CS513: Theory & Practice of Data Cleaning Final Project

Brad Ballard, Dhanendra Singh, and Jacob Rettig
University of Illinois at Urbana-Champaign Fall 2018

Abstract: This report describes a data cleaning workflow using the New York Public Library Rare Books Division historical menus dataset has an example to demonstrate various cleaning techniques. This project will use the following software and tools to clean and organize the dataset: OpenRefine, SQLite, and YesWorkFlow.

Team Member Contact Info

Brad Ballard - bjb3@illinois.edu
Dhanendra Singh - disingh2@illinois.edu
Jacob Rettig - jrettig2@illinois.edu

1. Dataset Overview and Initial Assessment

For the project, we use tools introduced in CS513 to clean and prepare a sample dirty dataset. Along the way we will document the steps from dirty to clean. The sections that follow describe this process in detail: (2.) Data cleaning with OpenRefine, (4.) Develop Rational Database Schema, and (5.) Create a Workflow Model.

An initial assessment of the New York Public Library dataset (referred to now forward as 'NY menus' dataset) is over 45,000 historical menus. The majority of these were organized by Frank E. Buttolph (Ref1) around 1900-1921. The dates on the menus range from the 1850s to 2010s. The data contains information on restaurant menu, but also other organizations like railroad or shipping companies. The dataset was digitized in 2011 via the "What's on the Menu?" project (Ref2). So far 17,500 the libraries' historical menus have been digitized. We are using the June 17, 2017 version for this project.

The data is in four files: Menu, MenuPage, MenuItem, and Dish. The 'Menu.csv' file has a unique id number, location information, venue, currency used, and other description-based information. The 'MenuPage.csv' file contains the id, plus an additional unique mean_id, image_id, height, width, and another with other image related information. The 'MenuItem.csv' file contains the id, plus an additional_page_id, dish_id, and other price related information for each dish. The 'Dish.csv' file contains the id, name of the dish, description, first/last appearance, and various price information.

A detailed description of each file's columns follows:

Menu.csv
id: unique id for this menu
name: name on menu, name of the restaurant, or blank
sponsor: sponsor, often the name of the restaurant

event: name of the meal or the event the menu was created for
venue: location where the food is served
place: often includes city, state, country, address, or name of venue
physical description: paper stock, dimensions, colors, design, etc. of menu
occasion: special occasion, holiday, daily, or blank
notes: additional details about the menu
call number: number within the NYPL collection
keywords: keywords on menu
language: language the menu is printed in
date: date the menu was collected, formatted as a string as YYYY-MM-DD
location: where the menu was used
location type: type of the location value
currency: money type charged for items on this menu
currency symbol: symbol for the currency
status: the digitization status of this menu – complete or under review
page count: number of pages on the menu
dish count: number of dishes on the menu

MenuPage.csv
id: unique designator for this menu item
menu id: specific id for menu
page number: page number in the menu
image id: a unique id for the scanned image of this menu, accessible on the NYPL site
full height: height of menu
full width: width of menu
uuid: another unique id for this page/image

MenuItem.csv
id: unique designator for this menu item
menu page id: id of the menu page that this item appears
price: the cost of the smallest portion of this item
high price: cost of the largest portion of this item
dish id: designator id of the dish that this menu item refers
created at: date/time that this database entry was created
updated at: the most recent date the database entry was updated
xpos: x-axis position of the item on the scanned image
ypos: y-axis position of the item on the scanned image

Dish.csv
id: a unique designator for this dish
name: the name of this dish
description: a description of this dish, always blank
menus appeared: number of menus this dish appears on
times appeared: number of times this dish appears (including additional sections)
first appeared: year this dish first appeared (also can be: 0, 1, or NA)
last appeared: year this dish last appeared (also can be: 0, 1, 2928, or NA)

lowest price: lowest price that this item was sold for
highest price: highest price that this item was sold for

Some hypothetical use cases for a dataset like this could be – How has the composition of restaurants changed over time? The density, clusters, or price changes? Can a model be made to predict food prices in certain areas? How have consumer food preferences changed over time? Can a model be made to estimate food prices based on ingredients/description?

The problem with answering all these questions is - the dataset is quite messy and need to be organized. For example, the date columns seem full of repeat or missing dates. However, some files are cleaner than others; we are not choosing to clean MenuPage.csv, as it is clean enough.

How clean does the dataset need to be? You can sort the columns and get some sense of the composition of the data, but there will need to be some prep work to answer practical questions. Having the data broken into multiple files makes it hard to compare across sheets. However, this format should be a straight forward to import into the SQL database.

Progress throughout his project can be tracked at our github repository (Ref3) and this associated webpage (Ref4). This platform was mostly used to aid in our team collaboration, but also provides a way to share our data cleaning techniques with a wider audience outside of this class.

2. Data cleaning with OpenRefine

OpenRefine, formally called Google Refine, is an open source desktop application the has many helpful features for data cleaning. It behaves like a database with rows and cells under columns which are similar to relational database tables. An OpenRefine project itself consists of one table. I plan to clean each of the four data file separately, some more than others as needed.

The major OpenRefine feature that will be helpful in cleaning our dataset is the clustering feature. The option allows the user to cluster similar text and replace it with a more standardized description. The common problem this solves the many variants of spelling but reference the same object.

