Excellent. Based on your full dataset inventory and your goal to build a *simplified but complete Swiss railway network* in SUMO — using lightweight geometries like those in linie\_mit\_polygon.csv — here is a **comprehensive and professional roadmap**, divided into **phases**, with detailed input/output expectations, purpose, and technical operations.

**🚆 Simplified Swiss Network Pipeline for SUMO (May 2025)**

**🎯 Goal**

Create a lightweight yet topologically correct Swiss rail network that:

* Covers the full national network (not just SBB),
* Preserves enough detail for simulating train movement and timetables,
* Is efficient enough for long-period and large-scale VC simulations in SUMO.

**🗂️ DATASETS OVERVIEW**

These datasets are available locally and provide the structural backbone:

| **Dataset Filename** | **Key Use** |
| --- | --- |
| linie\_mit\_polygon.csv | Primary geometry + logical line segment structure |
| linienkilometrierung.csv | Kilometrage reference (for positioning / debugging) |
| haltestellen\_2025.csv | Stops used in GTFS timetables (critical for routing logic) |
| dienststellen-gemass-opentransportdataswiss.csv | Station metadata including location, IDs, and types |
| haltestelle-haltekante.csv | Platform-level geometry, mostly for GTFS precision |
| actual\_date-world-traffic\_point-2025-04-05.csv | Real-world operational point structure and geometry |
| rollmaterial.csv | Technical vehicle specs (length, seating, vmax, etc.) |
| jahresformation.csv | Vehicle formations for trains (used in routing/vehicle modeling) |
| rollmaterial-matching.csv | Link between formation types and rolling stock entries |
| zugzahlen.csv | Train volumes per segment (used optionally for prioritization/weighting) |

**🧩 PIPELINE PHASES**

**PHASE 1: Line Geometry Extraction**

**Purpose:** Extract simplified network segments from linie\_mit\_polygon.csv.

**Input:**

* data/Swiss/raw/linie\_mit\_polygon.csv

**Output:**

* data/Swiss/processed/rail\_edges\_simplified.csv

**Key Operations:**

* Parse each row's LineString and sanitize attributes.
* Generate unique edge IDs from Linienname, von\_betriebspunkt, nach\_betriebspunkt.
* Store: edge\_id, from\_station, to\_station, gauge, shape, line\_id.

**PHASE 2: Node Extraction and Deduplication**

**Purpose:** Identify and assign a unique node ID and coordinate for each unique station (Betriebspunkt).

**Input:**

* rail\_edges\_simplified.csv (from Phase 1)
* dienststellen-gemass-opentransportdataswiss.csv
* optionally: haltestelle-haltekante.csv or actual\_date-world-traffic\_point...csv

**Output:**

* data/Swiss/processed/rail\_nodes\_simplified.csv

**Key Operations:**

* For each unique from\_station and to\_station, lookup geocoordinates.
* Resolve duplicates by preferring entries with operatingPoint == True.
* Assign a unique node ID per station name + line context.

**PHASE 3: Write SUMO Nodes & Edges**

**Purpose:** Generate SUMO-compatible XMLs.

**Input:**

* rail\_nodes\_simplified.csv
* rail\_edges\_simplified.csv

**Output:**

* data/Swiss/processed/sumo/simplified\_nodes.nod.xml
* data/Swiss/processed/sumo/simplified\_edges.edg.xml

**Key Operations:**

* Nodes: Convert each node to a <node> tag with ID, x, y.
* Edges: Convert each segment into a <edge> tag with from, to, shape.

**PHASE 4: Connection Building (Optional)**

**Purpose:** If required, create default <connection> entries for edge chaining.

**Input:**

* simplified\_edges.edg.xml

**Output:**

* simplified\_connections.con.xml

**Key Operations:**

* For each consecutive edge that shares a node, write a <connection> with default parameters.
* Can be skipped if SUMO routing handles it implicitly.

