells or the smoothed surface. Compare this to the crisp edges of the vector layers. Elevation and temperature are continuous fields; there is a value at every location, which is why rasters (or contour derivations) carry them.

6. Return to your paper answers and grade yourself. Where reality disagreed with your prediction, write one sentence explaining what the dataset publisher was optimizing for.

Success criteria: You can state, for each of the five phenomena, which model the real published layers use, and you can explain the two cases where scale changed the answer (rivers and cities). If your explanation includes the words "discrete" and "continuous" used correctly, you have the core of Chapter 1 internalized.

Stretch goal: Find one phenomenon that genuinely resists both models cleanly, such as a wetland boundary or a wildfire perimeter, and write three sentences on what is lost by forcing it into a polygon.

Exercise 2: Catch Web Mercator lying to you

Objective: Demonstrate projection distortion yourself by proving that two regions that look wildly different in size on the default basemap are actually comparable on the ground.

What you need: A free ArcGIS public account and Map Viewer's measurement tools. Background: Compendium Chapter 3 (Coordinate Systems and Projections).

Steps:

1. Open a new map with any standard basemap. The default web basemaps use Web Mercator, which preserves shape and direction locally but inflates area toward the poles. You are about to quantify that inflation instead of taking it on faith.
2. Zoom so both Greenland and Africa are visible. Eyeball it: on screen, Greenland looks roughly the size of the whole African continent. Write down your visual estimate of the ratio.
3. Open the measurement tool in Map Viewer (look for it among the map tools on the right side of the interface) and choose area measurement. Carefully trace an outline around Greenland. Record the reported area. The measurement tool computes geodesic values, meaning it measures on the ellipsoid, not on the distorted screen, so the number it gives you is honest even when the picture is not.
4. Now trace a rough outline around Africa and record that area. Compute the ratio. Compare it to your visual estimate from step 2. The gap between the two ratios is

the distortion, made concrete.

5. Repeat the experiment with distance instead of area: measure a straight east-west line of a few hundred kilometers near the equator, then measure a line of the same on-screen length across northern Canada or Scandinavia. Record both geodesic distances and note how different they are for the same screen length.
6. Write two sentences: one stating roughly how many times larger Africa actually is than Greenland, and one stating why the basemap cannot show that truthfully (hint: no flat map preserves everything; Web Mercator trades area fidelity for other properties, as Compendium Chapter 3 explains).

Success criteria: Your measured Africa-to-Greenland ratio should be dramatically larger than your on-screen visual estimate — by an order of magnitude in the difference of impressions. If your measured ratio came out close to your visual guess, your traces were too rough or you measured planar rather than geodesic values; retrace more carefully.

Stretch goal: Search Living Atlas or ArcGIS Online for a basemap or web map built on an equal-area or polar projection (they exist, though the default catalog leans Mercator), open it, and describe how the same two landmasses compare there.

Exercise 3: Make the same data tell four different stories

Objective: Restyle one choropleth layer four ways and observe how classification method, class count, and color choices change the apparent story without changing a single data value.

What you need: A free ArcGIS public account, Map Viewer, and one Living Atlas demographic layer with a numeric attribute, such as a census or American Community Survey layer of income or population by county. Background: Compendium Chapter 7 (Styling and Smart Mapping) for the mechanics and Compendium Chapter 4 (Cartographic Design) for the judgment.

Steps:

1. Add a county-level demographic layer from Living Atlas to a new map. Open the layer's styling pane, choose the numeric attribute (median income works well because it is skewed), and pick the counts-and-amounts color style.

2. Version one: switch the classification to equal interval with a moderate number of classes. Take a screenshot. Notice how skewed data under equal interval typically dumps most areas into one or two classes, making the map look uniform with a few extreme outliers.
3. Version two: change only the method to quantile, same class count. Screenshot. Quantiles force equal membership per class, so the map now shows vivid variation everywhere, including where real differences are small.
4. Version three: switch to natural breaks. Screenshot. This usually lands between the first two, clustering around genuine gaps in the data.
5. Version four: keep any method but set manual class breaks that you choose to make a deliberate point, for example placing a break just above or below a politically meaningful threshold, and swap the color ramp for one with an aggressive dark end. Screenshot.
6. Lay the four screenshots side by side. For each, write the one-sentence headline a newspaper would run if it published that map. The data never changed; only your choices did.
7. Decide which version you would actually publish and defend it in two sentences, referencing what the classification does to the distribution rather than which looks best.

Success criteria: You produce four visibly different maps from one unchanged attribute, and your four headlines genuinely differ in what they claim about the region. If two of your maps look the same, your data may be too uniform; switch to a more skewed attribute and rerun.

Stretch goal: Add a fifth version using an unclassed (continuous) color ramp and explain in three sentences why unclassed styling is more honest for some audiences and less readable for others.

Exercise 4: Interrogate a stranger's layer

Objective: Evaluate the fitness of a layer you did not make, using only its item page and its behavior in the map, and issue a written verdict on whether you would trust it for real work.

What you need: A free ArcGIS public account and the ArcGIS Online search. Background: Compendium Chapter 5 (Finding Data).

Steps:

1. Pick a topic you know something about, such as your city's parks, local trails, or schools in your state. Search ArcGIS Online content (not just Living Atlas — include public community content) for that topic and pick a layer that is not from an authoritative publisher. Deliberately choose something a random user shared.
2. Before adding it to a map, read the item page top to bottom. Score each of the following as pass, fail, or unknown: Who published it, and are they identifiable? Is there a description of the source and method, or is the summary empty or boilerplate? When was it last updated, and is that recent enough for the subject? Are terms of use stated? Is the geographic extent stated and plausible?
3. Add the layer to a map alongside an authoritative reference, such as an official government layer or a Living Atlas curated equivalent, plus an imagery basemap.
4. Do a completeness spot check: pick three places you personally know and verify the features exist, are positioned correctly against imagery, and have sensible attributes. Open a few pop-ups and look for empty fields, placeholder text, or obviously stale values.
5. Do a consistency check: compare feature counts or coverage against the authoritative layer in one area. Missing whole neighborhoods is a different failure than a few missing features, and it matters which one you have.
6. Write a five-line verdict: what the layer is good for, what it must not be used for, its biggest unknown, its likely vintage, and whether you would use it as-is, use it with caveats, or reject it.

Success criteria: Your scorecard has an explicit pass/fail/unknown for every question in step 2, and your verdict names at least one concrete defect or gap you found in step 4 or 5. Finding zero issues usually means the spot check was too gentle, not that the layer is perfect; check three more places.

Stretch goal: Repeat the same scorecard against the authoritative or Living Atlas equivalent and note which questions even well-curated layers leave unanswered.

Exercise 5: Build a bad map, then rescue it

Objective: Deliberately commit five classic cartographic sins in one map, then fix them one at a time so you can articulate exactly what each sin does to the reader.

What you need: A free ArcGIS public account, Map Viewer, and a county- or tract-level demographic layer from Living Atlas with both a raw count field (total population) and something to normalize by (area or households). Background: Compendium Chapter 4 (Cartographic Design), with styling mechanics from Compendium Chapter 7 (Styling and Smart Mapping).

Steps:

1. Start a new map and commit the sins on purpose. Sin one: style the layer by a raw count (total population) using a color-filled choropleth, so big empty counties look important. Sin two: choose a full-spectrum rainbow color ramp with no intuitive order. Sin three: crank the class count high enough that adjacent classes are indistinguishable. Sin four: put it all on a visually loud basemap, such as detailed imagery, so the thematic colors fight the background. Sin five: turn on labels for every feature at every scale.
2. Save this map as "Before" and take a screenshot. Show it to someone, or honestly ask yourself: what is the single takeaway? A good thematic map has one; this one should have none.
3. Fix the data sin first, because no styling can save the wrong variable: switch the choropleth to a normalized value, either by using a rate/density field or by setting the divide-by option in the style pane (population divided by area, or a count divided by total households). Counts belong in graduated symbols; rates and densities belong in color. This distinction is the load-bearing rule of the whole exercise.
4. Fix the color sin: replace the rainbow with a single-hue or two-hue sequential ramp where darker reliably means more. If your variable diverges around a meaningful midpoint, use a diverging ramp centered on that midpoint instead.
5. Fix the legibility sins: reduce to a small number of classes a reader can actually match to the legend, switch to a muted or light-gray basemap so the thematic layer owns the visual foreground, and restrict labels to larger scales or remove them entirely.

6. Save as "After," screenshot, and place the pair side by side. Write the one-sentence takeaway the After map communicates, then write one sentence per sin describing the specific damage it did in the Before map.

Success criteria: A stranger shown only the After map can state its main pattern in one sentence without prompting, and your sin-by-sin notes identify raw-counts-in-choropleth as the most serious of the five, since it misleads even careful readers rather than merely slowing them down.

Stretch goal: Make a third version tuned for a different audience, such as a colorblind-safe version with a ramp chosen for deuteranopia, and note which of your fixes survived unchanged.

Where to go next

If these exercises felt comfortable, you are ready for Volume B: Exercise 3 and 5 lead directly into Compendium Chapter 7 (Styling and Smart Mapping) and Compendium Chapter 9 (Pop-ups, Fields, and Labels), and the skeptical habits from Exercise 4 are the working posture for Compendium Chapter 10 (Hosted Feature Layers) when you start publishing layers of your own that strangers will interrogate right back.

Workbook: Volume B — ArcGIS Online Mastery

These five exercises drill the skills taught in Compendium Chapters 6 through 10: the Map Viewer, smart mapping, Arcade, pop-ups, and hosted feature layers. Work them in order — each one builds on the map you made in the last. Exercises 1 through 4 need nothing more than a free ArcGIS public account and the Living Atlas. Exercise 5 requires publishing privileges; if you don't have them, a read-along alternative is included so you still get the mental model. If you're unsure which kind of account you have, Compendium Chapter 2 (The ArcGIS Ecosystem) explains the account types.

One rule for the whole workbook: when a step doesn't work the way you expect, don't click randomly. Stop, name what you expected, and find the panel that owns that behavior — knowing which panel owns what is most of Map Viewer fluency.

Exercise 1: The Map Viewer grand tour — one map, every panel

Objective: Build and save a web map that deliberately touches every major Map Viewer panel, so you know where everything lives without hunting.

What you need: A free ArcGIS public account, signed in at arcgis.com. Living Atlas access is built in.

Steps:

1. Open Map Viewer with a new, empty map. Orient yourself first: one toolbar manages what's *in* the map (layers, basemap, tables, legend), the other manages *how the selected thing behaves* (properties, styles, filters, effects, labels, pop-ups). Every step below should use a panel you haven't touched yet — that's the tour.
2. Add data: `Add > Browse layers`, switch the source to Living Atlas, and search for a county-level demographic boundary layer — one of Esri's American Community Survey (ACS) boundary layers works well because it's polygon data with lots of numeric fields. Add it. (Compendium Chapter 5 covers judging what you find; here, an Esri-published ACS layer is a safe pick.)
3. Basemap: switch to a muted canvas basemap (a light or dark gray option) so thematic color will read clearly later. Notice the basemap is a slot, not a layer in your layers list.
4. Layer properties: in the layers pane, rename the layer to something human ("US Counties — Demographics"). Note that this renames the layer *in this map only* — the source item is untouched. Set a modest transparency and a visible range so the layer switches off when zoomed way out.
5. Filter: with the layer selected, build a filter that limits the display to one state or region. Confirm features outside the filter vanish from both the map and pop-up hit-testing.
6. Labels: enable labels using the county name field. Add a halo so labels survive busy backgrounds. Set a visible range on the labels tighter than the layer's own.
7. Effects: apply one effect (drop shadow or bloom), look at it, then remove it. The goal isn't to keep it — it's to know the panel exists so you reach for it deliberately later (Compendium Chapter 7 covers when effects help and when they're noise).

