Love the direction—this is shaping up like a proper “ingest → detect → map → fix → export” pipeline. Below is a concrete, build-from-scratch plan that matches your updated flow and plugs into the structures you’ve already been using (Admin/File\_Selector, Term\_Editor, Field\_Map, unified Output tab, etc.). I’ve split this into a minimum-viable build you can complete quickly, and “smart” layers you can add as you go.

**Architecture at a glance**

* **UI & Orchestration (VBA):** file/sheet picking, button events, wiring between steps, confidence flags, and export.
* **Data Handling (Power Query + Tables):** ingestion, structure detection helpers, canonicalization (types/headers/whitespace), transforms.
* **Mapping Intelligence (Tables + VBA):** synonym dictionary, format hints, fuzzy matching, confidence scoring.
* **Adjustment Tools (VBA + Tables):** term replacements, split/merge helpers, simple rule transforms.
* **Audit & Reproducibility:** log each run (inputs, heuristics, decisions) for traceability.

**Workbook layout (tabs you create now)**

1. **Admin** — control panel for the whole flow
   * Cells:
     + D2: “Is the source file currently open?” with **Yes/No** buttons (your existing pattern).
     + E4 (visible when “Yes”): dropdown of open workbooks (source).
     + E5 (visible when “Yes”): dropdown of open workbooks (carrier template).
     + E6: Output mode dropdown (**Row-Based / Column-Based**) — you already use this.
     + D8–E9: Source **Sheet** dropdown + auto-fill when only one exists.
     + D10–E11: Template **Sheet** dropdown + auto-fill when only one exists.
     + Buttons:
       - **Load Open Files** (populate dropdowns; unhides File\_Selector during use)
       - **Get Source Data**
       - **Get Template Fields**
       - **Auto-Detect Structure**
       - **Auto-Map Fields**
       - **Review Flags / Adjust**
       - **Export Output**
   * Named ranges (helpful): nmSourceFile, nmTemplateFile, nmSourceSheet, nmTemplateSheet, nmOutputMode
2. **File\_Selector** — scratch list of open workbooks (hide after use).
3. **q\_Ingest\_Source** — Power Query output table with canonicalized source data
   * Table name: tblSource (headers standardized, types enforced).
4. **q\_Ingest\_Template** — Power Query output with carrier template requirements
   * Table name: tblTemplateFields
   * Columns: TemplateField, Required?, FormatHint, Sample, Notes
5. **Dictionary** — brains for auto-mapping
   * Table tblSynonyms with columns:
     + CanonicalField (e.g., **Date of Birth**)
     + Variant (e.g., **DOB**, **Birth Date**, **Employee DOB**)
     + SourceSystem (optional; e.g., “Employee Navigator”)
     + Weight (0–1 for boosting)
   * Table tblFormatRules with columns:
     + CanonicalField, ExpectedType (Date/Text/Number/Currency/YesNo), RegexPattern (optional), Example, Strict? (TRUE/FALSE)
6. **Field\_Map** — the mapping workbench
   * Table tblFieldMap with columns:
     + TemplateField (from tblTemplateFields)
     + DetectedSourceField (auto-filled)
     + Confidence (0–100)
     + Flag (Blank/Needs Review/OK)
     + Notes (user notes)
   * You’ll bind the auto-mapping here, mark anything < threshold as **Needs Review**, and let users override.
7. **Term\_Editor** — your replace-in-column tool (you already spec’d this)
   * B1: dropdown selects a **CanonicalField** (from Field\_Map or Dictionary)
   * A3/B3 headers: **Current Term / New Term**
   * A4:A lists uniques for the chosen column (scripted), B4:B for replacements
   * Button: **Apply replacements** (with calc turned off during replace; then refresh uniques)
8. **Adjustments** — user-friendly transforms
   * Simple rule table tblTransforms with columns:
     + TargetField, TransformType (Split, Merge, Replace, Upper, Lower, Trim, MapValue, DateFormat), Param1, Param2, Param3
   * Example: Last, First → Split with delimiter , into LastName / FirstName.
9. **Output** — unified final table for export (driven by Admin!E6)
   * Table name: tblOutput
   * Populated by VBA after transforms; values only when exported.
10. **Logs** — every run gets a row
    * Columns: Timestamp, SourceFile, SourceSheet, TemplateFile, TemplateSheet, StructureDetected, Rows, Cols, AutoMapThreshold, FlagsCount, ExportFileName

