**End-to-End Name Screening Process (with FTS5 Integration)**

This solution processes sanctions data and payment messages, efficiently matches entity names, and helps flag possible risks using sophisticated text normalization and full-text search.

**1. Sanctions List Ingestion**

* **Download & Parse:** Get sanctions lists from trusted sources (e.g., OFAC, UN).[[1]](#fn1)
* **File Types:** Often CSV, XML, TXT, or other; sources may vary.[[2]](#fn2)[[1]](#fn1)
* **Parse & Structure:** Extracts fields: name, aliases, date of birth, nationality, ID numbers.[[1]](#fn1)[[2]](#fn2)

**2. Data Normalization and Cleaning**

* **Unicode Normalization:** Removes special characters, accents, inconsistent spacing, and script confusion.[[3]](#fn3)[[2]](#fn2) –
* Handle Literal “\uXXXX” Escapes in XML Text

Some XML files might have literal escape sequences (e.g. a backslash followed by u200b) instead of real Unicode characters:

"John Smith\u200b" (where \u200b is a zero-width space, invisible but problematic for matching).

* C) Normalize Common “Fancy” Punctuation

Sometimes names have curly quotes or special spaces (like “Smith” instead of "Smith", or non-breaking spaces).

* **Case Folding & Trimming:** Converts all text to lowercase and strips extra whitespace.[[2]](#fn2)[[3]](#fn3)
* **Diacritics Removal:** Standardizes text by removing accents (e.g., 'é' → 'e').[[3]](#fn3)[[2]](#fn2)
* Some names might have accents/diacritics (examples: José, François). To compare strings reliably, you want to strip those marks safely.
* **Alias Expansion:** Each entity’s aliases are fully normalized too.[[1]](#fn1)[[2]](#fn2) - (e.g. “Mohammed bin Salman”) can appear under multiple aliases or transliterations:
* Entity AliasesMohammed bin Salman MBS, Mohammad bin Salman, Mohamed Bin Salman Al Saud, محمد بن سلمان
* When we say “Alias Expansion: Each entity’s aliases are fully normalized too,” we mean:
* Each alias should go through the same normalization pipeline as the canonical name (Unicode cleanup, transliteration, tokenization, etc.).
* These normalized aliases are also indexed (FTS5, embeddings, phonetic blocks) — so any match against a variant still links back to the main entity.

**3. Knowledge Base (KB) Construction with FTS5**

* **Centralized Storage:** Structured records are saved into an SQLite database.[[1]](#fn1)
* **FTS5 Indexing:** Key fields (primary name, aliases, normalized name) are indexed using SQLite’s FTS5 extension. This enables instant, flexible full-text search through millions of records using advanced query syntax.[[4]](#fn4)[[5]](#fn5)[[1]](#fn1)
* **Tokenization:** FTS5 divides fields into searchable terms, supporting partial, wildcard, and fuzzy searches.[[5]](#fn5)
* **Scalability:** FTS5 ensures search efficiency for large volumes of sanctions data by returning relevant matches in real time.[[5]](#fn5)[[1]](#fn1)

**4. Ingest Payment or Customer Files**

* **File Types:** Payment messages, customer onboarding files, transaction logs (e.g., ISO 20022 XML, CSV).[[6]](#fn6)[[1]](#fn1)
* **Extract Fields:** Scripts extract payer/payee names, address, nationality, DOB, reference IDs.[[6]](#fn6)[[1]](#fn1)

**5. Name & Context Preprocessing**

* **Consistent Normalization:** Incoming data is normalized (same methods as KB) to ensure fair matching regardless of casing, accents, or input format.[[2]](#fn2)[[3]](#fn3)

