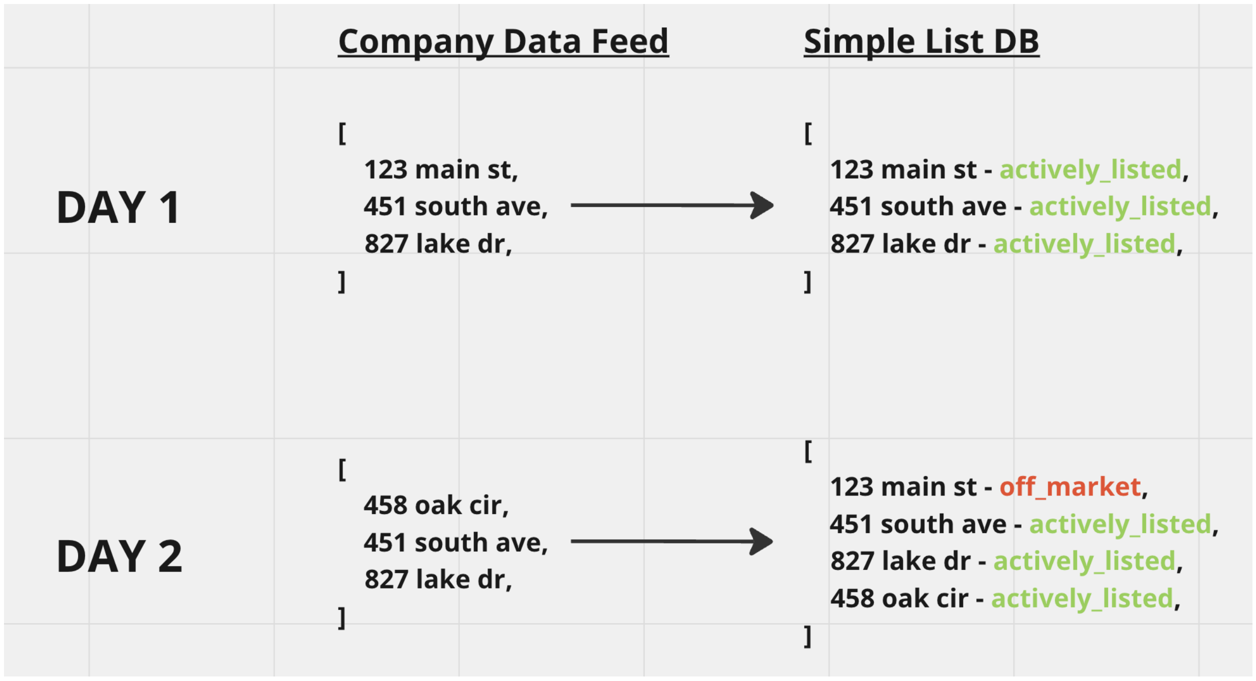
PS Code Challenge 1 - The following code challenge is intended to assess candidates on their ability to handle and clean multiple large datasets. The situation is as follows:

You run a real estate listing platform called Simple List, where customers can advertise their home on your platform for a fee. You have a few customers that want to automate the process of uploading listings, so you need to create an easy way for them to send you their data.

You decide to create an API that allows customers to easily send their data feed of properties via JSON to a single endpoint. Each day, this list of properties contains **only their actively listed properties**.

For example, a property that shows up in the feed on Day 1 might not be in the feed on Day 2 because the property gets rented and no longer needs to be advertised. You still want to store this property in your system, but you will need to keep track of its status as being "off_market" or "actively_listed".


Company Data Feed
Simple List DB
DAY 1
[123 main st, 451 south ave, 827 lake dr,]
[123 main st - actively_listed, 451 south ave - actively_listed, 827 lake dr - actively_listed,]
DAY 2
[458 oak cir, 451 south ave, 827 lake dr,]
[123 main st - off_market, 451 south ave - actively_listed, 827 lake dr - actively_listed, 458 oak cir - actively_listed,]

In the example above, notice how on Day 1 all the properties are considered "actively_listed" because they are being added for the first time. On Day 2, however, notice how "123 main st" does **not** appear in the company data feed. Thus, its status is reflected as "off_market" in the Simple List system.

The address "458 oak cir" is also a new address on Day 2, so it gets added as an "actively_listed" property in the Simple List system.

All addresses in the Simple List system are unique - no duplicates are allowed. You'll need to use the address of each property to handle duplicates as well as for updating the marketing status of each property. The full list of attributes that Simple List stores for each property is as follows:

Simple List Property Schema

address: string - street name and number only (lowercase)

market:?? string - the closest market to the property (camel-case & lowercase)

subMarket: string - the city of the property (camel-case & lowercase)

state: string - one of the 50 enforced US state abbreviations (lowercase)

zipCode: string - the zip code of the property, in 5 or 9 digit format

company: string - the name of the company that owns the property

numBeds: number - the number of beds

numBaths: number - the number of baths

squareFeet: number - the square footage of the property

price: number - the monthly rental price

description: string - a description of the property

images: string[] - a list of image URLs of the property

latitude: number - the latitude of the property

longitude: number - the longitude of the property

dateAdded: Date - the timestamp of when the property was first added

* does not include the "status" attribute ("actively_listed" vs. "off_market"), which is determined once the property has been added.

?? The closest Simple List market to the property, based on geodistance.

It will be up to you to map the above attributes properly. You **can** guarantee that the feed sent by the company will contain all the data you need to populate each of these fields, but you **cannot** guarantee that the typing, formatting, etc. will be consistent with the Simple List system.

Additionally, each time a new property is added to the Simple List system, it must be grouped into one of several of Simple List's **markets**. The starter code will provide you with this list of markets, in addition to the latitude/longitude values associated with each market.

If the company feed contains this "market" attribute AND the value for the "market" attribute is in the list of Simple List markets, you can directly map this field. However, if the field does not exist or the value cannot be found, you will need to calculate the closest market by geodistance using the latitude and longitude values of the property.

You may use any language of your choice to build the API. The starter code will provide you with three days worth of company feed data across three different companies. Instead of connecting to a database, you will instead capture a snapshot of the Simple List "DB" as a JSON file at the end of each day.

The starter code has provided this entire flow for you, all you have to do is include a call to the API you build. The POST endpoint you create should return the list of properties that are written/updated to the Simple List system.

Hint: In order to track the status of each property ("actively_listed" vs. "off_market"), use the previous day's snapshot to compare old vs. new properties.

Summary

Build an API using a language of your choice that contains a single endpoint capable of ingesting a data feed of properties. Using the Property Schema as a guide, map the properties from the company feed to the Simple List system, taking into account the following edge cases:

1. Track the status of each property as "actively_listed" or "off_market"
2. No duplicate addresses are allowed, especially across different companies
3. If the "market" attribute exists in the company data feed and the value of this field is one of the Simple List markets, map the field directly. Otherwise, determine the closest market by geodistance
4. Ensure typing is compliant with the Property Schema

You'll find the company data feeds in the "data" directory of the starter code - all are formatted in JSON. The "main.py" will provide you with the expected flow of events. All you need to do is include the call to the API endpoint you build.

Bonus - GET by geodistance

As an added benefit of creating an API, you want to be able to fetch a list of the closest properties based on a set of coordinates (latitude/longitude). Create a GET endpoint that fetches all properties in the Simple List system within a 65 mile radius of a given set of coordinates.

Use your snapshot from Day 3 as the data source for this endpoint.

Submission

Once you have completed the task(s), please zip the repository (including starter files) and email your completed code to llind@propertyshield.co. Good luck!