The Wranglers (Team28)

(1) Naveen Kumar Conjeevaram Baskaran (email id: nc42@illinois.edu)
(2) Alvin Do (email id: alvindo2@illinois.edu)
(3) Hemantika Dasgupta (email id: hd8@illinois.edu)

1. Description of Data Cleaning Performed

During our phase 1 submission, we had mentioned the below as our primary use case U1.

"Our target use case is that of a 'wedding wine selection where the wedding is being held in the US and the guests would prefer European wine'. As we know, weddings are a grand affair and drinks play a crucial role. Hence, in order to arrange an assortment of wines that should delight the guests and yet be within the wedding budget, we need to look at the points, price, title, description, winery and territory details(country, province, region_1, region_2). Barring the 'points' data, every other data point mentioned above needs to be cleaned to some extent to make it 'fit-for-purpose' for the wedding wine analysis."

STEP1: Cleaning accented & special characters

Our first step of cleaning involved the cleansing of the accented and other special characters. This was achieved using Python with regex patterns on the pandas columns (aka series) and using the 'unidecode' function. To start with, we loaded the csv file into a pandas dataframe using python.

We scanned (using regex strings) the text columns that have various data quality patterns & below are the columns that were found:

'description','designation','province','title','variety','winery', 'region_1', 'region_2'

When each column was scanned via regex using pandas series extractall() function, the below patterns were found.

- Single occurrence of non-word characters. Examples are , . " " ' ' ` `. These are predominantly found in the description where there is an elaborate detail on

the wine contents, smell, color and other attributes. These are valid characters and so we didn't clean them.

- Accented characters like Â,É,Ò,Ù ì for both upper and lower cases and many more. These characters are unicode characters and not ASCII characters. Though it is not valid in the English language, those are still legitimate characters in other cultures like German, Italy, Polish, and Turkey. This occurred in description, designation, title, variety and winery columns. Province and region_1 column had very few occurences. These were converted to English characters. For e.g., the designation "VulkÃ Bianco" was converted to "Vulka Bianco".

- Combination of consecutive non-word characters like ", . # $ % ' ' " " ( ) - / + *". Examples are …. !!!! **** ++++. These characters are consecutively repeated and don't make any sense. We replaced it with just a single occurrence of the non-word character. This change was applied to description, designation, title, variety and winery columns.

- Two count repetition of non-word characters. Examples are). %) +). These may be valid in columns like description and designation and so we didn't clean them. For example, in a descript like this - "A Cabernet-dominated (98%) wine", the presence of these characters perfectly make sense.

- Multiple consecutive repetition of non-word characters like '[+]' '(+)' '*%#&@!' '%@#\$!' which are not valid words or letters. These were found only in 'description' and 'designation' columns. These patterns were removed.

We also noticed that the Taster_name and taster_twitter_handle columns were mostly clean for the values that were present. There were only a few values present compared to the size of the csv file. These 2 columns are in the never enough use case and not critical for our 'fit for purpose' use case.

The rationale behind cleaning these characters from the above-mentioned fields is simply to enhance the readability. As we had mentioned before, one glance at the file revealed a lot of junk characters and it was really difficult to read through it.

Presence of these characters would also hinder the identification of unique records, hence it was necessary to clean these as the very first step.

STEP2: Cleaning the NULLs from the 'Price' column

The second step involved cleaning the NULL values we noticed in the Price column. This had to be cleaned as the 'price' column is crucial for our use case. Instead of populating it with just 0s, we opted for a 'mean value' ($35.3) of the prices and populated it with it. The price column had 8996 null values. This is an integer column and we needed non-null values in this column. Using the describe() function in pandas, we realized that the majority of the values were between 17 USD to 42 USD though we had some outliers like 3300 USD. We then utilized the fillna() function from pandas package to use the 'mean' of price value as the substitute values.

