CS 513 Data Cleaning Project Phase 2 Report

Team ID: Team6
Team Members: Hannah Benig (hhlim2@illinois.edu) and Shu Kit Tse (tse1@illinois.edu)
Project Github: https://github.com/hhlim2/cs513_team6
Project Box Link: https://uofi.box.com/s/7home4uje09zbehftfzglb339tesg8f4

Refining Target (Main) Use Case U1

Upon exploratory profiling of the Dish table (e.g., by extracting "name" entries in the Dish table using coffee related keywords), we were further able to understand the dataset and know it might contain as well as gain a better idea of how to use some of the information available (upon sufficient cleaning) to answer certain questions within the original scope of our target use case analysis (U1).

We originally titled U1 as "Understanding coffee in restaurants and cafes through time," where we hope to understand how coffee dishes may have evolved through time (based on available data) and listed the following example questions that fall under this broad use case analysis:

- Where and when (time of day, year, etc.) was coffee served?
- Based on the dataset, has coffee always been on restaurants' menus since the earliest record available?
- How popular was it (coffee dishes), or rather, when did it become popular as a menu item?
- When did different types of coffee drinks (i.e., not filter black or espresso) begin appearing on menus and become more prevalent?
- How did the prices change over time?

In order to better facilitate and focus our cleaning efforts, we have omitted the following original questions:

- Where and when (time of day, of year, etc.) was coffee served?
    - Why: While we have some data in the "event" column of the Menu table that would indicate if the menu was for different meals of the day, i.e., "BREAKFAST," "LUNCH," "DINNER," "SUPPER," there are also a lot of entries that are not directly relevant to our target use case, e.g., "ANNUAL BANQUET," "DAILY MENU," "WINE LIST," "CHRISTMAS DINNER" where we would likely filter out the latter types of entries and end up with an analysis that may not be conclusive.
    - As for "where," as most of these records are likely in New York, looking at the geographic region level would not be as informative or interesting. If we look at locations in terms of street address level, i.e., neighbourhoods, while we could probably use values under the "location" column unless we put in much more upfront effort and work to find a corresponding database for historical New York street addresses within the relevant range of years, there is no guarantee we would be cleaning the "location" names correctly to accurately cross-reference

such a potential database if it even exists and if it's readily available for us to obtain. Hence, we deem this question unworthy of pursuit for the purposes of this project.

- How popular was it (coffee dishes), or rather, when did it become popular as a menu item?
    - Why: Popularity can be measured in several ways, while we could make use of data in the "menus_appeared" or "times_appeared" columns or even "lowest_price" and "highest_price" as indicators (which we would be using in the other questions), these aren't necessarily direct measures. We, therefore, do not have adequate data for a proper analysis of popularity. An example of an adequate way popularity could be measured might be through the quantity of a dish (or rather a menu item) sold within a specified period of time, where we can then see if there are any trends to indicate an increasing or decreasing rate of such quantity sold over certain periods of time. As there isn't such sales or quantity sold data in the dataset, attempting to answer questions related to dish popularity won't be a fruitful endeavour.

U1 Narrative

We have thus scoped and refined our questions for U1 to the following:

1. When did different types of coffee drinks (dishes) begin appearing on menus and become more prevalent?
2. Based on the dataset, has coffee always been on restaurants' menus since the earliest record available?
3. How did coffee dish prices change over time?

Note that instead of excluding black coffee and espresso (as in the original question) we keep them because it is likely these are coffee dishes (specifically the former) with higher "menus_appeared" values (i.e., across different records once normalized).

Based on these questions, we are therefore interested in how different coffee dishes, e.g., "coffee" (i.e., black coffee, filter coffee), "espresso," "americano," "latte," "mocha," and so on, found in the "name" column under the Dish table may have changed in terms of following:

1. **Time,** i.e., years, where relevant columns are "first_appeared" and "last_appeared."
2. **Prevalence**, where we might define and measure prevalence here as how widespread and hence how many menus such coffee dishes have appeared, i.e., relevant column(s) may be "menus_appeared" and "times_appeared." Although specifically, "times_appeared" appears to refer to the total no. of times a dish has appeared, i.e., a dish could appear one or more times in a menu, in terms of measuring how prevalent a dish is across New York, i.e., the same dish appearing twice on the same menu does not make it "more prevalent," "menus_appeared" thus appears to be a better indicator than "times_appeared." We will, therefore, only make use of "menus_appeared" here.
3. **Price**, where relevant columns are "lowest_price" and "highest_price."

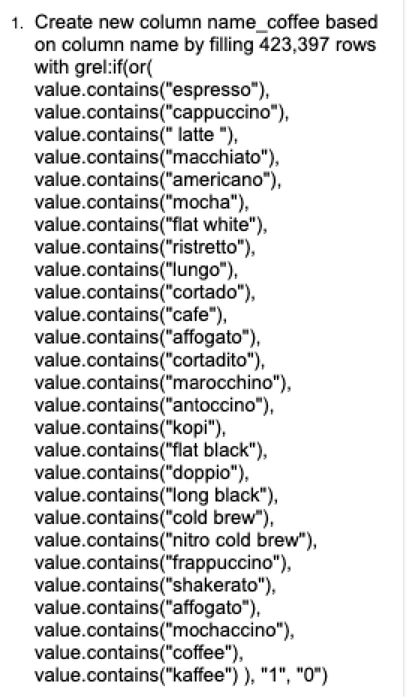
Note that all these columns are in the Dish table, and since we have deemed other original questions to not be worthwhile to work on and as we have the necessary relevant data found in just the Dish table to answer the above 3 questions, we would not be working with the rest of the tables i.e. Menu, MenuPage, MenuItem for this project.

Intuitively, we therefore want to clean and work with the following columns in the Dish table:

- name
- menus_appeared
- first_appeared
- last_appeared
- lowest_price
- highest_price

The “description” column appears to not have any entry in it, we remove this column for our project as part of the data cleaning actions discussed later. As for the “id” column, we would need to check that each “id” value is a unique integer within the column to ensure integrity constraints.

Specifically for the “name” column, we first extracted 1088 records containing common modern coffee keywords using OpenRefine (see key step below) in order to more directly look at, understand and essentially profile what is likely the majority of relevant records in our dataset (i.e. we first excluded the 3 tables Menu, MenuPage, MenuItem, then extracted a subset of records relevant to U1 in the Dish table).



```
1. Create new column name_coffee based
   on column name by filling 423,397 rows
   with grel:if(or(
   value.contains("espresso"),
   value.contains("cappuccino"),
   value.contains(" latte "),
   value.contains("macchiato"),
   value.contains("americano"),
   value.contains("mocha"),
   value.contains("flat white"),
   value.contains("ristretto"),
   value.contains("lungo"),
   value.contains("cortado"),
   value.contains("cafe"),
   value.contains("affogato"),
   value.contains("cortadito"),
   value.contains("marocchino"),
   value.contains("antoccino"),
   value.contains("kopi"),
   value.contains("flat black"),
   value.contains("doppio"),
   value.contains("long black"),
   value.contains("cold brew"),
   value.contains("nitro cold brew"),
   value.contains("frappuccino"),
   value.contains("shakerato"),
   value.contains("affogato"),
   value.contains("mochaccino"),
   value.contains("coffee"),
   value.contains("kaffee") ), "1", "0")
```

We say that this is likely the majority of relevant records and is a subset that is relevant to U1 because we do not know what keywords we may have potentially missed or if there may be different spellings, misspellings, or typos (e.g., during volunteers' input of data from menus) in any of these "name" entries. As it turns out, upon manually inspecting the "name" entries in this extracted set of records, besides in English, we realized there are a sizable number of entries written in German and French, and some also in Spanish. The main issue that we realized with potentially using such records, however, is that while various dish names are written in highly specific ways, in the context of U1, many of them ought to simply fall under the same normalized dish name.

For example, intuitively, for purposes of U1, the following entries could all potentially be renamed "Coffee":

- "coffee, per cup"
- "coffee, tea"
- "Cup of tea or coffee"
- "Cup coffee with cream"
- "Milchkaffee"
- "cafe (la tasse)"

As part of our cleaning process, we, therefore, need to normalize values in the "name" column for each relevant record in order to aggregate data in the other relevant columns (listed previously) for each normalized coffee dish. As to be explained later, while our approach could be considered quite manual and may not necessarily cover all such relevant records, we believe we have been quite thorough with the process.

We have thus established that we will be normalizing entries in the "name" column to coffee dishes, where such normalized coffee dishes will, therefore, be used in our downstream analysis in U1.

In order to better guide our cleaning efforts, we have come up with the following analysis examples for each of the 3 questions in U1.

1. When did different types of coffee drinks (dishes) begin appearing on menus and become more prevalent?
    - Example: Plot a cumulative line graph for each normalized coffee dish
        i. X-Axis: Year
        ii. Y-Axis: Total No. of Menus Appeared (since first appeared in this dataset)
    - Pre-processing for the Total No. of Menus Appeared
        i. Based on the available data, as we do not know in which exact years the normalized coffee dish in each record has appeared, we, therefore, first take the average of it across the range of years it may have appeared in and add this average to each relevant year in the range.
        ii. We do so by taking "menus_appeared" eg. "5" divided by the number of years it has appeared i.e. last_appeared - first_appeared + 1 eg.

1963-1900+1 = “64”, giving us 0.078125, then add this number to each of the years from 1900 to 1963, for the normalized dish name of this record.

iii. There may be multiple records with the same normalized coffee dish name, where the average of each record’s “menus_appeared” value will be added to the record’s associated years and then aggregated for this normalized coffee dish name.

iv. We thus have an “aggregated average no. of menus appeared” derived value for each associated year for all records of the same normalized coffee dish.

v. In order to tell if there may be a trend for each normalized coffee dish i.e., we want to know if over time there may be a rate of increase or decrease in the total no. of menus a dish has appeared, we therefore then derive a separate value for each associated year, that cumulatively adds to the “aggregated average no. of menus appeared” value starting from the first year it has appeared in.

vi. For example, for the hypothetical normalized coffee dish “Iced Coffee,”

“Iced Coffee”	1900	1901	1902	1903	1904
Aggregated Average No. of Menus Appeared	10	8	15	13	5
Cumulative Aggregated Average No. of Menus Appeared (renamed as Total No. of Menus Appeared)	10	18	33	46	51

○ Potential Analysis

i. Once we have pre-processed our cleaned dataset for each of these normalized coffee dishes, we can then plot a cumulative line graph for each of these dishes.

ii. We then try to find any trends associated with any particular normalized coffee dish, to see when they have first appeared, and if any of these dishes had increasing or decreasing trends during certain periods of time (years) i.e. increasing at an increasing rate, and increasing at a decreasing rate.

iii. Without visualization, i.e., plotting a chart, we could also do this by deriving slopes, finding multiple higher and lower slopes for each normalized coffee dish across years, and comparing them across normalized coffee dishes. Although intuitively, visualizing such derived data by plotting said chart might make it easier to see where the trends are before diving deeper into the data.

iv. While we know that such an analysis would not include factors such as restaurants closing down, or menu items being removed or changed, also because we do not have such relevant data, we believe the above would be informative enough for U1's purposes.

