Project 1: Web Scraper for County Recorder Portals

Background: Evictorbook.com, which is currently available for San Francisco and Oakland (in Alameda County), attempts to hold serial evictors accountable by matching eviction records to the property owner at the time of the eviction. To do so, the site relies on access to sales transfer data that contains the history of ownership transfers for every single parcel in a given county. While such data can be purchased from 3rd party vendors, it is prohibitively expensive, even though such data are public records. To allow Evictorbook to continue, we need to build web scrapers to collect this data from County Recorder's offices, which often maintain public portals that allow searches for records by date and type of record.

Deliverable: Web scraper(s) for the following counties (in ranked priority):

- San Francisco (link)
  - **Example Search:** Use Advanced Search, use Titles type "001 - DEED", adjust Document Date range
- Alameda (link)
  - **Example Search:** Change Date Filed From and To, select Document Type "DEED"
- Contra Costa (link)
  - **Example Search:** Select Document Type, adjust Start and End Dates, and select Document Type "DEED"

Requirements:

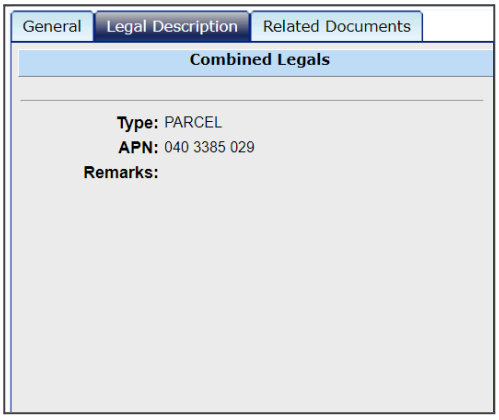
- Run parameters can be adjusted to limit search for 1) a specific date range, and 2) a list of recorded document types
- At least the following fields scraped: Identifier for transaction (e.g. instrument #), all grantor and grantee names (separated into individual records), recording / filing date, parcel identifier (e.g. APN), document type (e.g. deed). Recommended table schema (see example below):
  - **transactions**: instrument_no (int), recording date (datetime), parcel ID (string), document type (string)
  - **grantors**: instrument_no (int), grantor (string)
  - **grantee**: instrument_no (int), grantee (string)
- Scraped results written to CSVs
- Written in Python (e.g. Selenium or BeautifulSoup)
- Well-defined errors raised for common cases (e.g. network timeout, maximum search results reached), to be able to re-run scraper with adjusted parameters if needed
- Best practices to minimize scraping impact (e.g. delays between requests) and to avoid rejection by server (e.g. rotating IPs)
- Ability to re-start scraping run mid-way (e.g. parameters allow for ignoring the first n records), in the event of a failed run
- Code is readable, formatted, and well-documented to be used by a volunteer contributor base in a production codebase

Preferences:

- Use of data schema (e.g. Pydantic)
- Option to load data into a PostgreSQL database using SQLAlchemy
- Documentation of most successful run parameters (e.g. at what volume do runs begin to fail, is running small batches throughout the day more successful, etc.)

Example

For this recorded document from the Alameda County Recorder's office:



The ideal output would look like:

transaction_id	recording_date	parcel_id	record_type
2024031564	03/04/2024	"040 3385 029"	"DEED"

transaction_id	grantor
2024031564	"DEPINEDA ELBA Y MARTINEZ"
2024031564	"PINEDA RICARDO"
2024031564	"BURGOS ANGELA DOMITILA"

transaction_id	grantee
2024031564	"DEPINEDA ELBA Y MARTINEZ"
2024031564	"PINEDA RICARDO"

Project 2: Testing Framework for Business Networks

Background: Evictorbook creates business networks from state business filings to reveal the humans behind shell companies that own properties (see an example here). This is possible because businesses are registered with officer / CEO names and addresses, which can be used

to create a graph, where the nodes represent either businesses, officers, and addresses, and the edges represent when the two entities appear in the same business filing. Using this, we can create networks where each business, officer, and address is associated with an ID for the network it is part of. Evictorbook has created a method for producing these IDs, which involves filtering out edges and nodes that are “misleading” in the sense that they are not actually part of the same network (a common example are addresses that are the office locations for business filing services or law firms, which prepared the original business filing).

We need a testing framework for these outputs that will allow the Evictorbook team to quickly evaluate and verify the network outputs. Some examples include:

- Generating diffs of the network outputs between runs. *This should ignore changes to network IDs between runs*
- Generating high-level metrics of the networks for a given run (e.g. statistics about network sizes and composition)
- Using heuristics to flag nodes or edges that should be manually reviewed (e.g. high-degree / centrality / etc. nodes, bridge edges, etc.)
- Validating run results against manually labeled data (e.g. ground-truthed data on which nodes should be in the same network)

This task can be further defined and scoped if the team is interested and able to take this on.