|  |  |  |
| --- | --- | --- |
|  |  | **RIDERSHIP V. SALARIES**  *A case study of transportation and salaries in DTSF* |

15 May 2019



## **Extraction**

Our project's data was extracted from two different sources, BART.gov and Kaggle. To clean and combine this set, BART's data was manipulated first. The excel sheet was converted into a Panda's DataFrame, and then only the station name as well as the years 2011 through 2014 were copied into a new DataFrame. We then selected only the downtown San Francisco stations by their iloc location.

Next, the Kaggle data on San Francisco salaries was loaded into a Pandas DataFrame, and then the Base Pay, Benefits, Total Pay, and Year columns were copied out into a new DataFrame. We next drop all rows that are NaN, and then all rows that are non-numeric. Afterwards we are able to use the group\_by function to group the columns by year, and bumpy to round these figures down to two decimals.

Last, insert the BART DataFrame onto the Salaries Dataframes where we are then able to see all data grouped side-by-side by each year.

## **Transformation**

The main filtering we had to deal with was selecting only the years that coincide with our salary data 2012-2014 for BART ridership. We were able to join both data sources by year to better appreciate how/if salary fluctuation affected BART ridership.

We proceeded to clean the data by dropping all rows that were not numbers. As far as transformation is concerned we had to transform our index from “Year” to “Years”. Year is a keyword in SQL and it would have caused conflicts in the logic to have an index equal a keyword.

We chose a relational database for this project due to sources already being clean and no rows missing information. We also will not need to scale this project or add more databases, which make MySQL the right tool for the job.

## **Loading**

When deciding which database format to use, we came to the conclusion that MySql would fit our needs the best. Our data set did not need the extra features that a NoSql format like MongoDB brings, like storing dictionaries or lists. Since our data is easily stored in a table, it was a natural fit for MySql. We are also more confident when converting data for MySql use than the MongoDB format in python, which was a major factor for us. While our project could have been used to make a NoSql database, it was our belief that the added complexity would make it more difficult than was needed for the scope of our information.