- ○ How this clarifies or guides our cleaning work:
  i. With the above example, we thus have something more concrete in terms of knowing what we may be comparing and looking at in U1, which clarifies our intuitive decision that we should have sufficient data for analysis in U1 if we cleaned the aforementioned columns in the Dish table.
  ii. This further guides us to make certain renaming decisions where in principle we know for certain that such cleaning efforts on normalizing coffee dish names will definitely improve data quality.
  iii. For example, many entries in the "name" column could simply be renamed as "Coffee" for any dish directly referring to black coffee that we do not consider espresso or other types of coffee drinks. Doing so would lead to an overwhelming majority of records being normalized as "Coffee," whereas if we further rename/categorize such records by "portion sizes," it would provide additional information useful and interesting for our analysis in U1 and because we now have a clearer understanding of what U1 may entail, this has helped guide us in our renaming decision process. We now have "Coffee" referring to a cup of black coffee, and "Coffee (Small Pot)," "Coffee (Pot)," and "Coffee (Large Pot)" referring to different pot sizes of coffee being served. We could thus analyze data and trends by serving portions that are intrinsically relevant to U1.

2. Based on the dataset, has coffee always been on restaurants' menus since the earliest record available?
   - ○ Example: Upon cleaning our dataset, we could simply extract the smallest valid "first_appeared" across all dishes (that fulfill integrity constraints, e.g., it should be a valid year within the dataset's year range), as well as the smallest valid "first_appeared" year for each normalized coffee dish, to see if various coffee dishes were only introduced much later to the menus found in the dataset.
   - ○ How this clarifies or guides our cleaning work:
     i. Similarly, we now know that if we cleaned the aforementioned columns, we would have sufficient clean data to answer this question.
     ii. We also now know that it is important to ensure that integrity constraints are met, in particular to the "first_appeared" column; otherwise, the answer we would obtain for this question may be inaccurate.

3. How did coffee dish prices change over time?
   - ○ Example: Plot a graph displaying the min and max values for each year for each normalized coffee dish

- Similar to the way we pre-processed “menus_appeared” in answering question 1, we could also for example find the weighted averages (by “menus_appeared” for the record) of min or max values of such prices across range of relevant years for each normalized coffee dish (through “lowest_price” and “highest_price”), and see if there might be any trend prices.
- An obvious analysis would also simply be to compare prices within a range of years for various normalized coffee dishes.
- Additionally, we could also process and derive values from the “lowest_price” and “highest_price” columns to be used as an indicator of prevalence to help answer question 1 as well, in a manner similar to the example given in question 1.
- How this clarifies or guides our cleaning work:
    i. Once again, we know now that by cleaning entries in the aforementioned columns, we would be able to answer this question.
    ii. As it turns out, a lot of dishes’ “lowest_price” and “highest_price” values are 0, likely either because they simply weren’t inputted, or that information may not have been available for some reason.
    iii. In ensuring that our analysis for price changes may be meaningful, we could thus apply certain integrity constraints, such as setting entries with 0 in “lowest_price” to a value that isn’t 0.

Data Cleaning Workflow Approach and Steps (W: D-> D’)

To reiterate, these are the columns we will clean in the Dish table, upon which our cleaned dataset D′ would be deemed sufficiently clean and hence fit-for-purpose for U1. Note that once again, we only need the Dish table and are excluding the Menu, MenuPage, and MenuItem tables. Also note that while we had initially renamed “id” to “dish_id” in our report in Phase 1, as we are now only concerned with using data in the Dish table, this is no longer necessary. Again, as mentioned, our concept of what our cleaned dataset D′ should look like is based on the above analysis examples for each of the three questions in U1, alongside other notes mentioned.

Dish Table (to be cleaned for records relevant to U1 as defined by keyword list for “name”)

Column	**Data Type**	**Clean?**	**Integrity Constraints**	**Cleaning Notes**
id (Primary Key)	integer	Y	1. Data type constraint integer must be met 2. Each value must be unique within the column	1. Perform cleaning to meet integrity constraints

name	string	Y	1. The data type constraint string must be met 2. For the record to be considered relevant to U1, it needs to contain one of the various coffee-related string values in a predefined keyword list	1. A predefined keyword list is to be created upon data inspection and profiling 2. Use a mix of text filtering (keyword search, regex if necessary), clustering (all available methods), and cell editing via text facet in OpenRefine to manually clean as many records relevant to U1 as possible
description	String (theoretically)	Y	It must contain some kind of text	Remove this column because it does not contain any information
menus_appeared	integer	Y	1. Data type constraint integer must be met 2. Value cannot be negative 3. Value must be at least 1 4. Value cannot be more than 17545 (no. of records in Menu table, counted by Excel)	1. Perform cleaning to meet integrity constraints 2. Handle cases where the value is 0 or negative
times_appeared	integer	N	-	-
first_appeared	year	Y	1. Data type constraint year of format YYYY must be met, i.e., each Y is a digit. 2. The value must start with either 1 or 2 as the first digit 3. Value must be at least 1840 (based on the narrative description of the dataset on the NYPL menus website) 4. Value cannot be greater than 2021 (the year in which the dataset D was uploaded to Box, which we assume to be	1. Perform cleaning to meet integrity constraints 2. Handle cases where the value is 0 or negative or when the default value of first_appeared is larger than the value of last_appeared

			the year it was downloaded from NYPL) 5. The value must be smaller or equal to last_appeared in the same record	
last_appeared	year	Y	1. Data type constraint year of format YYYY must be met, i.e., each Y is a digit. 2. The value must start with either 1 or 2 as the first digit 3. Value must be at least 1840 (based on the narrative description of the dataset on the NYPL menus website) 4. Value cannot be greater than 2021 (the year in which the dataset D was uploaded to Box, which we assume to be the year it was downloaded from NYPL) 5. The value must be larger or equal to first_appeared in the same record	1. Perform cleaning to meet integrity constraints 2. Handle cases where the value is 0 or negative or when the default value of last_appeared is smaller than the value of first_appeared
lowest_price	float	Y	1. Data type constraint float must be met 2. Value cannot be negative or 0 3. The value must be smaller or equal to highest_price in the same record	1. Perform cleaning to meet integrity constraints 2. Handle cases where the value is 0 or negative or when the default value of lowest_price is larger than the value of highest_price 3. Determine and handle cases where the value is unreasonably high
highest_price	float	Y	1. Data type constraint float must be met 2. Value cannot be negative or 0 3. Value must be larger or equal to lowest_price in the same record	1. Perform cleaning to meet integrity constraints 2. Handle cases where the value is 0 or negative, or when the default value of highest_price is smaller than the value of lowest_price

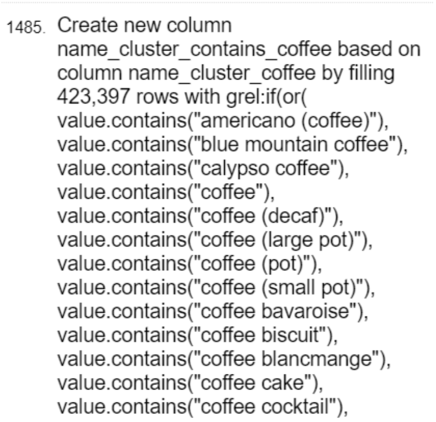
				3. Determine and handle cases where the value is unreasonably high.

Cleaning Steps (High Level)

1. Inspect and profile data using a set of common coffee keywords in OpenRefine and Excel.
    a. Before we conducted any cleaning, we first generated a list of coffee and coffee-related dishes that we would use as a starting point for the terms that we would want our final cleaned dataset to contain. This initial list was created by writing down common coffee names and quickly skimming the dataset to find some less common coffee dish names and names in other languages.
2. Clean (rename) the “name” column based on heuristic rules using OpenRefine (text filter using the predefined keyword list (see Coffee-related Keyword List below), then clustering methods and functions).
    a. Create a “name_cluster_coffee” column based on the “name” column. This is the column whose values we are going to rename (clean).
    b. Use the text filter function for each filter term (RegEx where necessary) in the predefined keyword list on the “name_cluster_coffee” column. This filters a subset of the dataset that is relevant to U1 (otherwise, 1) there will be too many clustering options, and 2) the dataset will require too much memory to process using OpenRefine).
    c. Create a text facet for the “name_cluster_coffee” column, and use all clustering methods and functions in OpenRefine, i.e., Key Collision (Fingerprint, n-Gram Fingerprint, Metaphone3, Cologne Phonetic, Daitch-Mokotoff, Beider-Morse) and Nearest Neighbor (Levenshtein, PPM). If a cluster of entries makes sense to be renamed together, do so. Otherwise, rename each entry individually. Do this based on the heuristic rules outlined further below (see Heuristic Rules and Examples).
    d. After using clustering methods and functions, scan through each row (of the text-filtered records) and rename where it makes sense (as clustering may not capture every record).
    e. To ensure we did not miss any records that are relevant to U1, use the choices shown in the text facet in OpenRefine to see if there are records we may have missed.
    f. While this process was highly manual, it was necessary to clean the dataset so that it would best suit our use case. We thus end up with records relevant to U1, each having one of the normalized coffee dish names (see Normalized Coffee Dish Names below) in the “name_cluster_coffee” column.
3. Set all values in the “name” column to lowercase.