8. Bookmarks: create two — one at full extent of your filtered region, one zoomed to a single metro area.
9. Save: `Save > Save as` with a real title, at least three tags, and a one-sentence summary. Then find the map's item page from your Content list and skim what got recorded there.

Success criteria: Sign out, sign back in, and reopen the map from `Content`. The filter, labels, transparency, bookmarks, and renamed layer all persisted. Without opening the app, you can say from memory which panel owns filtering, which owns labels, and which owns effects.

Stretch goal: Add a second Living Atlas layer, group the two layers, and set staggered visible ranges so one hands off to the other as you zoom — a generalized layer far out, a detailed one close in.

Exercise 2: Smart-mapping safari — one layer, five styles

Objective: Style the same numeric field five different ways and judge, for each, what it shows honestly and what it hides.

What you need: The map from Exercise 1 (or a fresh map with the same kind of county-level ACS polygon layer). You need one raw count field (total population works) and either an area field or a second count to normalize by.

Steps:

1. Open the styles pane for the layer and choose your count field. Watch the suggested style change — smart mapping inspects the data before offering options. That's the feature you're auditing in this exercise: are its suggestions right for *your question*? Fix the question now: "Where is population concentrated?"
2. **Style 1 — Counts and Amounts (size):** proportional symbols on the raw count. Judge it: do symbols pile up unreadably in dense regions? Does it at least avoid lying about totals?
3. **Style 2 — Counts and Amounts (color) on the raw count:** a choropleth of raw totals. Judge it harshly: big rural counties glow simply because they're big or populous in absolute terms. This is the classic misleading map — note *why* (Compendium Chapter 4 treats this in depth).

4. **Style 3 — Counts and Amounts (color), normalized:** same color style, but set the divide-by option to a normalization field (area, or total population if you switched to mapping a subgroup). Compare directly against Style 2: which counties changed class, and which map answers "where is it concentrated" rather than "where is there simply more land or more people"?
5. **Style 4 — Dot density:** available for polygons with counts. Judge it at three zoom levels: it can be evocative at one scale and unreadable static at another. Note the scale where it breaks.
6. **Style 5 — Above and Below:** a diverging theme centered on the average or another meaningful midpoint. Judge whether the midpoint is *meaningful* — a diverging ramp around an arbitrary center is decoration, not analysis.
7. Score all five styles from 1 to 5 against your fixed question, write the scores in the map's summary or a text note to yourself, and save the map with the winning style applied.

Success criteria: You can state, for each of the five styles, one thing it communicates well and one thing it distorts or hides. Your saved map keeps the style you judged best — and if that's a color style, it's the normalized one.

Stretch goal: Try the Relationship (bivariate) style with two related fields, and the combined Color and Size style. Decide whether either earns its added reading difficulty for your question. Chapter 7 (Styling and Smart Mapping) explains when multivariate styles pay off.

Exercise 3: Three Arcade expressions from scratch

Objective: Write three attribute expressions — a formatter, a classifier, and a guarded calculation — and wire each into the layer's pop-up.

What you need: The same map and layer, with at least two numeric fields you understand (a total and a subgroup count is ideal). Compendium Chapter 8 (Arcade from Zero to Fluent) is the companion text; this exercise is its gym.

Steps:

1. Open the pop-ups panel for the layer and find where attribute expressions are managed. Create a new expression and take stock of the editor: the `$feature`

profile variable, the function reference, the field list, and a way to test the expression against a real feature. Testing after every change is the habit to build.

2. **Expression 1 — the formatter.** Raw counts display like `1083459`. Return a readable string instead:

   ```
   Text(Round($feature.YOUR_COUNT_FIELD), '#,###')
   ```

   Replace the field name using the editor's field list rather than typing it — field names in ACS layers are not guessable. Test it, name the expression "Population (formatted)", and save.

3. **Expression 2 — the classifier.** Bucket a numeric value into named tiers using `When`:

   ```
   var v = $feature.YOUR_COUNT_FIELD;
   When(
     v >= 500000, "Large",
     v >= 100000, "Mid-size",
     v > 0, "Small",
     "No data"
   )
   ```

   Adjust the thresholds to fit your field's actual range (check the styles pane histogram for sensible breaks). Test it against a big county, a small one, and — important — one where the field might be empty.

4. **Expression 3 — the guarded calculation.** Compute a subgroup's share of the total, and refuse to divide by zero or null:

   ```
   var part = $feature.SUBGROUP_FIELD;
   var total = $feature.TOTAL_FIELD;
   IIf(total > 0, Round(part / total * 100, 1), null)
   ```

Returning `null` on bad input is deliberate: the pop-up shows nothing instead of `Infinity` or an error. Test on several features.

5. Add all three expressions to the pop-up's content and click around the map to see them in real use.

Success criteria: Clicking any feature shows a thousands-separated population, a tier name that matches the county's obvious size, and a percentage between 0 and 100 — with no error text, `Infinity`, or `NaN` on any feature you can find, including sparse ones.

Stretch goal: Rewrite the classifier using `Decode` and decide which reads better. Then move a copy of the classifier out of the pop-up entirely: use it as an expression-based style in the styles pane, so the map itself colors by your tiers.

Exercise 4: Pop-up makeover — text and chart blocks

Objective: Replace the default wall-of-fields pop-up with a designed one: a title that reads like a sentence, a text block that speaks plainly, and a chart that compares related fields.

What you need: The same map. Pick a set of related numeric fields that make a fair comparison — age brackets, income brackets, or similar breakdown fields in an ACS layer. Compendium Chapter 9 (Pop-ups, Fields, and Labels) is the reference; this is the practice.

Steps:

1. Audit the "before": click a feature, count how many fields a reader must scan to learn anything, and note the cryptic field names. That's your baseline.
2. Set the title. Use field substitution so it reads as a sentence fragment, e.g. `{NAME} County — {expression/...}` pulling in your formatted population from Exercise 3. The title should deliver the single most important fact by itself.
3. Remove or demote the default fields list. Add a **text** block and write one or two sentences that use `{field}` placeholders and your Exercise 3 expressions inline — for example: "About X% of residents here fall in the selected group, making this a *Tier* county." Prose with live values beats twenty labeled rows.

4. Add a **chart** block. Choose a column or pie chart, select your related breakdown fields as the series, and title the chart with the comparison it makes ("Population by age bracket"), not the dataset it comes from.
5. Order the blocks: text first, chart second. If you keep any fields list at all, trim it to the handful of fields a curious reader would genuinely want, and give each a plain-English display name in the fields configuration.
6. Test widely: click large counties, tiny ones, and edge cases. Charts built on near-zero values can render as an empty or misleading sliver — decide whether that's acceptable or whether the chart needs different fields.

Success criteria: A stranger clicking any county learns the headline fact within a few seconds without decoding a field name. The chart renders sensibly on every feature you test, and nothing in the pop-up shows a raw machine-readable field name.

Stretch goal: Configure number formatting (separators, rounded digits) on any remaining listed fields, and add an image or an Arcade element block to see what richer pop-up content looks like. Then compare your pop-up to the same layer's default in a fresh map — keep the before/after as your own proof of skill.

Exercise 5: Publish, view, edit, protect — the layer lifecycle drill

Objective: Publish a small hosted feature layer of your own, create a view layer from it, watch an edit flow from source to view, and use the view to hide fields and share safely.

What you need — read this first: This exercise requires an account that can publish hosted feature layers: an organizational account with a publishing role, or a free developer-tier account if one is available to you. A free public account cannot publish hosted layers. If that's you, skip to the read-along alternative at the end of this exercise — the mental model matters more than the clicks.

Steps:

1. Create the data yourself. In any text editor or spreadsheet, make a CSV with six rows of places you actually know — columns: `name`, `category` (two distinct values, e.g. "cafe" and "park"), `rating` (a number), `latitude`, `longitude`. Real-world

familiarity makes verification instant. (Compendium Chapter 11 covers data-creation paths; Chapter 12 covers doing schema properly.)

2. In `Content > New item`, add the CSV from your computer and choose to publish it as a hosted feature layer. At the field-configuration step, confirm `rating` is numeric and location is taken from your latitude/longitude columns.
3. Look at what you now own in Content: the CSV file item *and* a hosted feature layer item. Two items, one origin. Open the feature layer's item page and skim its settings — this page is the layer's control panel (Compendium Chapter 10 explains everything on it).
4. Create a view: from the feature layer's item page, create a view layer. In the view's settings, hide the `rating` field and add a definition filter so the view shows only one `category`. You now have a third item.
5. Make a new map from the **view**. Confirm: only one category appears, and pop-ups show no rating field. The view is a different front door to the same data.
6. Edit the source. Open the source layer (in a map with editing enabled, or via the data tab on its item page) and change a value or add a row that passes the view's filter. Reopen the view's map: the change is there. Views don't copy data — they share it.
7. Set the asymmetry that views exist for: share the view with everyone, keep the source private. Verify by opening the view's item in a private browser window — visible — and trying the source item's URL the same way — blocked.

Success criteria: Three items exist (file, layer, view); the view shows fewer fields and fewer rows than the source; an edit made through the source appears through the view without republishing; and the view is public while the source stays private.

Stretch goal: Create a second view with the opposite category filter — one source, two audiences. Then open the REST endpoint URLs of the source layer and the view from their item pages and compare them (Compendium Chapter 32 explains what you're looking at).

Read-along alternative (public accounts): Open any Esri-published hosted feature layer item from Living Atlas and walk its item page — overview, data tab, and whatever settings a viewer can see. As you read the steps above, map each onto what you're

looking at: publishing creates an item plus a live service; a view is a second service over the same stored data with its own field visibility, filters, and sharing; edits happen in the shared data, so every view sees them. When you later get publishing privileges, the drill will take minutes because the model is already in place.

Where this leaves you

You've now run the whole Volume B loop: find data, style it defensibly, compute what the data doesn't store, present it clearly, and control who sees what. Volume F (Chapters 26 through 30) is where these maps become apps; the map you built here is a ready input to that work.

Workbook: Volume C - Data Creation and Management

These five exercises walk the full life of a dataset: create data from nothing, give it structure, draw it by hand, deliberately damage it, then run the review loop that catches the damage. They exercise Compendium Chapter 11 (Creating Data), Chapter 12 (Schema Design), Chapter 13 (Editing Workflows), and Chapter 14 (Data Quality); when you get stuck on a concept, go back to the chapter and return here.

All five exercises share one scenario: a street-tree inventory for a park or a few blocks near where you live. Real trees you can see on the basemap make every step concrete, and each exercise feeds the next, so do them in order.

Before you start

You need a free ArcGIS public account (sign up at arcgis.com), a spreadsheet program or text editor, and a folder for the files you will make. Two steps require an organizational account and are flagged where they occur, each with a read-along alternative. Nothing requires ArcGIS Pro except stretch goals, which say so.

