## **Extract** (with BeautifulSoup, Splinter, Pandas)

1. url: <https://www.winespectator.com/vintage-charts/region/united-states>

A scraper was developed in order to scrape data from the above website to gather the vintage charts information. Each Country’s wine was broken down by State then category. To get the information for each wine category the scraper needed to navigate through several pages. Splinter was used to navigate the pages. A browser object was created that would find each link on the page and click the link.

The page for each category contained a table where each row was a different vintage year. A new BeautifulSoup object was created on each page. The soup object was then used to loop through each row of the table then a second loop to loop through each column of the row. The data was then stored in a dictionary which was used to create a DataFrame. The DataFrame was exported as a csv which was then sent off to be cleaned.

1. Wine Dataset from Kaggle

## **Transform** (with Datetime, Pandas)

### Wine Dataset from Kaggle

*Creating Pandas Dataframe*

1. Created a copy of the Dataframe to prevent any changes made from the original dataset
2. Created a filter to only bring specific columns
3. Renamed column header to create meaningful headers in the Dataframe
4. Filtered only US wines in the dataset

*Cleaning*

1. Dropping duplicates by wine titles and resetting the index
2. Dropped missing values from the DataFrame
3. Used the DateTime function to extract year from Vintage field (yyyy-mm-yyyy) and created another column to store the year
4. Dropped additional columns
5. Created a Pandas Dataframe to filter only US records and called it ‘cleaned\_df’
6. Sorted column ‘Points’ in descending order

### Webscraping CSV

*Cleaning*

1. Split ‘type’ column into - 'Province','Variety','Region'
2. Issues – 2 different sets of delimiters
3. Drop unnecessary columns
4. Check and drop null values
5. Rename columns to match other csv column names
6. Check ‘Year’ data type and change format to match other csv

*Merge*

1. Merge right (on webscraping csv) on 3 columns
2. Check to see null values in each column and drop
3. Check to find duplicates by name and drop duplicates
   1. Data from both csv’s did not match up as anticipated, rows were reduced to 3K
4. Export to csv

### Final Merged CSV

*Creating bins to assign Rating definition from scraped data*

1. 95-100 — Classic: Great wines
2. 90-94 — Outstanding: Wines of superior character and style
3. 85-89 — Very good: Wines with special qualities
4. 80-84 — Good: Solid, well-made wines
5. 75-79 — Mediocre: Drinkable wines that may have minor flaws
6. 50-74 — Not recommended

## 

## **Load** (with Pymongo, MondoDB)

### Database Creation

*Creating Pandas Dataframe*

1. converted final df to dictionary, reset the index to a string field
2. created a DB in Mongo called DataCondasWine\_db
3. created a collection in Mongo called final\_wine\_output
4. inserted wine\_dict to final\_wine\_output collection in mongo
5. row count in Mongo to confirm all 3K rows were populated