The purpose of this report is to explain the process of **E**xtracting, **T**ransforming, and **L**oading the datasets from the California Health and Human Services dept. (sourced through Data.World)

The data sets can be found here:

Transportation to Work –

* Data.World: <https://data.world/chhs/b1008bb6-2f54-4b49-a0e0-c4d1bc9440ff>
* CHHS: <https://data.chhs.ca.gov/dataset/transportation-to-work-2000-2006-2010>

Road Traffic Injuries –

* Data.World: <https://data.world/chhs/e0216fbb-3739-4d92-9630-88d9f5686ac6>
* CHHS: <https://data.chhs.ca.gov/dataset/road-traffic-injuries-2002-2010>

**Extract: CSV, Quickdatabase diagrams.**

We acquired the data and imported it to Jupyter Notebook to filter the dataset and create our database; then we loaded our database into post-gres SQL because of how easy it easily translates to through the interface and the management for this mid-to-large dataset.

We pulled the datasets using pandas and cleaned the data by regrouping: counties, region, transportation modes, and race and ethnicity.

**Transform & Clean:**

We first noticed that a large amount of rows contained null values, consequently, we filtered that out by dropping null values but showed too extreme of a measure; logically we wanted to keep the rows in order for us to create tables and/or group parameters to best represent relations between traffic injuries and modes of transportation. The dataset was split based on severity of injury – fatal injury and severe injury. We had to group the traffic injuries by severity and county name; then reformat the data to the population count that was organized based on county and severity of injury. This resulted in producing a table of *‘x number of rows and two columns*’, the rows for county and columns for severity of injury and then saved (isolated) severe injury column because it had numerical value and the other column had null values.

Population data was required from the traffic injuries dataset, which turns out it didn’t have total population per individual county; it measured population for individual cities within counties thus only show the population of each state. the numerical value was for a specific state instead of specific data for each individual county.

Re-indexing was used not to change the dataframe, but to drop *‘modes\_of\_transportation’* index to start from zero by locating null values to replace with zero.

We were only concerned with data corresponding to counties, so we disregarded rows with data corresponding to other geotypes. We used the *‘.isna’* function to show the null values but mapped them to *‘False’* values by using *‘.loc’* function to locate them; then replace those *‘False’* values with zeroes by using *‘.fillNa’.*

The dataframe tables and columns were renamed in accordance with the SQL schema to keep relevance. With the SQL Schema we renamed some columns to closely relate to the data set legend and description, for example: *‘injury\_mode’* was changed to *‘injury\_transportation\_mode’* so that the label is almost self-explanatory thus being relevant with a logical connection to the data set.

**Load the Database:**

Loading the csv converted DataFrame into post-gres SQL was our choice of database because the ease of translation.