Methodological Note

Stata Task

The Stata multi-table exercise was reproduced in Python (`pandas`) using the `.dta` inputs.

Approach

- **Data preparation:** All datasets (`tender`, `bid`, `bidder`) were imported, validated, and standardised. Identifiers were cast as integers to ensure clean joins, and checks confirmed that there were no duplicate tender IDs or missing city names.
- **Tenders per city and bidders per tender:** Frequency tables summarised tender counts by city and bidder counts per tender.
- **City-pair bidding flows:** Merging `bid` with `bidder` and `tender` produced origin–destination city pairs. Bid counts were aggregated by `(bidder_city, tender_city)`, showing local and cross-city activity.
- **Local experience variable:** For each potential bidder–tender pair, `local_experience = 1` if the bidder had placed a bid in the same tender city before the tender year. The earliest bid year per city is determined based on prior experience.

Challenges & Assumptions

- Stata was unavailable locally, so the procedure was executed in Python while replicating the exact Stata logic.
- In **Budapest (2016)**, two tenders shared the same year with no sequencing variable (e.g. bid date). Their temporal order was unknowable, both were conservatively assigned `local_experience = 0` for first-time bidders in that year.
- This rule avoids unsupported assumptions while maintaining internal consistency.

Python Task

The workflow was implemented in Python (`pandas`, `rapidfuzz`).

Approach

- **Normalisation:** Both sources were cleaned to lower case, stripped of accents, punctuation, and suffixes. Two standard views were created:
  - `display_name_std`: removes onomastic particles (e.g. *de*, *van*).
  - `display_name_ns`: retains them for full-string similarity.

  Middle initials were preserved.
- **Surname blocking & given-name logic:** Comparison occurred only within surname blocks (optionally two tokens for compound surnames). Given names had to match canonically, by nickname, or by initial. Missing middle names were tolerated, but inconsistent initials were rejected.
- **Aliases and fuzzy scoring:** A compact nickname dictionary (e.g. Tom–Thomas, Bill–William) generated variants. Similarity was computed via `RapidFuzz.token_sort_ratio` on both full and first–last views, the maximum score was used.
- **Thresholds and outputs:** Matches with `score` $\geq$ `90` were accepted (`match_type = fuzzy` or `exact`), keeping up to 35 candidates per member per cycle. Results include `final_matches.csv`, `unmatched_members.csv`, and `borderline_review.csv` (scores 80–89 for manual inspection).

Checks, Assumptions & Challenges

- **Specification constraints:** The `status` field was ignored by design, service years were used only as a review cue (not a hard filter) to avoid false negatives from specials/redistricting.
- **Ambiguities handled by rules:** The given name guard blocks plausible but wrong collisions such as Albert L. Smith vs Albert Alan Smith, and John E. Peterson vs John Charles Peterson, compound surnames (e.g. de la Garza) are handled via two-token blocking.
- **Borderline list for audit:** Near misses with scores 80–89 that pass surname and given name gates are exported for manual triage, several appear credible but require external identifiers (district/FEC/DOB).
- **Validation:** Row counts before/after normalisation were checked; empty election files were recorded. Small alias list can be extended (e.g. Peggy–Margaret) with no code changes.