CS513: Data Cleaning

Project Report

ABSTRACT

Project report on data cleaning performed on New York Public Library's crowd-sourced historical menus dataset to make the dataset usable for data analytics purpose

TEAM

Rohit Bansal (rbansal3@illinois.edu)
Parul Mainwal (mainwal2@illinois.edu)

Table of Contents

Introduction 3

Initial Assessment 3

- Structure Summary: 5
- Quality Issues 5
- Use Cases of the Dataset 6
- Data Cleaning Goals 6

Data Cleaning Process 7

- Step1: Python Script 7
- Step2: OpenRefine 8
- File: Menu.csv 8
- File: Dish.csv 16
- File: MenuItem.csv 18
- File: MenuPage.csv 18
- Step 3: Integrity Checks using SQLite database constraints 18
  - Database Model 18
  - Tables Creation and Data Import 19
  - Integrity Constraints check 20

Process Workflow (Provenance via YesWorkflow) 22

- Summary Workflow 23
  - Key Inputs/Outputs 23
  - Dependencies 23
  - Diagram 23
- Data Cleanup Detailed Workflow 23
- SQLite Detailed Workflow 25

Use Case Implementation 25

- Use Case 1: Top 10 Breakfast Dishes 25
- Use Case 2: Dishes Served in Hotel Eastman 26
- Use Case 3: Five Oldest Dishes 27

Challenges 28

- Ambiguity 28
- OpenRefine Efficiency 28

Results 29
Appendix 31
Dataset 31
Tools/Software Used 31
Project Deliverables 31

Introduction

This document is a project report for the end-to-end data cleaning process performed on New York Public Library's crowd-sourced historical menus dataset. It captures purpose of the data cleaning, details of the steps performed for data cleaning and our findings during the process. This document also highlights the provenance of the steps performed using a tool called "Yes Workflow". The comparison of the raw data and clean data is published via tableau dashboard here. The cleaned data can be accessed by downloading the dashboard and data source from the same link.

The New York Public Library's restaurant menu collection holds data about menus and dishes from 1840 to the present. This is a crowdsourced dataset collected through spreadsheets and APIs. Since the data is crowdsourced and collected via various means, the data quality is very poor. The objective of this project is to apply various data cleaning techniques and analyze their effectiveness on such a large data set. After performing data cleaning, it is expected that the data will be usable for various data analytics purpose effectively.

Initial Assessment

The New York Public Library's restaurant menu collection is one of the largest in the world. It holds about 45,000 menus dating from the 1840s to the present. It has almost half a million dish items listed in the database along with almost 20 thousand menus of various eateries. We downloaded the data in comma separated values format. As a result, we got 4 files:

- Dish.csv

  This file has name of the dish along with its lowest price and highest price. It also has some popularity related data where it informs when the dish first appeared in a menu and when it last appeared and in how many menus it appeared. Here is the list of columns:

  - id
  - name
  - description
  - menus_appeared
  - times_appeared
  - first_appeared
  - last_appeared
  - lowest_price
  - highest_price

- Menu.csv

  This file has details about the menus. The details include the name, place, occasion, event, location, venue, sponsor of the menu. It also has some other information such as number of pages in the menu, number of dishes in the menu, language of the menu, currency of payment and some other descriptive fields. Here is the list of columns:

    - Id
    - Name
    - Sponsor
    - Event
    - Venue
    - Place
    - physical_description
    - occasion
    - notes
    - call_number
    - keywords
    - language
    - date
    - location
    - location_type
    - currency
    - currency_symbol
    - status
    - page_count
    - dish_count

- MenuPage.csv

  This file refers to menu file and provides details of menu pages. It provides the image of the menu page, height and width of the page and page number. It also provides the menu id from the menu file so that page could be associated to specific menu. Here is the list of columns:

    - id
    - menu_id
    - page_number
    - image_id
    - full_height
    - full_width
    - uuid

- MenuItem.csv

  This file refers to the menu, dish and menu page files and provides the information related to each menu item. For each menu item, it provides the dish id from dish.csv and menu page id from menupage.csv. It also provides other details of menu items such as price, creation type, and the location coordinate of the place. Here is the list of columns:

    - Id
    - menu_page_id
    - price

- high_price
- dish_id
- created_at
- updated_at
- xpos
- ypos

Structure Summary:

Features	Menu	Dish	MenuPage	MenuItem
Row count	17547	423400	66937	1332726
Attribute count	20	9	7	9
Unique column	id	id	id	id
Refers			Menu-> Id	MenuPage -> Id Dish-> Id
Description	File with record for each menu and its details	File with records of dishes	File with menu page information along with reference to menu	File with menu item details along with reference to dish and menu page

Quality Issues

We analyzed each of the data files and identified below data quality issues:

- Leading/trailing spaces
- Additional spaces
- Inconsistent case
- Blank values for certain fields
- Inconsistent representation of missing information
- Special Characters like #%?\()[]
- Spelling mistakes
- In consistent format for values
- Some fields enclosed in double quotes.
- Inconsistent data -abbreviations/full form present for same field.
- Inconsistent date format for date type field.

- Key values mismatch between associated tables.
- Negative/Incorrect values assigned e.g. 0 for price
- Data integrity issues e.g. Created Date>Updated Date.
- Poor formatting
- Ambiguity

Use Cases of the Dataset

As highlighted above, the data had many quality issues. It had to be cleaned in order to make it usable for any use case. Cleaned data could be used for data analytics purpose. Some of the use cases that can be applied on the cleansed data are:

- Answering specific questions for researchers such as when did a dish first appeared on the menu, how has the price/location of a dish changed over the years.
- What dishes are contained in a Menu along with their price and other details.
- The change in demand of a dish over a period and possible contributing factors.
- Correlation between the demand of a dish with its price/location/other factors.

