

ETL Ornithology Project

Tom Levy

For this project, I chose to investigate an old flame from my childhood: ornithology. I have always admired birds, and when I was younger I consistently strived to identify bird species from just looking at a picture, as well as going birdwatching on occasion. While I fell out of love with ornithology, this project has become my attempt at rekindling my old flame, and the steps I have taken to approach this in the context of everything I have learned so far in this bootcamp through the ETL Process.

Extract

For this project, I have used four different datasets, and have used three methods to extract data used in this project. Two datasets were from Kaggle and pertain specifically to bird songs, and for these two datasets I extracted the data using the pandas read_csv method since both were in csv format. One of these datasets was based off of data from the Xeno Canto website, a public repository for bird song recordings, which measures factors such as bird species, bird locations, song lengths, etc. The other dataset was taken from a challenge for students in Cornell University's Ornithology Lab, which provided a summary of all the audio provided in the dataset, such as file type, rating of sound quality, bird species, etc. Outside of the CSVs, the extraction process got hairier. Cornell also has a site called ebird.org which contains observation API data. Instead of a JSON read using requests.get, the site provides a Python package called "ebird-api" which uses the function "get_observations" to print out observation data based on an API key and location, amongst other possible factors (see Fig. 1). Unfortunately, this is not a stable package and cannot handle large data extraction so my data for this part of the project had to be constrained to one week's worth of observations. My last method of extraction was using BeautifulSoup and Splinter together to extract a table of common names and scientific names of bird species from the Audubon website's bird guide (see Fig. 2).

Fig. 1 (top) and Fig. 2 (bottom)

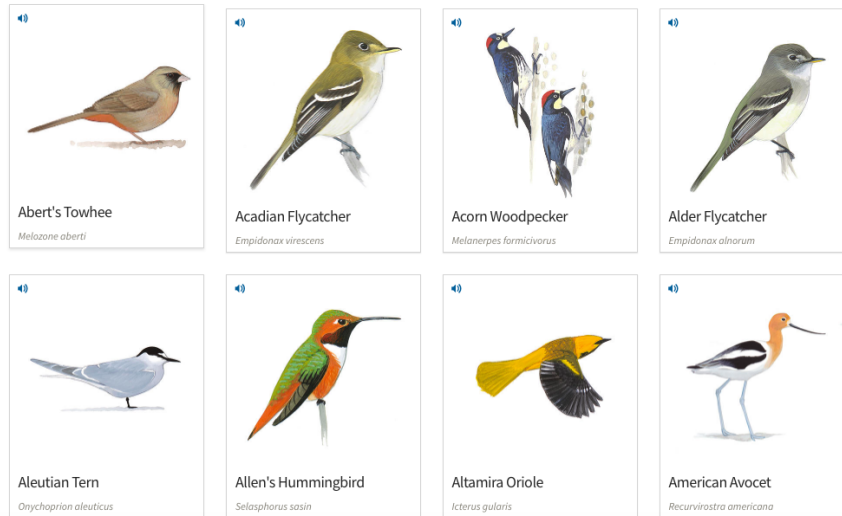
```
from ebird.api import get_observations
import pandas as pd
from sqlalchemy import create_engine
```

E-Bird API Extraction

```
records = get_observations("apikey", 'US', back=7)
```

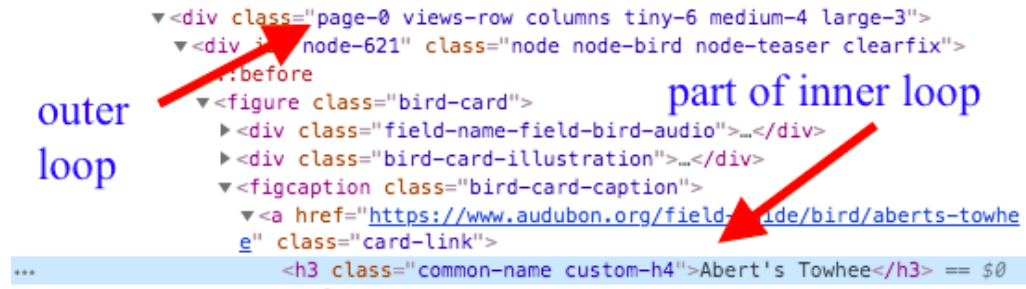
```
records
```

Some of Our Favorite Birds



Transform

For the Kaggle datasets, I needed to reduce the number of columns as there were many columns that contained extraneous information. For the Xeno Canto dataset, I kept the ID, genus, species, English name (common name), latitude and longitude coordinates, song length, and song date. Before eliminating duplicate rows and missing values, I transformed the song length column to fit a HH:MM:SS timestamp format using `rjust` and song date columns to fit a date object using the `datetime` library. For the Cornell dataset, I kept the ID, rating, species, scientific name (genus and species), latitude and longitude coordinates, type of song, elevation, and bitrate. Before eliminating duplicate rows and missing values, since units were included in the bitrate and elevation, I had to split these entries and convert them from string to integer types. Duplicate rows and missing values were then omitted from the final tables prior to the load step. For the API data, converting my observation JSON records to a dataframe was made extremely simple thanks to the `pd.DataFrame.from_dict` function. For the Audubon site, I was able to retrieve every div containing a class based on the page number (my outer loop) coupled with a bootstrap grid and from there, I obtained each `h3/p`, from which I extracted the text. I ended up with two lists, one containing the common name and another containing the scientific name, and created a dataframe from those lists.



```

▼<div class="page-0 views-row columns tiny-6 medium-4 large-3">
  ▼<div id="node-621" class="node node-bird node-teaser clearfix">
    .before
    ▼<figure class="bird-card">
      ▶<div class="field-name-field-bird-audio">...</div>
      ▶<div class="bird-card-illustration">...</div>
      ▼<figcaption class="bird-card-caption">
        ▼<a href="https://www.audubon.org/field-guide/bird/aberts-towhee"
          e" class="card-link">
***
      <h3 class="common-name custom-h4">Abert's Towhee</h3> == $0

```

Load

All my datasets were loaded into SQL using the SQLAlchemy create_engine method to establish the connection and then using the df.to_sql method to load the datasets into the schema. I created four schemas in SQL's pgAdmin tool for each of the datasets. All of these steps in the load process took place in my schema SQL file. I then joined both Kaggle datasets with each other based on the ID primary key and joined both my EBird API and Audubon BeautifulSoup data sets based on the common name for the bird species in my query SQL file.