STEP3: Cleaning the NULLs from the 'Country' column

We noticed that there were only 63 NULLs out of the ~130K records in the 'Country' column, so we extracted the winery column names, researched in public Google website to find the country of origin of these wines which are potentially named after the winery and then populated with those country names. We found a total of 27 unique winery names corresponding to the 63 null country values. This was required because we are interested in European nations only. We again used the Utilized the fillna() function from pandas package to use the dictionary values as substitute values.

STEP4: Cleaning data using OpenRefine

The next step involved making other cosmetic changes to the data file using the OpenRefine tool. The csv file was imported and cleaned for leading and trailing whitespaces, consecutive whitespaces, changed to 'Titlecase' for all the text columns. The 2 columns – 'Price' and 'Points' were converted to 'number' format as well. Lastly, the 'facet' and 'clustering' option was used and by this time we had a relatively clean csv file at hand. This cleaning was necessary because we had to ensure that we dealt with the syntactic errors as well. The price and points needed to be numeric values and having extra spaces can lead to duplicate records. Hence,

with the OpenRefine tool at our disposal, we leveraged it to do the remaining cleaning.

With the cleaning steps completed, it's now easy to look at the wines from the different European countries & territories, check the prices and points to make some decisions.

2. Document data quality changes

(a) Improvement from python cleaning steps: Below is the summary indicating the reduction in the NULL values on the cleaned columns. The five highlighted columns in green below had their NULL values populated(when there were no valued determined, we populated it with one whitespace character).

#	Column Name	Data Type	Non-NULL Counts (Before python cleaning)	Non-NULL Counts (After python cleaning)	No. of Records counts
1	country	String	129908 non-null	129971 non-null	63
2	description	String	129971 non-null	129971 non-null	
3	designation	String	92506 non-null	129971 non-null	37465
4	points	integer	129971 non-null	129971 non-null	
5	price	float	120975 non-null	129971 non-null	8996
6	province	String	129908 non-null	129971 non-null	63
7	region_1	String	108724 non-null	129971 non-null	21247
8	region_2	String	50511 non-null	50511 non-null	
9	taster_name	String	103727 non-null	103727 non-null	
10	taster_twitter_handle	String	98758 non-null	98758 non-null	
11	title	String	129971 non-null	129971 non-null	

12	variety	String	129970 non-null	129971 non-null	
13	winery	String	129971 non-null	129971 non-null	

The data quality improved a great deal for the columns ‘description’ and ‘designation’ too where the accented and repetitive non-word characters were cleaned. There were 186 different patterns.

- Accented character pattern - 50
- Single occurrence of non-word characters - 70
- Combination of consecutive non-word characters - 62
- Multiple consecutive repetition of non-word characters - 4

The remaining columns like province, region_1, variety, winery, title had very few (like 3-5) occurrences of the above mentioned patterns and those were cleaned as well.

(b) Improvement from open refine step

The cleaned step (a) dataset was passed through the below series of cleaning steps in OpenRefine. These steps further improved the syntactic errors and eliminated those.

Text transform on cells in column country using expression value.trim()
Text transform on cells in column country using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column description using expression value.trim()
Text transform on cells in column description using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column designation using expression value.trim()
Text transform on cells in column designation using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column points using expression value.toNumber()
Text transform on cells in column price using expression value.toNumber()
Text transform on cells in column province using expression value.trim()
Text transform on cells in column province using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column region_1 using expression value.trim()

Text transform on cells in column region_1 using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column region_2 using expression value.trim()
Text transform on cells in column region_2 using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column taster_name using expression value.trim()
Text transform on cells in column taster_name using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column taster_twitter_handle using expression value.trim()
Text transform on cells in column taster_twitter_handle using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column title using expression value.trim()
Text transform on cells in column title using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column variety using expression value.trim()
Text transform on cells in column variety using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column winery using expression value.trim()
Text transform on cells in column winery using expression value.replace(/[\p{Zs}\s]+/,' ')
Text transform on cells in column region_1 using expression value.toTitlecase()
Text transform on cells in column region_2 using expression value.toTitlecase()
Create column wine_id at index 1 based on column Column using expression cell.recon.match.id
Remove column wine_id