Because of the multiple issues with the data, the dataset was not fit enough for being used in any application. After cleaning the data, we will implement following use case on the cleaned data for demo:

- Use Case 1: Top 10 Breakfast Dishes
- Use Case 2: Dishes Served in Hotel Eastman
- Use Case 3: Five Oldest Dishes

Data Cleaning Goals

After analysis of the dataset, we decided to clean the data to a point that it can be loaded into the database and some basic data analysis could be performed. The goal of our cleaning process is to have it clean enough to be able to support basic user queries like menu structure, popularity of a dish over time etc. We also decided to make the clean data available via public tableau dashboards.

We **did not intend to remove the ambiguity** in the data. We did not want to remove ambiguity by making assumptions. For example, if we found an item as omelette 1 and omelette 2 then we did not try to merge it to “omelette”. The reason behind that was that these two could be a variant of omelette with different ingredient and sometime different size of servings. Assuming them to represent one dish could hamper the data integrity. So, we left those cases as it is. We only cleaned data when it had following issues:

- Leading/trailing spaces
- Additional spaces
- Inconsistent case
- Blank values for certain fields
- Inconsistent representation of missing information
- Special Characters like #%?\()[]
- Spelling mistakes
- In consistent format for values

- Some fields enclosed in double quotes.
- Inconsistent data -abbreviations/full form present for same field.
- Inconsistent date format for date type field.
- Key values mismatch between associated tables.
- Negative/Incorrect values assigned e.g. 0 for price
- Data integrity issues e.g. Created Date>Updated Date.
- Poor formatting

Data Cleaning Process

Step1: Python Script

We developed a python script in order to get rid of few of the common issues with various fields. These issues included:

- Leading/trailing spaces
- Additional spaces
- Inconsistent case
- Blank values for certain fields
- Inconsistent representation of missing information
- Special Characters like #%?\()[]

The reason for going for python script instead of Open Refine was that these issues were present with multiple columns in different files. With parallel processing architecture using map-reduce, these cleanups could be done very efficiently in one pass via python script. Using the script, we cleaned following items in one pass:

- File: Menu.csv (12 columns)
    - Name
    - Sponsor
    - Event
    - Venue
    - Place
    - Physical_Decription
    - Occasion
    - Notes
    - Date
    - Location
    - Status
    - Currency_Symbol
- File: Dish.csv (1 column)
    - Name

It is noticeable that menu.csv contains 17547 rows while dish.csv contains 423400 rows. Thus, with the script (12*17547) + (1*423400) = (210564 + 423400) = 633964 items were cleaned in one step within

few seconds. This turned out to be most efficient then any other method we considered including open refine transformations.

Step2: OpenRefine

We used OpenRefine to further clean the data. Menu, MenuItem and Dish file was cleaned using OpenRefine. MenuPage did not had any issues which could be removed by clustering of facet. Hence it was not considered for cleaning with OpenRefine. We used clustering and facet feature of open refine extensively. We considered following methods for clustering:

- Method: Key-Collision, Function: Fingerprint
- Method: Key-Collision, Function: n-gram fingerprint (n=2)
- Method: Key-Collision, Function: metaphone3
- Method: Key-Collision, Function: cologne-phonetic
- Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
- Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)

Not all the method returned good candidates and sometimes the method did not return any candidate. We picked and chose only good cluster to perform data cleaning while ignore the other clusters. Here is the list of detailed steps performed on each of the files:

File: Menu.csv

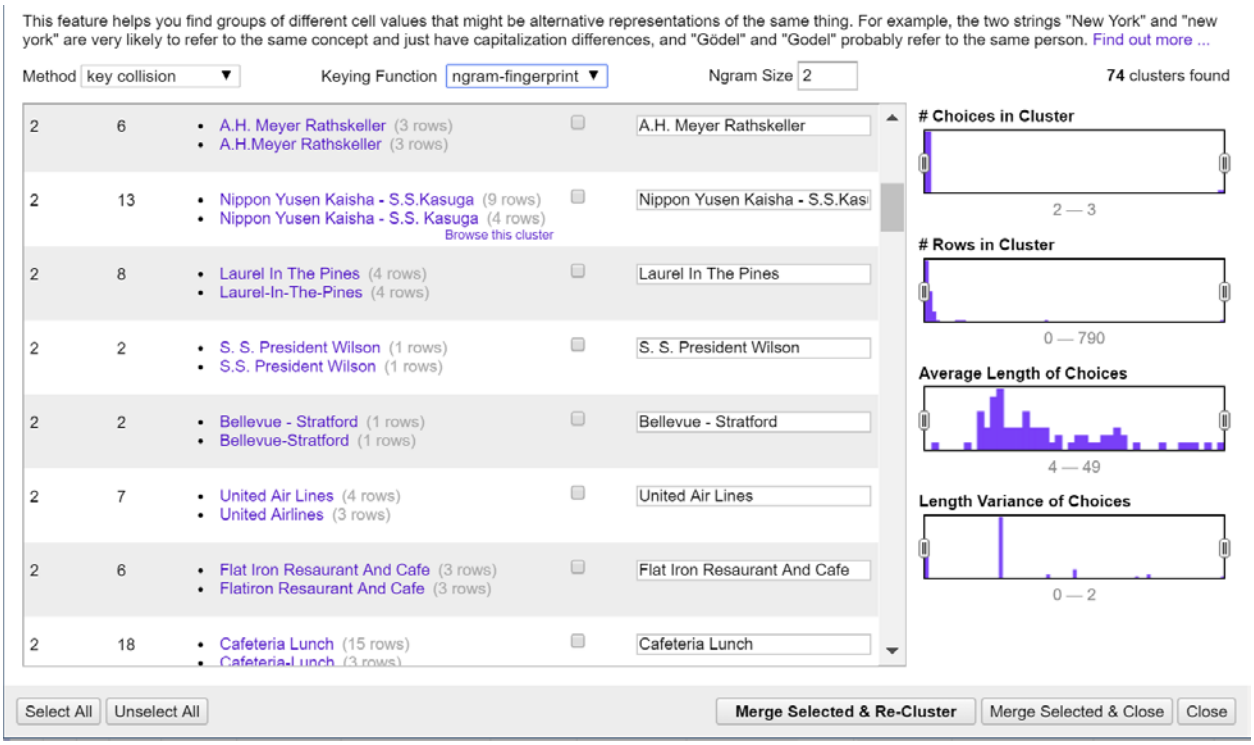
Step 1: Converted Date Field to Date

Step 2: Analyzed Date field outliers and based on that removed outliers from Date field