ArcGIS Online's interface shifts between releases, so steps are written at the level of intent. If a named button has moved, look for the same goal in the same area, or check Compendium Chapter 6 (Map Viewer Complete Reference) for the current layout.

Exercise 1: Build a CSV from nothing and import it two ways

Objective: Create a location table by hand and bring it into a map by the coordinate path and the address path, so you understand what each path needs and where each one fails.

What you need: A free public account, a spreadsheet or text editor, and Map Viewer. The address path's batch geocoder is a chargeable organizational service; if you only have a public account, read that branch and do the manual workaround in step 6 instead.

Steps:

1. In Map Viewer, zoom to your park or blocks on the imagery basemap and pick five trees you can actually see. For each one, get its latitude and longitude from the map itself: most map interfaces show coordinates for the cursor or a clicked location; if yours does not, drop a temporary sketch point on the tree and inspect it. Write the numbers down to five or six decimal places.
2. Build `trees_coords.csv` in your editor with columns `tree_id`, `latitude`, `longitude`, `species_guess`, `notes`, and one row per tree. Keep latitude and longitude as plain decimal numbers, negative for the southern and western hemispheres. This is the coordinate path from Compendium Chapter 11: the table already knows where everything is.
3. Add the CSV to your map using the map's add-layer-from-file option. When asked how to handle the file, choose to add it directly to the map rather than creating a hosted item; creating hosted feature layers requires an organizational account (Compendium Chapter 10 covers why). Confirm which columns hold the coordinates when prompted.
4. Check the result against the imagery. If all five points land in the ocean off West Africa, your latitude and longitude are swapped or a sign is missing. Fix the CSV, remove the layer, and re-add it.
5. Now the address path. Copy the file to `trees_addresses.csv`, delete the coordinate columns, and add an `address` column with the nearest street address for each tree, plus city and postal code columns. If you have an organizational account, add this file the same way and let the geocoder locate the rows; watch how it reports match quality, and inspect any row it placed with low confidence.

6. Public-account workaround: geocode by hand. Search for each address in the map's search box, read the coordinates of the result, and paste them into your CSV as new latitude and longitude columns. You have just done, manually and visibly, what the batch geocoder does invisibly, including the judgment calls on ambiguous matches.
7. Compare the two layers. Address-derived points typically land on the street centerline or the parcel front, not on the tree. Measure the offset on a couple of trees by eye against the imagery.

Success criteria: Both layers display; the coordinate-path points sit on the actual tree canopies in the imagery; you can state, in one sentence each, what the coordinate path requires that the address path does not, and vice versa.

Stretch goal: Add a sixth row with a deliberately malformed address (misspelled street, wrong postal code) and observe what the geocoder or the search box does with it. Bad-match behavior is the thing that bites real projects.

Exercise 2: Design a schema with domains for the tree inventory

Objective: Turn the ad-hoc CSV from Exercise 1 into a designed schema with field types, coded value domains, and a data dictionary you could hand to another person.

What you need: A spreadsheet or text editor. This exercise is design work first; applying the schema to a live hosted layer needs an organizational account, and that branch is flagged in step 5.

Steps:

1. List every question the inventory should answer: which trees need pruning, which species dominate, when was each tree last inspected, who inspected it. Fields exist to answer questions; write the questions first, per Compendium Chapter 12 (Schema Design).
2. Draft the field list. For each field decide a name (lowercase, no spaces, no reserved words), a display alias, a type (text, integer, decimal, date), and whether empty values are allowed. Aim for eight to twelve fields; a schema with thirty fields nobody fills in is worse than one with ten that everyone does.
3. Identify which fields should be constrained to a fixed list of values, then write those lists as coded value domains: a `condition` domain (for example Good, Fair, Poor,

Dead), a `species` domain from the species actually present in your area, maybe an `inspection_status` domain. For each domain, write both the stored code and the human-readable label. Ask of every text field: would free typing here create garbage? If yes, it wants a domain.

4. Decide on a range constraint for at least one numeric field, such as trunk diameter, and write down the plausible minimum and maximum. Write your reasoning next to it; the reasoning is what a future editor needs when a legitimate value falls outside the range.
5. Organizational accounts only: create an empty hosted feature layer (point geometry), open the item page's data tab, and add your fields, attaching each domain as you go. Public-account alternative: produce the same schema as a second sheet in your spreadsheet, one row per field, with columns for name, alias, type, domain values, and the question the field answers. That sheet is your data dictionary.
6. Stress-test the design by filling in three real trees from Exercise 1 using only the schema. Every time you hesitate about what to enter, the schema is underspecified; fix the definition, not just the entry.

Success criteria: A written data dictionary of eight to twelve fields, at least two coded value domains with codes and labels, one range constraint with reasoning, and three complete records entered without a single judgment call the dictionary could not answer.

Stretch goal: Add a second table for inspections, one row per visit, related to trees by `tree_id`, and write down why repeated inspections belong in a related table instead of extra columns. Compendium Chapter 12 treats relationships in depth.

Exercise 3: Digitize ten features with snapping

Objective: Draw ten connected features by hand with snapping turned on, so shared edges and endpoints actually coincide instead of merely looking close.

What you need: A free public account and Map Viewer. Use a sketch layer, which you can create without any org privileges; if sketch layers are unavailable in your setup, an editable hosted layer (org account) works identically, and Compendium Chapter 13 (Editing Workflows) covers both.

Steps:

1. In your Exercise 1 map, create a new sketch layer from the layer-adding controls. Zoom to your park at a scale where individual paths and lawn edges are clear in the imagery.
2. Open the snapping settings in the drawing toolbar and turn snapping on, including snapping to existing features and to the geometry you are currently drawing. Do this before drawing anything; retrofitting connectivity is the expensive way.
3. Draw one polygon for the park or block boundary, tracing the imagery. Place vertices at corners, not evenly; straight edges need only two.
4. Draw three path or sidewalk lines that start or end on the boundary. As each endpoint approaches the boundary, watch for the snap cue and let it grab the edge. Fight the urge to freehand it.
5. Draw two more polygons for interior areas (a playground, a pond, a lawn section) that share an edge with the boundary or with each other. Where edges are shared, snap vertex to vertex along the common stretch.
6. Add four point features for prominent trees, snapped onto or beside the paths as reality dictates. That makes ten features.
7. Audit your own connectivity: zoom in very close on every junction where two features should touch. At working scale everything looks connected; at close zoom, unsnapped work shows gaps and overshoots. Fix any you find by editing the offending vertex and letting it snap.

Success criteria: Ten features exist; at maximum zoom, every intended junction shows coincident geometry with no visible gap or crossing; you can point at one junction where you saw the snap cue engage and can say what it snapped to.

Stretch goal: Turn snapping off and draw one extra line connecting two features, as carefully as you can. Zoom in on its endpoints and compare with your snapped work. Keep the bad line; Exercise 4 wants it.

Exercise 4: Break the data, then find the breaks

Objective: Deliberately introduce duplicate features and sliver gaps, then detect all of them using inspection techniques rather than memory.

What you need: The map from Exercise 3 and the CSV from Exercise 1. All free. Rule-based topology validation is ArcGIS Pro territory (Compendium Chapter 14, Data Quality, covers it); here you do the manual version, which is also the version you can do anywhere.

Steps:

1. Break things first, and write each break on paper as you make it; the list is your answer key. Copy one row of your tree CSV and paste it as a new row with a new `tree_id`: a duplicate record at identical coordinates. Then add another row whose coordinates differ from an existing tree's by a tiny amount in the last decimal place: a near-duplicate, the kind field crews create by collecting the same tree twice.
2. In the sketch layer, draw a second polygon nearly on top of an existing interior polygon but offset slightly, leaving a thin gap along one edge and a thin overlap along another. Those thin shapes are slivers, and they are what unsnapped digitizing produces at scale.
3. Duplicate one line feature exactly on top of itself by drawing it again with snapping to the original, vertex by vertex. Perfectly coincident duplicates are invisible at every zoom level, which is what makes them dangerous.
4. Set the answer key aside. Now hunt as if you did not know. For the CSV duplicates: re-add the file, open its attribute table, and sort by each meaningful column in turn; exact duplicates land adjacent when sorted. Near-duplicates need a spatial check: zoom to each cluster of points and count features against visible trees.
5. For slivers: make the polygons semi-transparent in the layer styling so overlaps render darker, then pan the whole area slowly at close zoom looking for dark seams and hairline gaps of background color between shapes that should share an edge.
6. For the coincident duplicate line: click features to select them and watch the pop-up or selection indicator; two stacked features announce themselves when a single click returns more than one result. Dragging one aside temporarily also reveals its twin; drag it back or undo after.
7. Reconcile against the answer key. Anything you planted but failed to find tells you which detection technique you need to practice. Fix every defect: delete duplicates, and repair slivers by re-snapping the offset polygon's vertices to the neighbor.

Success criteria: Your hunt list matches your answer key exactly, every planted defect is repaired, and re-running the transparency pass and table sort turns up nothing new.

Stretch goal: If you have ArcGIS Pro, load the same features into a file geodatabase, build a topology with a no-gaps and a no-overlaps rule, and validate. Compare what the rules caught against what your eyes caught.

Exercise 5: Run a mini QA review loop

Objective: Put everything from Exercises 1 through 4 through one formal review cycle: checklist, findings log, fixes, and a verification pass by a second set of eyes.

What you need: Everything you have built so far, a spreadsheet for the QA log, and ideally one other person with a free account; a solo alternative is built into the steps.

Steps:

1. Write a QA checklist of eight to ten binary checks derived from this workbook: every point falls on a visible tree in imagery; no duplicate `tree_id` values; every domain-constrained value is from the domain list; every junction is snapped at close zoom; no slivers under transparency; no empty required fields; notes are intelligible to a stranger. Binary means each check is pass or fail, never "mostly."
2. Build the QA log spreadsheet with columns for check name, feature or row affected, finding, severity, status (open, fixed, verified), and reviewer initials. This little table is the whole machinery of Compendium Chapter 14's QA discipline at hobby scale.
3. Run the checklist yourself against every layer and table. Log each failure as a row; do not fix anything mid-review, because fixing while reviewing is how reviews lose their place. Expect findings; a review that finds nothing usually reviewed nothing.
4. Fix pass: work the log top to bottom, repair each finding in the data, and flip its status to fixed with a one-line note on what you changed.
5. Verification pass: share the map publicly or hand your files over, and have your second person re-run only the checks that had findings, flipping fixed to verified when they agree. Solo alternative: wait at least a day, then re-run those checks yourself; the delay is what makes you a different reviewer.
6. Close the loop with three sentences at the bottom of the log: which check caught the most, which defect escaped your Exercise 4 hunt only to surface here, and what

schema or digitizing change would prevent the top finding class entirely. That last sentence, defect to process change, is the entire point of QA.

Success criteria: A completed QA log where every finding reached verified status, and a three-sentence retrospective that names at least one upstream change rather than blaming individual mistakes.

Stretch goal: Run the loop a second time a week later after adding five new trees without looking at your checklist while collecting. Count findings per feature for the new batch versus the old. If the number dropped, the process improved you; that is the metric that matters.

Where to go next

You have now touched every stage Volume C describes: creation paths, structure and constraints, editing mechanics, and quality practice. The natural continuation is publishing this inventory as an editable hosted feature layer for a field device: Compendium Chapter 10 (Hosted Feature Layers) plus Chapter 30 (Field Maps, Survey123, QuickCapture), with the full worked pipeline in Compendium Chapter 37 (Worked Project - Field Collection).