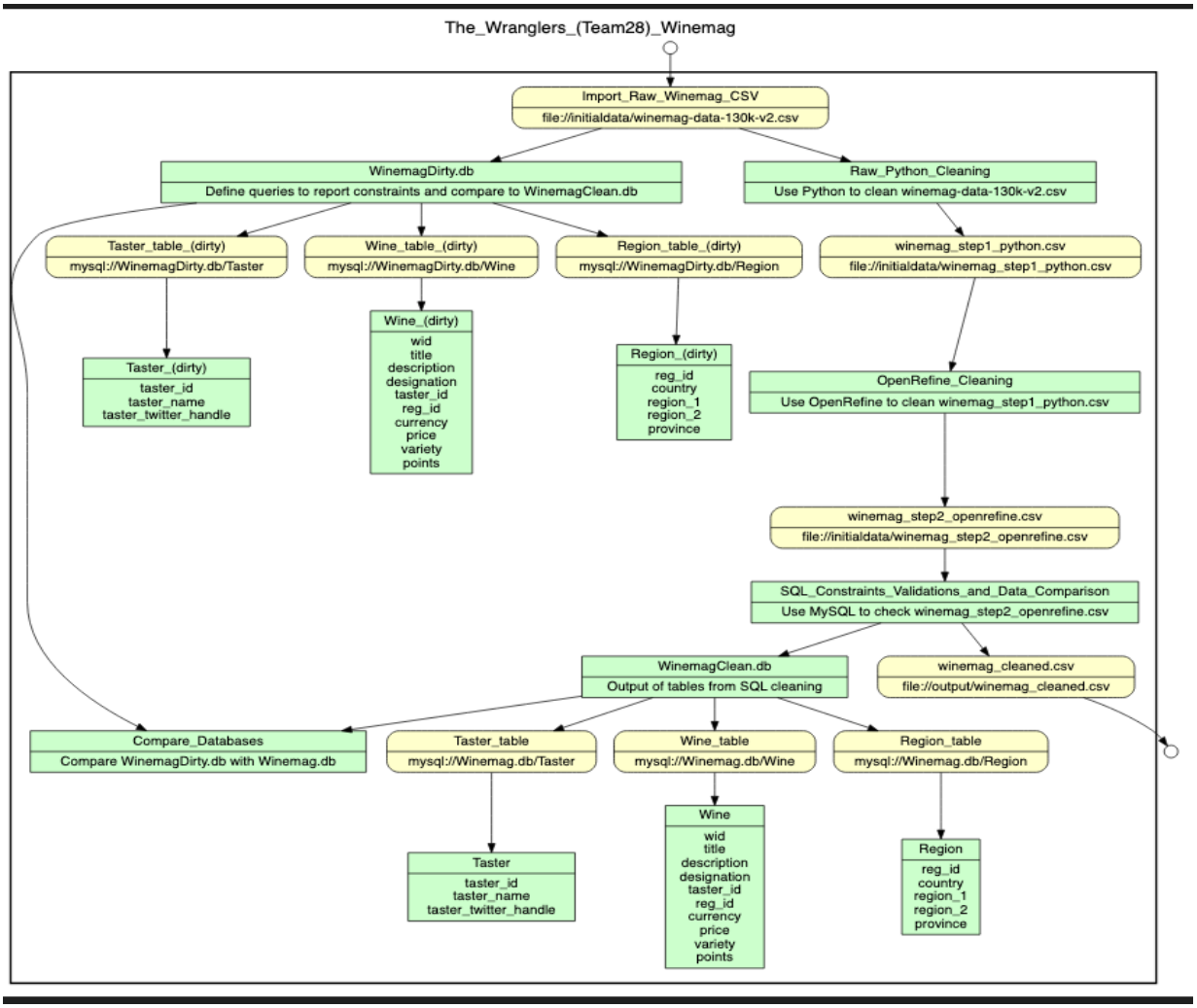
c) IC violations and Denial constraints

We were able to query and find titles that were associated with two or more wineries. We were also able to verify the 'check' constraints on price and points fields where no negative values were found for price & no points exceeded 100 as it was on a 0-100 scale. The NULL constraints were also verified as the country & price fields were no longer having NULL values.