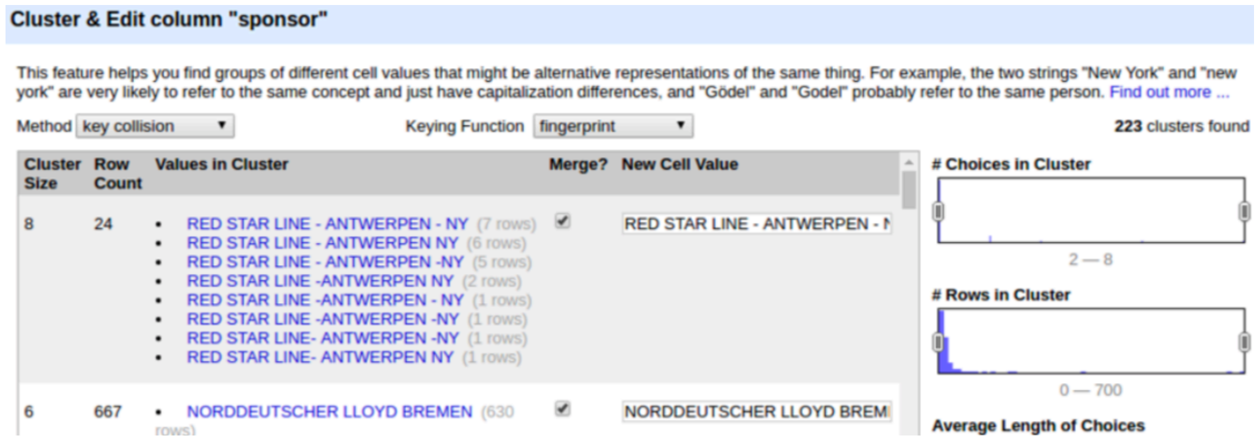
The following subsections will describe the step-by-step process from input file to output file. We will use UTF-8 encoding.

2.1 Menu.csv Cleaning

### Input File: Menu.csv #####################

For column: sponsor
(1) Trim leading and trailing white spaces
(2) Collapse consecutive white spaces
(3) Convert all column values to upper case
(4) Remove special characters using GREL (%, #, !, /, (,), [,], ?)
(5) Replace ";" with a space instead and then trim leading/trailing white spaces

(6) Make a facet and perform the cluster operation using the key-collision method and fingerprint function. Next merge the selected clusters.
(7) Repeat the previous step with n-gram, fingerprint, meta-phone3, and cologne-phonetic methods.
(8) Create a new facet. Next, use the cluster operation nearest neighbor method and levestein distance function. Then, merge the selected clusters.
(9) Create a new facet. Next, use the cluster operation nearest neighbor method and PPM distance function. Then, merge the selected clusters.

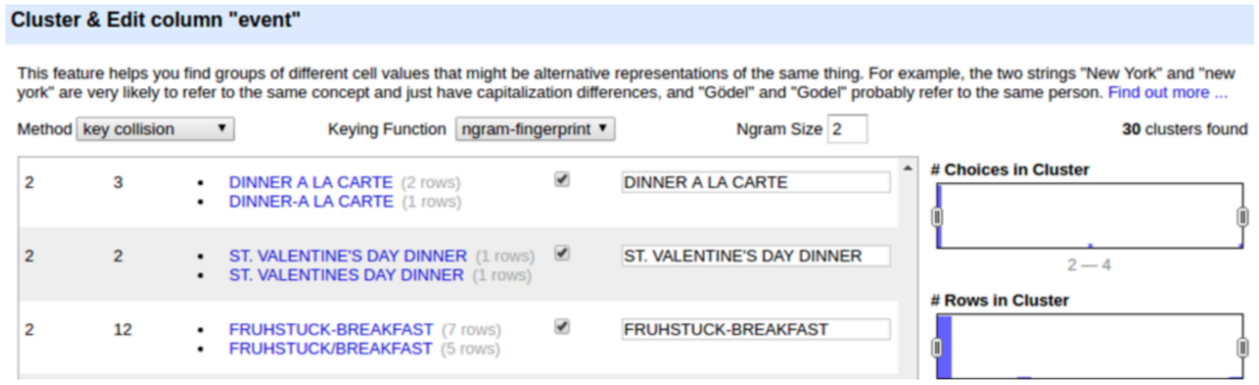


Cluster & Edit column "sponsor"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method: key collision — Keying Function: fingerprint — 223 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
8	24	• RED STAR LINE - ANTWERPEN - NY (7 rows) • RED STAR LINE - ANTWERPEN NY (6 rows) • RED STAR LINE - ANTWERPEN -NY (5 rows) • RED STAR LINE -ANTWERPEN NY (2 rows) • RED STAR LINE -ANTWERPEN - NY (1 rows) • RED STAR LINE -ANTWERPEN -NY (1 rows) • RED STAR LINE- ANTWERPEN -NY (1 rows) • RED STAR LINE- ANTWERPEN NY (1 rows)	✓	RED STAR LINE - ANTWERPEN - N
6	667	• NORDDEUTSCHER LLOYD BREMEN (630 rows)	✓	NORDDEUTSCHER LLOYD BREM



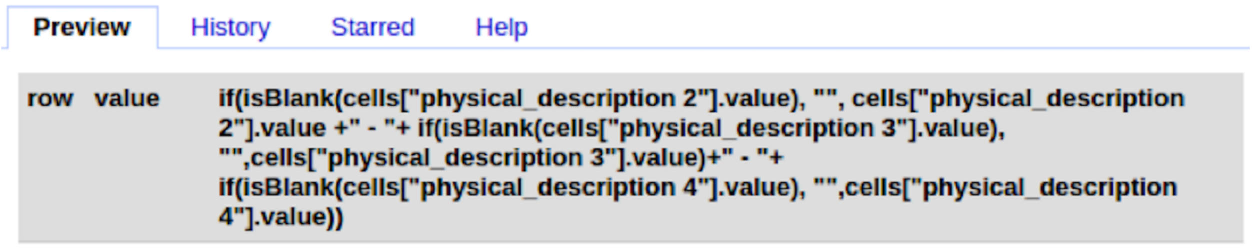
Cluster & Edit column "event"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method: key collision — Keying Function: ngram-fingerprint — Ngram Size: 2 — 30 clusters found