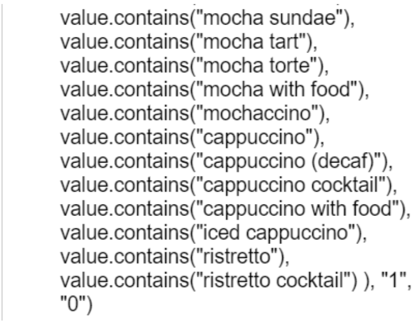
   a. During an initial evaluation of the dataset, we observed that some entries in the name column are all capitalized, some are all lowercase, and some are a mix. To make the dataset appear more uniform, we changed the values in name_cluster_coffee to be all lowercase.

4. Remove the description column.
   a. Some of the entries in the name column contain the name of the dish and a description, separated by a comma. We originally planned to separate these entries to place the description in the description column, however this proved to be more difficult than anticipated and because of the clustering and renaming that was done in step 2 above, many of the so-called descriptions that were in the name column were removed.
5. Filter for records containing keywords found in predefined list (from step 2) i.e. only records relevant to U1 are extracted, using OpenRefine. Export CSV file.
   a. Given the final list of normalized coffee dishes outlined in Cleaning Step 3 below, we used a GREL expression to filter our dataset down to entries that only contained one or more of the values in our predefined list. Then filtered our dataset to only show these values.
   b.



```
1485. Create new column
      name_cluster_contains_coffee based on
      column name_cluster_coffee by filling
      423,397 rows with grel:if(or(
      value.contains("americano (coffee)"),
      value.contains("blue mountain coffee"),
      value.contains("calypso coffee"),
      value.contains("coffee"),
      value.contains("coffee (decaf)"),
      value.contains("coffee (large pot)"),
      value.contains("coffee (pot)"),
      value.contains("coffee (small pot)"),
      value.contains("coffee bavaroise"),
      value.contains("coffee biscuit"),
      value.contains("coffee blancmange"),
      value.contains("coffee cake"),
      value.contains("coffee cocktail"),
```

... (many more entries here)



```
value.contains("mocha sundae"),
value.contains("mocha tart"),
value.contains("mocha torte"),
value.contains("mocha with food"),
value.contains("mochaccino"),
value.contains("cappuccino"),
value.contains("cappuccino (decaf)"),
value.contains("cappuccino cocktail"),
value.contains("cappuccino with food"),
value.contains("iced cappuccino"),
value.contains("ristretto"),
value.contains("ristretto cocktail") ), "1",
"0")
```

6. Remove rows where lowest_price or highest_price is missing using OpenRefine.
   a. Because we have no knowledge of what a “normal” price is for the coffee items in our dataset, if a row was missing a lowest_price or highest_price value, we removed this value from our dataset.
   b. To do this, we created new columns, lowest_price_missing and highest_price_missing, and then set the value to True if the value was missing from the row. We then filtered our dataset to show where both lowest_price_missing and highest_price_missing were False.

7. Clean the “menus_appeared” column using Python in a Jupyter Notebook to enforce the integrity constraints outlined above.
    a. Based on the integrity constraints outlined above, the menus_appeared value must be at least 1 and no greater than 17545. For the instances where the menus_appeared value is equal to 0, we assume because it is in the dataset, it appeared at least 1 time and set the menus_appeared value to 1.
8. Clean the “first_appeared” and “last_appeared” columns using Python in a Jupyter Notebook to enforce the integrity constraints outlined above.
    a. To address the invalid values in the first_appeared and last_appeared columns, if first _appeared is less than 1840, the value was set to 1840 and if first_appeared was greater than 2021, the value was set to the last_appeared value in the same row. If last_appeared is less than 1840, we set it to the first_appeared value in the row, if last_appeared is greater than 2021, we set the value to 2021.
9. Clean the “lowest_price” and “highest_price” columns using Python in a JupyterNotebook to enforce the integrity constraints outlined above.
    a. Having removed entries where the lowest_price or highest_price value was missing, we cleaned these columns to ensure they adhered to the integrity constraints of being non-negative, lowest_price <= highest price, and highest_price >= lowest_price.
10. Check integrity constraint violations of original dataset D and cleaned dataset D′ using SQL queries via Python in Jupyter Notebook.

Rationale for each Cleaning Step (High Level)

1. This is so that we understand what kinds of values are actually found in the “name” column that are actually relevant to U1.
2. This step cleans the entries in the “name” column for records relevant to U1. Without this, we would not have a dataset with a cleaned name column that contains normalized coffee dish names, where proceeding to analysis would be neither informative nor meaningful. OpenRefine is used here as the main tool because cleaning (renaming) entries in this column requires a lot of minor decisions based on various heuristic rules (outlined below), where doing so via text filtering, clustering methods and functions and manual editing likely proves to better cover as many relevant records as we could in a reasonable amount of time.
3. Changing data values in the “name_cluster_coffee” column to lowercase makes it more uniform and helps with pre-processing after cleaning and prior to analysis (U1).
4. As the description column is empty, keeping it would instead be misleading for downstream activities as it provides no additional information.
5. As mentioned in cleaning step 5, as we do not know what the “normal” price for coffee items in our dataset is, removing records where lowest_price or highest_price is missing makes more sense than deriving assumed values for them (also because the menus_appeared count for each record may be different).

6. This step is done so that we have a filtered dataset that contains only records relevant to U1.
7. The “menus_appeared” column is cleaned to enforce the integrity constraints outlined above.
8. Similarly, the “lowest_price” and “highest_price” columns are cleaned to enforce the integrity constraints outlined above.
9. And likewise, the “first_appeared” and “last_appeared” columns are cleaned to enforce the integrity constraints outlined above.
10. This is done so that we know the data quality of the cleaned dataset D′ has actually improved compared to the data quality of the original dataset D.

Coffee-related Keyword List (Cleaning Step 2)

This keyword list was created based on expanding a set of common coffee-related keywords after inspecting the set of extracted records potentially relevant to U1 (Cleaning Step 1).

Keyword(s)	**Related Text Filter Terms (or RegEx)**
affogato	affogat, afogat
americano	coffee american, cafe american, kaffee american (?=.*cof)(?=.*americ) (?=.*caf)(?=.*americ) (?=.*kaf)(?=.*americ)
antoccino	
cafe	caf
cappuccino	cappuc, capuc
coffee	coffe, cofe
cold brew	(?=.*cold)(?=.*brew)
cortado	corta
demi-tasse	demi-, demit
doppio	dopio
espresso	espres, expres
flat black	(?=.*flat)(?=.*black)
flat white	(?=.*flat)(?=.*white)
frappe	frap

frappuccino	frappuc
kaffee	kaffe, kafe
kopi	
latte	^(?=.*latte)(?!.*platte)(?!.*latter)(?!.*carlatte)(?!.*alatte) (The above filters away other common words containing the string “latte” within them)
lungo	
macchiato	macchiat, machiat
marocchino	
mocha	mocca, moca
mochaccino	
ristretto	ristret

As mentioned in cleaning step 2 above, we used this list of keywords and related terms in OpenRefine to first do a text filter either directly via the term itself or using regular expressions before renaming (cleaning) using clustering and via manual editing.

The following are some heuristic rules we followed (as well as some examples) when deciding how to rename values in the “name” column for records deemed relevant to U1. We did this mainly based on our own understanding of the kinds of coffee dishes we would like analyzed while keeping in mind that there are likely certain coffee dishes (or drinks) that may have been more prevalent during this time period in New York or restaurants in other locations found in the dataset. This brings us to another point: having domain knowledge or being a subject matter expert in coffee would help us to perform data cleaning much better, alongside understanding what the cleaned state of data should look like so that we could analyze it. The cleaning of the “name” column could thus be considered somewhat subjective. Regardless, we tried our best to adhere to certain heuristic rules we came up with so that our cleaning efforts remained consistent and as objective as possible throughout the entire process.

Heuristic Rules and Examples

1. If a dish name refers to a single cup of black coffee (no matter the method used, except for espresso), name it as “Coffee.” Include in this dish names that may contain black coffee with milk or sugar. This also refers to the coffee being hot.
2. Where portion size isn’t specified, assume this is a single cup.

3. Where the portion size of a small pot, normal pot, or large pot is specified, name it as the type of coffee followed by "(Small Pot)," "(Pot)," or "(Large Pot)," e.g., "Coffee (Pot)."
4. Where a coffee dish name contains "iced" or refers to it as being cold, add "Iced" in front of the coffee name eg. "Iced Coffee".
5. If a dish name contains an option of different drink options including coffee, name it as the type of coffee eg. "hot chocolate, tea or coffee" will be renamed "Coffee", as this actually potentially refers to 1 or more dishes where technically "hot chocolate, tea or coffee" could be considered 3 dishes.
6. As somewhat mentioned, besides espresso, we do not further categorize other types of brewing methods, e.g., filter, French press, moka pot, as such information may sometimes be lacking, and categorizing as such may end up providing inaccurate ratios of coffees made by different brewing methods. We are also not as concerned with different brewing methods being considered as different types of coffee (except for espresso).
7. Unless the variety or origin of a coffee used implies a particularly different price range, e.g., luwak coffee (which was sadly not found in the dataset), we do not name it as such, again because this information is highly lacking, and would contribute to a highly skewed ratio of coffees with named varieties or origins (i.e., exceedingly low) compared to those that do not have any named. We keep certain names, such as "Blue Mountain Coffee," because its price range would be quite different.
8. If a coffee dish refers to a known type of coffee drink (based on Internet search results), with a typical set of ingredients and recipe, keep it as such, e.g., "Coffee Glacé," "Dutch Coffee," or "New Orleans Coffee."
9. If a coffee drink isn't known (but contains more than just black coffee and may or may not contain milk or sugar), name it as "Coffee Drink."
10. If a coffee dish is a modern known espresso-based coffee eg. americano, latte, macchiato, keep their naming as such.
11. If a coffee dish originally has "demi-tasse" in its name (i.e. an espresso in a French espresso cup, where the roast might be considered French), it is renamed to be an espresso eg. "demi-tasse, served in a pot" will be renamed "Espresso (Pot)".
12. If a coffee drink contains alcohol but isn't of a specific known coffee cocktail name, name it "Coffee Cocktail."
13. If a dish name contains coffee but is part of a larger meal set that also contains food, e.g., cookies, fruits, or a full course meal, name it "Coffee with Food." This is relevant to U1 as it contains coffee, but the price for this would obviously be different (which we do not really care about, so long as it does not refer to just coffee on its own).
14. If a dish is for a coffee-flavored food, e.g., ice cream or cake, name it as such, i.e., "Coffee Ice Cream" or "Coffee Cake," as this would help us gauge coffee's prevalence regardless and is hence relevant to U1. Similar to a previous rule, if a dish is for a food with options for different flavours, including coffee, e.g., "Strawberry, Chocolate, Coffee, or Vanilla Ice Cream," rename it as the coffee version, i.e., "Coffee Ice Cream."
15. For such coffee-flavoured foods, if they can be reasonably categorized, do so, but otherwise rename them to keep their original meaning, e.g., "Mocha Torte."