**Flow (buttons → actions)**

1. **Load Open Files**
   * Enumerate open workbooks (excl. ThisWorkbook) into File\_Selector.
   * Set Admin dropdowns via data validation. Hide File\_Selector when done.
2. **Get Source Data**
   * Using nmSourceFile + nmSourceSheet, invoke PQ query **q\_Ingest\_Source**:
     + Promote headers → Trim/clean → Unmerge → Remove empty columns/rows
     + Enforce types using tblFormatRules where possible (safe coercions only)
     + Output to tblSource
   * If only one sheet exists, auto-set nmSourceSheet.
3. **Get Template Fields**
   * Using nmTemplateFile + nmTemplateSheet, PQ **q\_Ingest\_Template** to build tblTemplateFields.
4. **Auto-Detect Structure**
   * Inspect tblSource with heuristics (VBA reads the PQ table):
     + **Column-based dependents pattern:** repeated header stems like Dep1 First Name, Dep2 First Name, etc.
     + **Row-per-plan:** same person identifier (SSN/EmpID/Name+DOB) appearing on multiple rows with distinct plan/category values.
     + **Row-per-person:** unique person ID per row; plan fields already pivoted into repeated column blocks.
   * Write result to Admin (e.g., StructureDetected = Column-Based w/3 dependent blocks).
5. **Auto-Transform (if needed)**
   * Column-based → **Row-based** using PQ unpivot pattern (dependents to rows).
   * Row-per-plan → **Person-single-row** by grouping on person key and pivoting plan group with indexed suffixes.
   * Always aim to yield a **normalized person-centric table** for mapping. Output stays tblSource.
6. **Auto-Map Fields**
   * For each TemplateField:
     + Score all tblSource headers using:
       - **Exact/normalized match** (strip spaces/punct/case)
       - **Synonym hits** from tblSynonyms (boost by Weight)
       - **Token overlap** (Jaccard on word tokens)
       - **Type/regex compatibility** from tblFormatRules (bonus)
       - **SourceSystem-specific boosts** when SourceSystem matches (Employee Navigator)
     + Pick best candidate + score.
     + If score ≥ threshold (e.g., 80), set DetectedSourceField and mark **OK**; else **Needs Review**.
   * Write to tblFieldMap with confidences. Show counts on Admin.
7. **Review Flags / Adjust**
   * User fixes mappings with dropdowns of tblSource headers.
   * Apply Term\_Editor replacements and any Adjustments rules.
8. **Export Output**
   * Build tblOutput (columns in tblTemplateFields order).
   * Populate row values from normalized tblSource using tblFieldMap + transforms.
   * Write a **new workbook** (values only).
   * Log the run in **Logs**.

**Power Query: robust, parameterized ingestion**

Create two named cells on **Admin**:

* nmSourcePath, nmTemplatePath (the full paths resolved from selected open workbooks).
* nmSourceSheet, nmTemplateSheet (text).

**q\_Ingest\_Source (skeleton)**

let

SourceWB = Excel.Workbook(File.Contents(Excel.CurrentWorkbook(){[Name="nmSourcePath"]}[Content]{0}[Text]), null, true),

SheetName = Excel.CurrentWorkbook(){[Name="nmSourceSheet"]}[Content]{0}[Text],

Raw = try Table.SelectRows(SourceWB, each [Item] = SheetName and [Kind] = "Sheet"){0}[Data] otherwise error "Sheet not found",

Promo = Table.PromoteHeaders(Raw, [PromoteAllScalars=true]),

Trimmed = Table.TransformColumns(Promo, List.Transform(Table.ColumnNames(Promo), each {\_, Text.Trim, type text})),

Unmerged = Table.CombineColumns(Table.TransformColumnTypes(Trimmed, List.Transform(Table.ColumnNames(Trimmed), each {\_, type text})), {}, Combiner.CombineTextByDelimiter(" ", QuoteStyle.None), "Merged"), // safe no-op if no merges

Clean = Table.TransformColumnNames(Promo, each Text.Trim(Text.Clean(\_))),

NoEmptyCols= Table.RemoveColumns(Clean, List.Select(Table.ColumnNames(Clean), each List.NonNullCount(Table.Column(Clean, \_)) = 0)),