Workbook: Volume D - Spatial Analysis

These five exercises put the tools from Volume D (Compendium Chapters 16 through 20) into your hands. One honest note before you start: most spatial analysis tools in ArcGIS Online sit behind an organizational account with analysis privileges, and several of them consume credits (Compendium Chapter 2, The ArcGIS Ecosystem). Every exercise below tells you up front which parts need those privileges and gives you a free path: either a manual version built on the Map Viewer measurement and sketch tools, or a paper version where you reason through the same logic. The manual versions are not consolation prizes. Working a buffer question by hand teaches you what the tool actually computes better than clicking Run ever will.

For all five exercises, a free ArcGIS public account is enough to open Map Viewer, add Living Atlas layers, style them, measure, sketch, and save a map. Finding good layers is covered in Compendium Chapter 5 (Finding Data); Map Viewer mechanics are Chapter 6.

Exercise 1: One question, answered end to end

Objective: Answer a concrete proximity question, "How many schools in my county are within one mile of a major highway?", using Buffer and Summarize Within, and report the answer in a sentence a non-GIS person can trust.

What you need: Map Viewer and two Living Atlas layers: a point layer of schools and a line layer of major roads or freeways covering your area (search the Living Atlas for both; national coverage exists for the United States, and most countries have equivalents). The tool path needs an org account with spatial analysis privileges and a small credit cost. The free path needs only the measurement tool.

Steps:

1. Frame the question before touching any tool. Decide what counts as a school (the dataset decides for you; read its item page), what counts as a major highway (pick a value in the road-class field), and confirm you mean straight-line distance, not driving distance. Write the question down in one sentence.
2. Add both layers in Map Viewer and filter each to your county or state using the layer's filter settings. A smaller processing area means faster runs and lower credit use.
3. Tool path: open `Analysis > Tools` and run the buffer tool (Create Buffers) on the filtered roads layer at one mile, with the dissolve option on. Dissolving matters: without it, overlapping buffers around adjacent road segments would let one school be counted several times.
4. Run Summarize Within with the dissolved buffer as the summary area and the schools layer as the features to summarize. A plain count is all you need; skip the statistics options.
5. Open the result's table or pop-up, read the count, and write the answer sentence with its caveats: name the data source, its vintage, your definition of "major

highway," and the fact that one mile means straight-line distance. Compendium Chapter 16 (Proximity and Overlay) explains why each caveat is load-bearing.

Free path: shrink the question to one town. Add the same two layers, open the measurement tool from the map tools, and check each school in town against the nearest highway by measuring. Tally by hand. Same logic, smaller extent, zero credits.

Success criteria: The buffer draws as a continuous ribbon along the roads, not a pile of stacked shapes. The count survives a spot check: pick three schools sitting near the buffer edge, measure each one's distance to the road, and confirm the tool classified them the way your measurements say it should. Your written answer names the dataset and the distance.

Stretch goal: Rerun at half a mile and at two miles. The count will not scale linearly with distance. Explain why in two sentences, using what Chapter 16 says about how buffered area grows and how features cluster along corridors.

Exercise 2: One spatial join, three answers

Objective: Run the same join with different match options and explain, for one specific county, why the resulting counts differ.

What you need: A county boundaries polygon layer and a rivers or streams line layer from the Living Atlas, filtered to your state. The tool path uses Join Features, which needs analysis privileges. The paper alternative needs a pencil.

Steps:

1. Add both layers and filter the counties to your state. Look at the map first and pick out two or three rivers that visibly cross county lines. Predict which counties will be sensitive to the match option before you run anything.
2. Run Join Features with counties as the target, rivers as the join layer, the spatial relationship set to intersects, and a one-to-one join that produces a count of joined features per county.
3. Run it again identically, except set the spatial relationship to completely within.
4. Style both result layers by their count field and put them side by side (duplicate the map in a second browser tab if that is easier). The intersects counts will be greater

than or equal to the completely-within counts everywhere.

5. Pick one county where the gap is large and write the explanation: a river that crosses the county boundary is counted under intersects but excluded under completely within, because no part of the test asks how much of the river is inside. Chapter 16 covers the full menu of spatial relationships.

Paper alternative (no privileges needed): Draw three adjacent counties and two rivers, one contained entirely in a county and one crossing all three. Build the count table for intersects and for completely within by hand. The whole lesson fits on an index card.

Success criteria: Intersects counts are never smaller than completely-within counts, and you can name one specific river responsible for the difference in one specific county.

Stretch goal: Rerun the intersects version as a one-to-many join and look at the result's table. Explain what happened to the number of rows and why one-to-many output is the wrong input for a choropleth map but the right input for a table of county-river pairs.

Exercise 3: Hot spots you can defend

Objective: Produce a hot-spot analysis and write an interpretation paragraph that claims only what the statistics actually support.

What you need: A polygon layer carrying both a count-style field and a population field; the American Community Survey layers in the Living Atlas work well (tract or county level). The tool path uses Find Hot Spots, which needs analysis privileges and credits. The free path uses only smart mapping.

Steps:

1. Before running anything, decide whether your question is about counts or rates. This is the single biggest honesty decision in the exercise: raw counts of almost any human phenomenon cluster where people are, so a hot-spot map of counts is usually just a population map wearing a costume. Chapter 17 (Statistical and Pattern Analysis) treats this in depth.

2. Tool path: run Find Hot Spots on the raw count field. Then run it again on a rate, either by using the tool's option to divide by a population field where the interface offers one, or by analyzing a pre-computed rate field in the data.
3. Compare the two results. Note where hot spots moved, shrank, or vanished once population was accounted for.
4. Write one interpretation paragraph containing all four of these elements: (a) what was measured, at what unit of aggregation, from what vintage of data; (b) what "statistically significant hot spot" means, roughly that clustering this strong is unlikely to appear by random chance at the stated confidence, and explicitly not that anything caused anything; (c) whether you analyzed a count or a rate and why; (d) one plausible alternative explanation for the pattern that your analysis cannot rule out.

Free path (no privileges): You cannot compute the statistic, but the core lesson survives. Style the count field and the rate field with smart mapping (Chapter 7) in two copies of the map and compare where each draws the eye. Then write the same four-part paragraph about the visual pattern, replacing the significance language with an honest statement that no significance test was run.

Success criteria: The count map and the rate map disagree somewhere, and you can say why. Your paragraph contains zero causal language, states the aggregation unit, and includes at least one alternative explanation.

Stretch goal: Rerun (or re-style) at a different aggregation level, counties instead of tracts, and note where conclusions change. You have just demonstrated the modifiable areal unit problem; name it, and check its entry in Compendium Chapter 40 (Mega-Glossary).

Exercise 4: Drive time versus the circle

Objective: Show, for a real location, where a straight-line buffer overstates access, and quantify the gap.

What you need: A point of interest near a barrier: a fire station, library, or grocery store close to a river, rail corridor, or limited-access highway. The tool path needs an org account with network analysis privileges, because travel areas call a network

service and consume credits (Chapter 2). The free manual path needs only the measurement tool and is genuinely the more instructive version.

Steps (tool path):

1. Search for or sketch your point in Map Viewer.
2. Create a straight-line buffer around it at a fixed distance.
3. Run the travel-area tool (named along the lines of Generate Travel Areas or Create Drive-Time Areas depending on interface) around the same point, set to driving distance at the same distance value.
4. Overlay the two shapes. Find places that are inside the circle but outside the travel area, and name the barrier responsible for each. Compendium Chapter 18 (Network Analysis) explains why the network result is jagged and asymmetric.
5. Compare the two areas from the result attributes and state the ratio in a sentence.

Steps (free manual path):

1. Using the measurement tool, mark spots roughly one mile from your point in four to six compass directions, deliberately including at least one across the barrier.
2. For each spot, trace the actual road route from your point with the measurement tool and record the road distance.
3. Build a small table: direction, straight-line distance, road distance, ratio. The directions with the highest ratios point at your barriers.

Success criteria: You found at least two locations where road distance is more than double the straight-line distance, and you can name the barrier for each. Your write-up states which question each shape answers: the circle answers "as the crow flies," the travel area answers "as the car drives."

Stretch goal: Reason about time instead of distance. A location across a freeway interchange can be close by road distance yet slow at rush hour. Write two sentences on when you would insist on a time-based travel area with traffic, using Chapter 18's treatment of travel modes.

Exercise 5: Weighted overlay on paper

Objective: Design and execute a small weighted suitability model by hand, so you understand exactly what the raster overlay tools in Compendium Chapter 19 automate.

What you need: Paper or a spreadsheet, plus Map Viewer with the imagery basemap and the measurement tool. No analysis privileges, no Pro, no credits. If licensing has blocked you out of every tool so far, this exercise is your equalizer: it is the full suitability workflow with your brain as the geoprocessor.

Steps:

1. State a siting goal in one sentence: for example, choose the best location for a weekend farmers market in your town. Pick four criteria you can observe from a web map, such as distance to public parking, visibility from a main street, flat open ground, and distance from an existing competing market.
2. For each criterion, define a 1-to-9 scoring scale before you look at any candidate site. Write down exactly what earns a 9 and what earns a 1 (for parking: a 9 might be a public lot within a short walk, a 1 no parking within several blocks). This step is reclassification, in raster terms, and doing it blind to the candidates keeps you honest.
3. Assign each criterion a weight, with the four weights summing to 100, and justify each weight in one sentence.
4. Pick three or four real candidate sites in your town. Score each site on each criterion by inspecting the imagery basemap and measuring distances in Map Viewer.
5. Multiply scores by weights, sum per site, and rank the sites.
6. Run a sensitivity test: move ten points of weight from your heaviest criterion to your lightest and recompute. Note whether the winner changed.

Success criteria: A complete table of sites by criteria with scores, weights, and totals; a declared winner; and one sentence stating whether the ranking survived the sensitivity test. If the winner flips on a modest weight change, your honest conclusion is "it depends on priorities," which is a real and useful finding.

Stretch goal: Identify the criterion that was hardest to score consistently and rewrite its scale so two different people would assign the same score. If you have access to

ArcGIS Pro, rebuild the model with distance rasters, reclassification, and weighted overlay following Chapter 19, and compare the software's answer to your paper one.

Where this leaves you

You have now run, or hand-simulated, the four workhorse patterns of Volume D: buffer-and-summarize, spatial join, statistical pattern detection, and multi-criteria overlay, plus the network-versus-Euclidean distinction that separates careful proximity work from lazy circles. The worked projects in Compendium Chapters 36 and 37 chain these same moves into complete deliverables; the recipes in Chapter 38 give you compressed versions to adapt. When a result looks wrong, start with Chapter 39 (Troubleshooting Encyclopedia) before assuming the tool lied. It almost never did; the inputs usually did.

Workbook: Volume E - ArcGIS Pro Deep Dive

This workbook turns Volume E into hands-on practice. Every exercise here requires ArcGIS Pro on a Windows machine — there is no browser workaround for the desktop application itself. If you do not have Pro, you have two honest options: get a time-limited free trial or a low-cost personal-use license (Compendium Chapter 2, The ArcGIS Ecosystem, explains the licensing paths), or read along. Each exercise includes a read-along alternative that tells you what to watch for so the material still sticks.