16. If a dish name is actually a store product (rather than a restaurant dish), one which contains ground coffee or coffee beans, name it “Coffee in Tin” (as it appears they were typically sold in tins). Again, this contributes to the prevalence of coffee, which is hence relevant to U1.
17. Where a dish name looks like it may be a coffee dish but we are unable to tell what it is, search for it on the Internet to determine how we should rename or categorize it.
18. Where a dish name looks like it may be a coffee dish but is in a non-English language, use Google Translate to understand what it may be in English before determining if it is relevant to U1.
19. All dish names that are in non-English languages relevant to U1 are renamed in English eg. “kaffee” and “cafe” (where cafe here refers to black coffee in French) are renamed “Coffee”.
20. Where a dish name contains the word “cafe,” distinguish and decide between whether it refers to a type of coffee (as in the French word cafe) or if it refers to a cafe as a location, e.g., a cafe restaurant.

Following our approach and heuristics rules, we thus cleaned entries in the “name” column for records relevant to U1, and ended up with the following list of normalized coffee dish names based on the different values we found and renaming decisions made.

Normalized Coffee Dish Names (Cleaning Step 2)

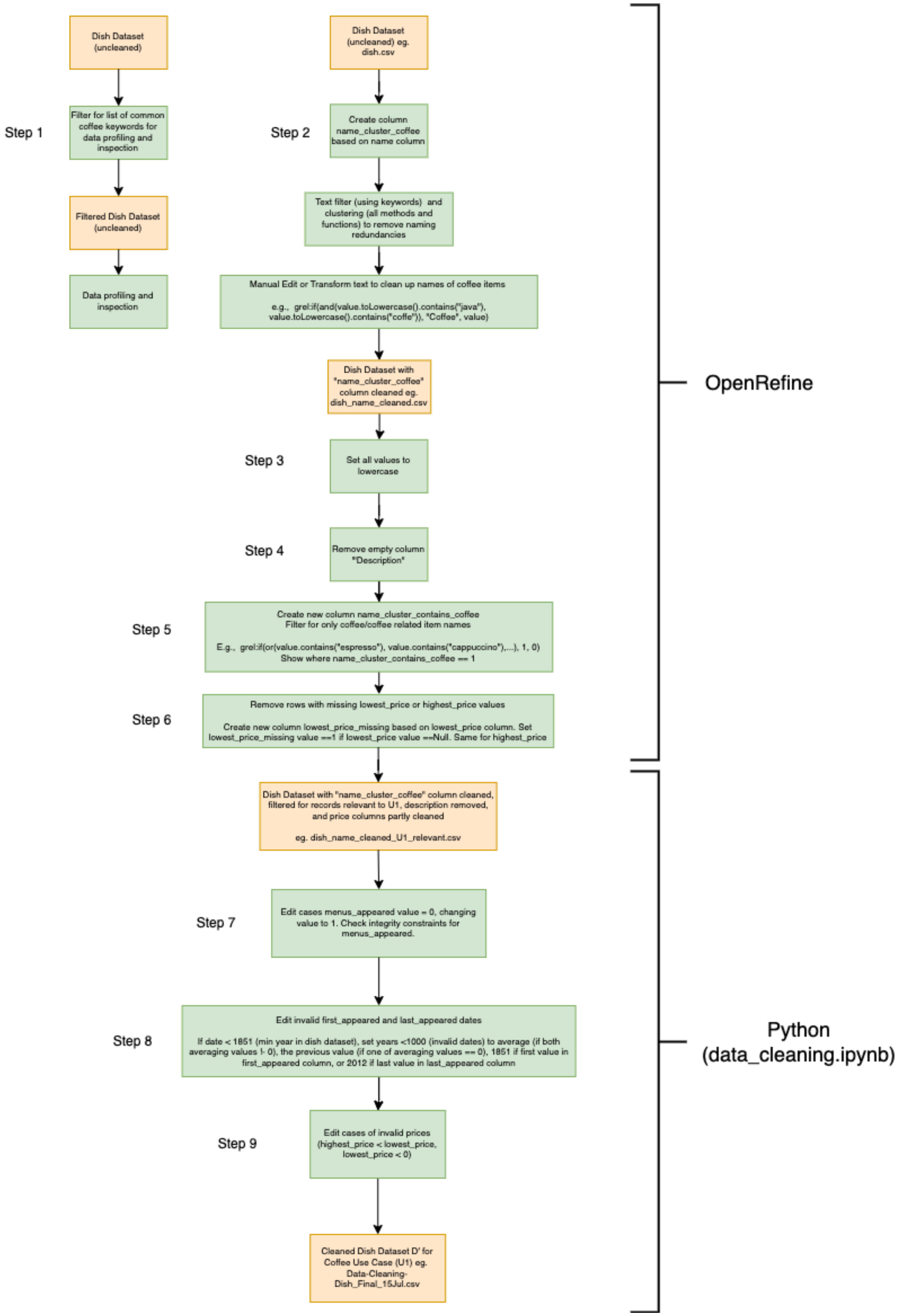
Keyword	Normalized Coffee Dishes	No. of Records
Coffee	Americano (Coffee) Blue Mountain Coffee Calypso Coffee Coffee Coffee (Decaf) Coffee (Large Pot) Coffee (Pot) Coffee (Small Pot) Coffee Bavaroise Coffee Biscuit Coffee Blancmange Coffee Cake Coffee Cocktail Coffee Drink Coffee Eclair Coffee Glacé Coffee Granita Coffee Ice Cream Coffee in Tin Coffee Jelly	15 2 2 1841 134 1 507 5 9 4 1 104 77 24 43 69 3 218 45 22

	Coffee Liqueur	17
	Coffee Mousse	8
	Coffee Parfait	65
	Coffee Pastry	15
	Coffee Pudding	6
	Coffee Sambuca	2
	Coffee Souffle	2
	Coffee Sundae	24
	Coffee with Cigar	1
	Coffee with Food	2191
	Cuban Coffee	3
	Dutch Coffee	5
	Frappe (Coffee)	18
	French Coffee	38
	French Coffee (Decaf)	5
	Iced Americano (Coffee)	1
	Iced Coffee	197
	Iced Coffee (Pot)	27
	Iced Frappe (Coffee)	1
	Irish Coffee	55
	Kona Coffee	19
	L'Absinthe Coffee	1
	Latte (Coffee)	3
	Macchiato (Coffee)	4
	Mocha (Coffee)	33
	New Orleans Coffee	3
	Postum (Coffee Substitute)	47
	Russian Coffee	4
	Spanish Coffee	4
	Swiss Coffee	2
	Toulouse Lautrec Coffee	1
	Turkish Coffee	58
	52 terms	5986 records total
Cafe	Cafe au Lait	16
	Cafe au Lait (Pot)	2
	Cafe Brulot	18
	Cafe con Leche	1
	Cafe Liegeois	1
	Iced Cafe Au Lait	1
	6 terms	39 records total
Espresso	Espresso	273

	Espresso (Decaf)	8
	Espresso (Pot)	16
	Espresso Cake	3
	Espresso Cocktail	12
	Espresso Drink	3
	Espresso Pastry	2
	Espresso Wafers	1
	Espresso with Food	2
	Iced Espresso	2
	Tiramisu (Espresso)	2
	11 terms	324 records total
Mocha	Iced Mocha	1
	Mocha (Pot)	1
	Mocha Bavaroise	1
	Mocha Cake	17
	Mocha Drink	1
	Mocha Eclair	6
	Mocha Glacé	1
	Mocha Ice Cream	20
	Mocha Jelly	1
	Mocha Macaron	2
	Mocha Mousse	2
	Mocha Parfait	5
	Mocha Pastry	5
	Mocha Pie	9
	Mocha Praline	1
	Mocha Pudding	8
	Mocha Souffle	2
	Mocha Sundae	1
	Mocha Tart	4
	Mocha Torte	6
	Mocha with Food	2
	Mochaccino	2
	22 terms (The term “Mocha (Coffee)” is included previously under “Coffee”)	98 records total
Cappuccino	Cappuccino	34
	Cappuccino (Decaf)	2
	Cappuccino Cocktail	6
	Cappuccino with Food	1
	Iced Cappuccino	4
	5 terms	47 records total
Ristretto	Ristretto	29

	Ristretto Cocktail 2 terms	1 30 records total

Overall (“outer”) Workflow W1


Dish Dataset (uncleaned)
Step 1
Filter for list of common coffee keywords for data profiling and inspection
Filtered Dish Dataset (uncleaned)
Data profiling and inspection
Dish Dataset (uncleaned) eg. dish.csv
Step 2
Create column name_cluster_coffee based on name column
Text filter (using keywords) and clustering (all methods and functions) to remove naming redundancies
Manual Edit or Transform text to clean up names of coffee items
e.g., grel:if(and(value.toLowercase().contains("java"), value.toLowercase().contains("coffe")), "Coffee", value)
Dish Dataset with "name_cluster_coffee" column cleaned eg. dish_name_cleaned.csv
Step 3
Set all values to lowercase
Step 4
Remove empty column "Description"
Step 5
Create new column name_cluster_contains_coffee
Filter for only coffee/coffee related item names
E.g., grel:if(or(value.contains("espresso"), value.contains("cappuccino"),...), 1, 0)
Show where name_cluster_contains_coffee == 1
Step 6
Remove rows with missing lowest_price or highest_price values
Create new column lowest_price_missing based on lowest_price column. Set lowest_price_missing value ==1 if lowest_price value ==Null. Same for highest_price
OpenRefine
Dish Dataset with "name_cluster_coffee" column cleaned, filtered for records relevant to U1, description removed, and price columns partly cleaned
eg. dish_name_cleaned_U1_relevant.csv
Step 7
Edit cases menus_appeared value = 0, changing value to 1. Check integrity constraints for menus_appeared.
Step 8
Edit invalid first_appeared and last_appeared dates
If date < 1851 (min year in dish dataset), set years <1000 (invalid dates) to average (if both averaging values != 0), the previous value (if one of averaging values == 0), 1851 if first value in first_appeared column, or 2012 if last value in last_appeared column
Step 9
Edit cases of invalid prices (highest_price < lowest_price, lowest_price < 0)
Cleaned Dish Dataset D' for Coffee Use Case (U1) eg. Data-Cleaning-Dish_Final_15Jul.csv
Python (data_cleaning.ipynb)