Step 3: Used Clustering from open refine on column “Sponsor”. Following methods were considered:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

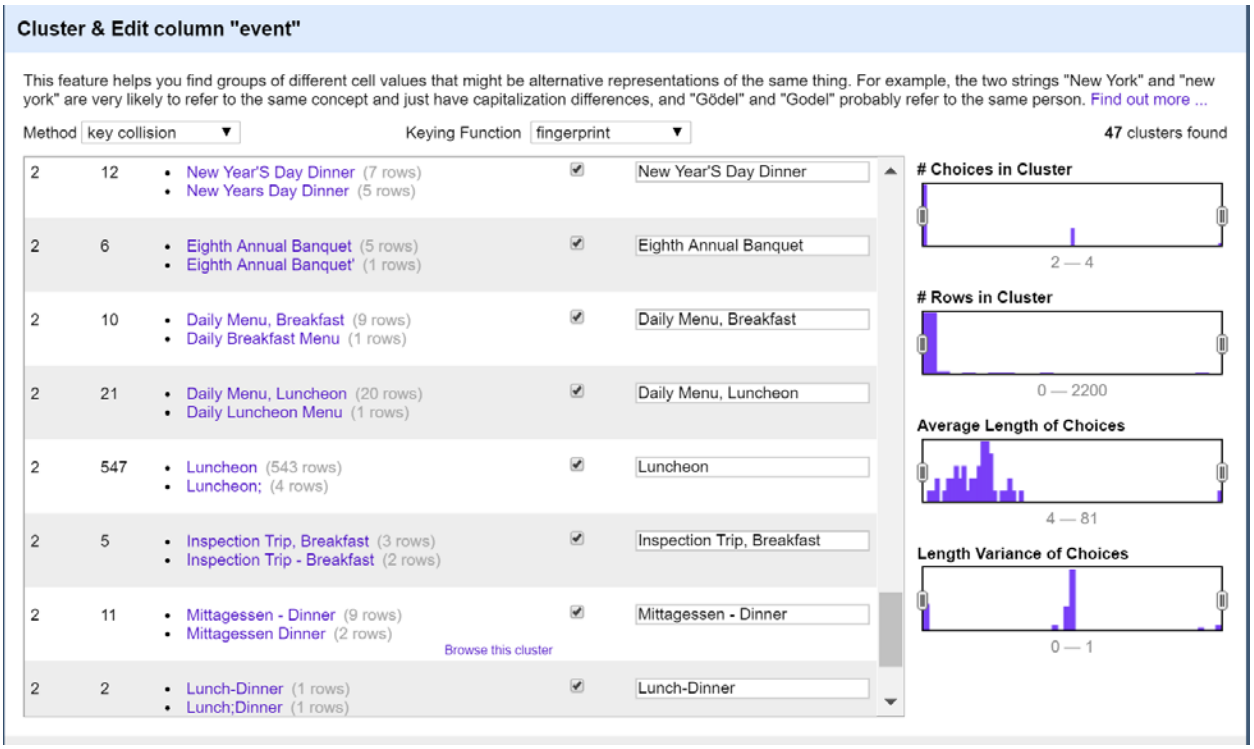
Screenshot:



Step 4: Used Clustering from open refine on column “Event”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

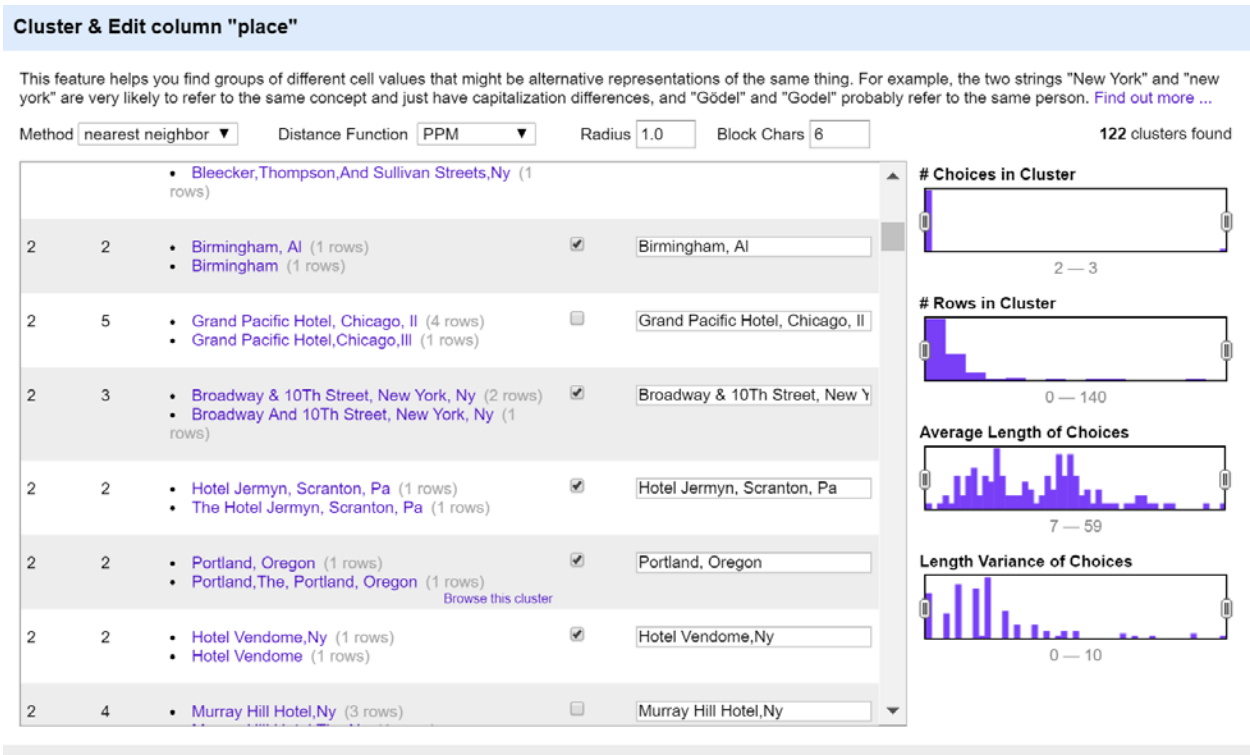
Screenshot:



Step 5: Used Clustering from open refine on column “Place”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

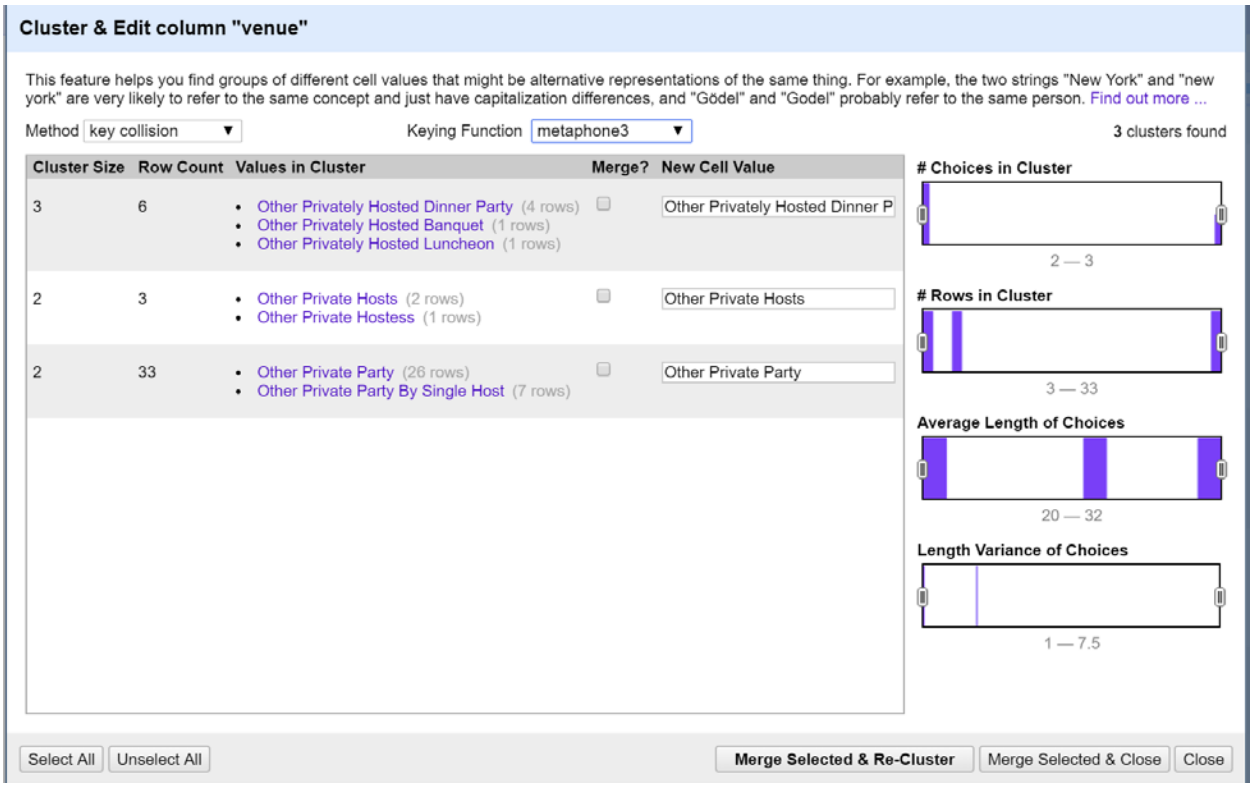
Screenshot:



Step 6: Used Clustering from open refine on column “Venue”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

