# **ETL Project**

# Preparing wine lovers to visit the wineries recognized by *Wine Enthusiast* for top-rated wines

### Group Members: Zach Elson, Martin Wehrli, Katherine Sullivan

## **Extract**

Data for this project came from two sources, Wine Enthusiast Magazine ([www.winemag.com](http://www.winemag.com)) and Google ([www.google.com](file:///C:\Users\koegs\Desktop\class\ETL%20project\ETL-project\ETL-current\www.google.com)). The wine data was downloaded from Kaggle.com in the form of 16 separate CSVs. The data in Kaggle was originally sourced from the Wine Enthusiast Magazine website.

Hotel and restaurant data was extracted using Google’s Geocode API platform (<https://maps.googleapis.com/maps/api/geocode/json>) and performing JSON requests . Wine region and country information from the wine dataset was used to return latitude and longitude for each winery. This latitude and longitude data was then used to pull the nearest lodging and restaurants to each winery using Google’s Place API platform (<https://maps.googleapis.com/maps/api/place/nearbysearch/json>).

## **Transform**

Step 1: Build and clean combined wine dataset*Transforming* – Initially, using Python we combined 16 CSV files into one larger dataframe (“all\_wines\_df”) in a Jupyter Notebook.  
  
*Cleaning* – Each of the 16 CSV files required UTF-8 encoding during the building of the larger dataframe. The only other cleaning required at this point was to add an index (“ID”) to the combined wine dataframe.

#### Step 2: Create a dataframe for wines less than $30 and rated 95 points

*Transforming* – In this step, we queried the “all\_wines” SQL table in PostGres to create a table (“thirty\_dollars\_95”) of all wines with prices of $30 or less that are also rated 95 points. After cleaning as described below this table was saved as “thirty\_dollars\_clean.csv” to be used in the following transforming steps.

*Cleaning* – This table (“thirty\_dollars\_95”) required cleaning before the data could be used to pull Google API longitude and latitude values. Specifically, some of the “country” and “region” columns were modified, so Google APIs would recognize the relevant city.

The “country” column for all wines produced in the United States was manually changed from “US” to the specific state where the wine was produced. This was done so the Google API’s would correctly identify the state of origin.

The “region” column reflects the dynamic within wine region definition where sometimes a wine region is both an actual town and a geographically defined wine region. (e.g. Burgenland in Austria is both a city and a recognized wine region). In this case, no cleaning was required. However, some wine regions are geographically defined but are not actual cities. In this case, the wine region was replaced with a city within the wine region (e.g. Kremstal is an Austrian wine region but Krem is the city within the Kremstal wine region recognized by Google APIs).

Additionally, the region listed for U.S. wines was the respective state, but the wine region and city were reflected in the “subregion” and “subsubregion” columns. In these instances, the “region” column was changed to reflect the corresponding city.

Finally, both the original and cleaned data are reflected in the csv to demonstrate the before and after cleaning data values. The original date is reflected in the “country” and “region” columns. The cleaned data is reflected in the “country\_clean” and “region\_clean” columns.

