Data Analytics Bootcamp

toronto BICYCLIST SAFETY  
etl projeCt report

Alexis Summey, Andrew Davis, Ellen Colborn, Justin Butler, Timisha Martin, Scott Caldwell

## The city of Toronto has increased the number of established bike lanes on public roads, and the number of pedestrian and bike related accidents has significantly dropped in the past five years. In light of observable trends and public transportation improvements, relevant data was extracted, transformed, and loaded into a database, much like it would be processed in industry.

## Extract

|  |  |
| --- | --- |
|  | Before being able to manipulate the data, it had to be found and downloaded. There were several good sources for data with relevance to this project. |

1. **Toronto Collision Data CSV**

The first data found was a conglomeration of traffic collision data including motor vehicles, pedestrians, and bicycles. The file was available as a CSV on Kaggle and was downloaded as such. The CSV file was successfully loaded into a Jupyter notebook using the “read\_csv” function in pandas. This resulted in a dataframe containing all types of collision data with all possible fields included.

1. **Red Light Camera Location CSV**

Much like the Toronto Collision data, the red light camera location was available as a CSV file. The file was downloaded from Toronto’s Open Data website and was loaded into a Jupyter notebook in the same way. The resulting dataframe contained all fields in the original CSV with all data included as well.

1. **Bike Lane Shapefile**

Toronto’s Open Data also contained a Shapefile with the locations and pathways of bike lanes throughout the city. Geopandas was necessary to open and use the file, and after attempting to make use of the bike lane file, Geopandas was installed along with fiona, which Geopandas depends on as well. Unfortunately, this file did not end up being used because of the unfamiliarity of various functions with this breed of pandas.

## Transform

|  |  |
| --- | --- |
|  | After extracting the necessary data on both Toronto collisions and red light camera locations from their respective sources, the next step was to transform the data to make it more streamlined. |

1. **Toronto Collision Data CSV**

After loading the initial dataframe of collision data, two separate dataframes were created based on the type of collision reported. One dataframe contained all pedestrian collision information and the other contained all bike collision information. Unnecessary columns were also dropped after this splitting of the data. Some of the dropped columns included ‘Ward\_Name’, ‘Hood\_Name’, and several other keys that identified specific information about the incidents. The longitude and latitude of each collision were rounded to four decimal places in order to increase the number of locational matches. After doing that, the two resulting numbers were concatenated into one column so that the dataframes could be merged based on matching coordinates and thus matching locations.

1. **Red Light Camera Location CSV**

The red light camera was transformed in a very similar way to the previous dataset. However, only one dataframe was garnered from this dataset, and similarly, unnecessary columns were dropped from the dataframe. The latitudes and longitudes were processed in the same way; they were rounded to four decimal places and then concatenated to occupy a new column containing this concatenated latitude and longitude.

1. **Merging the Data**

Because of the transformation on the latitude and longitude into the single concatenated column, the datasets were able to merge on that column. The pedestrian collision dataframe was merged with the red light data using an inner join on the concatenated latitude and longitude. In the same way, the bike collision dataframe was merged with the red light dataframe using an inner join on the same field.

## Load

|  |  |
| --- | --- |
|  | Once the transformation of the data was complete, the data was loaded into an SQL database, specifically Postgres. |

After transforming the dataframes, a database was created in Postgres called toronto\_etl\_db. The create\_engine module was imported from SQLAlchemy in the Jupyter notebook. The variable engine was created using the create\_engine module which established a connection to the Postgres database via user credentials and database path. Using the to\_sql function, the unmerged and merged dataframes were uploaded to Postgres as part of the already existing database.