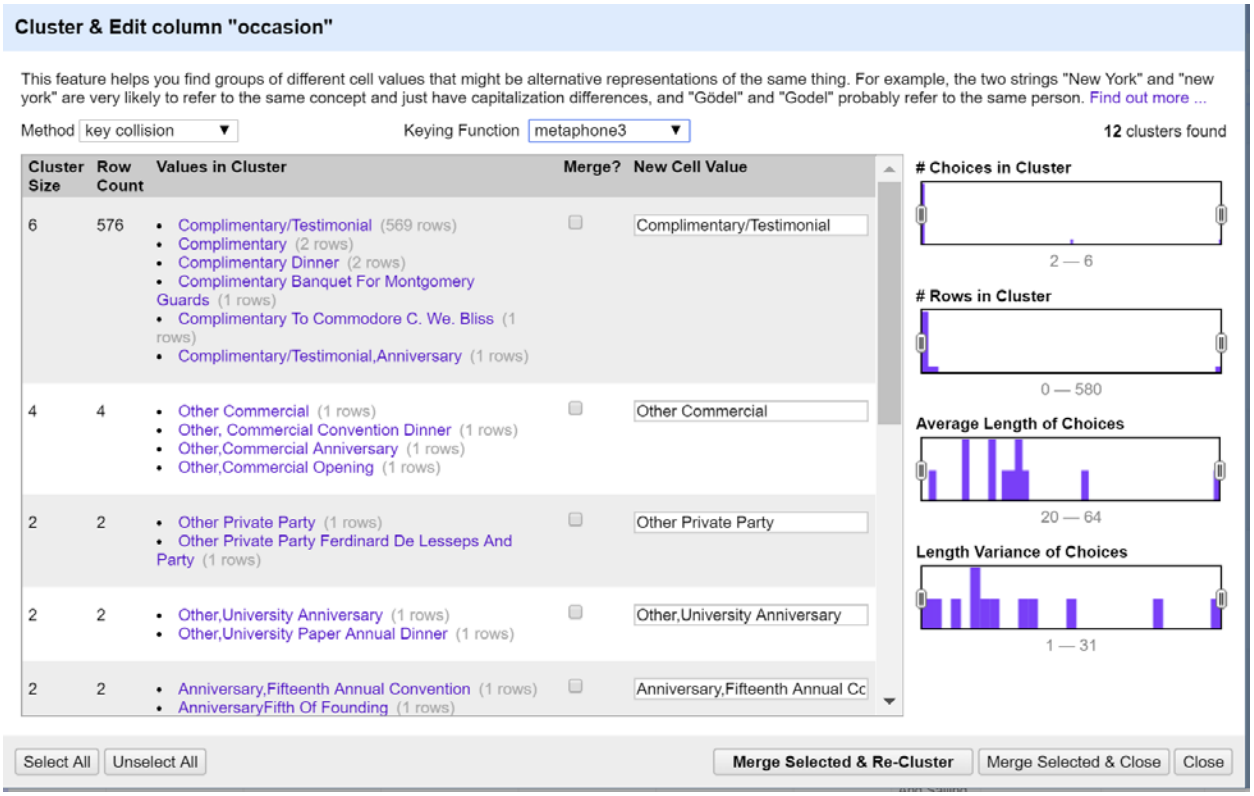
Screenshot:



Step 7: Used Clustering from open refine on column “Occasion”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

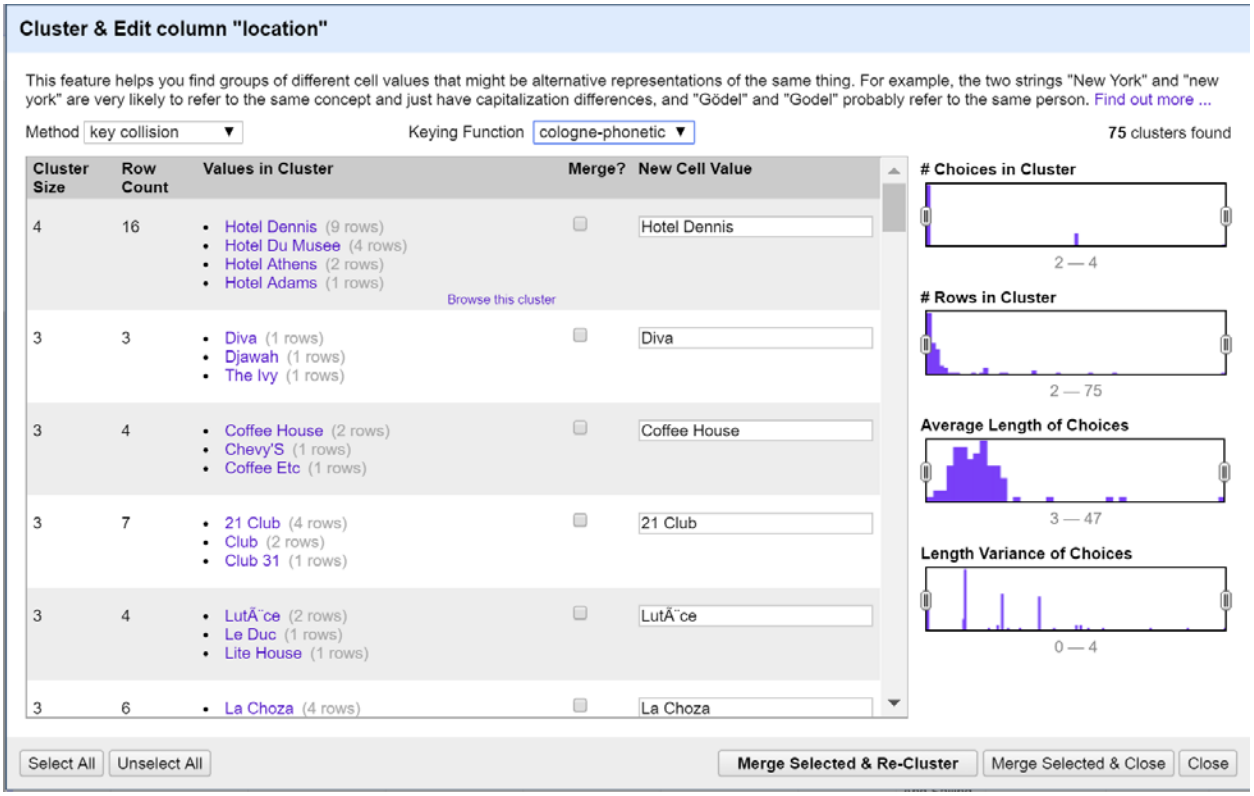
Screenshot:



Step 8: Used Clustering from open refine on column “Location”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