3. Create a workflow model

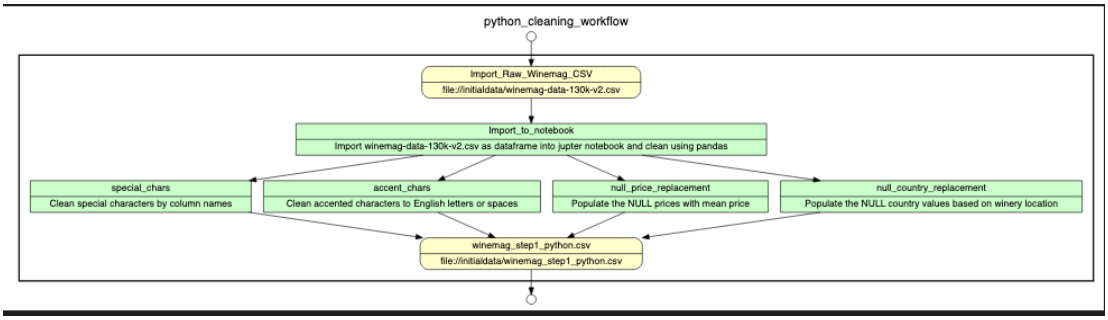
Below is a screenshot of the outer workflow model W1. The key input to this workflow model is the Winemag-data-130k-v2.csv file. The high level steps involve updating the SQLite database(WinemagDirty.db) with the raw data which is then saved as 3 different tables in the database as shown below. The same input file is also the input to the Python Jupyter notebook where the special characters and NULL values are cleaned using Python codes & the intermediate Step 1 csv file is generated as output. This in turn is fed to the OpenRefine_Cleaning step as in input and from that the final step2 csv file is

generated. Post that this step2 csv is fed as an input for the SQL constraints and IC violations check so, the cleaned data is ingested back to SQLite in the WinemagClean.db for comparison purposes. The different URI's are also shown in the below model.

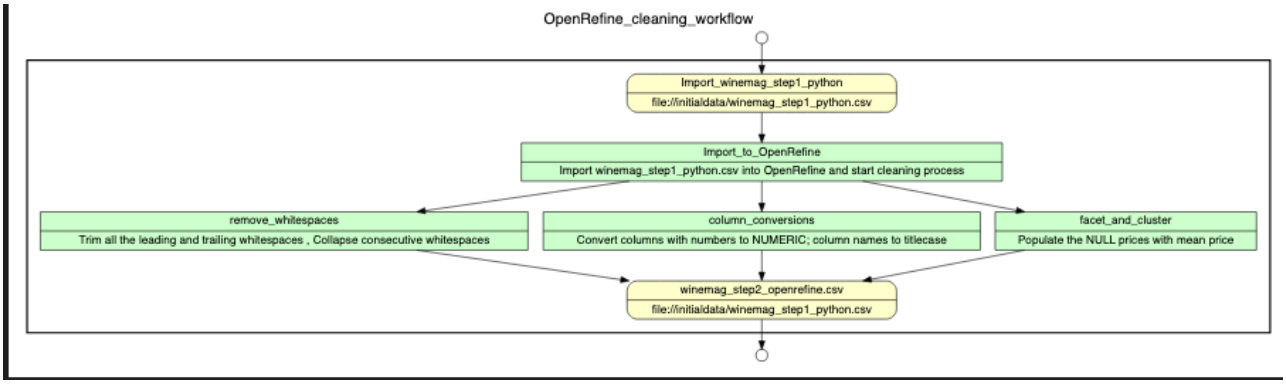


The SQL constraints validation step where we would upload it back to SQLite is dependent on the Python & OpenRefine cleaning steps and no other major dependency was observed.