2	3	• DINNER A LA CARTE (2 rows) • DINNER-A LA CARTE (1 rows)	✓	DINNER A LA CARTE
2	2	• ST. VALENTINE'S DAY DINNER (1 rows) • ST. VALENTINES DAY DINNER (1 rows)	✓	ST. VALENTINE'S DAY DINNER
2	12	• FRUHSTUCK-BREAKFAST (7 rows) • FRUHSTUCK/BREAKFAST (5 rows)	✓	FRUHSTUCK-BREAKFAST

<u>For column: physical_description</u>
(1) Slit the columns using ‘;’
(2) Then rename the first column: physical_description_type
(3) Use GREL to join ‘physical_description 2’, ‘physical_description 3’, and ‘physical_description 4’ into one column named: physical_description_additional.
(4) Use ‘ - ‘ to separate the values from the different columns (Remember the space before and after the dash). Note: if the cell in the column is empty, leave that value as a blank space. For example, if a column is blank, make the new column in ‘physical_description_additional’ blank also.



```
row   value   if(isBlank(cells["physical_description 2"].value), "", cells["physical_description
              2"].value +" - "+ if(isBlank(cells["physical_description 3"].value),
              "",cells["physical_description 3"].value)+" - "+
              if(isBlank(cells["physical_description 4"].value), "",cells["physical_description
              4"].value))
```

For column: date
(1) Convert date format to YYY-MM-DD
(2) Remove date outliers, where the year is either less than 1851 or greater than 2012. Make these flagged cells blank.

For column: call_number
(1) Trim leading/trailing white spaces
(2) Collapse consecutive white spaces

Unchanged columns: id, name, keywords, language, status, page_count, dish_count

> Output File: CMenu.csv

2.2 MenuPage.csv Cleaning

### Input File: MenuPage.csv ##################

File not cleaned in OpenRefine
Unchanged columns: id, menu_id, page_number, image_id, full_height, full_width, uuid

> Output File: CMenuPage.csv

2.2 MenuItem.csv Cleaning

### Input File: MenuItem.csv ##################

For column: created_at
(1) Convert date format to YYYY-MM-DD
(2) Remove date outliers, where the year is either less than 1851 or greater than 2012. Make these flagged cells blank.

For column: updated_at
(1) Convert date format to YYYY-MM-DD
(2) Remove date outliers, where the year is either less than 1851 or greater than 2012. Make these flagged cells blank.

> Output File: CMenuItem.csv

2.2 Dish.csv Cleaning

### Input File: Dish.csv ##################

For column: name
(1) Use key-collision to cluster values. Note: cannot do nearest_neighbours cluster method because the time required to do this too computationally expensive

> ***> Output File: CDish.csv***

3. (Optional) Clean data further with an alternative tool

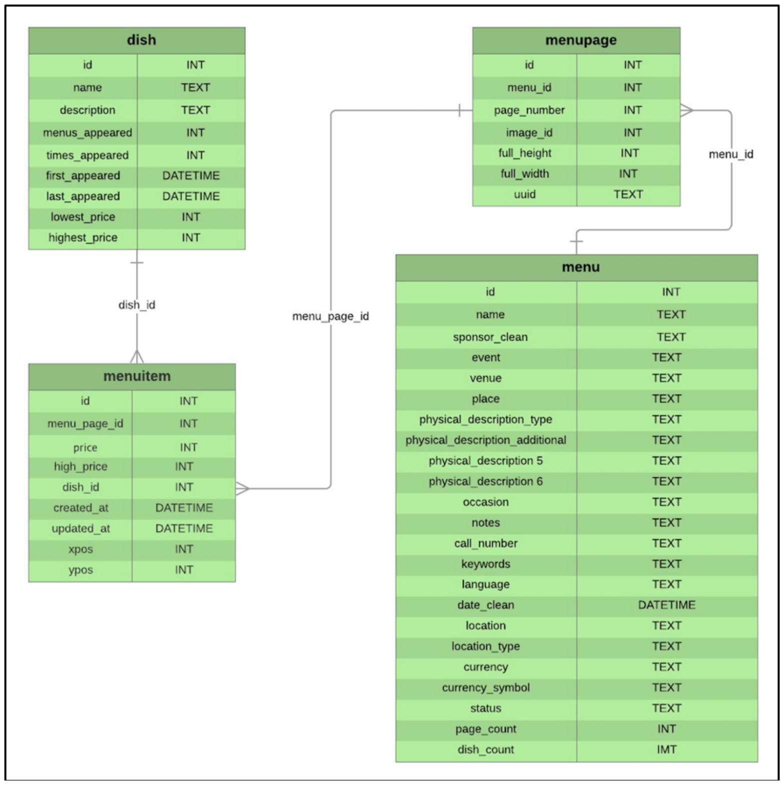
We choose not to do this optional part.

4. Develop Rational Database Schema

The next step in our data cleaning workflow is to import the cleaned data into a database. We are choosing to use SQLite as it is open source, simple, and straightforward to work with. When first designing a database, we must think about integrity constraints and the relational schema.