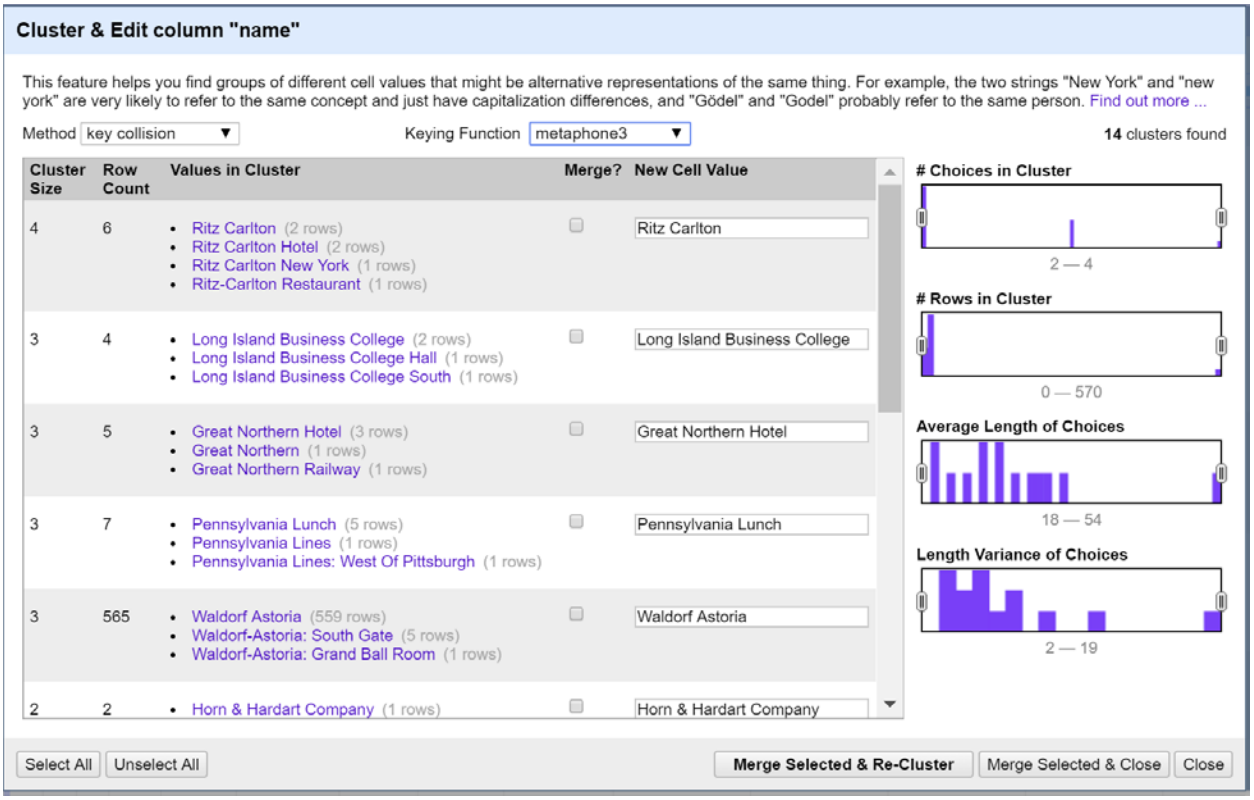
Screenshot:



Step 9: Used Clustering from open refine on column “Name”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

Screenshot:



Step 10: Split the physical_description column. It resulted in 7 new columns:

physical_description 1

physical_description 2

physical_description 3

physical_description 4

physical_description 5

physical_description 6

physical_description 7

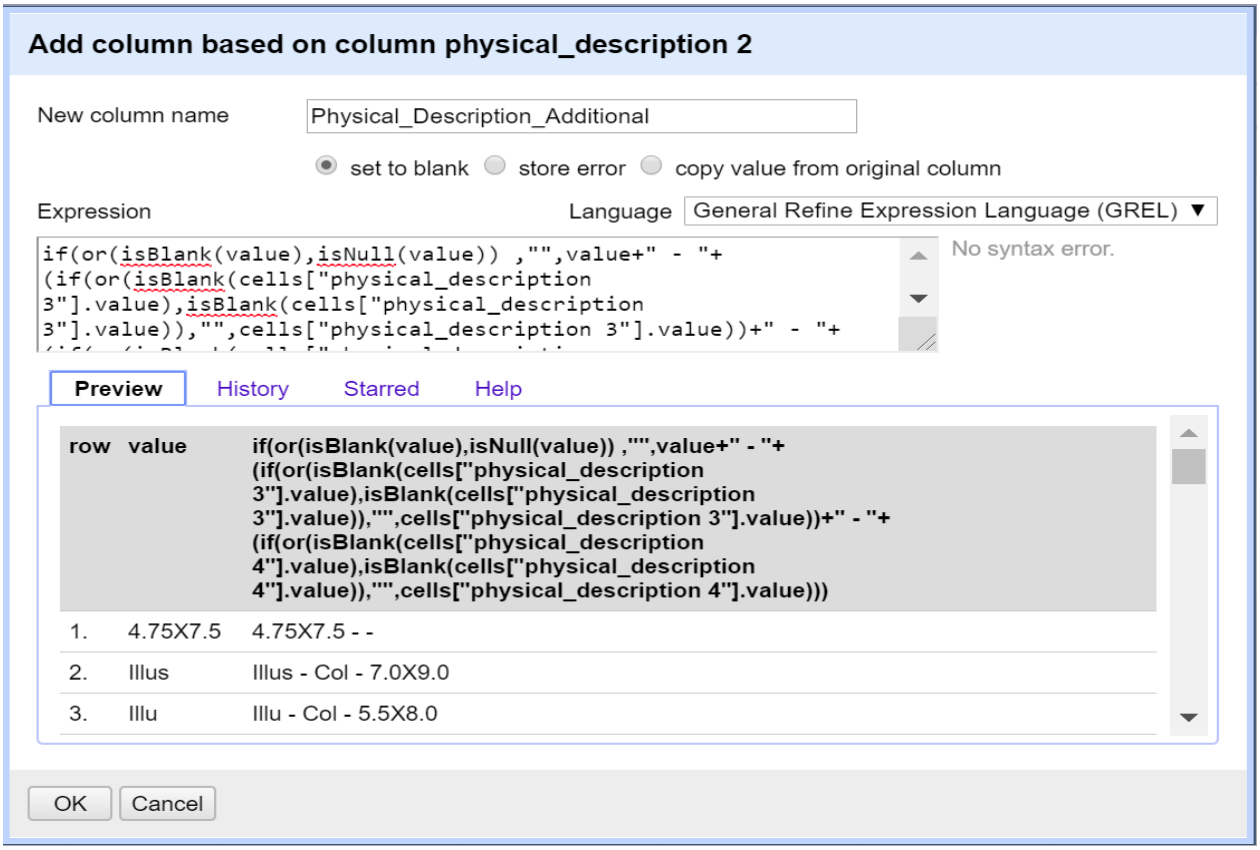
We updated the column physical_description 1 to physical description type and then created a new column Physical_description_additional by merging following columns:

physical_description 2

physical_description 3

physical_description 4

Screenshot:



File: Dish.csv

Step 1: We found even after cleaning with python script, there were few different kinds of special characters were present in the file. So, we transform data to remove unwanted characters. Formula:

value.replace(/[>:<%#@!\\()\[\]\?\"\-*,\.\+]/, " ").replace(/\s+/," ").trim()

Step 2: Used Clustering from open refine on column “Name”. Following methods were used:

a) Method: Key-Collision, Function: Fingerprint
b) Method: Key-Collision, Function: n-gram fingerprint (n=2)
c) Method: Key-Collision, Function: metaphone3
d) Method: Key-Collision, Function: cologne-phonetic
e) Method: Nearest Neighbor, Function: PPM (Radius: 1, Block Chars: 6)
f) Method: Nearest Neighbor, Function: Levenshtein (Radius: 1, Block Chars: 6)
g) Group updates

Screenshot:

Cluster & Edit column "name"

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 ▼ **396** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
5	7	• Canadian Cheese Imported (2 rows) • Imported Canadian Cheese (2 rows) • Canadian Imported Cheese (1 rows) • Cheese Canadian Imported (1 rows) • Imported Cheese Canadian (1 rows)	☐	Canadian Cheese Imported
5	13	• Iced Tea Or Coffee (6 rows) • Iced Tea Or Iced Coffee (3 rows) • Iced Coffee Or Tea (2 rows) • Coffee Or Tea Iced (1 rows) • Iced Coffee Or Iced Tea (1 rows)	☐	Iced Tea Or Coffee
5	17	• Imported Swiss Cheese (6 rows) • Swiss Cheese Imported (5 rows) • Cheese Imported Swiss (2 rows) • Imported Cheese Swiss (2 rows) • Swiss Imported Cheese (2 rows) Browse this cluster	☐	Imported Swiss Cheese
5	13	• Club Soda C & C Imported (5 rows) • Imported Club Soda C & C (4 rows) • Club Soda C &C Imported (2 rows) • C & C Club Soda Imported (1 rows) • Club Soda Imported C & C (1 rows)	☐	Club Soda C & C Imported

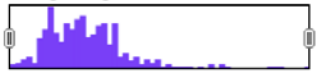
 # Choices in Cluster

2 — 5

 # Rows in Cluster

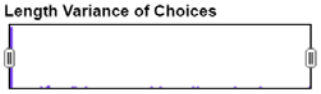
2 — 40

Average Length of Choices



8 — 63

Length Variance of Choices



0 — 4

Select All | Unselect All | **Merge Selected & Re-Cluster** | Merge Selected & Close | Close

Cluster & Edit column "name"

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 nearest neighbor Distance Function levenshtein Radius 1.0 Block Chars 6 **20642** clusters filtered from **21970** total

3	3	• Antedeluvian (1 rows) • Antediluvian (1 rows) • Antidiluvian (1 rows)	☑	Antedeluvian
3	3	• Schwepp'S Imported Ginger Ale Pint 50 (1 rows) • Schwepp'S Imported Gingerale Pint (1 rows) • Schweppe'S Imported Gingerale Pint (1 rows)	☐	Schwepp'S Imported Ginger
3	3	• Tomates Genoise (1 rows) • Tomates GÃ©noise (1 rows) • Tomatoes Genoise (1 rows) Browse this cluster	☐	Tomates Genoise
3	3	• Guiness Export Stout (1 rows) • Guinnes Export Stout (1 rows) • Guinness Export Stout (1 rows)	☐	Guiness Export Stout
3	5	• Manhattan Salad (3 rows) • Manhatam Salad (1 rows) • Mountain Salad (1 rows)	☐	Manhattan Salad
3	5	• Orange Pekoe (3 rows) • Orange PekoÃ« (1 rows) • Orange PÃ©koÃ« (1 rows)	☐	Orange Pekoe
3	5	• Consomme Florentine (3 rows)	☐	Consomme Florentine

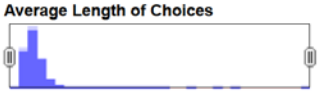
 # Choices in Cluster

0 — 150

 # Rows in Cluster

0 — 10

Average Length of Choices



0 — 330

Length Variance of Choices

0 — 15

Select All | Unselect All | **Merge Selected & Re-Cluster** | Merge Selected & Close | Close

File: MenuItem.csv

Step 12: Transform created_at field to date by creating a new field created_date

Step 13: Transformed updated_at field to date by creating a new field updated_date.

(Note: both created_at and created_date field exists in the csv file. Similarly, both updated _at and updated _date field exists in the database.)