Below is a screenshot of the 3 inner workflow models that we created using YesWorkflow. From the high level steps mentioned in the outer workflow model, we came up with these 3 inner workflow models – one each for the Python cleaning, the OpenRefine cleaning and the SQL constraints workflow. Let's look at the screenshot of the first inner workflow model.



As this was the first step, so the key input is still the raw csv file (the data element in the top yellow box) and the next step involves importing this to a Python Jupyter notebook as a dataframe. Once it's identified by the python notebook, the python functions are used to clean the special characters, accented characters, replacing NULL values with data for the 'country' and 'price' fields. Finally, the key output from these steps is generated in the form of 'winemag_step1_python.csv'.

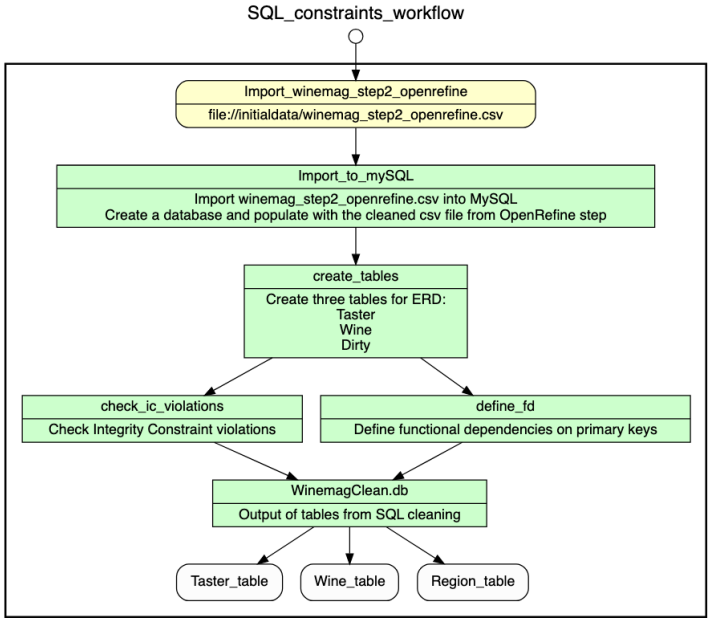


The second inner workflow model(shown above) is related to the OpenRefine cleaning. The main input in this step is the cleaned file from the Python cleaning step i.e the winemag_step1_python.csv and then it goes through the data conversion processes of eliminating whitespaces, changing datatypes to numeric for the point and price fields and finally the faceting & clustering steps. At the end, the key output from this step is the step2 csv file.

The final inner workflow model is that of the SQL constraints workflow(as shown below). This workflow is for the cleaned dataset and its key input is the CSV obtained from the OpenRefine output. The steps involve creation of the tables and then uploading the cleaned data in the database. The final tables are the outputs of this workflow.

The main tools we used for cleaning was Python(because of the flexibility and the varied set of functions it provides), OpenRefine(an open source tool that can easily

be used to clean large datasets), YesWorkflow(for creating the workflow diagrams), SQLite (for being an easy setup open-source tool which can easily handle csv files).



4. Conclusions & Summary

This project was an eye-opener for all of us in the team. We will surely be approaching data in a different way going forward. There were so many steps involved to ensure that data was relatively cleaner than the raw dataset. In the process, we actually realized that data profiling and wrangling is actually time-consuming but very easily glanced over by many people working in the data industry.

As for our primary use case,

We would like to recognize each other for the tremendous dedication towards this project. Naveen was instrumental in performing the Python and OpenRefine steps, Hemantika helped with the project summarization & identifying IC violations whereas Alvin helped with data profiling as well as prepared all the inner and outer workflow diagrams in YesWorkflow.

5. Submission of supplementary materials

We are submitting the below documents as the supplementary materials in the zip file:

a) YesWorkflow outer model (a .png file, a *.gv file and *.yw file)
b) YesWorkflow inner models (a .png file, a *.gv file and *.yw file)
c) OpenRefine operation history (as a JSON file - winemag_step2_OpenRefine_recipe)
d) Screenshots of Python & SQL queries
e) A text file with the raw and clean dataset paths.