## Finding Data Your project must use 2 or more sources of data. We recommend the following sites to use as sources of data:data.worldKaggleYou can also use APIs or data scraped from the web. However, get approval from your instructor first. Again, there is only a week to complete this!Data Cleanup & Analysis Once you have identified your datasets, perform ETL on the data. Make sure to plan and document the following:The sources of data that you will extract from.The type of transformation needed for this data (cleaning, joining, filtering, aggregating, etc).The type of final production database to load the data into (relational or non-relational).The final tables or collections that will be used in the production database.You will be required to submit a final technical report with the above information and steps required to reproduce your ETL process.Project Report At the end of the week, your team will submit a Final Report that describes the following:Extract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).Transform: what data cleaning or transformation was required.Load: the final database, tables/collections, and why this was chosen.Please upload the report to Github and submit a link to Bootcampspot.

## ETL Project Transformation

**Data Sources Used**

| **Dataset** | **Description** | **Extract Method** | **Transformations Performed** |
| --- | --- | --- | --- |
| Boston Crime Data  <https://www.kaggle.com/AnalyzeBoston/crimes-in-boston>  **Original Format:** csv | (from Kaggle)  Crime incident reports are provided by Boston Police Department (BPD) to document the initial details surrounding an incident to which BPD officers respond. This is a dataset containing records from the new crime incident report system, which includes a reduced set of fields focused on capturing the type of incident as well as when and where it occurred.  Records begin in June 14, 2015 and continue to September 3, 2018. | **Manual pull**: data from Kaggle website | After the data was pulled from Kaggle via .CSV format it was converted to a Pandas Dataframe. After scrubbing the dataframe the individual dates were deemed unusable since they didn’t include the “day” so it was decided to scrap those individual columns and keep 'OCCURRED\_ON\_DATE' so that it can parsed as need in future steps.  Various columns were removed. These were kept:   1. 'OFFENSE\_CODE' 2. 'OFFENSE\_CODE\_GROUP' 3. 'OFFENSE\_DESCRIPTION' 4. 'DISTRICT' 5. 'REPORTING\_AREA' 6. 'SHOOTING' 7. 'OCCURRED\_ON\_DATE' |
| Zip Codes  <https://www.unitedstateszipcodes.org/ma/>  **Original Format:** html | Website which lists US postal zip codes based on query criteria entered (we leveraged the URL which queried zip codes for the state of MA) | **Web Scrape:** Scraped the URL to harvest the zip codes from the website along with associated information needed (City, State, County etc)  **Scraping Code:** Web\_Scrape\_Zipcodes.ipynb | I had to call the csv parser directly to force Pandas not to truncate the leading zero in the zipcodes. Once done. No further transformation needed |
| Boston Police Districts  <https://www.boston.gov/departments/police#districts>  **Original Format:** html | Website Lists Police Districts with associated District Codes and District Names | **Web Scrape:** Scraped the Website for District Codes and Precinct Addresses  **Scraping Code:** Scrape Districts.ipynb | After pull of districts from the website, broke up the grouping of A1 and A15 in the code so they could be represented in our dataset as two separate districts and locales  Once loaded into a DataFrame I had to strip the district names to isolated the proper district names for a future join with the Census data. |
| Demographic Stats  ACS API  <https://www.census.gov/data/developers/data-sets/acs-5year.html>  **Original Format:** json | Leveraged the ACS API Census to pull ethnic and race demographics for the key Boston zip codes identified | **API Pull:** Leveraged the ACS API to pull data for summarized groups by race/ethnicity per census specification  **API Code:** Census API.ipynb | Census data had zipcode field so I had to call the csv parser directly to instruct to deal with this field as string. I also had to employ DROPNA to Clean the dataset |

**Load Method**

Ultimately, python was used to create “clean” datasets via the transforms described in the specs above. The dataframes created in python were exported to csv files (in the Database\_Files folder in our repo).

A python script was written to push the “clean” dataset exports to a postgres DB (**DB Code:** Create Postgres Tables.ipynb)

To join our datasets in a meaningful manner, we created a SQL view. We chose a view so the joined view could be dynamically updated in the event that our datasets were refreshed

**SQL for view:** Join\_SQL

The dataset from our join query can be found in the repo. As part of our join, we omitted crime statistics which did not have a recorded location (as this would not allow us to get the associated demographic info) – this was 1,765 records out of 319,073.

**Dataset:** Joined\_Master\_Dataset

***We saved the SQL source file for the postgres DB in the repo—can be used to replicate the DB***

**Lessons Learned and Next Steps**

1. We finally got to use the Census API—now that we understand how it works and the amount of data to be found, we are looking forward to using it further
2. The hardest part of this project was meaningfully correlating Boston Police Districts with geographic locations—we relied heavily on zip codes and scraping but we would like to explore using lat/lon coordinates to get more accurate. This would be good practice for other geographic districts that are arbitrary and dynamic (congressional districts?)

**Geetha Added 10/29/2019**

1. **Download, scrape data from data sources (creates csv files referenced)**
2. **Push scraped and downloaded data to postgres:** Create Postgres Tables.ipynb
3. **Use Join SQL to run against the created DB:** Join\_SQL
4. **Final Output of the Join SQL:** Joined\_Master\_Dataset