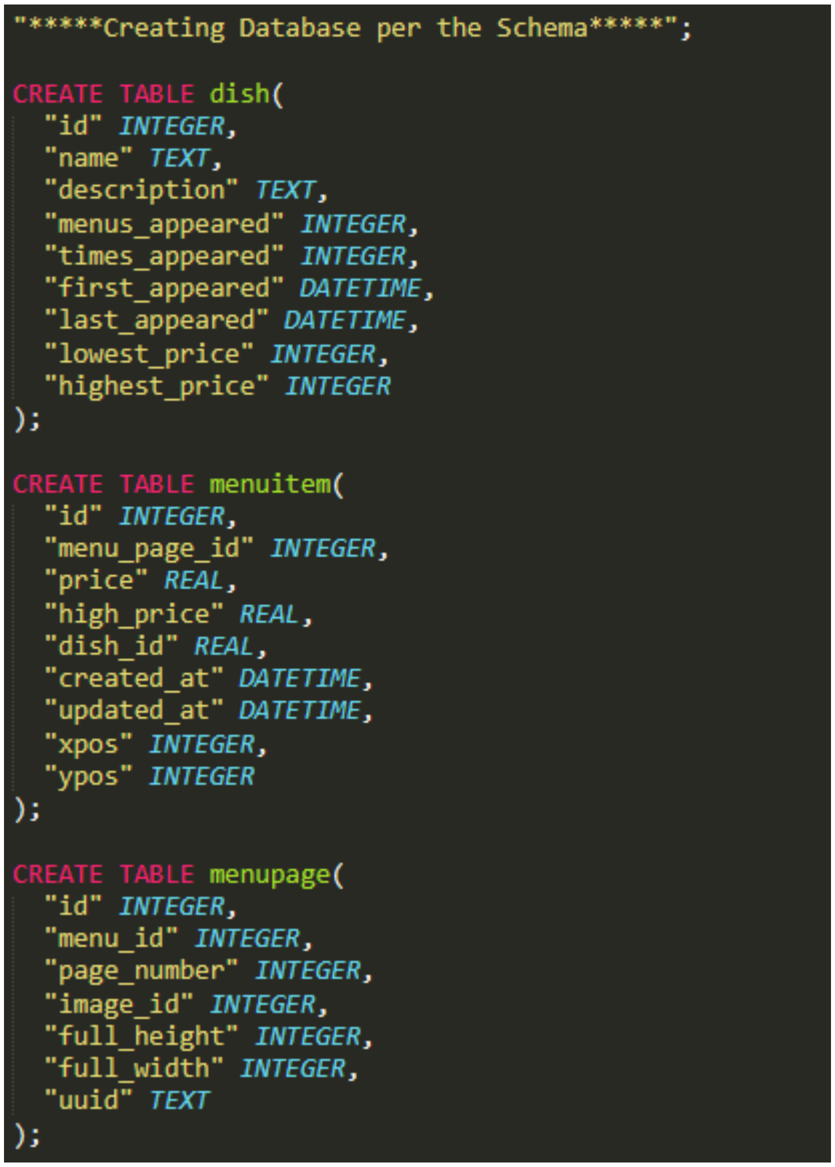
4.1 Schema

Below, is the schema in which there are four tables (one for each input file): dish, menuitem, menupage, and menu.

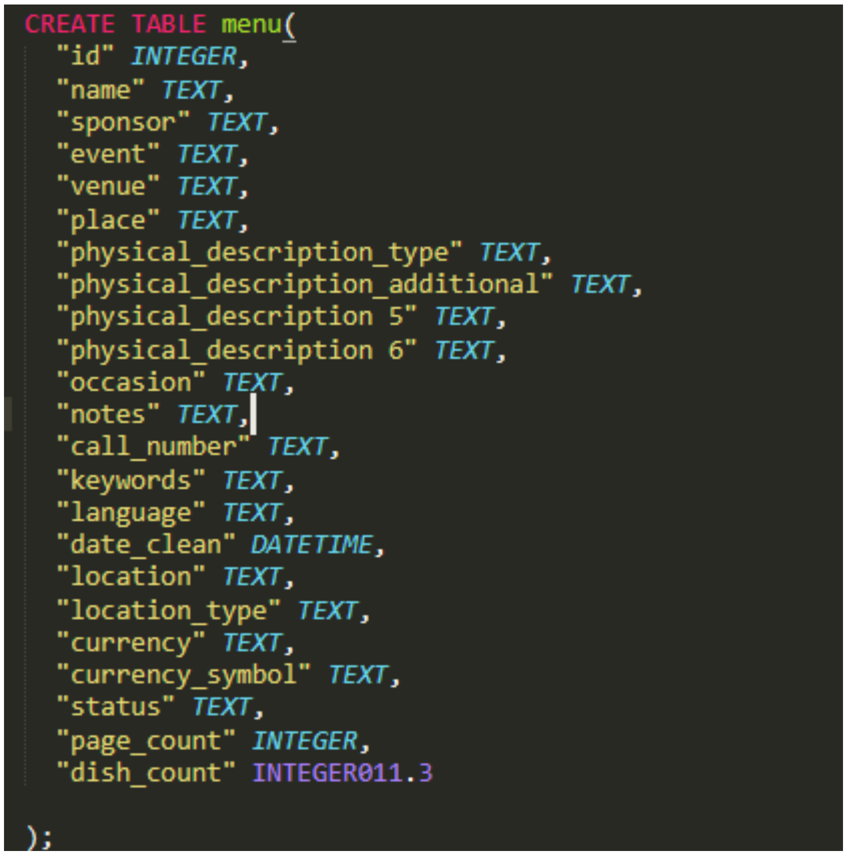


4.2 Table creation and Data load

SQL to create Database as per the Schema defined above.

