Data Foundation Cleanup Scoping Worksheet

Domain Methods

WHAT THIS WORKSHEET IS FOR

Use this in a leadership, RevOps, finance, marketing, or data working session before you start a vendor search or open a hiring req. The goal is to name the failure layer, first success condition, and phase-one boundary before the cleanup brief starts to sprawl.

1. NAME THE DECISION THAT KEEPS BREAKING

- Meeting or workflow in scope: _____

- What answer is not trustworthy enough today? _____

- What expensive decision keeps stalling because of that? _____

- Which team feels the pain first? _____

2. SORT THE FAILURE LAYER

- Definition conflict: yes / no

- Source-system quality: yes / no

- Modeling or reporting logic: yes / no

- Workflow ownership: yes / no

- Which layer should lead phase one, and why? _____

3. WRITE THE FIRST SUCCESS CONDITION

- In 30 to 45 days, what should feel cleaner in one real meeting? _____

- Which metric family or reporting moment should improve first? _____

- What caveat or side spreadsheet should disappear or shrink? _____

4. DOCUMENT OWNER EXPECTATIONS

- Executive sponsor: _____

- Business context owner: _____

- System or data owner: _____

- Who can approve a definition change or scope tradeoff? _____

- Who inherits the operating rhythm after phase one? _____

5. SET THE PHASE-ONE BOUNDARY

- Fix now: _____

- Scope now, fix later: _____

- Do not pull into phase one: _____

- What full redesign temptation should stay out of this first pass? ___

6. USE THE BRIEF IN THE NEXT HIRING CONVERSATION

- What kind of help does this scope actually need first? _____

- What should a freelancer, partner, or hire own in phase one? _____

- What should they explicitly not be asked to absorb? _____

- What would make you say the problem is still not scoped tightly enough? _

SUGGESTED ROUTES

- Use Translate the Ask when the request is still fuzzy and the business problem needs a tighter brief before build work or hiring starts.

- Use Data Foundation when the business can name the right first win but the systems, models, or source logic underneath it are still too brittle to support it.