1. **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.

2. **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.

3. **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.

4. **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).

5. **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).

6. **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).

7. **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

8. **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).