As the workflow for part of the “inner” data cleaning workflow W2, in particular to OpenRefine, is highly detailed, i.e., a lot of steps, we thought it would make more sense to link it instead. Please see https://github.com/hhlim2/cs513_team6/blob/main/ORYW_python_workflow.pdf

We chose to clean the “name” column first as this would help us identify records that are relevant to U1, i.e., records whose “name” column is deemed irrelevant would not be cleaned. By outputting a CSV with a “name_cluster_coffee” column after cleaning step 2 in OpenRefine, where only records relevant to U1 would contain one of the normalized coffee dish names as defined above, we could thus then only clean records relevant to U1 for the rest of the relevant columns. The rationale for each cleaning step was explained previously above, where the order of certain columns that were cleaned isn’t actually important, e.g., “menus_appeared” and “lowest_price,” as they do not have dependencies. After all cleaning steps are completed, we then run SQL queries to ensure integrity constraints are enforced, and data quality has indeed improved (see the section below). The overall design of the workflow thus ensures that we clean a subset of the original dataset D into a cleaned dataset D′ that is relevant to U1 and ensures that data quality has improved, providing a dataset directly useful for pre-processing and analysis in U1.

As for tools used, Excel and OpenRefine were first used to inspect and profile the dataset, as these are tools that help to quickly filter and visualize records in a table/CSV format.

We then make use of OpenRefine again to take advantage of its features in text filtering, regular expressions, text facet, and clustering methods (and functions) for ease of cleaning (renaming) the “name” column, particularly also because this is text data.

After the “name” column is considered clean enough, for records relevant to U1, we make use of OpenRefine’s features again to proceed with steps 3 to 6, yielding a filtered dataset with only records relevant to U1.

We then load this filtered dataset (where “name” is considered cleaned) into a Jupyter notebook and clean the “menus_appeared”, “lowest_price”, “highest_price”, “first_appeared” and “last_appeared” columns using Python (Pandas library) to enforce integrity constraints as outlined, and to ensure it is fit for use in U1, yielding our cleaned dataset D′. Pandas is used here as it contains functions that help with handling and cleaning numerical data.

Lastly, we use SQL via Python in a Jupyter notebook to check integrity constraints of both the original dataset D and cleaned dataset D′ as SQL is designed for doing so efficiently, in particular to relational databases.

Comparison of Work Done (Phase 1 Plan to Phase 2 Actions)

In Phase 1, our cleaning actions consisted of the following steps:

In the name column, some entries are all capitalized. We will change the entries to be all lowercase.

In the name column, some of the entries contain the name of the dish and a description, separated by a comma. We will separate these entries to place the description in the description column.

In the first_appeared and last_appeared columns we see some invalid values such as 1 and 2928. Assuming that all values in these two columns are valid years, we will replace the invalid values with the average of the 4 values surrounding them.

In the lowest_price and highest_price, we have some missing values and some invalid values, such as 0 for the highest_price. To resolve this issue, we will remove the rows with invalid values.

To successfully complete UC1 outlined above, we will need to extract dish names containing or related to “coffee.” Data quality issues that we will need to address to complete this task successfully are deduplicating dishes based on name and identifying names that are semantically similar or even the same, depending on the extent of analysis we would like to perform.

Here is a comparison with the work done in Phase 2.

The capitalization step remained the same and was completed.

The name column splitting step was removed in lieu of simplifying a majority of the dish names. For example, “Coffee, which makes the politician wise and see through all things, with his half-shut eyes” became “coffee”.

When we fixed the invalid dates, rather than replacing invalid years with the average of the 4 surrounding cells, we opted to simply replace values < 1840 with 1840 and values > 2021 with 2021.

The step to remove the rows with missing lowest_price or highest_price remained the same and was completed.

The step to extract coffee dish names remained mostly the same but proved to be a much more manual task than anticipated, as we did not anticipate the amount of variation in the dish names and the addition of dish names in other languages.

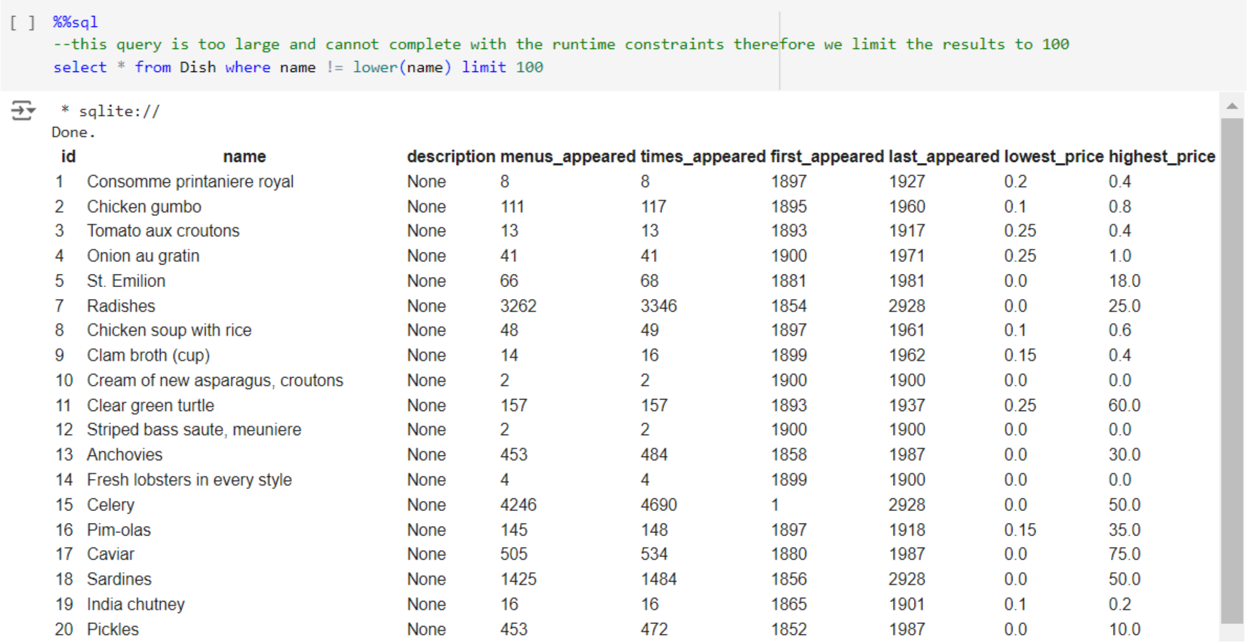
Our actual cleaning steps/workflow when doing data cleaning in Phase 2, as compared to our original plan in Phase 1, was also different. While we initially anticipated in Phase 1 to work in an iterative approach in terms of finding new data quality problems as we cleaned, we later realized this to be a part of data profiling, which we have then done as our first step in Phase 2. We were able to do this in Phase 2 as we have since more clearly defined our main use case U1, where our potential analysis examples served as a guide in identifying what to do with various values in the “name” column in cleaning steps 1 and 2, as well as in principle ensuring we have the relevant data to perform U1 once our dataset is cleaned through the entire workflow.

Data Quality Changes and Evidence that Data Quality Was Improved

We used SQL to validate that the data cleaning we completed was successful. Some of the SQL commands are quite lengthy and will not fit in a single screenshot. Please refer to the sql_queries.ipynb notebook for the full queries and returned values

Cleaning Action 1 - Set all values in the name column to lowercase.

Before (a subset of violations, the whole returned table will not fit in one screenshot)

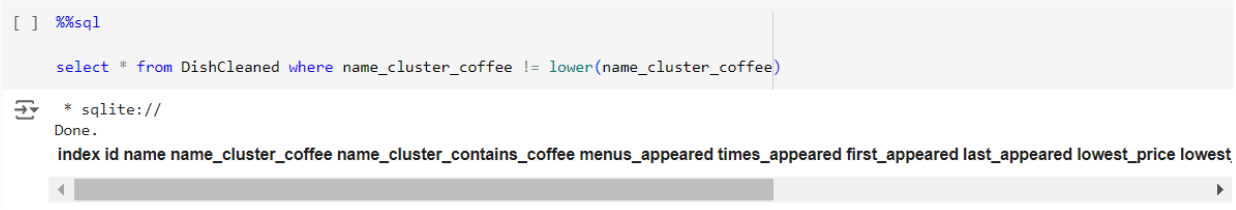


```
%%sql
--this query is too large and cannot complete with the runtime constraints therefore we limit the results to 100
select * from Dish where name != lower(name) limit 100
```

```
* sqlite://
Done.
```

id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
1	Consomme printaniere royal	None	8	8	1897	1927	0.2	0.4
2	Chicken gumbo	None	111	117	1895	1960	0.1	0.8
3	Tomato aux croutons	None	13	13	1893	1917	0.25	0.4
4	Onion au gratin	None	41	41	1900	1971	0.25	1.0
5	St. Emilion	None	66	68	1881	1981	0.0	18.0
7	Radishes	None	3262	3346	1854	2928	0.0	25.0
8	Chicken soup with rice	None	48	49	1897	1961	0.1	0.6
9	Clam broth (cup)	None	14	16	1899	1962	0.15	0.4
10	Cream of new asparagus, croutons	None	2	2	1900	1900	0.0	0.0
11	Clear green turtle	None	157	157	1893	1937	0.25	60.0
12	Striped bass saute, meuniere	None	2	2	1900	1900	0.0	0.0
13	Anchovies	None	453	484	1858	1987	0.0	30.0
14	Fresh lobsters in every style	None	4	4	1899	1900	0.0	0.0
15	Celery	None	4246	4690	1	2928	0.0	50.0
16	Pim-olas	None	145	148	1897	1918	0.15	35.0
17	Caviar	None	505	534	1880	1987	0.0	75.0
18	Sardines	None	1425	1484	1856	2928	0.0	50.0
19	India chutney	None	16	16	1865	1901	0.1	0.2
20	Pickles	None	453	472	1852	1987	0.0	10.0