**PHASE 5: Network Generation**

**Purpose:** Compile the raw XMLs into a usable .net.xml SUMO network.

**Input:**

* simplified\_nodes.nod.xml
* simplified\_edges.edg.xml
* simplified\_connections.con.xml *(optional)*

**Output:**

* april\_2025\_swiss\_simplified.net.xml

**Key Command:**

netconvert -n simplified\_nodes.nod.xml -e simplified\_edges.edg.xml -x simplified\_connections.con.xml -o april\_2025\_swiss\_simplified.net.xml

**PHASE 6: GTFS Stop-to-Node Mapping**

**Purpose:** Match GTFS stops (haltestellen\_2025.csv) to network nodes.

**Input:**

* haltestellen\_2025.csv
* rail\_nodes\_simplified.csv

**Output:**

* stop\_id\_to\_node\_id.csv (GTFS stop ID → SUMO node ID)

**Key Operations:**

* Match via geocoordinates (lat/lon) with nearest node search (Euclidean).
* Use dienststellen...csv as fallback to improve location accuracy.

**PHASE 7: Route Definition via GTFS or Jahresformation**

**Purpose:** Define train trips across the network.

**Input:**

* GTFS trip files OR jahresformation.csv
* stop\_id\_to\_node\_id.csv
* optionally: zugzahlen.csv for weighting

**Output:**

* train\_routes.csv: sequence of node IDs per trip
* route\_to\_vehicle\_type.csv: map trip to vehicle type (from formation)

**Key Logic:**

* Each row = one trip → ordered list of nodes
* Use rotation and bitmap in jahresformation.csv to assign timetable windows

**PHASE 8: Vehicle Type Definitions**

**Purpose:** Define vehicle types for SUMO using technical specs.

**Input:**

* rollmaterial.csv
* rollmaterial-matching.csv
* jahresformation.csv

**Output:**

* vehicle\_types.veh.xml

**Key Logic:**

* Merge formation data with rolling stock specs.
* For each unique mapped train type, define SUMO vehicle attributes: length, vMax, accel, decel, seats, type.

**PHASE 9: Route File Generation**

**Purpose:** Create .rou.xml file for SUMO simulation.

**Input:**

* train\_routes.csv
* vehicle\_types.veh.xml
* route timetables (from jahresformation.csv or GTFS)

**Output:**

* full\_routes.rou.xml

**Key Elements:**

* <route>: defines edge sequence
* <vehicle>: assigned to route, includes departure time and type

**PHASE 10: Run SUMO Simulation**

**Purpose:** Execute a full-day or sampled simulation to evaluate route logic and network usability.

**Input:**

* april\_2025\_swiss\_simplified.net.xml
* full\_routes.rou.xml
* vehicle\_types.veh.xml

**Output:**

* SUMO logs
* Optional: FCD output, tripinfo, emission logs, etc.

**Command Example:**

sumo -n april\_2025\_swiss\_simplified.net.xml -r full\_routes.rou.xml --fcd-output fcd.xml --tripinfo-output tripinfo.xml

**🧪 Optional Add-ons (Post-MVP)**

* Use zugzahlen.csv to generate synthetic train flows if timetable is missing.
* Integrate linienkilometrierung.csv for validation/debugging of segment distances.
* Add delay models or VC-specific logic after Phase 10.

**PHASE 1 – Extract Simplified Edges and Nodes from linie\_mit\_polygon.csv**

**Goal:** Create a basic edge and node representation of the Swiss railway network using simplified geometries for efficiency.

**📥 Input:**

* D:/PhD/prog\_report\_2025\_June\_project/data/Swiss/raw/linie\_mit\_polygon.csv

**📤 Output:**

* Nodes CSV: D:/PhD/prog\_report\_2025\_June\_project/data/Swiss/interim/simple\_nodes.csv
* Edges CSV: D:/PhD/prog\_report\_2025\_June\_project/data/Swiss/interim/simple\_edges.csv

**⚙️ Logic:**

* Parse the geo\_shape field (WKT LineString) to extract coordinates.
* Create nodes at start and end of each LineString.
* Generate unique node IDs from the coordinate values.
* Save edge definitions including edge ID, source and target nodes, length, and metadata.

**📜 Script: extract\_simple\_edges\_and\_nodes.py**

**Path:** D:/PhD/prog\_report\_2025\_June\_project/scripts/simple\_network\_creators/