#### Step 3: Create a heatmap of the locations of the wineries with wines $30 or less and rated 95 points.

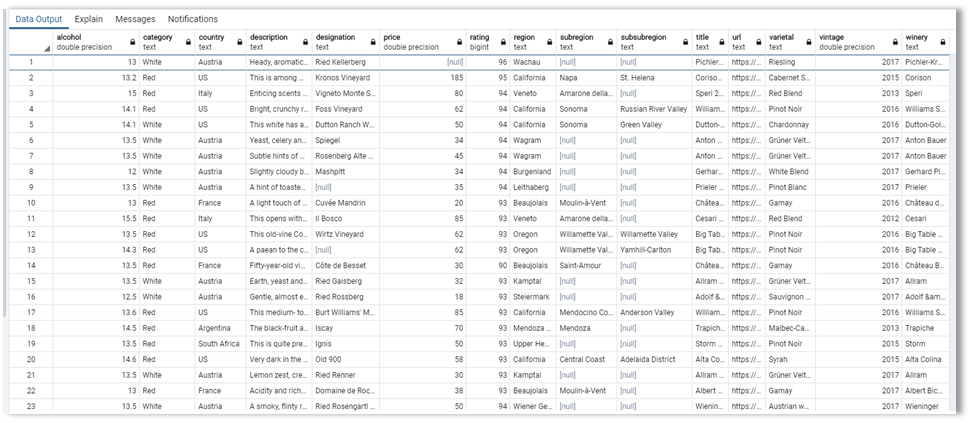
*Transforming* – The cleaned dataset “thirty\_dollars\_clean.csv” was read into the Jupyter Notebook. The “country\_clean” and “region\_cleaned” columns were used to develop latitude and longitude values for each winery represented in the data. This was achieved using Google’s geocoding API (<https://maps.googleapis.com/maps/api/geocode/json>). These values were then used to create a heatmap (see image below) of the locations of all the wineries included in the data.



Step 4: Build and clean a dataframe of the closest hotel and restaurant to each winery for wines $30 or less and rated 95 points  
*Transforming* – The same latitude and longitude data developed in Step 3 above was used to query Google’s “Nearby” mapping API (“<https://maps.googleapis.com/maps/api/place/nearbysearch/json>) to find the nearest hotel and restaurant to each winery in the data. Several columns of data were dropped as there are not as relevant to the listing of nearest hotel and restaurant. Additionally, this wine and winery data can be accessed using ‘ID’, ‘winery’ or other keys in both tables.

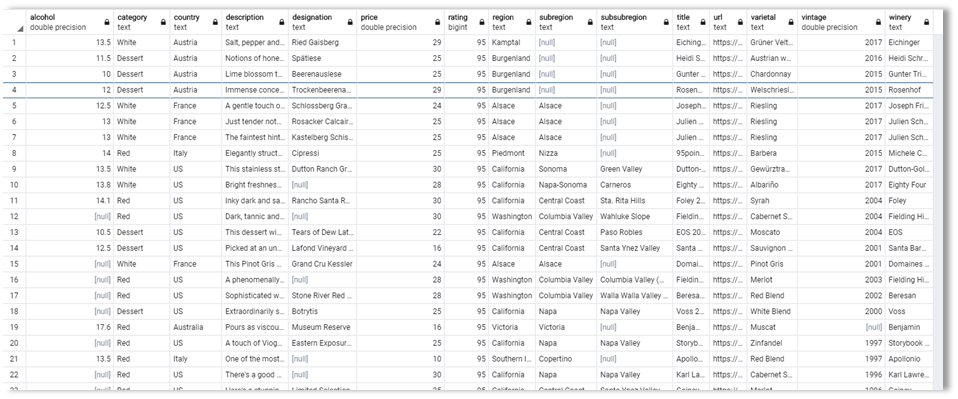
## **Load**

Step 1: Load table of all wine data in PostGres  
We created a database within Postgres to store the wine data for future access. The Pandas dataframe “all\_wines\_df” is quite large but also full of a lot of useful information about wines (see image below). We created an engine, connection and then session with pgAdmin to convert the dataframe to a SQL table (“all\_wines”) so it can be accessed and queried easily. Below is a screen shot from Postgres of the “all\_wines” table we created:



#### Step 2: Load table of wines $30 or less that are rated 95 points in PostGres

The query used in Step 2 of the Transform section returned a table of 70 wines. This table was loaded to pgAdmin as “thirty\_dollar\_95” and includes all wine data for wines $30 or less that are rated 95 points. Below is a screen shot from Postgres of the “thirty\_dollar\_95” table we created:



#### Step 3: Load table of wines $30 or less that are rated 95 points and their nearest hotels and restaurants in PostGres

The query used for this step of the Transform section returned the nearest hotel and restaurant to the wineries from ‘Step 2’ above. This table was loaded to pgAdmin as “thirty95\_hotels\_rests” . Below is a screen shot from Postgres of the “thirty\_dollars\_95” table we created with hotel and restaurant information:

