Read Me - Harris RP (Real Property) Scrape

Harris RP (Real Property) Scrape

Version: 12.1
Date: 2025-05-26
Primary Developer(s): [Your Name/Handle Here]
AI Collaborator(s): The Masterclass Room (facilitated by Claude-3 Opus)

1. Project Overview

This script (Script 1 of Project Phoenix) is designed to identify real property re cords potentially associated with deceased individuals. It takes a list of probat e leads as input, searches the Harris County Clerk Real Property online portal, extracts detailed information about property transactions, and outputs a struct ured, flattened CSV file. This output serves as the primary input for Script 2, w hich performs advanced record linkage and relevance scoring.

The core challenge addressed by this script is the automated and robust extra ction of data from a web portal that presents information in multiple, sometime s inconsistent, formats, particularly concerning legal descriptions and party d etails.

2. Core Functionality

*   **Input:** Reads probate leads from a specified CSV file (default: `harris_s ample.csv`). Expects columns like `decedent_last`, `decedent_first`, `filing_dat e`, `case_type_desc`, etc.
*   **Tiered Web Scraping:**
    *   Navigates to the Harris County Clerk Real Property portal.

    *   Employs a multi-tier search strategy for each decedent to maximize relev
ant hits while managing search scope:
        *   **Name Standardization:** Cleans input names (uppercase, removes
suffixes, standardizes first name part for searching).
        *   **Tier 1:** Searches `LAST_NAME STANDARDIZED_FIRST_NAME_PAR
T` (e.g., "SMITH JOHN").
        *   **Tier 2:** Searches `LAST_NAME FIRST_INITIAL` (e.g., "SMITH J").
        *   **Tier 3:** Searches `LAST_NAME` only (configurable, typically for les
s common surnames).
        *   **Grantor-Focused:** Searches primarily target the decedent as a "Gr
antor."
        *   **Date Range:** Searches within a configurable window (+/- 1 year by
default) around the probate filing date.
        *   **Pagination:** Handles multiple pages of search results (configurable
maximum pages per tier).
*   **Data Extraction:**
    *   **Real Property (RP) Document Info:** File Number, Document Filing Dat
e, Instrument Type.
    *   **Party Information:** Parses Grantors, Grantees, and Trustees. Prioritiz
es structured sub-row data if available, with a robust fallback to parse concat
enated party strings from the main "Names" column of a record.
    *   **Legal Description:** Uses a hybrid approach:
        1. Attempts to parse structured HTML sub-tables.
        2. If HTML table parsing fails or is incomplete, scans multiple subsequen
t table cells for plain-text legal descriptions.
        3. Extracts detailed fields (Desc, Lot, Block, Sec, Subdivision, Abstract, S
urvey, Tract) using refined regular expressions with lookaheads.
        4. Includes post-processing to clean common suffixes (e.g., "Related Do
cs").
*   **Output:**
    *   Generates a CSV file (e.g., `harris_rp_targeted_matches_YYYYMMDD_H
HMMSS.csv`) in the `data/targeted_results/` directory.
    *   **Flattened Structure:** Each row represents a single party's involvemen
t in a single property transaction. Common property details are repeated.
    *   **Contextual Enrichment:** Output rows include key data from the input
probate lead and metadata about the search process (e.g., search tier, search

term used).
    *   **Column Naming Convention:** Uses `probate_lead_...` for fields from t
he input CSV and `rp_...` for fields scraped from the Real Property portal for cl
arity.
*   **Resilience & Debugging:**
    *   Configurable retries for search operations.
    *   Form state resets between search tiers.
    *   Extensive timestamped console logging.
    *   Screenshots on error.
    *   HTML page dumps for debugging specific search results.

3. Setup and Installation

Prerequisites:
*   Python 3.7+
*   Playwright library and its browser drivers.

Installation:
1. **Clone the repository (if applicable) or download the script.**
2. **Install Python dependencies:**
   ```bash
   pip install pandas playwright beautifulsoup4
   ```
3. **Install Playwright browser drivers** (if running for the first time):
   ```bash
   playwright install
   # or playwright install chromium
   ```

4. Configuration

Key configurations are at the top of the script (`.py` file):

*   **`INPUT_PROBATE_LEADS_CSV`**: Path to the input CSV file containing p
robate leads.
    *   **Required columns (example):** `decedent_last`, `decedent_first`, `filing

_date` (formats like MM/DD/YYYY, YYYY-MM-DD accepted), `case_type_desc`, `county`, `case_number`, `subtype`, `status`, `signal_strength` (these latter ones are used for enriching output).
*   **`OUTPUT_DIR`**: Directory where output CSVs and debug files will be saved.
*   **`TIER_SETTINGS`**: Dictionary to control the tiered search:
    *   `enable_tier_3`: Boolean, to enable/disable "Last Name Only" searches.
    *   `max_pages_per_tier`: Integer, max number of result pages to scrape per search tier.
    *   `common_surnames`: Set of strings, surnames for which Tier 3 search will be skipped.
*   **Various Timeout Constants:** (e.g., `DEFAULT_ELEMENT_TIMEOUT`, `PAGE_LOAD_TIMEOUT_INITIAL`) can be adjusted if needed for different network conditions.
*   **`MAX_ROWS_TO_DEBUG_HTML`**: Controls how many initial records per page get detailed row structure logging.
*   **`STOP_AFTER_FIRST_SUCCESSFUL_LEAD`**: Boolean (in `run_targeted_rp_scrape`), useful for testing. Set to `False` for full runs.

5. Usage

Run the script from the command line:

```bash
python your_script_name_v12.1.py
```

- The script will process leads from the `INPUT_PROBATE_LEADS_CSV`.
- Output will be saved to a timestamped CSV file in the `OUTPUT_DIR`.
- Console logs will provide detailed progress and debug information.
- Debug screenshots and HTML dumps may be created in `OUTPUT_DIR` on errors or specific events.

6. Output CSV Columns (Key Fields)

The output CSV will contain rows flattened by party, with columns including:

- **Probate Lead Information (prefixed with `probate_lead_`):**
  - `probate_lead_county`
  - `probate_lead_case_number`
  - `probate_lead_filing_date`
  - `probate_lead_decedent_first`
  - `probate_lead_decedent_last`
  - `probate_lead_type_desc`
  - `probate_lead_subtype`
  - `probate_lead_status`
  - `probate_lead_signal_strength` (original signal strength from input)
- **Real Property Record Information (prefixed with `rp_`):**
  - `rp_file_number`
  - `rp_file_date`
  - `rp_instrument_type`
  - `rp_party_type` (Grantor, Grantee, Trustee, N/A)
  - `rp_party_last_name`
  - `rp_party_first_name`
  - `rp_legal_description_text`
  - `rp_legal_lot`
  - `rp_legal_block`
  - `rp_legal_subdivision`
  - `rp_legal_abstract`
  - `rp_legal_survey`
  - `rp_legal_tract`
  - `rp_legal_sec`
- **Search Metadata & Scoring:**

- `rp_signal_strength` (calculated score for the RP record's relevance)
- `rp_found_by_search_term` (the actual name string used in the portal search)
- `rp_search_tier` (TIER_1, TIER_2, or TIER_3)

7. Key Functions & Logic Flow

- **`run_targeted_rp_scrape()`**: Main orchestration function. Reads leads, iterates through them, calls search functions, and handles final DataFrame creation and CSV output.
- **`search_rp_for_decedent_and_extract()`**: Orchestrates the search for a single lead, including retries and calling the tiered search execution. Enriches results with lead context.
- **`execute_tiered_rp_search()`**: Manages the TIER_1, TIER_2, TIER_3 search logic for a lead.
- **`_execute_single_search()`**: Performs a single search on the portal for a given name and tier, including form filling, clicking search, and handling pagination of results. Calls `extract_data_from_current_page_rp`.
- **`extract_data_from_current_page_rp()`**: Core data extraction logic for a single page of results. Identifies main record rows and initiates parsing for basic info, legal descriptions, and parties. Implements the flattening logic.
- **`extract_legal_description_from_html_table()`**: Parses structured HTML (`<table>`) legal descriptions using BeautifulSoup.
- **`parse_plain_text_legal_description()`**: Parses legal descriptions from plain text strings using refined regex.
- **`parse_parties_from_names_column()`**: Fallback parser for concatenated party strings found in a single "Names" column.
- **Helper functions:** `standardize_name_for_search`, `parse_probate_filing_date_from_input`, `compute_signal_rp_score_for_record`, etc.

8. Known Limitations / Future Improvements (for this script)

- **Party Parsing from `k_sub_loop`:** The current `k_sub_loop` (iterating through `<tr>` elements *after* a main record row) is designed for 2-cell "Label: Value" format. Its effectiveness for parsing party data in this specific site's varied sub-row structures could be further investigated if the `parse_parties_from_names_column` fallback proves insufficient for some record types. However, current evidence suggests the fallback is the primary successful mechanism.
- **"Exploded" Legal Descriptions in Sub-Rows:** While the `k_sub_loop` now attempts to catch these, their prevalence and the completeness of this capture versus the plain-text scan (Attempt 2) could be further analyzed. It currently acts as a tertiary gap-filler.
- **HTML Table Detection (Attempt 1 for Legal Desc):** The heuristic for identifying a "Type A" HTML legal description table is robust but might require tuning if new variants of such tables are discovered.

9. Next Steps (Project Phoenix - Script 2)

The output of this script (v12.1) will be the input for **Script 2: Probate Lead Matching & Scoring**. Script 2 will focus on:

- Advanced fuzzy name matching (`thefuzz`) between `probate_lead_...` names and `rp_party_...` names.
- Implementing probabilistic record linkage principles (e.g., Fellegi-Sunter concepts).
- Developing a more sophisticated overall linkage/relevance score.
- Producing a final, scored, and match-indicated dataset.

How to Use This `README.md`:

1. Save this content as a file named `README.md` in the same directory as yo
ur `your_script_name_v12.1.py` script.
2. Fill in `[Your Name/Handle Here]`.
3. If you put your script(s) in a Git repository (e.g., on GitHub, GitLab), this `RE
ADME.md` will be automatically displayed on the repository's main page, maki

ng it instantly accessible.
4. You can provide this file or a link to it to your new "Project Manager" AI inst ance along with the Phase 3 prompt.

This documentation provides a solid overview for anyone needing to understa nd, use, or maintain this script. It covers the "what, why, and how" effectively.