No paid data is used anywhere. Everything comes from the Living Atlas (Compendium Chapter 5, Finding Data) or from data you create during the exercise. Work through the exercises in order — Exercise 2 reuses the project from Exercise 1, and Exercises 4 and 5 reuse the map from Exercise 3.

Exercise 1: The project setup discipline drill

Objective: Build a Pro project whose structure you could hand to a stranger — or to yourself in six months — without an apology email.

What you need: ArcGIS Pro with any license level. No org account needed; you can sign in with a Pro license and still complete this exercise entirely locally. *Read-along alternative:* study the project anatomy section of Compendium Chapter 21 (Pro Interface and Projects) and sketch on paper what folders and files a new project creates.

Steps:

1. Before opening Pro, create a folder on disk where this project will live — something like a `GIS-Practice` folder with a subfolder for this workbook. The discipline starts outside the software: projects that get dumped into default locations are projects that get lost.
2. Launch Pro and create a new project from the Map template. Point it at the folder you just made, name it something descriptive rather than `MyProject1`, and leave "create a folder for this project" checked so everything nests cleanly.
3. Open the Catalog pane (`View > Catalog Pane`) and inspect what Pro made for you: a project file, a default file geodatabase, and a default toolbox. Right-click each and read its properties. Understand that the geodatabase is where your outputs will land unless you say otherwise — this default matters constantly in Exercise 2.
4. Add a folder connection (`Insert > Connections > Folder > Add Folder Connection`) to one other location you actually use, and deliberately do not add connections to your entire drive. A project with three meaningful connections is navigable; one with fifteen is noise.
5. Rename the default map to describe its contents, not its existence — "Practice Area Overview," not "Map." Do the same for anything else you create from now on. Add one Living Atlas layer (`Map > Add Data`, then browse the Living Atlas section) so the map is not empty.
6. Save the project, close Pro, then move the entire project folder to a different location on disk and reopen the project from there. Watch what survives the move and what breaks.

Success criteria: The reopened project loads with no broken data sources (red exclamation marks) for anything stored inside the project folder. You can explain, without looking, where a new geoprocessing output would land by default and why.

Stretch goal: Create a second project from the Catalog template instead of the Map template, compare what each template gives you, and decide which you would make your personal default. Then explore `Project > Options` and set your default project location so future projects start disciplined automatically.

Exercise 2: Geoprocessing by hand, then rebuilt in ModelBuilder

Objective: Run a four-tool geoprocessing chain manually, then rebuild it as a reusable model and prove both produce identical results.

What you need: The project from Exercise 1, plus two Living Atlas layers that overlap: one polygon layer (a counties or administrative-boundaries layer works well) and one line layer (rivers, railroads, or major roads). Any overlapping polygon-and-line pair works — the chain is the point, not the data. *Read-along alternative:* Compendium Chapter 22 (Geoprocessing in Depth) walks the same tools conceptually, and Chapter 25 (ModelBuilder) shows a finished model; trace the data flow with a pencil.

Steps:

1. Add both Living Atlas layers to your map. Pick one state or region as your study area. Select its polygons using `Map > Selection > Select By Attributes`, then export the selection to your project geodatabase (right-click the layer, then look for the export or copy-features option). Working from a local copy keeps the rest of the chain fast and keeps you from hammering a shared service — Compendium Chapter 10 (Hosted Feature Layers) explains why that courtesy matters.
2. Run the chain by hand from the Geoprocessing pane (`Analysis > Tools`), letting each output land in the project geodatabase: first Dissolve your exported polygons into a single study-area boundary; second, Clip the line layer to that boundary; third, Buffer the clipped lines at a distance that makes sense for your data; fourth, use a select-by-location or an overlay tool to find which of your original polygons intersect the buffer. Compendium Chapter 16 (Proximity and Overlay) covers what these tools actually compute — do not re-derive that here, just run them deliberately.
3. After each tool, open its entry in the geoprocessing history (`Analysis > History`) and read the parameters it recorded. This history is your lab notebook; the whole exercise fails silently if you skip it.

4. Now create a new model in your project toolbox (right-click the toolbox in Catalog, then create a new model). Drag the same four tools into the model, chain outputs to inputs, and set the study-area selection as a model parameter so the model can be rerun for a different region.
5. Validate and run the model. Let its outputs land beside your manual outputs with distinct names.

Steps continued — the comparison: Open the attribute tables of the manual final output and the model final output side by side.

Success criteria: Feature counts match exactly between the manual run and the model run, and rerunning the model with a different region parameter completes without edits to the model itself. If counts differ, the history pane will show you which parameter diverged — that diagnosis is the real lesson.

Stretch goal: Expose the buffer distance as a second model parameter, then run the model as a geoprocessing tool from the pane rather than from the model canvas. If you want to go further, Compendium Chapter 31 (Python for ArcGIS) shows how to export the same chain to a Python snippet from the history pane.

Exercise 3: The symbology recreation challenge

Objective: Reproduce another cartographer's map styling closely enough that a side-by-side comparison takes effort to tell apart.

What you need: Pro, plus a reference map: browse the Living Atlas in a web browser and pick a published web map with a graduated-color (choropleth) polygon style you like — demographic maps are plentiful and free. Keep it open in the browser as your answer key. Add the same or a similar polygon layer to a new map in your Pro project. *Read-along alternative:* Compendium Chapter 23 (Symbology and Labeling) with the reference web map open beside it; for each styling choice in the web map, find the Pro control that would produce it.

Steps:

1. Study the reference map before touching anything. Write down what you observe: the field being mapped, roughly how many classes, whether the breaks look equal-

interval or quantile-like, the color ramp's character, outline treatment, transparency, and how labels behave as you zoom. Reverse-engineering the spec is most of the skill.

2. In Pro, open the Symbology pane for your layer and choose graduated colors. Match the field, the classification method, and the class count to your observations. Compendium Chapter 7 (Styling and Smart Mapping) covers the web-side equivalents if you want to compare vocabularies.
3. Match the color ramp. If no built-in ramp matches, edit individual class colors — use the browser's zoom or a color-picker utility to sample the reference map's actual colors rather than trusting your memory.
4. Match the polygon outlines and any transparency. Outlines are where most recreations visibly fail: check color, width, and whether the reference uses no outline at all.
5. Add labels and match their font character, size behavior, halo, and placement as closely as Pro's labeling controls allow.
6. Zoom your Pro map to the same extent as the reference, put the two windows side by side, and hunt the differences like a spot-the-difference puzzle. Fix what you find.

Success criteria: At matching extents, a person glancing at both maps for five seconds cannot immediately say which is which. Class breaks land on the same values or defensibly close ones, and you can articulate every deliberate deviation you kept.

Stretch goal: Save your recreated symbology as a style or layer file so it is reusable, then restyle the same layer with an unclassed color scheme and write two sentences on which communicates the data more honestly — Compendium Chapter 4 (Cartographic Design) gives you the evaluation vocabulary.

Exercise 4: A two-page layout with a map series

Objective: Build a print-quality layout with a main map and an overview locator, then convert it into a map series that generates a page per feature automatically.

What you need: Pro and the styled map from Exercise 3, whose polygon layer will double as the map series index. *Read-along alternative:* Compendium Chapter 24

(Layouts and Map Series) covers every element used here; sketch the two-frame layout on paper and annotate which properties each element needs.

Steps:

1. Create a new layout (`Insert > New Layout`) at a standard page size in landscape. Add a map frame covering roughly the right two-thirds of the page, pointed at your Exercise 3 map.
2. Add a second, small map frame in an upper corner as an overview locator. Point it at the same map but zoom it well out, and add an extent indicator to the locator frame so it draws a box showing where the main frame is looking. This two-frame pattern — detail plus context — is the workhorse of report cartography.
3. Furnish the page: title, legend, scale bar, north arrow if the projection justifies one (Compendium Chapter 3 explains when it does not), and a small credits block naming your data sources. Position elements with guides rather than by eye.
4. Enable a spatial map series in the layout properties, using your polygon layer as the index layer. Set the title text to draw from an attribute field dynamically, so each page names its own feature.
5. Page through the series using the series controls and watch what updates automatically (extent, dynamic title) versus what stays fixed (legend, credits). Fix anything that breaks on odd-shaped features — a long thin polygon will stress-test your margins.
6. Verify at least two consecutive pages look presentation-ready, since those two pages are the deliverable Exercise 5 will export.

Success criteria: Paging through the series changes the main frame's extent and the title with zero manual edits, the locator's extent indicator tracks correctly on every page, and no element overlaps or clips on the two pages you chose.

Stretch goal: Add dynamic text that reports an attribute value from the current index feature (a population figure, an area), so each page carries its own statistic. Then adjust the series' extent behavior — the margin or scale rounding around each feature — and compare fixed-scale pages against best-fit pages.

Exercise 5: The export settings shoot-out

Objective: Export the same layout under several settings profiles and learn, from evidence, which settings matter for size, quality, and downstream use.

What you need: Pro and the finished layout from Exercise 4. A PDF reader and any image viewer for inspection. *Read-along alternative:* the export section of Compendium Chapter 24, plus this exercise's comparison table drawn from its description — predict the winners before reading on.

Steps:

1. Create an exports subfolder inside your project folder so the comparison files stay together and named consistently: include the format and the key setting in each filename.
2. From `Share > Export Layout`, export the current page as a PDF at a deliberately low resolution setting, then again at a high one. Keep every other setting identical.
3. Export a third PDF with vector output disabled (rasterized), if your export dialog offers the vector/raster choice, at the high resolution. This isolates the single setting that most changes PDF behavior.
4. Export the same page as a PNG at the same high resolution, and once more as a JPEG. Now you have five files representing the realistic decision space.
5. Compare file sizes in your file browser first. Then open each and inspect three things: zoom deep into label text (does it stay crisp or pixelate?), try selecting text in the PDFs (vector PDFs keep selectable text; rasterized ones do not), and look at flat color areas in the JPEG for compression smudging.
6. Write a three-line decision rule for yourself: which settings for print, which for email, which for embedding in a slide deck. If your organization ever asks for "a high-res map," you now know which questions to ask back.

Success criteria: You can point at the largest and smallest files and explain why each landed there, and you can demonstrate one export where text stays selectable and one where it does not.

Stretch goal: Use the map series export options to export both of your Exercise 4 pages into a single multi-page PDF, then compare its size against the sum of two single-page

exports. Then export one page at a physically larger page size versus a higher resolution and decide which lever actually bought you more usable detail.

Where to go next

If these five exercises felt comfortable, the two worked projects in Compendium Chapters 36 and 37 chain these same skills into end-to-end builds, and Chapter 38 (Cookbook) gives you forty more reps in smaller bites. If ModelBuilder was the exercise that clicked, Chapter 31 (Python for ArcGIS) is the natural escalation.

Workbook: Volume F — Apps and Field Operations

These five exercises put the Volume F chapters to work: Instant Apps (Compendium Chapter 26), StoryMaps (Chapter 27), Dashboards (Chapter 28), and the field apps (Chapter 30). Exercises 1 and 2 are fully doable with a free public ArcGIS account. Exercises 3 and 4 need an organizational account — each includes a read-along alternative so you still get the reasoning practice without one. Exercise 5 needs nothing but this page. Before you start, build one reusable asset: a web map you will carry through the first three exercises. If you have not made a web map before, work through Compendium Chapter 6 (Map Viewer Complete Reference) first.

Exercise 1: One map, three Instant Apps — judge the fit

Objective: Publish the same web map through three different Instant Apps templates and articulate, in writing, which template fits which audience and why.