**6. Matching Step (Core Screening with FTS5)**

* **FTS5 Search:** For each transaction or customer entity, normalized names are first located using FTS5 in the KB for fast, efficient candidate retrieval.[[7]](#fn7)[[4]](#fn4)[[1]](#fn1)
* **String Similarity Algorithms:** Candidate results from FTS5 are further compared using Levenshtein, Jaro-Winkler, and Jaccard similarity scoring.[[8]](#fn8)[[9]](#fn9)
* Let's walk through three common string similarity metrics—how each works and what they capture, with examples. All outputs are normalized to a range from 0 (no similarity) to 1 (exact match), making them easy to compare and combine.
* **1. Levenshtein Similarity (Normalized)**
* What it does: Counts the minimum single-character edits (insertions, deletions, substitutions) needed to turn a into b.
* Normalized so that 0 = no similarity and 1 = exact match.
* Formula:
* Examples:
* a = "kitten", b = "sitting"
* Levenshtein distance = 3; max length = 7
* Normalized similarity:
* a = "Anna", b = "Ana"
* Distance = 1; max length = 4
* Similarity:
* **2. Jaro-Winkler Similarity**
* What it does:
* Measures how similar two strings are, giving extra weight if they share the same prefix (helpful for names).
* Score is always between 0 and 1.
* Examples:
* a = "MARTHA", b = "MARHTA"
* High similarity (characters matched, just swapped), result: ~0.96
* a = "Robert", b = "Rupert"
* Result: ~0.93 (overlap on prefix 'R', some transpositions)
* a = "DWAYNE", b = "DUANE"
* Result: ~0.84
* **3. Token Overlap (Jaccard Similarity)**
* What it does:
* Considers sets of tokens (usually words separated by spaces), and measures their overlap.
* , where A and B are sets of tokens from a and b.
* Examples:
* a = "John Smith", b = "Smith John"
* Token sets: {John, Smith} and {Smith, John}
* Overlap = 2, Union = 2
* Similarity = 1.0
* a = "Anna Maria Jones", b = "Anna M. Jones"
* Token sets: {Anna, Maria, Jones} and {Anna, M., Jones}
* Overlap = 2 (Anna, Jones), Union = 4 (Anna, Maria, M., Jones)
* Similarity = 0.5
* a = "Robert Brown", b = "Ronald Brown"
* Overlap = 1 (Brown), Union = 3 (Robert, Ronald, Brown)
* Similarity = 1/3 ≈ 0.33
* Quick Recap Table

| * **Metric** | * **Captures** | * **Example** | * **Score** |
| --- | --- | --- | --- |
| * Levenshtein (norm) | * Typos, edit distance | * "kitten" vs "sitting" | * 0.57 |
| * Jaro-Winkler | * Transpositions, prefixes | * "MARTHA" vs "MARHTA" | * ~0.96 |
| * Token overlap | * Word/subset variations | * "Anna Maria" vs "Anna M. Jones" | * 0.5 |

* **Contextual Features:** Date of birth, country, and ID numbers are also compared and included in the scoring.[[9]](#fn9)[[10]](#fn10)

**How Context Features Behave (and Why They Help)**

**Names Alone Can Be Ambiguous**

String similarity is great for matching names, but it's common for popular or short names to produce several candidates with very similar scores. For example, "Li Wei" might return several database entities: multiple "Li Wei" entries, and even "Wei Li". The name alone can't reliably distinguish which is the correct match.

**Context Flips Borderline Cases**

Context features—like date of birth, country, or ID matches—act as "tie-breakers" among these similar candidates. Each such feature provides a small, explainable 'nudge' to the overall score.

**Feature Examples**

* Date of Birth (DOB) Match:  
  If the query DOB exactly matches an entity's normalized DOB:
  + Adds  to the score.
* Country Match:  
  If the query country (in ISO2/ISO3/name format) is found in the entity's set of nationalities or address text:
  + Adds .
* Soft ID Match:  
  If the last 4-6 alphanumeric characters of the query ID appear in any of the entity's IDs:
  + Adds .

