**INTRODUCTION**

The purpose of this report is to explain the **ETL** (**E**xtracting, **T**ransforming, and **L**oading) process performed on the following datasets:

* [Transportation to Work](https://data.chhs.ca.gov/dataset/transportation-to-work-2000-2006-2010/resource/fc5eafc8-0648-4a30-9bf5-e06f09e8320d) (TTW)
* [Road Traffic Injuries](https://data.chhs.ca.gov/dataset/road-traffic-injuries-2002-2010/resource/cdb50347-6fe1-456e-a336-d7daf0aba595) (RTI)

Both were acquired from the [California Health and Human Services dept.](https://data.chhs.ca.gov/) via data.world.

**EXTRACT: CSV, QUICKDATABASE DIAGRAMS**

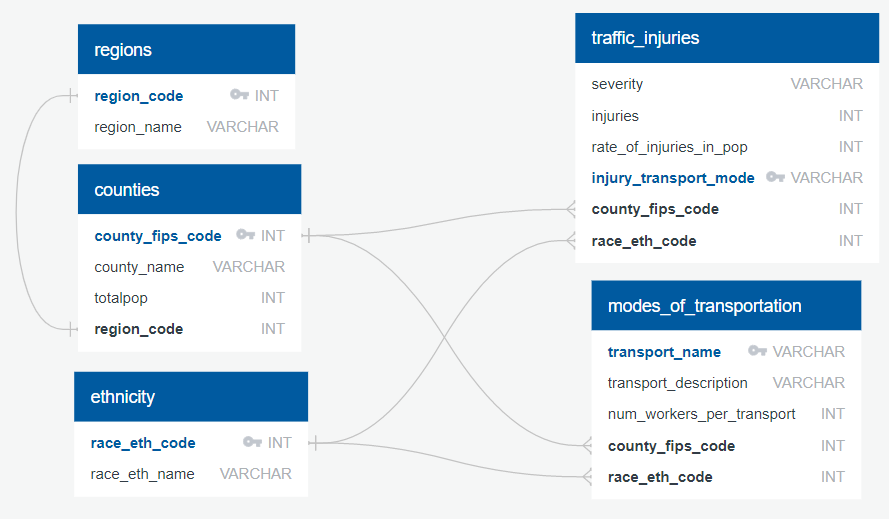
We downloaded the datasets and imported them into Jupyter Notebook from a local folder, so as to explore and filter the datasets as well as create our database. Following the data processing, we loaded our database into post-gres SQL because its interface is ideal for the management of this mid-to-large dataset.

We read the CSV files using pandas (­) and proceeded to transform the data.

**TRANSFORM & CLEAN:**

We first noticed that a large number of rows contained null values. Upon closer examination, we realized that these null values were constrained to three particular columns: , , and . In the (population rate) column, null values were meant to indicate that 0% of the population met a given condition. Consequently, we replaced all null values with 0. As for the and columns, they contained null values in the rows collecting data applicable to a different type of geographic region (e.g. state, census tract, etc., NOT counties). For simplicity’s sake, we chose to construct our database with information specific to counties only.

Our database included 5 tables, and there were related in the manner depicted below:



The regions, counties, and ethnicity tables were constructed in the following manner:

1. We grouped TTW by , , and and created a new dataframe from this group.
2. We grouped TTW by and and created a new dataframe from this group.
3. We grouped TTW by and and created a new dataframe from this group.
4. We performed steps 1-3 on RTI, and then performed three outer joins.
   1. To create , we joined the appropriate dataframes on .
   2. To create , we joined the appropriate dataframes on .
   3. To create , we joined the appropriate dataframes on .
5. Population data () for the table was acquired from RTI. This dataset contained total population per city, rather than total population per county. To compute the latter, we grouped TTI by and used the function.
6. We created a new dataframe with and columns, and joined it to the table on .
   1. Note that some rows in contain null values. This is due to the fact that RTI included fewer counties than TTW, and TTW did not have any population data.

The steps below outline the process used to construct the and tables.

1. was built by filtering RTI for the desired columns.
   1. The first 4 columns were chosen because they were the most uniquely descriptive statics in the data set.
   2. The last 2 columns were included so as to link with the three previously created reference tables.
2. was built by filtering TTW for the desired columns.

As is evident, we stored our data in a relational database. It seemed an ideal choice given that it would allow us to craft a well-defined architecture.

**LOAD: DATAFRAMES TO SQL**

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