What you need: A free public ArcGIS account. A web map you build in this exercise from a Living Atlas layer — no uploads, no publishing privileges. If a particular template is not offered to your account type, read along for that one and configure the other two.

Steps:

1. Sign in to ArcGIS Online and open Map Viewer. Add one Living Atlas layer with rich attributes — a demographic boundary layer, a parks layer, or a hazards layer all work. Pick something where individual features are worth clicking on.
2. Configure a decent pop-up for the layer (title from an attribute, a handful of meaningful fields — see Compendium Chapter 9 for pop-up craft) and save the map with a clear name. This map is your constant; only the app wrapper will change.
3. From the map's item page or from Map Viewer, choose `Create app > Instant Apps`. Browse the template gallery and read the stated purpose of each template before touching anything. Note how the gallery describes each one in terms of audience task, not features.
4. Create three apps from the same map: one **Media Map** (or the most minimal "just show the map" template available to you), one **Sidebar** (a template with an exploration panel, legend, and tools), and one **Nearby** or similar search-first template (built around "find things close to a location").
5. For each, accept the express-setup defaults first, preview it, and only then toggle two or three settings that seem to matter. Resist deep configuration — the point is to feel each template's opinion, not to fight it.
6. Open all three published apps side by side in browser tabs. For each, write two sentences: who is this for, and what task does it make easy? Then write one sentence about what each template makes *hard*.

Success criteria: You have three working app URLs driven by one web map. Your notes name a distinct primary audience for each template, and you can state one task where each template beats the other two. If your three write-ups sound interchangeable, you configured too much and homogenized them — reset one to defaults and look again.

Stretch goal: Pick the template you judged the worst fit for your map and try to redeem it: change the map (not just app settings) until that template works. Notice how much of "app fit" is really "map design fit" — a lesson Chapter 26 states and this exercise proves.

Exercise 2: A five-block StoryMap with one sidecar and map actions

Objective: Build a short, complete story that uses exactly five block types, including one sidecar with working map actions.

What you need: A free public ArcGIS account (sign in at the ArcGIS StoryMaps site). The web map from Exercise 1. One or two images you own or that are openly licensed.

Steps:

1. Open ArcGIS StoryMaps and start a new story. Write a real title and a one-line subtitle in the cover — treat it as the promise the story must keep, per Compendium Chapter 27's guidance on story structure.
2. Add block one: a **text** block of two or three short paragraphs framing a question your map can answer ("Where are the gaps in park access?" beats "This is a map of parks").
3. Add block two: an **image** block with a caption and alt text. The image should earn its place — a photo of the subject, not decoration.
4. Add block three: a **map** block embedding your Exercise 1 web map at a deliberate extent. Set the extent so the reader's first view answers "where are we?"
5. Add block four: a **sidecar**. Choose the docked layout. Build three slides: each slide pairs a short narrative panel with the same web map at a different extent or with different layers visible. This is the spine of the story — spend most of your time here.
6. Inside one sidecar narrative panel, add **map actions**: select a phrase in your text and attach an action that moves the map to a specific place, or add action buttons that toggle layer visibility. Test that clicking the text changes the map beside it.
7. Add block five: a closing **text** block (or a quote block) that answers the opening question in one or two sentences. Preview the whole story on a narrow window to check the mobile reading experience, then publish (share to Everyone if you want a public link).

Success criteria: The published story reads top to bottom in a few minutes with no dead blocks. Every map action, when clicked, visibly changes the map. A friend who knows nothing about GIS can tell you what the story's point was — if they describe the *map* instead of the *point*, your text blocks are captions, not narrative.

Stretch goal: Duplicate the story and rebuild the sidecar in the floating layout instead of docked. Compare scroll feel on a phone. Write one sentence on when you would choose each layout.

Exercise 3: A dashboard with two wired selectors

Objective: Build a dashboard where two selectors filter the map and at least one data element, so a viewer can slice the data without touching the map.

What you need: An organizational account with app creation privileges — public accounts cannot author dashboards. A layer with at least one categorical field and one numeric or date field; a Living Atlas feature layer works, or reuse your Exercise 1 map. **No org account?** Read the steps anyway, then open any public dashboard you can find (search ArcGIS Online for dashboards shared publicly), interact with its selectors, and reverse-engineer which element is wired to which — write down the action mapping you infer. That analysis is most of the learning.

Steps:

1. Create a new dashboard from the app launcher or via `Create app > Dashboards` from your web map's item page. Add the map as the first element.
2. Add two data elements that summarize the layer: an indicator (a single big number, such as a count or a sum) and a chart (serial or pie, grouped by your categorical field). Confirm both show sensible values before wiring anything — debugging data and actions at the same time is misery (Compendium Chapter 28 covers element configuration in depth).
3. Open the dashboard header or side panel settings and add your first selector: a **category selector** driven by the categorical field (grouped values, not static lists, so it stays current as data changes).
4. In that selector's **Actions** settings, wire it to filter the map layer, the indicator, and the chart. This is the core skill: a selector does nothing until you explicitly tell it what to filter.
5. Add a second selector of a different kind — a number selector on your numeric field, or a date selector if the layer has dates. Wire it to the same targets.
6. Test the combination logic: set both selectors at once and verify the elements reflect *both* conditions. Then clear each selector and confirm everything returns to the unfiltered state.
7. Check the layout at a laptop-width window: selectors visible without scrolling, map dominant, numbers legible from arm's length.

Success criteria: Changing either selector visibly updates the map, the indicator, and the chart; using both together narrows results further; clearing them restores the full view. If an element ignores a selector, the wiring gap is in that selector's action targets — not in the element.

Stretch goal: Add a list element wired so that *selecting a row in the list* flashes and pans to that feature on the map. You have now wired element-to-element actions in both directions: control-to-data and data-to-map.

Exercise 4: A Field Maps form with conditional visibility, tested on a phone

Objective: Build an editable layer with a smart form where one question appears only when a previous answer requires it, and prove it works in the Field Maps mobile app.

What you need: An organizational account that can publish hosted feature layers and use Field Maps Designer, plus a phone with the ArcGIS Field Maps app installed and signed in to the same account. **No org account?** Read the steps, then on paper design the schema and visibility rule for this scenario: a tree-inspection form where "Pest observed?" (yes/no) controls whether "Pest type" and "Photo of damage" appear. Write the fields, domains, and the condition in plain English — that design is the transferable skill.

Steps:

1. Create a new editable point layer (in Map Viewer or via Field Maps Designer's new-layer flow). Give it these fields: a category field with a domain (for example, issue type: Pothole / Streetlight / Graffiti / Other), a text field named something like "Other description", and a severity field. Domains are doing real work here — see Compendium Chapter 12 (Schema Design) for why coded values beat free text in the field.
2. Open the layer's map in **Field Maps Designer** (from the app launcher, or from the map's item page) and go to the form builder.
3. Drag your fields into the form. Set sensible input types: the domain field becomes a choice list automatically; make severity a choice list too.
4. Select the "Other description" element and add a **conditional visibility** expression: the element is visible only when the issue-type field equals "Other". The builder lets

you define this as a simple condition; under the hood it is an Arcade expression (Compendium Chapter 8), and you can open the expression editor to see what got generated.

5. Save the form. On your phone, open Field Maps, find the map, and start collecting a new point. Walk the form: pick "Pothole" and confirm the description question is absent; switch to "Other" and watch it appear; switch back and confirm any entered text handling behaves the way you expect.
6. Submit one test record, then verify it back at your desk: open the layer's data in Map Viewer and confirm the attributes arrived intact.

Success criteria: The conditional field appears and disappears on the phone exactly per your rule, and a submitted record shows correct attributes on the desktop. If the form on the phone looks stale after edits, pull down to refresh the map in Field Maps — designers commonly forget the mobile app caches the form.

Stretch goal: Add a second condition: make severity required only when issue type is "Pothole". Then test the failure path — try to submit a pothole with no severity and confirm the form blocks you with a comprehensible message.

Exercise 5: Choose the right field app — five scenarios

Objective: For five realistic field-data scenarios, choose Field Maps, Survey123, or QuickCapture, and defend each choice in two or three sentences.

What you need: Nothing but Compendium Chapter 30 fresh in your mind. No account required.

Steps:

1. Read each scenario. Before writing, identify the deciding factors: is the work map-centric or form-centric? Are assets revisited or captured once? Is speed of capture the constraint? Who are the users and how much training will they get?
2. **Scenario A:** A utility crew inspects the same hydrants every quarter, navigating to each one and updating its condition record.
3. **Scenario B:** A public health team interviews residents door-to-door using a long, branching questionnaire; the location is incidental, captured once per interview.

4. **Scenario C:** A volunteer in a moving vehicle logs roadkill sightings along rural highways — one tap per observation, no stopping, no typing.
5. **Scenario D:** After a storm, untrained community volunteers report damaged structures; the org wants photos, a short form, and submissions from people who will never install training or sign in.
6. **Scenario E:** An environmental technician monitors stream sites: navigates to fixed stations, records water readings on a structured form, and occasionally adds a brand-new station.
7. For each, write your pick and justification, then compare against the reasoning below. Disagreement is fine if your justification holds — several scenarios have a defensible second choice.

Success criteria: Check yourself: **A — Field Maps** (asset-centric, revisit-and-update, navigation to existing features is the core motion). **B — Survey123** (form-first with branching logic; the survey is the product, the point is metadata). **C — QuickCapture** (single-tap capture at speed; any form would be a safety hazard). **D — Survey123 via a public link** (no sign-in, no app training, works in a browser — the sharing model decides it, not the form). **E — Field Maps** (map-centric navigation to stations plus structured editing; a strong second answer is Field Maps launching a linked Survey123 form for the readings — if you said that, you have understood the chapter better than the exercise). You pass if at least four picks match and every justification names a deciding factor rather than a feature list.

Stretch goal: Write a sixth scenario of your own where the honest answer is "two apps working together," and specify which app hands off to which, and what data travels across the handoff.

Workbook: Volume G - Automation and Administration

Volume G is where ArcGIS stops being an application and starts being a system: REST services you can read like documents, Python commands you can capture and replay, and administrative decisions about who gets what. This workbook gives you five

exercises to practice all three. Two run entirely in a web browser with no account at all. One needs ArcGIS Pro, with a full read-along alternative if you do not have it. Two are paper exercises, because administration is mostly thinking, and thinking is free.

Work them in order. Exercises 1 and 2 build on each other directly.

Exercise 1: Read a feature service in the browser

Objective: Open a live feature service's REST endpoint in a plain browser tab and extract the facts an administrator or developer would need before trusting the service.

What you need: Any web browser and the Living Atlas website (browsing requires no sign-in — see Compendium Chapter 5, Finding Data). No account, no software.

Steps:

1. Browse the Living Atlas site, filter to feature layers, and pick a layer with familiar content — a country, state, or county boundary layer works well. Open its item page.
2. Find the service URL. On the item's overview page there is a URL section (usually along the right side) with a copy button. Copy it.
3. Paste the URL into a new browser tab. You land on the Services Directory: a plain HTML page generated by the server itself, describing the service. This page is the subject of Compendium Chapter 32 (REST Services) — here you just learn to read it.
4. Answer the service-level questions first: how many layers does this service contain, and what index number does each have? Notice the URL ends in `FeatureServer` — the layers hang off it as `/0`, `/1`, and so on.
5. Click into the first layer (or add `/0` to the URL). Now answer, from the page alone: What is the geometry type? What is the spatial reference WKID (Compendium Chapter 3 explains what that number means)? What is the maximum record count the server will return per request? Which capabilities are listed — Query only, or editing operations too?
6. Read the fields list. Identify the ObjectID field, one string field, and one numeric field, noting the declared type of each.
7. Add `?f=pjson` to the end of the layer URL and reload. Same information, now as formatted JSON — the shape a script or app would consume.