After

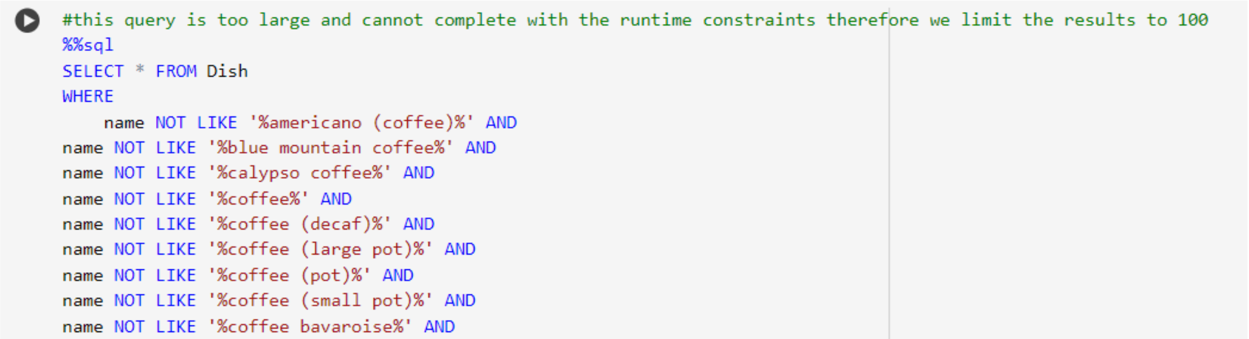


```
%%sql

select * from DishCleaned where name_cluster_coffee != lower(name_cluster_coffee)
```

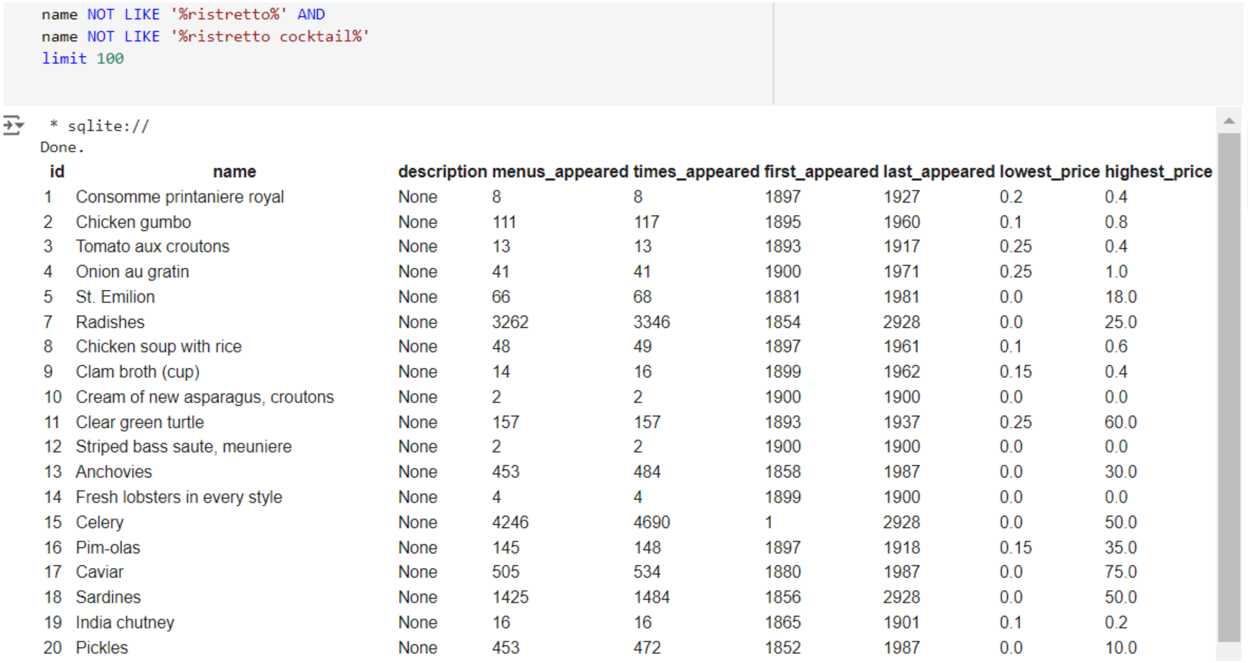
```
* sqlite://
Done.
```

index	id	name	name_cluster_coffee	name_cluster_contains_coffee	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	lowest

Data Cleaning Action 2 - Filter the dataset so that only coffee dishes remain.
(This SQL query is too lengthy and will not fit in one screenshot.)



```
#this query is too large and cannot complete with the runtime constraints therefore we limit the results to 100
%%sql
SELECT * FROM Dish
WHERE
    name NOT LIKE '%americano (coffee)%' AND
name NOT LIKE '%blue mountain coffee%' AND
name NOT LIKE '%calypso coffee%' AND
name NOT LIKE '%coffee%' AND
name NOT LIKE '%coffee (decaf)%' AND
name NOT LIKE '%coffee (large pot)%' AND
name NOT LIKE '%coffee (pot)%' AND
name NOT LIKE '%coffee (small pot)%' AND
name NOT LIKE '%coffee bavaroise%' AND
```

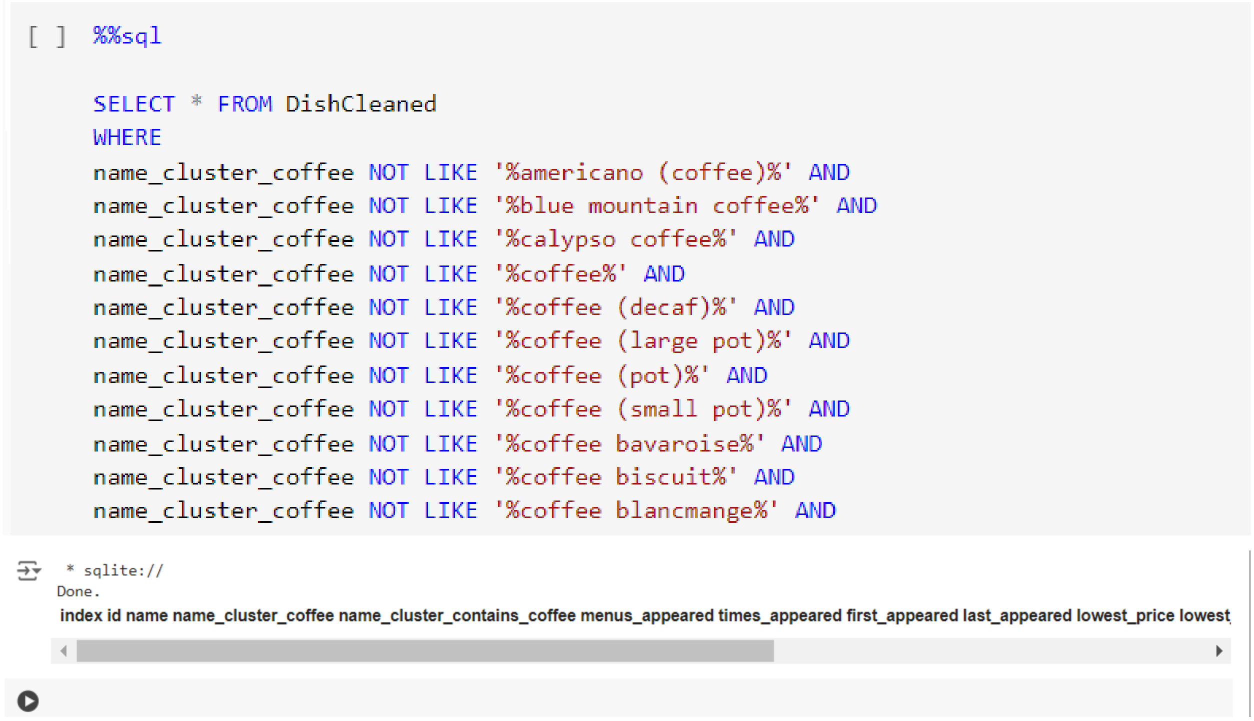


```
name NOT LIKE '%ristretto%' AND
name NOT LIKE '%ristretto cocktail%'
limit 100
```

```
* sqlite://
Done.
```

id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
1	Consomme printaniere royal	None	8	8	1897	1927	0.2	0.4
2	Chicken gumbo	None	111	117	1895	1960	0.1	0.8
3	Tomato aux croutons	None	13	13	1893	1917	0.25	0.4
4	Onion au gratin	None	41	41	1900	1971	0.25	1.0
5	St. Emilion	None	66	68	1881	1981	0.0	18.0
7	Radishes	None	3262	3346	1854	2928	0.0	25.0
8	Chicken soup with rice	None	48	49	1897	1961	0.1	0.6
9	Clam broth (cup)	None	14	16	1899	1962	0.15	0.4
10	Cream of new asparagus, croutons	None	2	2	1900	1900	0.0	0.0
11	Clear green turtle	None	157	157	1893	1937	0.25	60.0
12	Striped bass saute, meuniere	None	2	2	1900	1900	0.0	0.0
13	Anchovies	None	453	484	1858	1987	0.0	30.0
14	Fresh lobsters in every style	None	4	4	1899	1900	0.0	0.0
15	Celery	None	4246	4690	1	2928	0.0	50.0
16	Pim-olas	None	145	148	1897	1918	0.15	35.0
17	Caviar	None	505	534	1880	1987	0.0	75.0
18	Sardines	None	1425	1484	1856	2928	0.0	50.0
19	India chutney	None	16	16	1865	1901	0.1	0.2
20	Pickles	None	453	472	1852	1987	0.0	10.0