**Why Are These Nudges Small?**

They're "deliberately small" so that a strong name similarity still decides most matches, but context can break ties or give a slight boost — not overwhelm the score. They help avoid false matches without hiding the meaning of the result or introducing hard rules that would ignore name similarity.

Tuning Magnitudes

* If your IDs are highly reliable: increase the soft ID match weight .
* If DOB is often missing or noisy: lower the DOB match weight .

Illustrative Examples

**Example 1: Ambiguous Name, Context Resolved**

Suppose you search for "Li Wei" with DOB = "1990-09-18", country = "CN", ID ends in "1234".

| **Candidate** | **Name Score** | **DOB Match** | **Country Match** | **ID Soft Match** | **Total Score** |
| --- | --- | --- | --- | --- | --- |
| Li Wei (CN, 1990-09-18, ID ...1234) | 0.95 | +0.05 | +0.03 | +0.07 | 1.10 |
| Li Wei (CN, 1991-02-05, ID ...8321) | 0.94 | 0 | +0.03 | 0 | 0.97 |
| Li Wei (SG, 1990-09-18, ID ...1234) | 0.94 | +0.05 | 0 | +0.07 | 1.06 |

Here, context neatly separates the best match (first candidate) from near-ties, making scoring more interpretable and reducing errors.

**Example 2: Weaker Context, Lower Impact**

If DOB is often missing or noisy, its boost drops:

* "Jane Doe" — DOB match adds only +0.02, so the name score dominates unless context is fully reliable.

**Example 3: ID Is Highly Reliable**

If analyst feedback confirms IDs are nearly unique, raise the boost:

* "John Smith" — ID soft match adds +0.12, which quickly breaks ties when present.

Summary

* Context features break ties and help make screening decisions more accurate and explainable.
* Small boosts keep outcomes interpretable, so context helps but never overrides extremely good/bad name matches.
* Tuning lets you adapt to data reliability: make context more important as fields become reliable, less so when noisy/missing.

**7. Decision Logic (Scoring & Thresholds)**

* **Aggregate Score:** Weighted sum of name similarities and context matches.[[9]](#fn9)
* **Thresholds:**
  + *Block:* Score above high threshold signals a strong match.
  + *Review:* Medium score triggers manual investigation.
  + *Clear:* Low score is not a likely match.[[9]](#fn9)

**8. Audit Log & Reporting**

* **Audit Trail:** Logs all raw and normalized fields, similarity scores, and context comparison per match candidate.[[6]](#fn6)
* **Reporting:** Outputs in CSV/XML/JSON for compliance teams, detailing hits and decision paths.[[6]](#fn6)[[1]](#fn1)

**9. Output and Feedback Loop**

* **Screening Result:** Each record is marked as cleared, review, or blocked.
* **Feedback (Optional):** False positives/negatives flagged by humans can be used to adjust scoring or heuristics.[[1]](#fn1)[[6]](#fn6)

**Quick Recap Table (With FTS5 Added)**

|  |  |
| --- | --- |
| Step | Purpose |
| Sanctions Ingestion | Parse, normalize, store KB |
| Data Normalization | Standardize all text fields |
| KB Construction | Structured, indexed database for fast search |
| **FTS5 (Full-text Search)** | Provides instant, scalable search and candidate retrieval over millions of records using virtual tables and advanced query syntax [[5]](#fn5)[[1]](#fn1) |
| File Ingestion | Parse payment/customer data |
| Preprocessing | Normalize new input names |
| Matching | Score similarity & context; FTS5 results are filtered and ranked |
| Decision Logic | Block, Review, or Clear |
| Audit & Output | Log and report results |

FTS5 is the backbone of real-time, scalable search in this solution, providing the necessary speed and flexibility for rapid compliance screening and investigation at financial scale.FTS5 (Full-text Search) is a key component in the end-to-end name screening solution, enabling fast and scalable matching of names and aliases within the Knowledge Base (KB) by leveraging advanced text indexing and querying capabilities in SQLite.[[4]](#fn4)[[7]](#fn7)[[5]](#fn5)[[1]](#fn1)