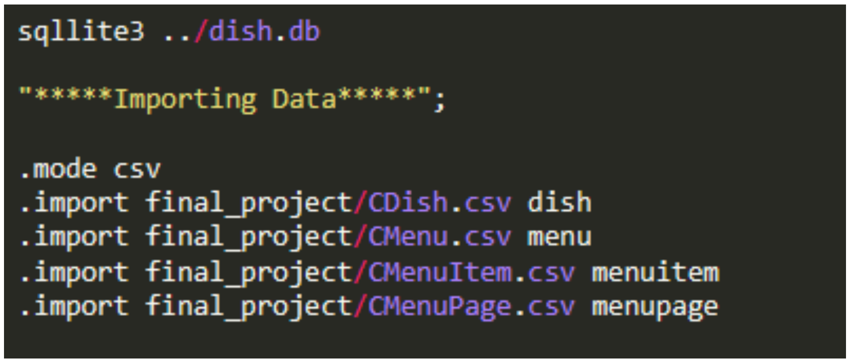


```
"*****Creating Database per the Schema*****";

CREATE TABLE dish(
  "id" INTEGER,
  "name" TEXT,
  "description" TEXT,
  "menus_appeared" INTEGER,
  "times_appeared" INTEGER,
  "first_appeared" DATETIME,
  "last_appeared" DATETIME,
  "lowest_price" INTEGER,
  "highest_price" INTEGER
);

CREATE TABLE menuitem(
  "id" INTEGER,
  "menu_page_id" INTEGER,
  "price" REAL,
  "high_price" REAL,
  "dish_id" REAL,
  "created_at" DATETIME,
  "updated_at" DATETIME,
  "xpos" INTEGER,
  "ypos" INTEGER
);

CREATE TABLE menupage(
  "id" INTEGER,
  "menu_id" INTEGER,
  "page_number" INTEGER,
  "image_id" INTEGER,
  "full_height" INTEGER,
  "full_width" INTEGER,
  "uuid" TEXT
);
```



```
CREATE TABLE menu(
  "id" INTEGER,
  "name" TEXT,
  "sponsor" TEXT,
  "event" TEXT,
  "venue" TEXT,
  "place" TEXT,
  "physical_description_type" TEXT,
  "physical_description_additional" TEXT,
  "physical_description 5" TEXT,
  "physical_description 6" TEXT,
  "occasion" TEXT,
  "notes" TEXT,
  "call_number" TEXT,
  "keywords" TEXT,
  "language" TEXT,
  "date_clean" DATETIME,
  "location" TEXT,
  "location_type" TEXT,
  "currency" TEXT,
  "currency_symbol" TEXT,
  "status" TEXT,
  "page_count" INTEGER,
  "dish_count" INTEGER011.3

);
```

Importing .CSV data into the tables



```
sqllite3 ../dish.db

"*****Importing Data*****";

.mode csv
.import final_project/CDish.csv dish
.import final_project/CMenu.csv menu
.import final_project/CMenuItem.csv menuitem
.import final_project/CMenuPage.csv menupage
```

4.3 Integrity Constraints

The following are the integrity constraints:

Menu (menu) Table

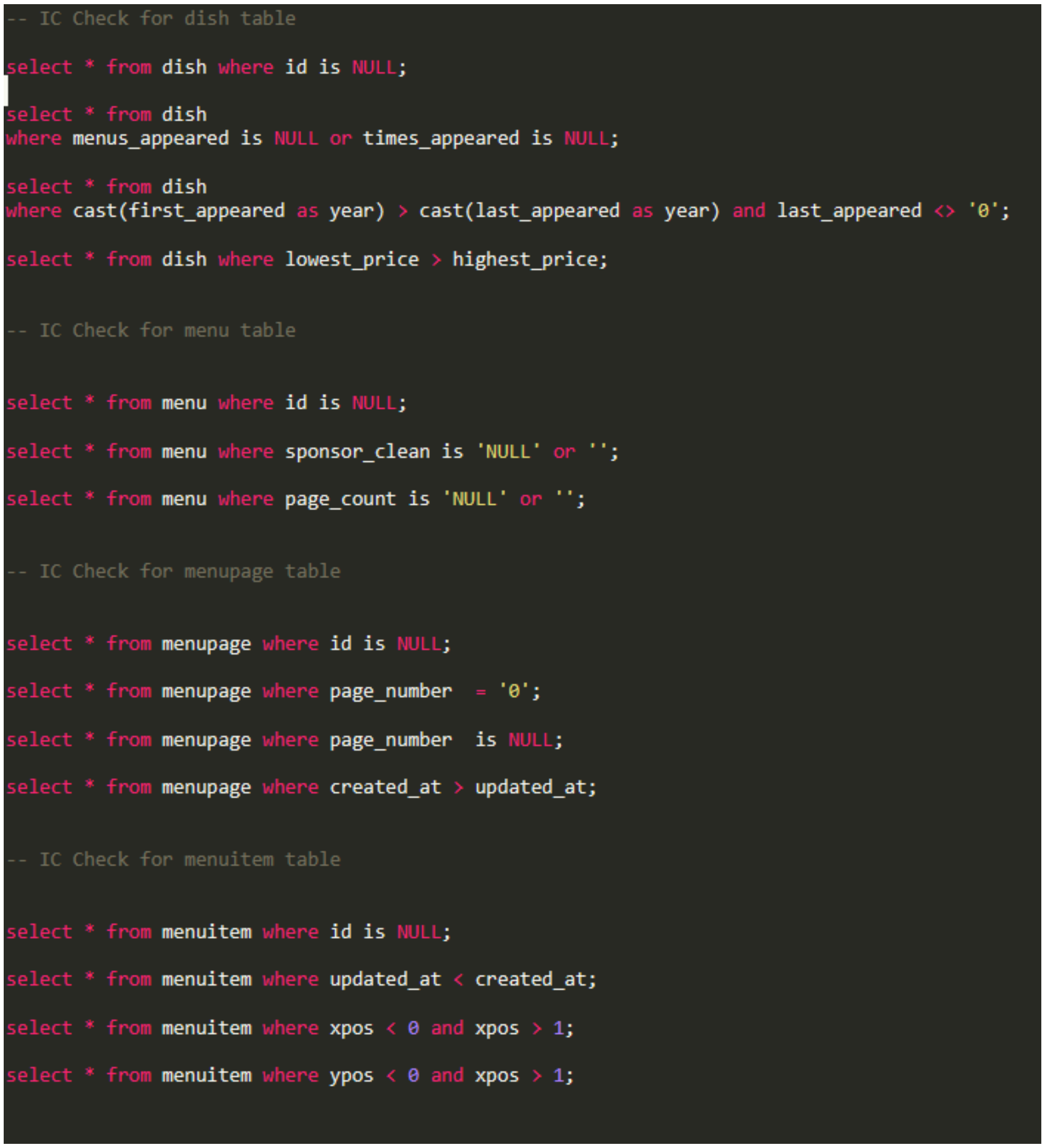
- Id cannot be Null
- Identify rows where sponsor is NULL. Sponsor cannot be NULL so deleted these rows as they are violating the Integrity Constraint.