Result = NoEmptyCols

in

Result

*(Do the same for q\_Ingest\_Template, but you only need the template’s field list and format hints.)*

**VBA: essential building blocks (module plan)**

Create modules:

* modUI — button handlers + UI state
* modFiles — open workbook enumeration, sheet enumeration, path resolution
* modPQ — refresh specific queries, read tables
* modDetect — structure heuristics
* modMap — auto-mapping & confidence
* modTransform — apply Term\_Editor and Adjustments
* modExport — build tblOutput and save new workbook
* modUtil — helpers (normalize strings, token sets, Jaccard, regex via VBScript.RegExp, safe coercions)

**List open workbooks → File\_Selector (core you can paste)**

Public Sub LoadOpenFiles()

Dim wsFS As Worksheet, wb As Workbook, i As Long

Set wsFS = ThisWorkbook.Sheets("File\_Selector")

wsFS.Visible = xlSheetVisible

wsFS.Cells.ClearContents

i = 1

For Each wb In Application.Workbooks

If wb.Name <> ThisWorkbook.Name Then

wsFS.Cells(i, 1).Value = wb.Name

wsFS.Cells(i, 2).Value = wb.FullName

i = i + 1

End If

Next wb

' Bind validation on Admin E4/E5 from wsFS column A range actually populated.

wsFS.Columns.AutoFit

wsFS.Visible = xlSheetHidden

End Sub

**Resolve selected file → path named range**

When a file is chosen in Admin E4/E5, write the **FullName** to nmSourcePath / nmTemplatePath. If only one sheet in that workbook, set nmSourceSheet automatically.

**Refresh queries (source/template) on demand**

Public Sub RefreshIngest()

ThisWorkbook.Queries("q\_Ingest\_Source").Refresh

ThisWorkbook.Queries("q\_Ingest\_Template").Refresh

End Sub

**Structure detection heuristics (sketch)**

* **Column-based dependents:** scan header names; if repeated stems like Dep1, Dep2 or Spouse\_1, Child\_1 patterns exist across multiple attribute names → column-based.
* **Row-per-plan:** compute count of distinct rows per PersonKey (SSN/EmpID or Name+DOB). If many >1 with distinct PlanType/CoverageLevel → row-per-plan.
* **Row-per-person:** otherwise.

Write a short routine that:

* Builds PersonKey using best-available fields (prefer SSN/EmpID; fallback to Name+DOB).
* Tallies distribution and sets Admin status.

**Auto-mapping confidence (practical scoring)**

Score = 0

* +60 exact normalized header match
* +40 synonym match (× Weight)
* +20 token Jaccard similarity (>0.6 → +20, >0.4 → +10)
* +10 expected type match (from tblFormatRules)
* +10 source-system boost (if variant tagged for “Employee Navigator”)  
  Capped at 100. Threshold to **auto-accept**: 80. 60–79 → **Needs Review**.

Normalization: lowercase, remove non-alnum, collapse spaces, strip “employee/ee/er” prefixes/suffixes when generic.

**Transforms you can ship in v1**

* **Split “Last, First”**: detect comma; write to LastName/FirstName.
* **Yes/No coercions**: map {Y,Yes,True,1}→Yes; {N,No,False,0}→No.
* **Enum mappings**: Employee → EE, Spouse → SP, Child → CH.
* **Date normalization**: try parse, then format mm/dd/yyyy.
* **Currency**: strip symbols/thousands; cast number; round to 2.
* **Tier normalization**: map vendor variants to {EE, ES, EC, FAM}.

These should be **rule-driven** via Adjustments so users can see and tweak.

**Column-based → Row-based (dependents) pattern (PQ recipe)**

1. Identify repeated dependent blocks: e.g., stems {Dep1, Dep2, Dep3} with same subfields {First Name, Last Name, DOB, Relationship, SSN}.
2. Unpivot all dependent columns into rows; parse the stem index (1..N) into a DepIndex.
3. Keep Employee columns static; output **Employee** + **Dependent** rows with a new column Role = Employee/Dependent.
4. Filter out fully blank dependent rows (all subfields empty).

