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.

**Overview:**

For this project, we were interested in seeing what kind of information there is available on mass shootings and crime rate. Once we gathered the data, the next step for a fully built out project would then be to see if there is any correlation between areas of crime and mass shootings.

We had two datasets, one mass shooting dataset which came from Mother Jones through Data World,

<https://data.world/awram/us-mass-shootings>,

as well as our crime dataset which again came from Mother Jones through Data World,

<https://data.world/carlvlewis/u-s-metro-areas-violent-crime-rates-by-type-1970-2015>.

Within both datasets we knew that we had to clean up the data to only select columns we thought were important. Since we are only looking at raw numbers, summaries of the shootings or descriptions aren’t necessary. Using SQL, we limited our columns on the shooting side down to fatalities, injured, and total victims, then exported our new information as a CSV. From this cleaned up data we then wanted to separate by years. Taking our pandas data frame with the CSV data we then reduced it by the years we looked at by creating a new data frame based on a list of dictionaries. With this new data frame, we ran a python command to then insert the data into a mongo database.

**Extract:**

<https://data.world/awram/us-mass-shootings>,

<https://data.world/carlvlewis/u-s-metro-areas-violent-crime-rates-by-type-1970-2015>

Both links were queried from Data.World using SQL and downloaded/formatted as CSV files.

**Transform:**

Ran SQL queries to filter our spreadsheets to show only the columns we needed.

Loaded CSV files into a Pandas data frame with which we could manipulate the data further.

Created a new data frame based on a list of dictionaries highlighting each year’s relevant data.

Joined the two cleaned up data frames to combine information.

**Load**:

Ran a Python script with the module Pymongo to insert our information into a mongo database.

We felt a non-relational database would work best for this project because using Mongo allowed to have our data as a JSON file or a dictionary for easier importing and exporting. We felt it was best to have more control over our data as a dictionary (CRUD) and upload into MongoDB.

<https://www.pluralsight.com/blog/software-development/relational-non-relational-databases>