Success criteria: Without opening any map, you can state the layer's geometry type, its WKID, its per-request record limit, whether it can be edited, and the names of one text and one numeric field. You can also explain what each segment of the URL means, from the server address down to `/0`.

Stretch goal: Find a second public layer from a different publisher that does allow editing. Confirm it from the Services Directory alone: the capabilities line will include editing operations, not just Query. Note anything else that differs — record limits, field counts, whether attachments are enabled.

Exercise 2: Run a query with URL parameters

Objective: Query a feature layer using nothing but the browser address bar, controlling the filter, the returned fields, and the output format.

What you need: The layer URL from Exercise 1, or any public feature layer URL ending in `/FeatureServer/0`.

Steps:

1. Append `/query` to the layer URL and load it. You get an HTML form — the server's built-in test harness for the query operation.
2. In the form, set the where clause to `1=1` (which matches everything), set the option that returns only a count, and submit. Read the number. Then look at your address bar: every form field you touched became a URL parameter. That URL is the API call; the form just built it for you.
3. Now bypass the form. Edit the URL directly so it contains `where=1=1`, `returnCountOnly=true`, and `f=json`. Reload. Same count, machine-readable format.
4. Write a real filter against a field you identified in Exercise 1. For a string field the value takes single quotes — something in the shape of `where=STATE_NAME='Texas'`, using your layer's actual field name and a value you know exists. The browser will encode spaces and quotes for you.
5. Trim the payload: set `outFields` to just two field names, add `returnGeometry=false`, keep `f=json`. Confirm the response contains only the attributes you asked for and no geometry.

6. Swap `f=json` for `f=pjson` (pretty-printed) and then `f=html`, and note how the same request renders three ways.

Success criteria: You can hand-build one URL that returns only your chosen attributes, with no geometry, for only the features matching your filter — and you can say what every parameter in it does. Sanity check: your filtered count should be smaller than your `1=1` count.

Stretch goal: Ask the server for a statistic instead of features. Use the query form's output-statistics input to request a count or sum grouped by a field, submit, and study the URL it builds. Check first that the layer page lists statistics support among its advanced query capabilities — not every layer has it. Compendium Chapter 32 covers statistics queries in depth.

Exercise 3: Capture and modify a Python command

Objective: Convert a tool you ran with mouse clicks into a line of Python you can edit and rerun — the shortest reliable path from GIS user to GIS automator.

What you need: ArcGIS Pro (any license level) with a project containing at least one feature layer; if you have none, digitize a few points first (Compendium Chapter 11 covers every path). **This exercise requires Pro.** If you do not have it, use the read-along alternative below — do not skip the exercise.

Steps:

1. Run the Buffer tool on any layer from `Analysis > Tools`, with a distance such as 500 meters. Let it finish.
2. Open the geoprocessing history from the `Analysis` ribbon (the History button). Every tool run in this project is logged here — this is the memory that makes capture possible (Compendium Chapter 22, Geoprocessing in Depth).
3. Right-click your Buffer entry and choose `Copy Python Command`.
4. Open the Python window from the `Analysis` ribbon's Python dropdown, paste, but read before you run: the tool is now a function call in the shape `arcpy.analysis.Buffer(...)`, and every choice you made in the tool dialog is an argument.

5. Modify two arguments: change the output name so you do not collide with the first run, and change the distance to `"1 Kilometers"`. Notice the unit lives inside the distance string — that is how geoprocessing passes linear units.
6. Press Enter to run. Confirm the new layer appears in Contents, and check History: your Python run was logged too. The loop is closed — clicks become code, code becomes history, history becomes code again.

Read-along alternative (no Pro): Study this representative captured command and answer the questions below.

```
arcpy.analysis.Buffer(
    in_features="Schools",
    out_feature_class=r"C:\Projects\Demo\Demo.gdb\Schools_Buffer",
    buffer_distance_or_field="500 Meters",
    dissolve_option="ALL"
)
```

Which argument would you change to buffer a different layer? Which to write the result somewhere else? Why is the unit inside the distance string rather than a separate argument? What do you predict happens if you rerun this unchanged — and what setting governs whether the existing output is overwritten? Compendium Chapter 31 (Python for ArcGIS) answers all four; try before you look.

Success criteria: Your modified command runs without errors and produces a second, distinct output — and you can point to the argument that controls distance without opening the help.

Stretch goal: Run a second tool (for example Clip) against your buffer output, copy its Python command too, and paste both lines into the Python window in sequence so the first tool's output feeds the second tool's input. That is a two-line script — you are halfway to Compendium Chapter 31's standalone scripts and Chapter 25's models.

Exercise 4: Design user types and groups for a 12-person organization

Objective: Produce a one-page licensing and sharing plan that gives twelve fictional staff exactly the access their work requires and nothing more.

What you need: Pen and paper or a blank document, and Compendium Chapter 34 (Administration) for the current user type and role definitions. No software, no account. This is deliberately a paper exercise: administrators commit real money and real risk with these decisions, so the design habit matters more than the clicks.

The organization — Ridgeline Water District, 12 people:

- General manager: wants dashboards on a phone, never edits anything
- Office administrator: manages accounts and resets passwords
- GIS coordinator: publishes layers, builds apps, administers the org
- Two engineers: run analysis, occasionally publish results
- Four field technicians: collect and update asset data on phones
- Customer service representative: looks up addresses and parcels, read-only
- Finance officer: opens one dashboard, monthly
- Communications officer: edits StoryMaps and publishes public content

Steps:

1. For each person, write one sentence naming what they actually do with GIS: view, edit in the field, analyze, publish, or administer. Resist job-title inflation — describe the work, not the rank.
2. Assign each person a user type from the current lineup (check Chapter 34 or your organization's licensing page for current names — the roster changes over time; historically it runs from a viewer tier through field-editing tiers to full creator and professional tiers). Apply one rule ruthlessly: never assign a capability the person will not use in the next ninety days.
3. Assign each person a role (the defaults run from viewer through data editor and publisher up to administrator). Remember the constraint from Chapter 34: the user type caps which roles are available.
4. Design the groups. List every group you would create, and for each: its members, the content shared to it, and whether it is a shared-update group (members can edit each other's items). Aim for the smallest set of groups that covers the flows — a field-editing group, internal-viewing group(s), a public-content group, and whatever else the org genuinely needs.

5. Trace two flows through your design as a test. First: a field technician updates a hydrant record from a phone — which user type, role, group membership, and layer sharing make that possible (Compendium Chapters 10 and 30 cover the layer and app sides)? Second: the communications officer publishes a StoryMap to the public — and what in your design prevents a field technician from doing the same?
6. Find your single point of failure. If the GIS coordinator is unreachable, who can administer? If your answer is "no one," fix the design and note the tradeoff you accepted.

Success criteria: Every person has exactly one user type and one compatible role; nobody holds publishing rights who never publishes; both traced flows work end to end through your groups; and you can defend every above-viewer assignment in a single sentence.

Stretch goal: A summer intern joins to help with field collection: write down what you grant and what you deliberately withhold. Then Ridgeline merges with a neighboring district's 8-person team: identify which parts of your plan scale unchanged and which need redesign.

Exercise 5: Online vs Enterprise decision memo

Objective: Write a short decision memo recommending ArcGIS Online, ArcGIS Enterprise, or a hybrid for each of three scenarios, naming the deciding factor explicitly.

What you need: Compendium Chapter 35 (ArcGIS Enterprise) and Chapter 2 (The ArcGIS Ecosystem) as references; a blank document. Paper exercise — no software.

The scenarios:

- **A.** A regional land trust: five staff, no IT department, public-facing story maps, and volunteer photo collection a few weekends per year.
- **B.** An electric utility: the regulator requires the network model to stay on infrastructure the utility controls; deep integration with an on-premises asset management system; a capable dedicated IT team.
- **C.** A county government: an existing Online organization serving public maps, now facing internal demand for heavy routine imagery processing and internal-only apps

handling sensitive human-services data.

Steps:

1. For each scenario, list the constraints in two columns: pressures toward Online (no IT staff, public reach, elastic hosting, minimal maintenance) and pressures toward Enterprise (data residency, control over infrastructure, integration with internal systems, sensitive data behind the firewall).
2. Identify the deciding constraint: the single requirement that eliminates one option regardless of everything else in the column. Not every scenario has one — if a scenario is genuinely balanced, say so, because "either works, here is the cheaper-to-operate choice" is itself a legitimate finding.
3. Write each memo in five parts, half a page at most: the recommendation in one sentence; the deciding factor; what the organization gives up by this choice; the ongoing burden it accepts (qualitatively — patching, upgrades, and server administration on one side; subscription management and credit budgeting on the other, per Chapter 2); and a revisit trigger ("revisit this decision if...").
4. For scenario C specifically, evaluate the hybrid: an Online organization and an Enterprise deployment working together is a supported pattern (Chapter 35 covers collaboration between the two). If you recommend both, state which workload lives where and why.

Success criteria: Each memo leads with its recommendation, and each deciding factor is a real constraint quoted from the scenario, not a preference. The hard test: someone reading only your three deciding-factor sentences could reconstruct all three recommendations.

Stretch goal: Take your most confident recommendation and write the counter-memo arguing the opposite choice as honestly as you can. If the counter-memo turns out persuasive, your deciding factor was not actually decisive — find the one that is.

Where to go next

Exercises 1 through 3 are the raw ingredients of Compendium Chapter 31's scripting workflows and Chapter 32's service integrations; exercises 4 and 5 are the daily reality of Chapter 34. To see these skills combined under project pressure, work Compendium

Chapters 36 and 37 (the worked projects), and keep Chapter 38 (Cookbook) nearby for the recipes you will now recognize on sight.

Workbook: Volume H — Capstones

Everything before this point handed you a recipe. These two capstones hand you a spec. That difference matters: a portfolio reviewer or hiring manager does not care whether you can follow numbered steps — they care whether you can take a loosely defined problem, make defensible decisions, and ship something that works. The two worked projects in Compendium Chapter 36 (Worked Project - Dashboard) and Compendium Chapter 37 (Worked Project - Field Collection) are the guided versions of what you are about to do. The rule for this workbook: do not open those chapters until your self-assessment is done. Work from the spec, and when you get stuck, go to the reference chapters cited in each rubric — not to the worked answer.

Accounts, time, and rules of engagement

Both capstones need app-authoring privileges that a free public account does not include: creating dashboards, publishing editable hosted feature layers, and signing in to field apps are organizational-account capabilities. The practical free path is the ArcGIS Online free trial, which is time-limited but costs nothing and covers everything both capstones require. Sign up for it only when you can run both capstones back to back within the trial window. If a trial is not an option, each capstone below includes a read-along alternative that preserves most of the learning with a public account.