*(You already asked for that label; include Role in the output.)*

**Row-per-plan → Person-single-row pattern (PQ recipe)**

1. Group by PersonKey.
2. Within each group, add PlanIndex = 1..k.
3. Pivot “plan attributes” by PlanIndex so you get PlanName\_1, CoverageLevel\_1, … PlanName\_k, etc.
4. Join back static person fields from the group’s **first non-blank** row per field.

This avoids your earlier “still multiple rows per person” outcome by collapsing at the group stage before pivot.

**Export (values-only)**

* Create a new workbook.
* Add one sheet named after the carrier template.
* Write headers exactly in tblTemplateFields order.
* Loop the normalized source rows, map through tblFieldMap, apply transforms, write cell **Values** (no formulas).
* Save as CarrierName\_Export\_YYYYMMDD.xlsx.

**Logging**

Every run writes to **Logs** with: file names, sheets, structure, row/col counts, threshold used, number of flags, export filename. This will save you later when someone asks “why did EE Tier say EC that one time?”

**Build order (2–3 focused work sessions)**

**Session A: Skeleton & ingestion**

1. Create tabs/named ranges and all tables (empty is fine).
2. Implement **Load Open Files**, bind Admin dropdowns, auto-select sheet if only one.
3. Wire **q\_Ingest\_Source** and **q\_Ingest\_Template** to nmSourcePath/nmTemplatePath + sheets.
4. Confirm tblSource and tblTemplateFields load clean.

**Session B: Detect → Transform → Map**

1. Implement structure heuristics (write status to Admin).
2. Build PQ steps for **one** structure path you’ll see most (Employee Navigator: usually row-per-plan).
3. Create Dictionary (seed with your “benefits in hand / employee navigator” aliases).
4. Implement modMap.AutoMap scoring → fill tblFieldMap with confidences & flags.

**Session C: Adjustments & Export**

1. Term\_Editor (unique list + Apply replacements w/ calc off/on).
2. Adjustments rules (start with Split, Replace, Enum map).
3. Build tblOutput and **Export Output** button.
4. Add **Logs** write at each step.

**Seed the Dictionary fast**

From your Data Guide, add:

* Canonicals: First Name, Last Name, Middle Name, Suffix, DOB, DOH, Gender, Address 1/2, City, State, Zip, SSN, Email, Relationship, Coverage Level, Plan Name, Benefit Type, Approved Benefit Amount, etc.
* Variants you see in Employee Navigator and carriers:
  + DOB|Birth Date|Date of Birth|EE DOB
  + DOH|Hire Date|Date of Hire
  + Relationship|Rel|Dependent Type
  + Coverage Level|Tier|Cov Tier|Coverage Tier|Lvl
  + SSN|Social|Social Security|S.S.N.  
    Give Employee Navigator variants a Weight boost (e.g., 1.2).

**Guardrails & polish**

* **Threshold slider** on Admin (e.g., cell E14) to tune auto-map aggressiveness.
* **Type cross-check**: if TemplateField expects Date but the candidate column fails parse on ≥30% rows, auto-lower confidence.
* **Empty-column warning** if mapped source field has >95% blanks.
* **Undo/backup**: every export writes a sibling CSV for quick diff/debug.
* **Performance**: turn off screen updating and calc during big writes; for term replacement, use arrays not cell-by-cell edits.

**Where to start right now**

1. Stand up the **Admin** tab with the named ranges and buttons.
2. Paste the **LoadOpenFiles** routine and bind to your button.
3. Create the two PQ queries using the parameterized skeleton above.
4. Build tblTemplateFields from your carrier template file; confirm it loads.
5. Seed Dictionary and FormatRules from your Data Guide (just a dozen high-value fields to start).
6. Implement the auto-mapping scorer using the simple rules listed; write to tblFieldMap.
7. Test on your 10–15 row Employee Navigator sample.

When you’re ready, I can drop in ready-to-paste VBA for:

* Normalization/tokenization + Jaccard similarity
* The auto-mapping scorer filling tblFieldMap
* Structure detection (including dependent-block detection)
* The row-per-plan collapse (PQ) and column-based → row-based (PQ)
* Term\_Editor unique-list + apply-replacements routine
* Export output (values-only) with logging

From there, we can harden the heuristics with your real files and ratchet up the “auto” part of auto-mapping as your dictionary grows. The machine gets smarter the more synonyms and format rules you feed it—delightfully human, but without the mood swings.