**End-to-End Name Screening Process (with FTS5)**

**1. Sanctions List Ingestion**

* Sanctions lists are downloaded (e.g., OFAC, UN) and parsed to extract key fields such as name, aliases, date of birth, nationality, and IDs.[[2]](#fn2)[[1]](#fn1)

**2. Data Normalization and Cleaning**

* Unicode normalization, case folding, whitespace stripping, diacritics removal, and alias expansion ensure consistent representation of all fields.[[3]](#fn3)[[2]](#fn2)[[1]](#fn1)

**3. Knowledge Base (KB) Construction**

* All cleaned records are loaded into an SQLite database.
* *FTS5 Full-text Search:* The KB uses FTS5 virtual tables indexed on names, aliases, and normalized forms. FTS5 tokenizes these fields and enables advanced text queries, allowing for fuzzy, partial, and wildcard matches, dramatically speeding up the retrieval process even with millions of records.[[4]](#fn4)[[5]](#fn5)[[1]](#fn1)

**4. Ingest Payment or Customer Files**

* Payment messages and customer data files are parsed for relevant fields (names, address, nationality, DOB, IDs).[[6]](#fn6)[[1]](#fn1)

**5. Name & Context Preprocessing**

* All input fields are normalized identically to the KB to enable fair and accurate matching.[[3]](#fn3)[[2]](#fn2)

**6. Matching Step (Core Screening with FTS5)**

* FTS5 search identifies candidate matches by querying the indexed KB for close text matches to each input record.[[7]](#fn7)[[1]](#fn1)
* Candidates returned by FTS5 are further evaluated using string similarity (Levenshtein, Jaro-Winkler, Jaccard) and contextual features (DOB, country, IDs).[[8]](#fn8)[[9]](#fn9)

**7. Decision Logic (Scoring & Thresholds)**

* A weighted aggregate score of name and context similarity determines the action: block (strong match), review (partial match), or clear (no match).[[9]](#fn9)

**8. Audit Log & Reporting**

* Every action and heuristic is logged, audit trails enable traceability, and results are reported in CSV/XML/JSON for compliance teams.[[1]](#fn1)[[6]](#fn6)

**9. Output and Feedback Loop**

* Each record receives a clearance status, with human feedback optionally integrated to fine-tune future heuristic and scoring decisions.[[6]](#fn6)[[1]](#fn1)

**Table: Quick Recap (with FTS5 Detail)**

|  |  |
| --- | --- |
| Step | Purpose |
| Sanctions Ingestion | Parse, normalize, store KB |
| Data Normalization | Standardize all text fields |
| KB Construction | Indexed database for fast search |
| **FTS5 (Full-text Search)** | Enables rapid, flexible, and scalable candidate retrieval over large name datasets [[4]](#fn4)[[5]](#fn5)[[1]](#fn1) |
| File Ingestion | Parse payment/customer data |
| Preprocessing | Normalize new input names |
| Matching | Score similarity & context; FTS5 yields candidates for fuzzy matching |
| Decision Logic | Block, Review, or Clear |
| Audit & Output | Log and report results |

FTS5 provides the scale and speed needed for real-time compliance screening, ensuring millions of entities can be checked against incoming transactions instantly and reliably.[[5]](#fn5)[[4]](#fn4)[[1]](#fn1)

⁂

1. load\_kb.py
2. normalize\_sanctions.py

1. [preprocess.py](http://preprocess.py)

1. <https://www.sqlite.org/fts5.html>

1. <https://www.geeksforgeeks.org/sqlite/sqlite-full-text-search/>
2. iso20022\_preprocess\_audit.py

1. <https://www.sqlitetutorial.net/sqlite-full-text-search/>
2. features\_text.py

1. [screen.py](http://screen.py)
2. features\_context.py