ETL (Extract, Transform, and Load) Project Overview

Extraction

In this step of ETL project, data was extracted from the sources, being;

- Road Crashes for five Years Victoria as published by Department of Transport (VIC).
- Estimated Resident Population by Local Government Area (LGA) as published by Australian Bureau of Statistics

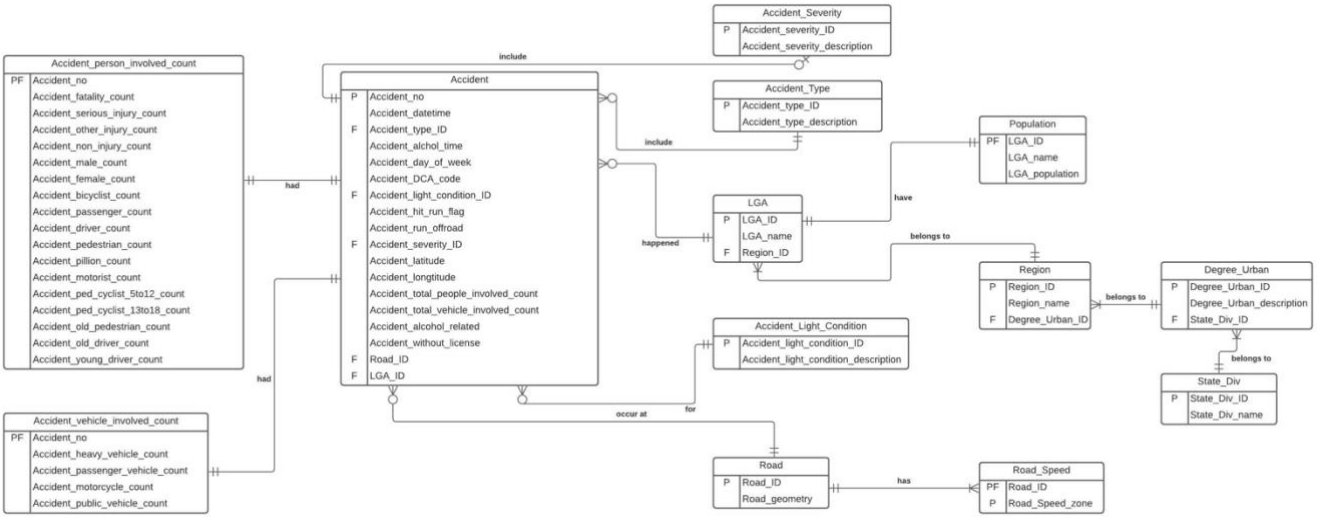
CSV files of the raw data can be found in the Resources folder in this repository.

Some validations were done during extraction stage such as;

- Reconciliation of the records with the source data
- Ensuring that no spam/unwanted data loaded
- Data type check
- Checking the dataframe for duplicate/fragmented data
- Checking whether all the primary/foreign keys are in place within the data frame or not

Data extracted from the data sources were raw and not usable in its original form. Therefore, it needed to be cleansed, mapped and transformed.

We utilised LucidChart to perform data modelling and our database schema was defined as below;



After data modelling, we predefined the database schemas in PostgresSQL.

Transformation

At this stage, we went through the process of changing the format, structure, or values of data. Below is a list of actions that we performed.

- Data Cleaning
    - Dropped any rows that have NaN values
    - Dropped any rows that have missing data and are blank
    - Removed all unnecessary columns

- Data Transformation
    - Removed columns that were not required for the purpose of this analysis
    - Transformed a column into multiples
    - Assigned IDs to individual rows in the dataframe
    - Performed table normalisation to structure our tables in database
    - Grouped our data by the day of accidents

Loading

Loading data into our target data warehouse database, PostgreSQL, was the last step of the ETL process.

PostgreSQL, also known as Postgres, which is a free and open-source relational database management system emphasising extensibility and SQL compliance.

For loading the tables to the database, firstly we created tables in our database by using the schemas.sql, followed by creating a SQLAlchemy connection.

We then proceeded to install a database connector by running "pip install psycopg2" in terminal and created an engine to store our Pandas dataframes in the relevant tables in our database.