Our project was designed to gather and save the weather and alcohol consumption data from the US states into our SQL database.

We scoured the web for datasets regarding alcohol in the US. We found a csv file named alcohol\_data.csv and we decided to use this along a Weather API to see if there could possibly be a correlation between the weather and the amount of alcohol consumed per state, per year.

Why:

We thought it would be interesting to see if weather actually affected how much people drank. We believe it may also depend on where you live. California has Napa Valley, so do they drink more wine than beer? Idaho and Colorado also produce wine though, but does the weather differences affect if they drink more beer than California? Through using all 3 data sources, we can see if, depending on the state, what alcohol they drink the most and how much of it they drink, and any correlations with the weather for that state.

In order to call the weather API, first we had to upload a file with latitude and longitude for the states, as we need to add this information into the URL.

We found the file with Latitude, Longitude and States in Kaggle website:

<https://www.kaggle.com/washimahmed/usa-latlong-for-state-abbreviations>

After uploading it into out Jupyter Notebook, we had to do a small transformation just to use the name of the state instead of its abbreviation. We did this to align with our data from the Alcohol Consumption and Weather, that has the state name.

Where we found the alcohol\_data.csv: <https://www.openicpsr.org/openicpsr/project/105583/version/V1/view?path=/openicpsr/105583/fcr:versions/V1/alcohol_data.csv&type=file>

Steps taken to load the Alcohol Data into SQL Database:

Step 1:

We started by saving the csv we found into a folder we could reference later in the Jupyter notebook.

Step 2:

SQL – We created a DB called booze\_db, then added a table “state\_alcohol” with renamed columns to eventually push the csv information into.

Step 3:

In Jupyter notebook, we imported everything necessary, then read the alcohol\_data.csv file.

Step 4:

We renamed the columns to match what we had labeled in the “state” SQL table.

Step 5:

We realized that we needed to combine 2 of the columns, ‘state\_name’ and ‘year’, so that we have a primary key. We then created a new dataframe that merged ‘state\_name’ and ‘year’.

Step 6:

We added it to the original dataframe as another column, and renamed it ‘combined’.

Step 7:

Created a database connection with SQL ‘booze\_db’.

Step 8:

Confirmed the table that I created in SQL to make sure there was a table I could then push the information I just cleaned to.

Step 9:

Loaded the final dataframe into the ‘state’ SQ

Regarding to the API, we have decided to use an API from the website worldweatheronline.com and store the historical data of the weather for all the US states. The API call takes really long time to run when you are trying to run it for a lot of data. It makes sense to hold it in a SQL database so we can easily access it when we need to perform any analysis with weather data.

The API returns a lot of information related to weather such as temperature, percentage of cloudiness, sunrise, sunset, total snow, precipitation, humidity, wind speed for each day of a period given to the URL.

The biggest period we can send is a month. So, for example, to run the 51 US states for one year we have to call the API 612 times. For our project we decided to store 5 years of data (2012, 2013, 2014, 2015 and 2016) in order to analyze the consumption of alcohol and check if there is any relation with the weather and also if the consumption has been changing along the years because of external factors like the Global Warming.

To achieve our goal in this project we have to call the API 3060 times. As we have a limit of 500 calls per day, we decided to store the data for only 8 states, 2 years and 3 months. In a real world we would pay for unlimited calls per day. But calling and storing the data for theses 8 states, we can demonstrate what and how we can do it.

This is an example of the API return:

**{**"data": **{**"request": **[{**"type": "LatLon","query": "Lat 32.60 and Lon -86.68"**}]**,"weather": **[{**"date": "2012-01-01","astronomy": **[{**"sunrise": "06:49 AM","sunset": "04:52 PM","moonrise": "11:29 AM","moonset": "No moonset","moon\_phase": "New Moon","moon\_illumination": "0"**}]**,"maxtempC": "19","maxtempF": "66","mintempC": "10","mintempF": "50","totalSnow\_cm": "0.0","sunHour": "0.0","uvIndex": "0","hourly": **[{**"time": "24","tempC": "19","tempF": "66","windspeedMiles": "10","windspeedKmph": "17","winddirDegree": "296","winddir16Point": "WNW","weatherCode": "266","weatherIconUrl": **[{**"value": "<http://cdn.worldweatheronline.net/images/wsymbols01_png_64/wsymbol_0017_cloudy_with_light_rain.png>"**}]**,"weatherDesc": **[{**"value": "Light drizzle"**}]**,"precipMM": "0.4","humidity": "76","visibility": "8","pressure": "1024","cloudcover": "23","HeatIndexC": "12","HeatIndexF": "54","DewPointC": "8","DewPointF": "46","WindChillC": "11","WindChillF": "52","WindGustMiles": "17","WindGustKmph": "28","FeelsLikeC": "11","FeelsLikeF": "52"**}]}**,**{**"date": "2012-01-02","astronomy": **[{**"sunrise": "06:49 AM","sunset": "04:52 PM","moonrise": "12:01 PM","moonset": "12:50 AM","moon\_phase": "Waxing Crescent","moon\_illumination": "0"**}]**,"maxtempC": "9","maxtempF": "48","mintempC": "1","mintempF": "33","totalSnow\_cm": "0.0","sunHour": "0.0","uvIndex": "0","hourly": **[{**"time": "24","tempC": "9","tempF": "48","windspeedMiles": "13","windspeedKmph": "21","winddirDegree": "299","winddir16Point": "WNW","weatherCode": "113","weatherIconUrl": **[{**"value": "<http://cdn.worldweatheronline.net/images/wsymbols01_png_64/wsymbol_0001_sunny.png>"**}]**,"weatherDesc": **[{**"value": "Sunny"**}]**,"precipMM": "0.0","humidity": "53","visibility": "10","pressure": "1029","cloudcover": "0","HeatIndexC": "5","HeatIndexF": "40","DewPointC": "-5","DewPointF": "23","WindChillC": "1","WindChillF": "33","WindGustMiles": "20","WindGustKmph": "32","FeelsLikeC": "1","FeelsLikeF": "33"**}]}**,

Steps taken to load the Weather Data into SQL Database:

Step 1:

Using Jupyter notebook we read the csv file with 8 US states. The API is called using longitude and latitude. This file is not the completed file. It is the one we created with less states so we can hold to our 500 calls per day.

Step 2:

SQL – We created a table called “weather\_state\_year” in our “booze\_db” database, then added a table with the columns that we want to store in order to perform our analyses later on. We also created a column called “State\_Year\_Month” which is a unique field, so we can have a proper primary key,

Step 3:

We created 2 lists: month and year. So in our code we checked the months and year we were going to run so we can figure out what is the first and last day.

Step 4:

Then we created our URL: "http://api.worldweatheronline.com/premium/v1/past-weather.ashx?" + "key=" + api\_key + "&q=" + str(lat) +"," + str(lon) + "&date=" + firstday + "&enddate=" + lastday + "&tp=24&format=json"

Step 5:

With the API return we started to do the transformation with the data. We decided to hold the data per month. For this project we would analyze per year as we do not have the alcohol data per month but we believe it is worth storing it per month. Most of the analysis would make more sense to be done per month like checking if the alcohol consumption changes in the summer / winter or when it is sunny, for example.

We are calculation how many hours of sun by doing: (sunrise – sunset) \* 1 – (percentage of cloudiness).

The API returns the data per day. For our table we are holding the following values:

* Total of sunlight
* Average temperature
* Average Cloudiness
* Total of precipitation

Step 6:

We then created a dataframe to hold the summary data.

Step 7:

Created a database connection with SQL ‘booze\_db’.

Step 8:

Confirmed the table that I created in SQL to make sure there was a table I could then push the information I just cleaned to.

Step 9:

Loaded the final dataframe into the ‘weather\_state\_year’ SQL table using “grouped\_weather\_month\_df.to\_sql(name='weather\_state\_year', con=engine, if\_exists='replace', index=False)”

Having these data into our SQL Database, we can start to think about other types of analysis, such as suicide related to weather and suicide related to alcohol problems.