Three rules:

1. Timebox each capstone. A weekend per capstone is realistic; if you are past double that, you are polishing instead of finishing.
2. Keep a decision log — a plain text file where you record every nontrivial choice and why. You will need it for the portfolio checklist at the end.
3. Done beats perfect. A finished project with documented limitations demonstrates more competence than a half-built project with beautiful symbology.

Capstone 1: A dashboard about a place you know

Objective. Design and build a live, interactive dashboard on a topic from your own life or neighborhood, end to end, from data sourcing through published app.

What you need. An ArcGIS Online trial account (or organizational account); the Living Atlas; optionally an open data portal for your city or region. Read-along alternative: with a free public account you can complete steps 1 through 4 as a styled web map with configured pop-ups, then read Compendium Chapter 28 (Dashboards) against your map and sketch the dashboard layout on paper — you lose the app assembly but keep the data and map work, which is most of the grade.

The spec. Pick a personal domain: somewhere you have real knowledge and can judge whether the output looks right. Good candidates: restaurants or coffee shops you have actually visited, your running or cycling routes, street trees on your block, farmers markets in your county, playgrounds you have taken kids to, houses sold in your neighborhood this year. Your dashboard must:

- Combine at least one layer you created yourself with at least one layer sourced from the Living Atlas or an open data portal.
- Show a map, at least two numeric indicators, at least one chart, and a list or table.
- Respond to interaction: selecting or filtering in one element changes what other elements show.
- Answer a question a real person would ask, stated as a subtitle on the dashboard itself.

Steps.

1. Write the brief before touching software. One paragraph: who the dashboard is for, the single question it answers, and the three things the viewer should learn in ten seconds. Every later decision gets tested against this paragraph.
2. Source the reference layer. Search the Living Atlas or a local open data portal for a layer that gives your topic context — boundaries, demographics, streets, land use. Evaluate it the way Compendium Chapter 5 (Finding Data) teaches: who published it, when it was updated, whether the geometry and attributes are trustworthy at your working scale.

3. Create your own layer. Use whatever path from Compendium Chapter 11 (Creating Data) fits: sketch features directly in Map Viewer, geocode a spreadsheet of addresses, or build a small CSV with coordinates. Before you digitize a single point, design the schema on paper — field types, a coded-value domain for any category field, sensible field names (Compendium Chapter 12, Schema Design). Aim for a few dozen features; enough for charts to mean something.
4. Build the web map. Style both layers deliberately rather than accepting defaults (Compendium Chapter 7, Styling and Smart Mapping), configure pop-ups that show only what matters in a readable order (Compendium Chapter 9), and set a default extent that frames your area of interest. This map is the dashboard's foundation; time spent here pays off double.
5. Assemble the dashboard. From your web map's item page, create a new dashboard. Lay out the map, indicators, chart, and list against a rough paper sketch first. Follow the composition guidance in Compendium Chapter 28: most important element largest, related elements adjacent, no element you cannot justify.
6. Wire the interactions. Add a category selector or use map-driven actions so that filtering one element filters the rest. Test every interaction path, including "what does the viewer see when nothing is selected."
7. Publish and share. Set the dashboard and every layer it depends on to shared, then open it in a private browser window to verify a stranger sees what you see. Broken sharing on one dependency is the single most common way finished dashboards die in public.

Milestone checklist.

- [ ] Brief written and saved in the decision log
- [ ] Reference layer chosen, with a one-line quality justification
- [ ] Own layer published with domains and clean field names
- [ ] Web map styled, pop-ups configured, default extent set
- [ ] Dashboard laid out with map, two indicators, chart, list
- [ ] At least one working cross-element interaction
- [ ] Shared publicly and verified in a private browser window

Common failures.

- Chart soup: six charts that each restate the same count. If two elements answer the same question, delete one.
- The self-symbolizing default: publishing with smart mapping's first suggestion untouched. Reviewers can tell.
- Indicator lies: a big number with no unit, no timeframe, and no comparison. "47" means nothing; "47 open requests, up from 31 last month" means something.
- Sharing rot: dashboard public, one of its layers private. Always test logged out.
- Scope creep: adding a second question halfway through. Ship the first dashboard; make the second one your next portfolio piece.

Self-assessment rubric. Score yourself honestly on each row; anything below "competent" tells you exactly which chapter to reread before revising.

DIMENSION	CHAPTERS	COMPETENT LOOKS LIKE
Data sourcing	Ch 5, Ch 10	Reference layer's provenance stated; your layer performs without warnings
Schema design	Ch 12	Correct field types; category fields use domains; no fields named `field_1`
Map craft	Ch 4, Ch 7	Symbology encodes the data's actual structure; legible at the default extent
Pop-up quality	Ch 9	Only relevant fields, formatted, in a deliberate order
Dashboard composition	Ch 28	Layout matches the brief's ten-second test; no orphan elements
Interactivity	Ch 28, Ch 8	Selections cascade correctly; empty-selection state is sensible

Success criteria. Someone who knows your neighborhood but not GIS can open the link cold, understand the question within ten seconds, and answer it correctly by

interacting with the dashboard — and nothing they click produces a blank or broken element.

Stretch goal. Add one Arcade expression that computes a value not stored in any field — a per-area rate, a days-since-visit count, a category derived from two other fields — and drive an indicator or pop-up with it (Compendium Chapter 8, Arcade from Zero to Fluent).

Capstone 2: The two-person field job, played solo

Objective. Run a complete field data collection operation in which you play both roles — the GIS lead who designs and manages the job, and the field worker who executes it — and let each role's friction teach you what the other did wrong.

What you need. An ArcGIS Online trial account; a smartphone with the ArcGIS Field Maps app installed; something inspectable within walking distance — street trees, sidewalk defects, fire hydrants, park benches, storefronts, litter hotspots. Read-along alternative: without a trial or a phone, run the same two-role structure using browser-based editing in Map Viewer (Compendium Chapter 13, Editing Workflows) — you lose GPS and the mobile form, but the schema-design, role-separation, and QA lessons survive intact.

The spec. The lead designs a collection schema, publishes an editable layer, and configures the field app. The field worker collects at least twenty real features over two separate outings, following the form exactly as configured — no mental workarounds. Between outings, the lead reviews the incoming data, documents every defect, fixes the schema and form, and sends the worker back out. The final deliverables are a QA'd dataset, a defect log, and a simple results map.

Steps.

1. As the lead: write the field brief. What is being collected, what attributes matter, what a field worker must be able to answer in under thirty seconds per feature while standing on a sidewalk. Keep the schema to a handful of fields.
2. As the lead: build the layer. Create an editable hosted feature layer with the schema from step 1 — coded-value domains for every category, required fields only where

truly required, attachments enabled for photos (Compendium Chapter 12; Compendium Chapter 10, Hosted Feature Layers).

3. As the lead: configure the field experience. In Field Maps Designer, build the form: field order matching the physical inspection order, domains rendering as pick lists, a photo prompt. Decide your editing settings deliberately — can the worker update existing features or only add new ones? (Compendium Chapter 30 covers Field Maps.)
4. Switch roles. As the field worker: sign in to Field Maps on your phone and collect the first ten features. Follow the form exactly as configured. Every time you improvise, hesitate, or want a value the pick list does not offer, write it down — that note is the deliverable of this outing, not just the points.
5. As the lead: run the mid-job QA review. Open the collected data in Map Viewer or a table view. Check completeness, positional plausibility, attachment presence, and whether attribute values cluster suspiciously (everyone picking the first pick-list option is a form-design smell, not a data pattern). Log every defect and its root cause per Compendium Chapter 14 (Data Quality).
6. As the lead: fix the operation. Amend domains, reorder the form, adjust required fields, tighten the brief — whatever the defect log demands. Record each change in the decision log with the defect that motivated it.
7. As the field worker: collect the second ten features with the revised setup. The comparison between outing one and outing two friction is the core evidence of this capstone.
8. As the lead: close out. Final QA pass, then build a small results web map that shows the collected features styled by their key category, with a pop-up including the photo.

Milestone checklist.

- [ ] Field brief written; schema designed on paper before publishing
- [ ] Editable layer live with domains, required fields, attachments
- [ ] Form configured in Field Maps Designer and tested on the phone before outing one
- [ ] Outing one complete: ten or more features plus a friction log
- [ ] Mid-job QA review done; defect log with root causes

- [ ] Schema and form revisions applied and documented
- [ ] Outing two complete; friction compared against outing one
- [ ] Final QA pass and results map shared

Common failures.

- Free-text category fields. If the worker can type anything, they will, and your QA review becomes archaeology. Domains exist for exactly this.
- Everything required. Ten required fields per feature means the worker fabricates values to dismiss the form. Require only what you would reject the record without.
- Skipping the phone test. A form that looks fine in the designer can be unusable on a phone in sunlight. Test on the actual device before outing one.
- Playing both roles simultaneously. The exercise only works if the worker refuses to compensate for the lead's mistakes. When the form is bad, collect bad data and log why — that is the lesson.
- No mid-job checkpoint. Reviewing only at the end converts recoverable defects into a spoiled dataset.

Self-assessment rubric.

DIMENSION	CHAPTERS	COMPETENT LOOKS LIKE
Schema and domains	Ch 12	Every category field domained; required fields defensible
Layer configuration	Ch 10	Editing settings intentional; attachments working
Form design	Ch 30	Field order matches inspection order; sub-thirty-second capture
Field execution	Ch 13, Ch 30	Twenty-plus real features; friction log kept honestly
QA discipline	Ch 14	Defects logged with root causes, not just fixed silently

DIMENSION	CHAPTERS	COMPETENT LOOKS LIKE
Iteration	Ch 14, Ch 12	Outing two measurably smoother, and you can prove it from the logs

Success criteria. A stranger reading your defect log and decision log can reconstruct what went wrong in outing one, what you changed, and why outing two went better — and the final dataset passes your own documented QA checks.

Stretch goal. Feed the collected layer into a live dashboard of collection progress (features per day, counts by category, latest photo), so the lead role could have monitored the field role in real time — then note in your log what you would add to the schema to make that dashboard better, which is how real field operations evolve.

The portfolio checklist

A capstone becomes a portfolio piece when someone else can evaluate it without you in the room. For each finished project, assemble:

- **A live, public link** that works in a private browser window. If it needs a login, it is not a portfolio piece.
- **A one-page project summary**: the brief, the data sources with provenance, the tools used, and the question the project answers. Write it for a non-GIS reader.
- **Your decision log**, cleaned up. Three to ten entries showing choices you made and rejected alternatives. This is what separates "followed a tutorial" from "exercised judgment."
- **A limitations section.** Two or three honest sentences about what the data cannot support and what you would do with more time. Stated limitations signal competence; hidden ones signal the opposite.
- **Evidence of process**, especially for Capstone 2: the defect log and the outing-one-versus-outing-two comparison are more impressive than the final map.
- **Complete item pages.** Every layer, map, and app has a real thumbnail, summary, description, and terms of use. Reviewers who know ArcGIS Online will click through to your items; empty metadata undoes everything above it.

- **One screenshot per project** exported at readable resolution, for contexts where the live link cannot travel.

If both capstones clear their rubrics and this checklist, you have working evidence of the full stack this compendium teaches: sourcing, schema, creation, cartography, apps, field operations, and quality control. From here, Compendium Chapter 38 (Cookbook) is your recipe box for the next projects, and Compendium Chapter 39 (Troubleshooting Encyclopedia) is where you go when the next spec — the one a client hands you — breaks in ways this workbook never mentioned.
