### Data sources and gating

* Required DB tables:
  + proposal, proposer, insured\_member, documents, optionally previous\_insurance\_details, and rule\_engine\_trail.
* Gating conditions:
  + documents.validated = TRUE
  + rule\_engine\_trail.finreview\_required = TRUE for the same proposal\_number
* Outcome: Only proposals meeting both conditions proceed to scoring.

### SQL extraction (data\_extraction.py)

* Uses CTEs:
  + validated\_documents: proposal→proposer links with validated documents.
  + finreview\_required\_set: proposals marked finreview\_required = TRUE.
  + member\_sum\_assured: SUM of insured\_member.sum\_insured per proposer.
  + previous\_insurance\_summary: SUM of other insurance per proposer (0 if table missing).
* Final SELECT returns only the fields required to score:
  + proposal\_number, proposer\_id, stated\_age, dob, occupation, annual\_income, premium, sum\_assured, other\_insurance\_sum\_assured
* Defensive behavior:
  + If optional tables don’t exist (e.g., previous insurance), aggregates default to 0.
  + Schema/table names are env-overridable; code quotes identifiers and introspects columns to avoid name mismatches.

### Preprocessing and validation (finance\_score\_engine.py)

* Preprocessing:
  + Safe numeric casting for annual\_income, premium, sum\_assured, other\_insurance\_sum\_assured (stripping commas, negative to None).
  + Normalizes occupation to lowercased trimmed strings.
* Validation (FR2):
  + Flags missing critical fields per proposal in validation\_issues (e.g., missing\_annual\_income).
  + These flags appear in the scored JSON for auditability and can be enforced by your consumers.

### Ratios (derived features)

* sar\_income\_ratio = sum\_assured / annual\_income
* tsar\_income\_ratio = (sum\_assured + other\_insurance\_sum\_assured) / annual\_income
* premium\_income\_ratio = premium / annual\_income
* All ratios guard against division by zero or missing values (result becomes None if not computable).

### YAML rule configuration (finance\_score\_rules.yaml)

* Weights (must sum to 1.0):
  + sar\_income\_ratio: 0.5
  + tsar\_income\_ratio: 0.25
  + premium\_income\_ratio: 0.25
* Bands per component:
  + Each ratio has ordered, non-overlapping ranges that map to 1–5 scores.
  + Example (sar): <2 → 1, 2–3 → 2, 3–4 → 3, 4–5 → 4, >=5 → 5.
* Decisions:
  + risk\_categories: maps final score 1→Safe … 5→Reject.
  + underwriting\_flags: conditional rules based on final score and component scores (e.g., score 2 with premium\_score ≤ 4 → Pass; 5 → Decline).

### Scoring logic

* Component scores:
  + Each ratio is looked up against its YAML bands to get a component score 1–5.
* Weighted total:
  + weighted\_score = sar\_score*w\_sar + tsar\_score*w\_tsar + premium\_score\*w\_prem.
  + final\_finance\_score = round(weighted\_score).
* Decisions:
  + risk\_category from score→label mapping (YAML).
  + underwriting\_flag from conditional rules (YAML).
* Explainability:
  + score\_factors: top three contributors computed via component score × weight, sorted desc.
  + This shows why a score ended up where it did (e.g., SAR contributed most).

### Outputs (per proposal)

* Inputs JSON (traceability):
  + Path: finance\_scores/YYYYMMDD/inputs/finance\_input\_.json
  + Fields: proposal\_number, proposer\_id, stated\_age, dob, occupation, annual\_income, premium, sum\_assured, other\_insurance\_sum\_assured
* Score JSON (decision package):
  + Path: finance\_scores/YYYYMMDD/finance\_score\_.json
  + Fields: proposal\_number, proposer\_id, ratios, component scores, final\_finance\_score, risk\_category, underwriting\_flag, score\_factors, validation\_issues

### Guarantees and safeguards

* Only proposals satisfying document validation and finreview\_required = TRUE are processed.
* Safe handling of missing or malformed numeric fields.
* Optional tables handled with zero-value fallbacks to keep scoring operational.
* Full auditability via per-proposal inputs snapshot and scored output, plus validation\_issues and score\_factors.

### Summary

* Extracts only proposals that are both validated and flagged for finance review.
* Computes three ratios from extracted fields (SAR, TSAR, Premium vs Income).
* Scores each ratio using YAML-defined bands, weights them, and rounds to a final score.
* Derives risk category and underwriting flag from YAML decisions.
* Writes two JSONs per proposal: inputs (for traceability) and scored outputs (for decisions).