- Page count should not be NULL
- Deleted physical_description 7 as it had all the NULL values

Dish (dish) Table
- Id should not be Null
- menus_appeared or times_appeared cannot be NULL
- Lowest price of the dish should be less than the highest price
- Identify number of rows of data where last_appeared is “0”. These rows will not be deleted as these dishes may be part of the other data.

Menu Page (menupage) Table
- Id cannot be Null
- Page number should not be NULL or “0”. No such rows were identified
- created_at date should be always greater than updated_at. Few rows were identified with this constraint
-

Menu Item (menuitem) Table
- Id cannot be Null
- created_at date should be always greater than updated_at. No such rows were violating this IC check.
- Xpos and ypos values should always be between 0 and 1.



```
-- IC Check for dish table

select * from dish where id is NULL;

select * from dish
where menus_appeared is NULL or times_appeared is NULL;

select * from dish
where cast(first_appeared as year) > cast(last_appeared as year) and last_appeared <> '0';

select * from dish where lowest_price > highest_price;


-- IC Check for menu table


select * from menu where id is NULL;

select * from menu where sponsor_clean is 'NULL' or '';

select * from menu where page_count is 'NULL' or '';


-- IC Check for menupage table


select * from menupage where id is NULL;

select * from menupage where page_number  = '0';

select * from menupage where page_number  is NULL;

select * from menupage where created_at > updated_at;


-- IC Check for menuitem table


select * from menuitem where id is NULL;

select * from menuitem where updated_at < created_at;

select * from menuitem where xpos < 0 and xpos > 1;

select * from menuitem where ypos < 0 and xpos > 1;
```

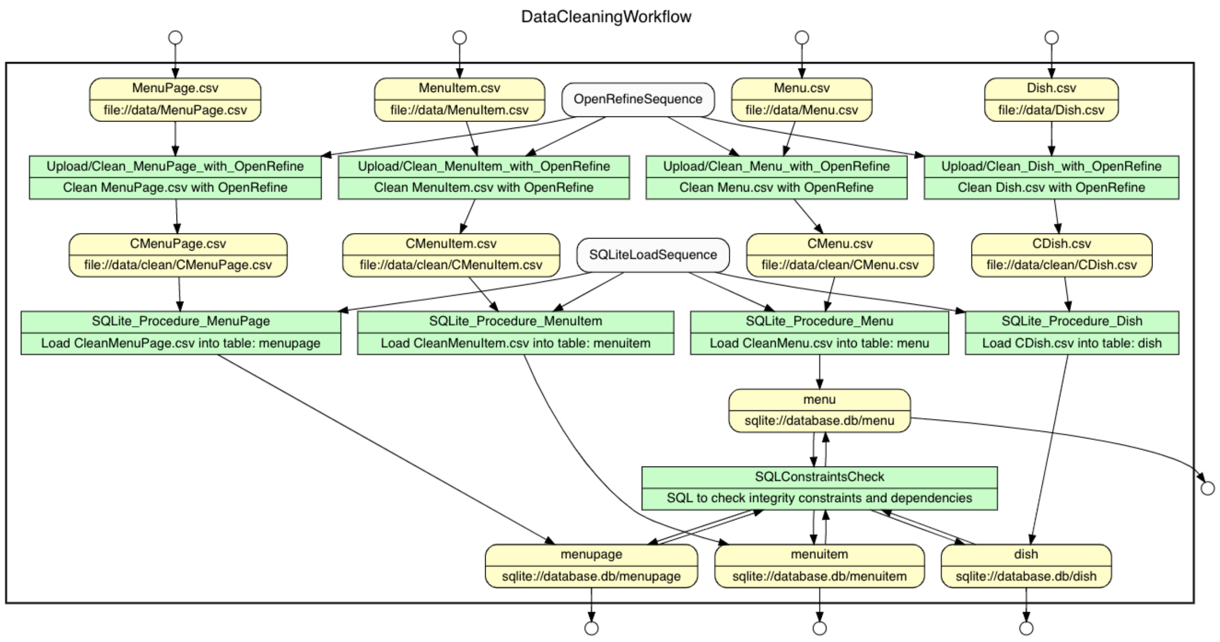
5. Create a Workflow Model

YesWorkFlow is a solution to annotating your data workflow. It's easy to script and doesn't require you to re-write any of your existing code. Simply add a special (YW) comments to your existing script. Later, these comments are used to declare how the data was transformed step-by-step.