After



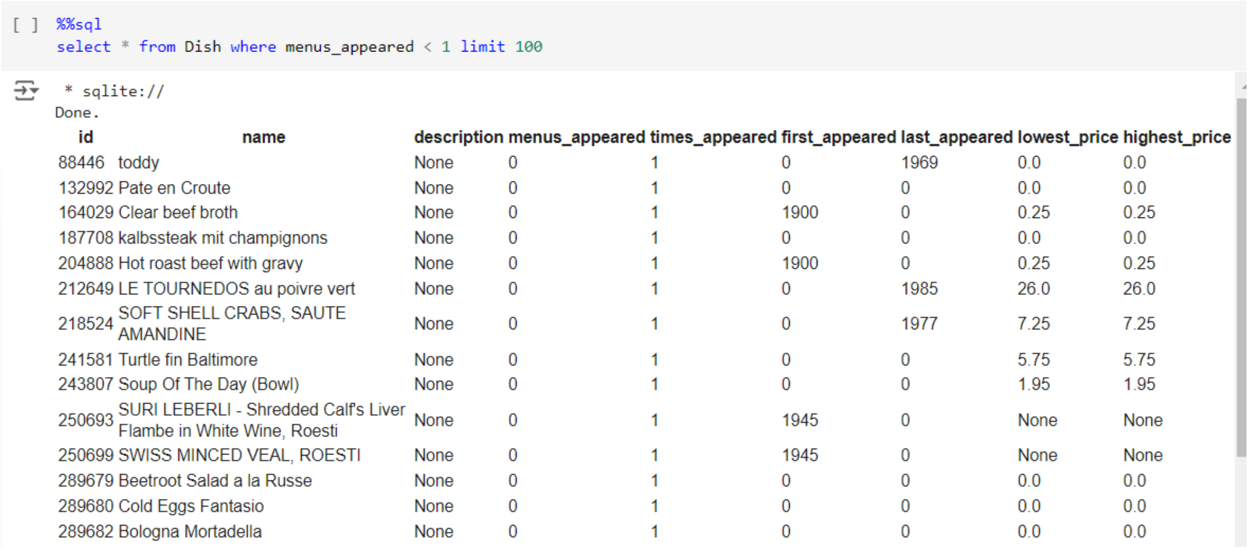
```
[ ] %%sql

    SELECT * FROM DishCleaned
    WHERE
    name_cluster_coffee NOT LIKE '%americano (coffee)%' AND
    name_cluster_coffee NOT LIKE '%blue mountain coffee%' AND
    name_cluster_coffee NOT LIKE '%calypso coffee%' AND
    name_cluster_coffee NOT LIKE '%coffee%' AND
    name_cluster_coffee NOT LIKE '%coffee (decaf)%' AND
    name_cluster_coffee NOT LIKE '%coffee (large pot)%' AND
    name_cluster_coffee NOT LIKE '%coffee (pot)%' AND
    name_cluster_coffee NOT LIKE '%coffee (small pot)%' AND
    name_cluster_coffee NOT LIKE '%coffee bavaroise%' AND
    name_cluster_coffee NOT LIKE '%coffee biscuit%' AND
    name_cluster_coffee NOT LIKE '%coffee blancmange%' AND
```

```
* sqlite://
Done.
```

index id name name_cluster_coffee name_cluster_contains_coffee menus_appeared times_appeared first_appeared last_appeared lowest_price lowest

Data Cleaning Action 3 - Ensure that menus_appeared >= 1

Before (a subset of violations, the whole returned table will not fit in one screenshot)

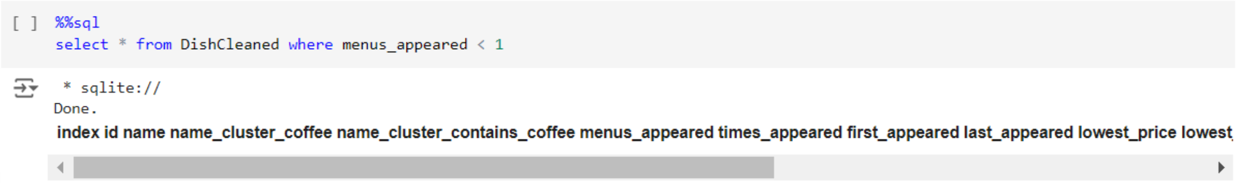


```
[ ] %%sql
    select * from Dish where menus_appeared < 1 limit 100
```

```
 * sqlite://
Done.
```

id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
88446	toddy	None	0	1	0	1969	0.0	0.0
132992	Pate en Croute	None	0	1	0	0	0.0	0.0
164029	Clear beef broth	None	0	1	1900	0	0.25	0.25
187708	kalbssteak mit champignons	None	0	1	0	0	0.0	0.0
204888	Hot roast beef with gravy	None	0	1	1900	0	0.25	0.25
212649	LE TOURNEDOS au poivre vert	None	0	1	0	1985	26.0	26.0
218524	SOFT SHELL CRABS, SAUTE AMANDINE	None	0	1	0	1977	7.25	7.25
241581	Turtle fin Baltimore	None	0	1	0	0	5.75	5.75
243807	Soup Of The Day (Bowl)	None	0	1	0	0	1.95	1.95
250693	SURI LEBERLI - Shredded Calf's Liver Flambe in White Wine, Roesti	None	0	1	1945	0	None	None
250699	SWISS MINCED VEAL, ROESTI	None	0	1	1945	0	None	None
289679	Beetroot Salad a la Russe	None	0	1	0	0	0.0	0.0
289680	Cold Eggs Fantasio	None	0	1	0	0	0.0	0.0
289682	Bologna Mortadella	None	0	1	0	0	0.0	0.0

After



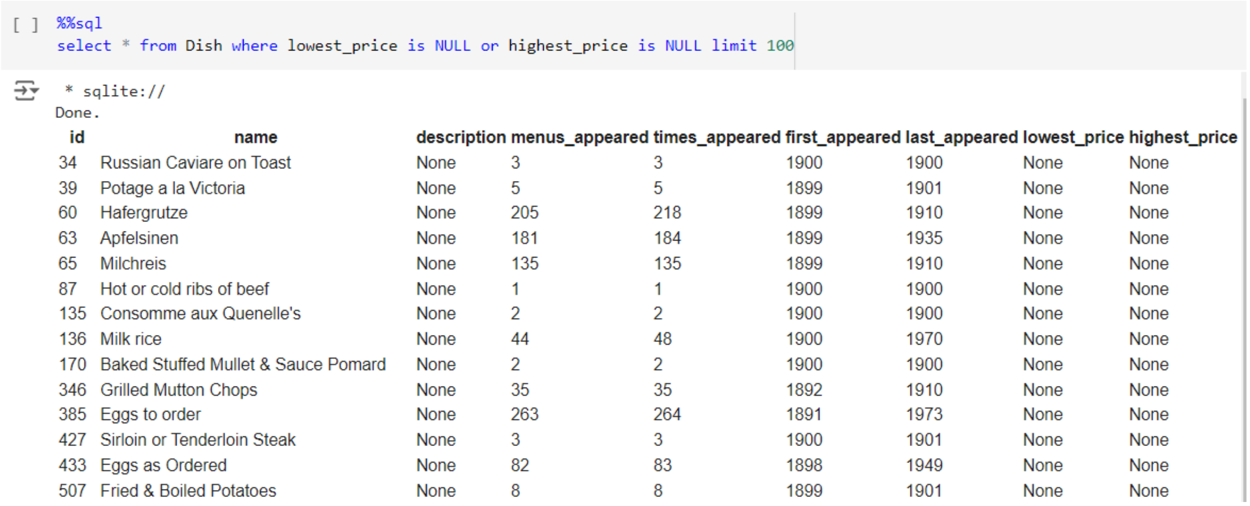
```
[ ] %%sql
    select * from DishCleaned where menus_appeared < 1
```

```
 * sqlite://
Done.
```

index	id	name	name_cluster_coffee	name_cluster_contains_coffee	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	lowest

Data Cleaning Action 4 - Remove rows where lowest_price or highest_price is missing

Before (a subset of violations, the whole returned table will not fit in one screenshot)

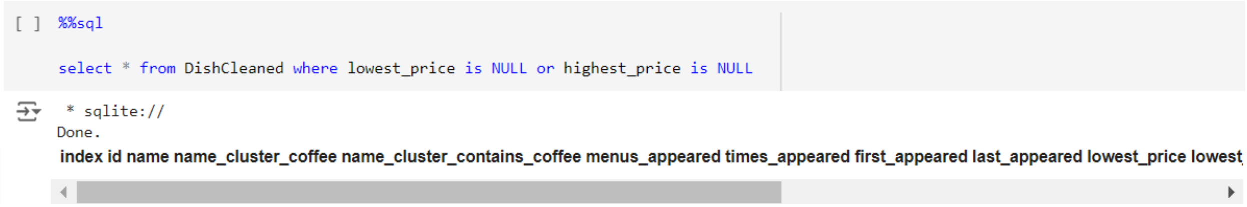


```
[ ] %%sql
    select * from Dish where lowest_price is NULL or highest_price is NULL limit 100
```

```
 * sqlite://
Done.
```

id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
34	Russian Caviare on Toast	None	3	3	1900	1900	None	None
39	Potage a la Victoria	None	5	5	1899	1901	None	None
60	Hafergrutze	None	205	218	1899	1910	None	None
63	Apfelsinen	None	181	184	1899	1935	None	None
65	Milchreis	None	135	135	1899	1910	None	None
87	Hot or cold ribs of beef	None	1	1	1900	1900	None	None
135	Consomme aux Quenelle's	None	2	2	1900	1900	None	None
136	Milk rice	None	44	48	1900	1970	None	None
170	Baked Stuffed Mullet & Sauce Pomard	None	2	2	1900	1900	None	None
346	Grilled Mutton Chops	None	35	35	1892	1910	None	None
385	Eggs to order	None	263	264	1891	1973	None	None
427	Sirloin or Tenderloin Steak	None	3	3	1900	1901	None	None
433	Eggs as Ordered	None	82	83	1898	1949	None	None
507	Fried & Boiled Potatoes	None	8	8	1899	1901	None	None

