**ETL Project Report: Airport Weather Delays**  
Kayla St. Germain, April Lagnevall, Stefano Imperiale

February 2nd, 2021

**E**xtract:

In this project, three separate data sources were used: flights.csv1, airports.csv2, and Past Weather API3. The two CSVs came from Kaggle and the Past Weather API came from World Weather Online. You can find the sources below, in the Footer.

To extract the Past Weather API data, it is recommended to sign up for the 60 Day Trial. However, the user should be aware that they are allowed only 500 calls a day. As a result, the user will only be grabbing a sample size of the data, rather the full year’s worth of weather.

The API calls are dependent on two lists, the Dates and Airport Code. To create a list of dates, it is recommended the User input the Start Date. From here, the code assumes 365 days in a year and uses quick calculations with the datetime module. In order to create a list for the Airport Codes, the User will need to load and clean the airports.csv file first. The main goal is to only create a list with US-NY airports. This can be done with a .loc method. Unfortunately, since the trial account only allows 500 call/day, the Airport Code list is only intentionally limited to two large airports, JFK and LGA. This will at least give the User a data table with enough data.

Once the Dates and Airport Code Lists are made, API Calls are ready to be performed. World Weather Online has many parameters and queries the User can input into their query URL. In this code, the format\_output is set to ‘json’, with a time interval (tp) equal to 6 hours, and include locations on. After the query URL is created, the User can begin calling the data with a couple of for loops. This data can later be stored into a DataFrame and a CSV file.

**T**ransforming:

All three CSV files were cleaned using Pandas. Since this project is only interested in New York State, the airports.csv was drastically cut down with a simple .loc method, only locating “US-NY”. To further clean the CSV, columns like “Continent”, “Home\_link”, “Wikipedia\_Link”, and “keywords” were dropped. Lastly, any NaNs were dropped using the .notnull method.

The flights.csv file is too large for GitHub, as a result, a sample file is provided, flights\_sample1.csv . To trim down the unnecessary data, similar methods were used in the airports.csv . The biggest change to the flights\_sample1.csv was merging the month, day, and year columns into one string. This was an important task so that the dates matched the format on the historical\_wx.csv file.

Transforming the historical\_wx.csv file was an easy task. The biggest obstacle was renaming the column headers so that they were SQL friendly. For example, the code would throw a KeyError when ever the headers had parenthesis or percentages. It is also important to note, that each table had an Airport Code, this will end up being the Primary Key across all three tables.

**L**oading:

The primary focus of our project centered on the weather delays for the New York state airports. The availability of data for that selection was the main impetus for continuing research. In order to obtain this information several methods were used under the Extraction step:

* World Weather Online API generation was the first resource needed to begin compiling data.
* We then used World Weather Online to obtain the archived weather data in combination with an Airport CSV obtained by Kaggle, link provided in the bibliography.

Through the use of Pandas via Jupyter Notebook, the data was verified for duplicates, blank fields, and sorted down to a manageable list in order to standardize the results to a more uniform format. On this note, separating the airports into large, medium, and small rankings for the whole state was of particular use. Other points of contention for this step included:

* Latitude and longitude, location
* Type of Airfield, this expanded airports to include facilities such as helipads and closed fields that may not have been indicated at first
* Delay type, some delays were not weather-related
* Search constraints due to the number of pulls from sources that could be made for a day’s data usage cap for API keys

The ‘Load’ step was made possible by using a mixture of PostgresSQL in PG Admin and the SQLAlchemy functions in Jupyter Notebook. The formation of the tables in this way gave us creative control over its structure and layout. By the end of this process, a comprehensive list of 2015 New York state airport delays was created with pertinent information for future clients to view.