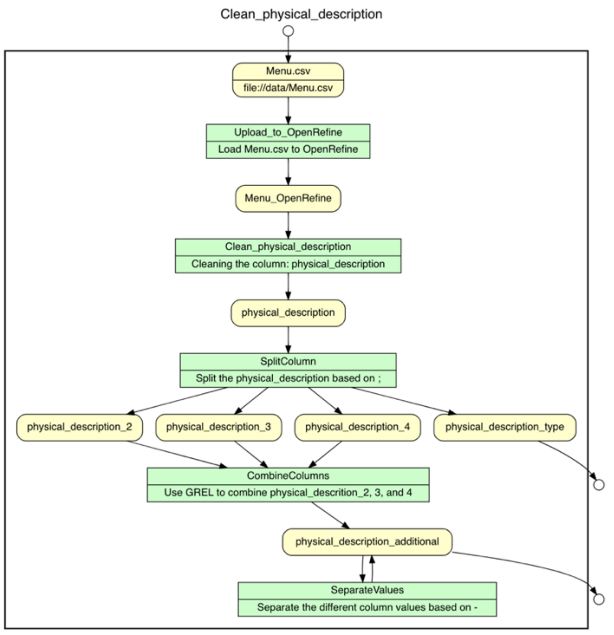
We used the online editor [Ref5] to generate various workflow diagrams shown below. The first diagram is the complete YW graph including the data and operations steps.

The second diagram down is a more detailed example of how the data was cleaned using OpenRefine. In order to provide a detailed example of how the data was cleaned, we selected a single column from the Menu.csv file, physical_description, and visualized how we prepared the data.

Below, is the overall workflow for cleaning the dataset as a whole. The code can be seen in Appendix 1.


DataCleaningWorkflow
MenuPage.csv
file://data/MenuPage.csv
MenuItem.csv
file://data/MenuItem.csv
OpenRefineSequence
Menu.csv
file://data/Menu.csv
Dish.csv
file://data/Dish.csv
Upload/Clean_MenuPage_with_OpenRefine
Clean MenuPage.csv with OpenRefine
Upload/Clean_MenuItem_with_OpenRefine
Clean MenuItem.csv with OpenRefine
Upload/Clean_Menu_with_OpenRefine
Clean Menu.csv with OpenRefine
Upload/Clean_Dish_with_OpenRefine
Clean Dish.csv with OpenRefine
CMenuPage.csv
file://data/clean/CMenuPage.csv
CMenuItem.csv
file://data/clean/CMenuItem.csv
SQLiteLoadSequence
CMenu.csv
file://data/clean/CMenu.csv
CDish.csv
file://data/clean/CDish.csv
SQLite_Procedure_MenuPage
Load CleanMenuPage.csv into table: menupage
SQLite_Procedure_MenuItem
Load CleanMenuItem.csv into table: menuitem
SQLite_Procedure_Menu
Load CleanMenu.csv into table: menu
SQLite_Procedure_Dish
Load CDish.csv into table: dish
menu
sqlite://database.db/menu
SQLConstraintsCheck
SQL to check integrity constraints and dependencies
menupage
sqlite://database.db/menupage
menuitem
sqlite://database.db/menuitem
dish
sqlite://database.db/dish

Below, is the workflow for cleaning the 'phsyical_description' column for the Menu.csv dataset. The code can be seen in Appendix 2.


Clean_physical_description
Menu.csv
file://data/Menu.csv
Upload_to_OpenRefine
Load Menu.csv to OpenRefine
Menu_OpenRefine
Clean_physical_description
Cleaning the column: physical_description
physical_description
SplitColumn
Split the physical_description based on ;
physical_description_2
physical_description_3
physical_description_4
physical_description_type
CombineColumns
Use GREL to combine physical_description_2, 3, and 4
physical_description_additional
SeparateValues
Separate the different column values based on -

6. (Optional) Develop provenance queries

We choose not to do this optional part.

7. Conclusion

Any scientist, engineer, or researcher spends a substantial amount of time cleaning data for their research. Further, at conferences or when publishing research papers, often detailing steps documenting where you got the data from and how it was cleaned are required. These steps are required in order to potentially replicate your research/models later. Tools described in this paper aim to help document these cleaning steps in a simple and easy way.

In this project we used OpenRefine and SQLite for the cleaning steps. The challenges we faced generally is handling all the duplicate names or invalid/NA entries. Without much information about how to fill in the gaps, we often had to delete these entries. One tries to avoid deleting data if possible, but in our case, we felt it an appropriate solution.

Due to the size of the datasets in this project, it is hard to verify integrity of each clustering step. In some cases, we were not able to see all the clusters that were suggested by OpenRefine and had to rely on its decision on those.

Another problem is the slow run time and manual man hours - imputing GREL commands, reviewing cluster groups, documenting steps – data cleaning still requires a lot of time even given tools like these. OpenRefine, like excel and some other tools, has limitations in the size of datasets it can work with. This is an example of where other open source programming languages help, such as Python or R. There is no ‘silver bullet’ in data cleaning. It remains a piecewise blend of various tool as the data moves through a line of steps.

