

|  |  |
| --- | --- |
| Project 2  ETL |  |
| **Group members**   * Chadi * Ruchi * Hamza * Minnie |  |

|  |  |
| --- | --- |
| Sources  <https://www.mavenanalytics.io/data-playground?page=3>  <https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data?resource=download>  Extract:  We used two datasets in this project where one was taken from Kaggle and the other from Maven Analytics and both where in the form of csv files. The dataset from Kaggle originally had 16 columns and the Maven dataset had 32. Both files were opened in a Jupyter notebook using Pandas.  Transform:  The first step in the transformation process, is removing all the columns that were unnecessary and kept the key columns such as those related to the location, price and review scores.  The dataset from Kaggle only had listings from New York so to ensure that the datasets had parity we removed all listings outside of New York on the Maven dataset.  Thinking ahead about the tables that would be eventually used in data analytics, the datasets should have the same named values to be able to follow the process of creating FACTs and DIMS. We noticed that some neighborhoods appeared in one dataset and not the other, so we stored all the unique values of the maven dataset which had more values into a list and compared it to the unique neighborhoods of the Kaggle dataset. If the item did not appear in both the data frames it was appended to a variable called outliers.  The three outlier neighborhoods were 'Neponsit', 'Bay Terrace, Staten Island' and 'Breezy Point' and were removed from the Kaggle dataset.  We made sure as well that the districts between both files are the same by calling the unique values of each dataset and comparing them together. It was a match. The final step in transformation process was renaming the columns intuitively.  Load:  First, we created the database in Postgres as it’s a structured database engine. Then we created the two fact tables based on the final datasets created in the transform phase. We matched the column names as per the datasets and set the data types accordingly.  In Jupyter notebook, we’ve created a connection with the Postgres engine to be able to push the data to the respective tables. And finally ran the code and checked that the data is available in the database created.  A relational database was chosen as our data is in the form of structured data and is strictly text, and we need a simple model to be created. In addition, it is perfect for normalization process with easy access to data through SQL query. |  |
|  |
|  |