After

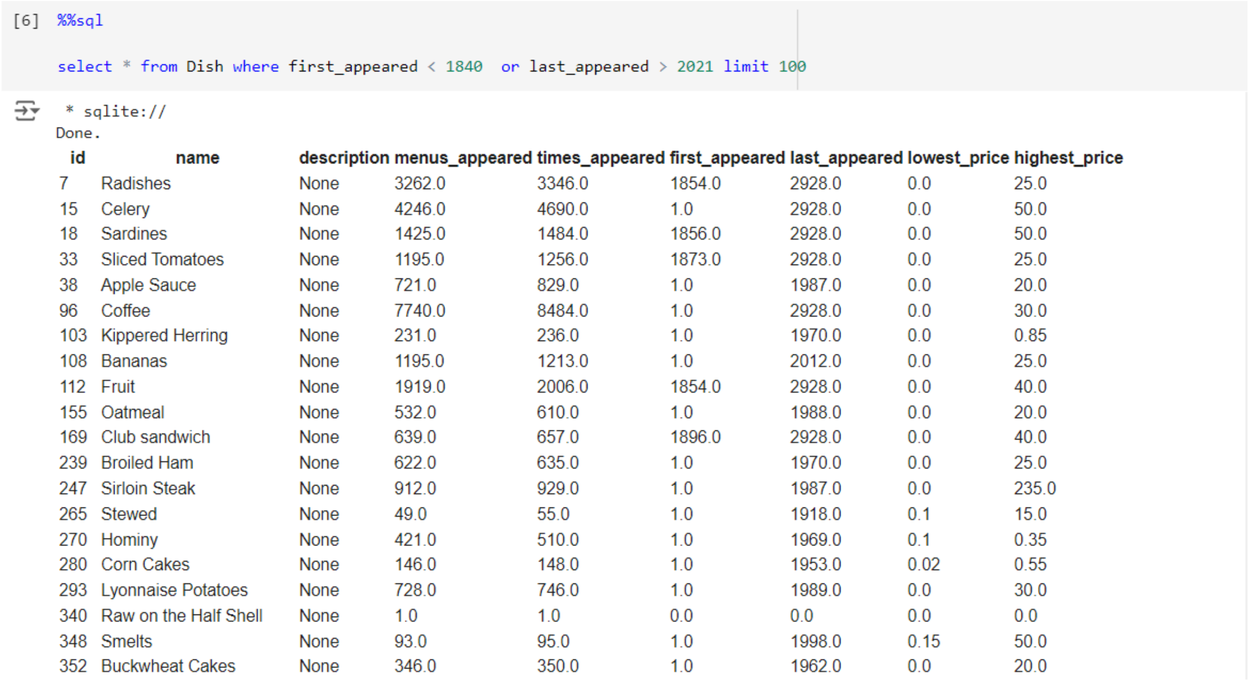


```
[ ] %%sql

    select * from DishCleaned where lowest_price is NULL or highest_price is NULL
```

```
 * sqlite://
Done.
index id name name_cluster_coffee name_cluster_contains_coffee menus_appeared times_appeared first_appeared last_appeared lowest_price lowest
```

Data Cleaning Action 5 - Fix invalid dates.

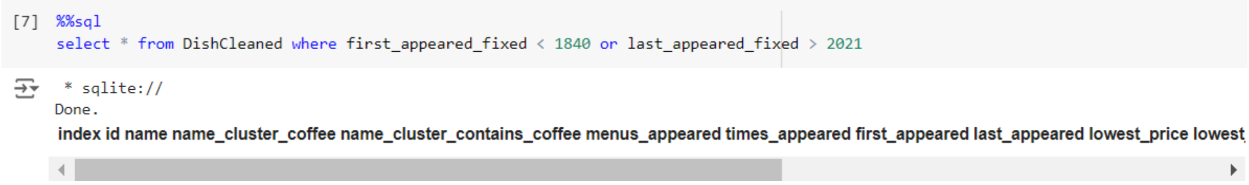
Before (a subset of violations, the whole returned table will not fit in one screenshot)



```
[6] %%sql

    select * from Dish where first_appeared < 1840  or last_appeared > 2021 limit 100
```

```
 * sqlite://
Done.
 id          name          description menus_appeared times_appeared first_appeared last_appeared lowest_price highest_price
 7    Radishes             None        3262.0         3346.0         1854.0         2928.0        0.0          25.0
 15   Celery               None        4246.0         4690.0         1.0            2928.0        0.0          50.0
 18   Sardines             None        1425.0         1484.0         1856.0         2928.0        0.0          50.0
 33   Sliced Tomatoes      None        1195.0         1256.0         1873.0         2928.0        0.0          25.0
 38   Apple Sauce          None        721.0          829.0          1.0            1987.0        0.0          20.0
 96   Coffee               None        7740.0         8484.0         1.0            2928.0        0.0          30.0
 103  Kippered Herring     None        231.0          236.0          1.0            1970.0        0.0          0.85
 108  Bananas              None        1195.0         1213.0         1.0            2012.0        0.0          25.0
 112  Fruit                None        1919.0         2006.0         1854.0         2928.0        0.0          40.0
 155  Oatmeal              None        532.0          610.0          1.0            1988.0        0.0          20.0
 169  Club sandwich        None        639.0          657.0          1896.0         2928.0        0.0          40.0
 239  Broiled Ham          None        622.0          635.0          1.0            1970.0        0.0          25.0
 247  Sirloin Steak        None        912.0          929.0          1.0            1987.0        0.0          235.0
 265  Stewed               None        49.0           55.0           1.0            1918.0        0.1          15.0
 270  Hominy               None        421.0          510.0          1.0            1969.0        0.1          0.35
 280  Corn Cakes           None        146.0          148.0          1.0            1953.0        0.02         0.55
 293  Lyonnaise Potatoes   None        728.0          746.0          1.0            1989.0        0.0          30.0
 340  Raw on the Half Shell None       1.0            1.0            0.0            0.0           0.0          0.0
 348  Smelts               None        93.0           95.0           1.0            1998.0        0.15         50.0
 352  Buckwheat Cakes      None        346.0          350.0          1.0            1962.0        0.0          20.0
```

After



```
[7] %%sql
    select * from DishCleaned where first_appeared_fixed < 1840 or last_appeared_fixed > 2021
```

```
 * sqlite://
Done.
index id name name_cluster_coffee name_cluster_contains_coffee menus_appeared times_appeared first_appeared last_appeared lowest_price lowest
```

Summary and Quantified Results of Data Cleaning Efforts

Column Changed	Action Summary	# Cells Changed
Name	Cluster and rename “name” to a simplified version, filter to	6530

	only show coffee dishes	
Description	Remove Column	All
Menus_appeared	If menus_appeared = 0, set menus_appeared = 1	40
First_appeared	If first _appeared is less than 1840, the value was set to 1840 and if first_appeared was greater than 2021, the value was set to the last_appeared value in the same row.	879
Last_appeared	If last_appeared is less than 1840, we set it to the first_appeared value in the row; if last_appeared is greater than 2021, we set the value to 2021.	874
Lowest_price	Remove the row if the value is missing	420
Highest_price	Remove the row if the value is missing	420

Summary of Contribution of Each Team Member

Shu completed the highly manual task of identifying, filtering, and cleaning the “name” column in the dataset so that we could easily identify coffee dishes. Hannah continued the cleaning process by performing the cleaning tasks done in a Jupyter Notebook, validated that the cleaning was successful with SQL queries, and created the workflow diagram. Both Shu and Hannah contributed to the writing of this report.

Findings and Problems Encountered

The problems we encountered were mainly with 1) finding out which records were relevant to U1 and figuring out how to filter for them, and 2) understanding the meaning of various values in the “name” column in the Dish CSV and figuring out how to rename them appropriately for U1.

In profiling our dataset relevant to U1, and via a highly manual cleaning approach in cleaning step 2, we have found a process that worked for us in cleaning (and obtaining) what was likely most records in the dataset that are relevant to U1. We have also found that the dataset in particular to the “name” column was highly “dirty” (for U1), and decided to create and use a list of normalized coffee dish names in order to clean our dataset.

In doing so, we have thus also encountered a major problem in automating such cleaning efforts, where understanding and devising rules to be used via, for example, Python with RegEx to clean the dataset using a script may have otherwise proved inaccurate or too time-consuming for a one-off cleaning process (as opposed to manually cleaning them through OpenRefine’s clustering methods). As such, with highly manual cleaning, reproducibility might also become an issue (as it would take a similar amount of effort to perform the same steps), even with workflow documentation.

Lessons Learned

1. Clearly defining our use case has helped us go a long way in conceptually defining what is considered “good enough” or “clean enough” and making various data cleaning decisions.
2. Domain knowledge and subject matter expertise can help a lot in making various data cleaning decisions, and in this case, in particular, renaming decisions in the “name_cluster_coffee” column (i.e., cleaning the “name” column), as it requires knowing how to rename certain dishes the same or different.
3. Based on what we learned from working on this project, we could have potentially gone for a more structured, automated approach using Python scripts (or just in a Jupyter notebook) to clean the “name” column. Intuitively, an example would be to use a mix of Excel, OpenRefine, and Python to inspect and profile the data. Understand what “types of categories” of “name” values are there, and make decisions on how to rename them (before actually doing so). Then, write Python scripts using a mix of clustering methods, regular expressions, and potentially other methods to clean the “name” column based on relevant keywords, related filter terms, and specific rules for this dataset that we could have devised for filtering, clustering, and renaming (e.g., first figuring out the previously stated heuristic rules based on all records relevant to U1, then turning them into the rules in code). We could then use OpenRefine to make sure we have renamed most of the relevant records and manually edit those we may have missed in Python, including using OpenRefine’s various clustering methods and functions to check this. This would help ensure that cleaning efforts are relatively scalable and reproducible as well.
4. Data cleaning is hard work. Wherever possible, we should try to automate steps so that it is scalable and reproducible, especially as datasets get larger and when databases are live and constantly updated. However it would appear that there is just a portion of the work in this field that may not necessarily make sense to fully automate.
5. As it appears, the data cleaning process itself can be messy (as structured and automated as we want it to be), as we do not know what exactly we are dealing with until we get into cleaning the dataset. As such, the importance of workflow and provenance naturally presents itself, as when we have to perform downstream pre-processing and

analysis (e.g., U1 in this project) on the cleaned dataset, a lot of times we really want to know how certain values have gotten to the state that they are in, so that we can trust the data and therefore our analysis.

Possible Next Steps

As described in the potential analysis examples for U1 above, we would likely proceed with using our cleaned dataset D′ directly, going through the pre-processing efforts followed by similar analyses mentioned. We would therefore be able to find out the answers to our 3 questions in U1, and perhaps develop more detailed and deeper analyses from there!