8. Appendix

Appendix 1: DataCleaningWorkflow

```
# @begin DataCleaningWorkflow  @desc The overall workflow for cleaning the NYPL Menus Dataset
# @in Menu.csv  @uri file://data/Menu.csv
# @in MenuPage.csv  @uri file://data/MenuPage.csv
# @in MenuItem.csv  @uri file://data/MenuItem.csv
# @in Dish.csv  @uri file://data/Dish.csv

#     @begin Upload/Clean_Menu_with_OpenRefine  @desc Clean Menu.csv with OpenRefine
#     @in Menu.csv  @uri file://data/Menu.csv
#     @param OpenRefineSequence
#     @out CMenu.csv  @uri file://data/clean/CMenu.csv
#     @end Upload/Clean_Menu_with_OpenRefine

#     @begin Upload/Clean_MenuPage_with_OpenRefine  @desc Clean MenuPage.csv with OpenRefine
#     @in MenuPage.csv  @uri file://data/MenuPage.csv
#     @param OpenRefineSequence
#     @out CMenuPage.csv  @uri file://data/clean/CMenuPage.csv
#     @end Upload/Clean_MenuPage_with_OpenRefine

#     @begin Upload/Clean_MenuItem_with_OpenRefine @desc Clean MenuItem.csv with OpenRefine
#     @in MenuItem.csv  @uri file://data/MenuItem.cv
#     @param OpenRefineSequence
#     @out CMenuItem.csv  @uri file://data/clean/CMenuItem.csv
#     @end Upload/Clean_MenuItem_with_OpenRefine

#     @begin Upload/Clean_Dish_with_OpenRefine  @desc Clean Dish.csv with OpenRefine
#     @in Dish.csv  @uri file://data/Dish.csv
#     @param OpenRefineSequence
#     @out CDish.csv  @uri file://data/clean/CDish.csv
#     @end Upload/Clean_Dish_with_OpenRefine

#     @begin SQLite_Procedure_Menu  @desc Load CleanMenu.csv into table: menu
#     @in CMenu.csv  @uri file://data/clean/CMenu.csv
#     @param SQLiteLoadSequence
#     @out menu  @uri sqlite://database.db/menu
#     @end SQLite_Procedure_Menu

#     @begin SQLite_Procedure_MenuPage @desc Load CleanMenuPage.csv into table: menupage
#     @in CMenuPage.csv  @uri file://data/clean/CMenuPage.csv
#     @param SQLiteLoadSequence
#     @out menupage  @uri sqlite://database.db/menupage
#     @end SQLite_Procedure_MenuPage

#     @begin SQLite_Procedure_MenuItem @desc Load CleanMenuItem.csv into table: menuitem
#     @in CMenuItem.csv  @uri file://data/clean/CMenuItem.csv
#     @param SQLiteLoadSequence
#     @out menuitem  @uri sqlite://database.db/menuitem
#     @end SQLite_Procedure_MenuItem

#     @begin SQLite_Procedure_Dish  @desc Load CDish.csv into table: dish
#     @in CDish.csv  @uri file://data/clean/CDish.csv
#     @param SQLiteLoadSequence
#     @out dish  @uri sqlite://database.db/dish
#     @end SQLite_Procedure_Dish

#     @begin SQLConstraintsCheck  @desc SQL to check integrity constraints and dependencies
#     @in menu  @uri sqlite://database.db/menu
#     @in menupage  @uri sqlite://database.db/menupage
#     @in menuitem  @uri sqlite://database.db/menuitem
#     @in dish  @uri sqlite://database.db/dish

#     @out menu  @uri sqlite://database.db/menu
#     @out menupage @uri sqlite://database.db/menupage
#     @out menuitem  @uri sqlite://database.db/menuitem
#     @out dish  @uri sqlite://database.db/dish
#     @end SQLConstraintsCheck

# @out menu  @uri sqlite://database.db/menu
# @out menupage  @uri sqlite://database.db/menupage
# @out menuitem  @uri sqlite://database.db/menuitem
# @out dish @uri sqlite://database.db/dish
# @end DataCleaningWorkflow
```

Appendix 2: Clean_physical_description

```
# @begin Clean_physical_description @desc Workflow for cleaning physical_description in Menu.csv
# @in Menu.csv  @uri file://data/Menu.csv

#   @begin Upload_to_OpenRefine @desc Load Menu.csv to OpenRefine
#   @in Menu.csv
#   @out Menu_OpenRefine
#   @end Upload_to_OpenRefine

#   @begin Clean_physical_description @desc Cleaning the column: physical_description
#   @in Menu_OpenRefine
#   @out physical_description
#   @end Clean_physical_description

#   @begin SplitColumn @desc Split the physical_description based on ;
#   @in physical_description
#   @out physical_description_type
#   @out physical_description_2
#   @out physical_description_3
#   @out physical_description_4
#   @end SplitColumn

#   @begin CombineColumns @desc Use GREL to combine physical_descrition_2, 3, and 4
#   @in physical_description_2
#   @in physical_description_3
#   @in physical_description_4
#   @out physical_description_additional
#   @end CombineColumns

#   @begin SeparateValues @desc Separate the different column values based on -
#   @in physical_description_additional
#   @out physical_description_additional
#   @end SeparateValues

# @out physical_description_type
# @out physical_description_additional
# @end Clean_physical_description
```

References

[Ref1] https://en.wikipedia.org/wiki/Frank_E._Buttolph
[Ref2] http://menus.nypl.org
[Ref3] https://github.com/bradjballard/dc-workflow
[Ref4] https://bradjballard.github.io/dc-workflow
[Ref5] https://try.yesworkflow.org