Step 14: Transform xpos field to number

Step 15: Transformed ypos field to number

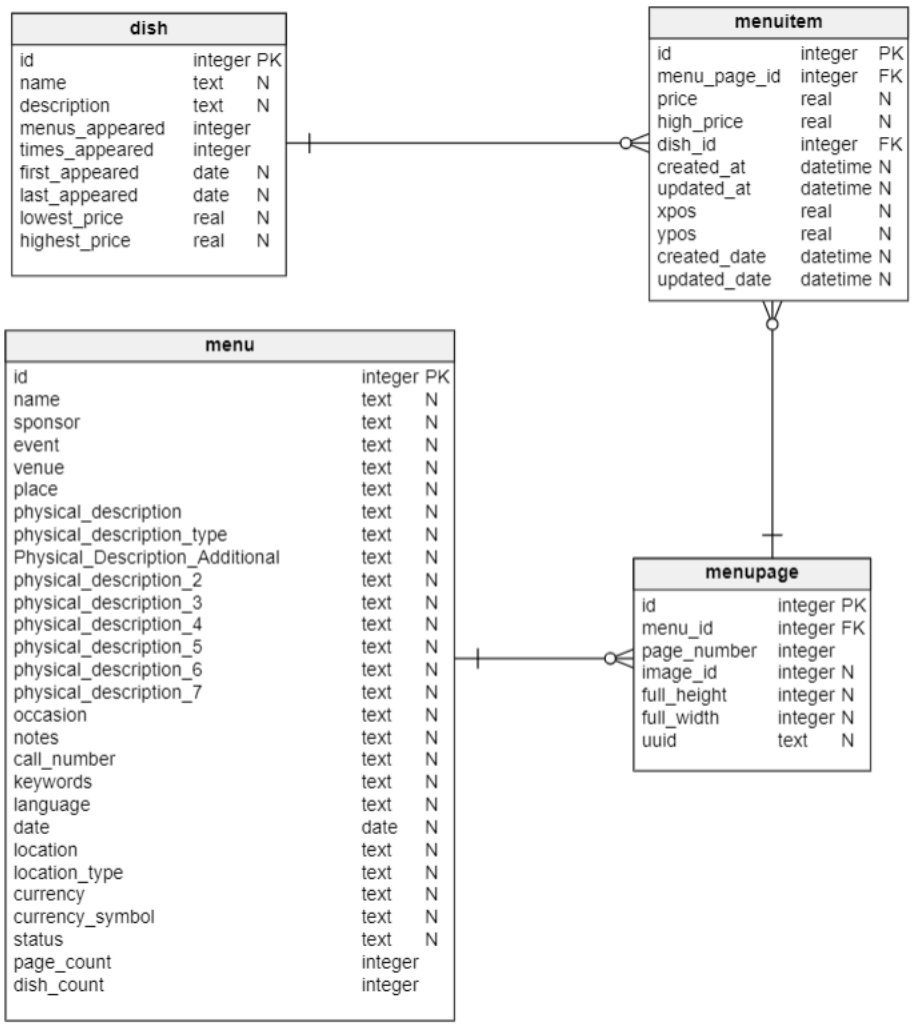
File: MenuPage.csv

No transformations were performed for this csv in OpenRefine.

Step 3: Integrity Checks using SQLite database constraints

Database Model

Based on the cleaned data from step 2, we created a database model which included the logical integrity constraints of 'Primary Key', 'Foreign Key' and 'Not Null'. The database model also defined the one-to-one and one-to-many relationships between the four tables. Building the schema based on this model ensured that only the records satisfying all the defined constraints are loaded into the tables and the dirty data is discarded. Here is the ER diagram of our DB model:

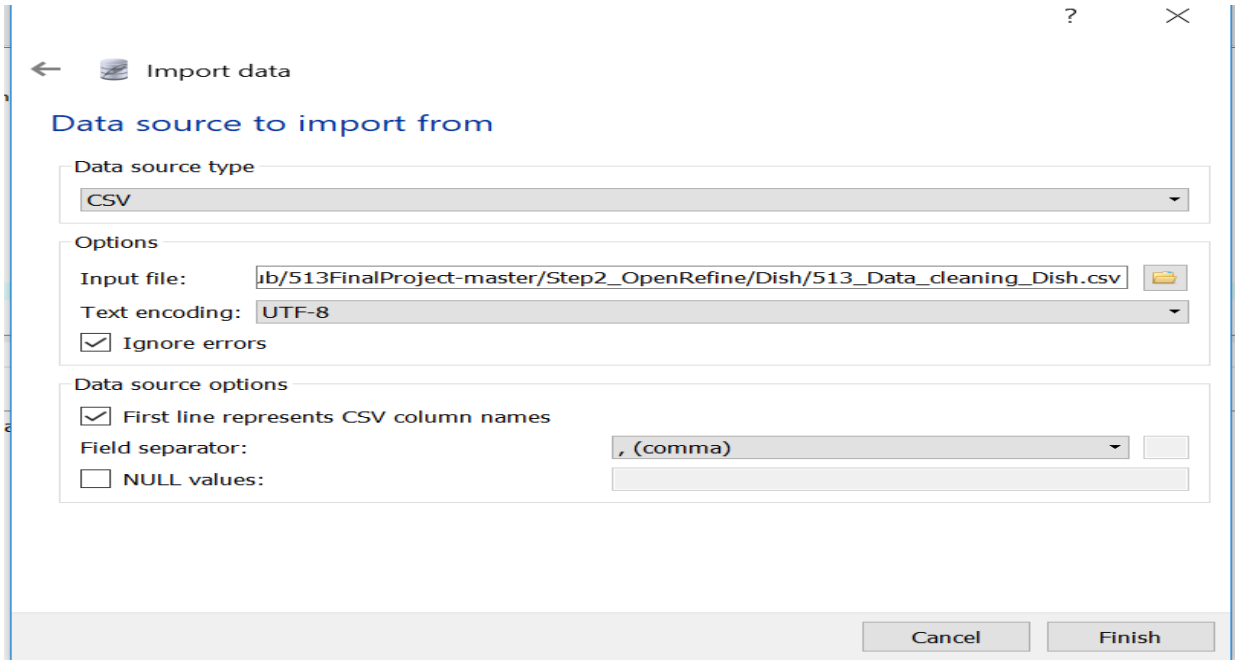


Tables Creation and Data Import

A database schema 'nypl' was created using SQLiteStudio. We created 4 tables and imported data into these tables from the cleaned csv files. As a result, following 4 tables were created:

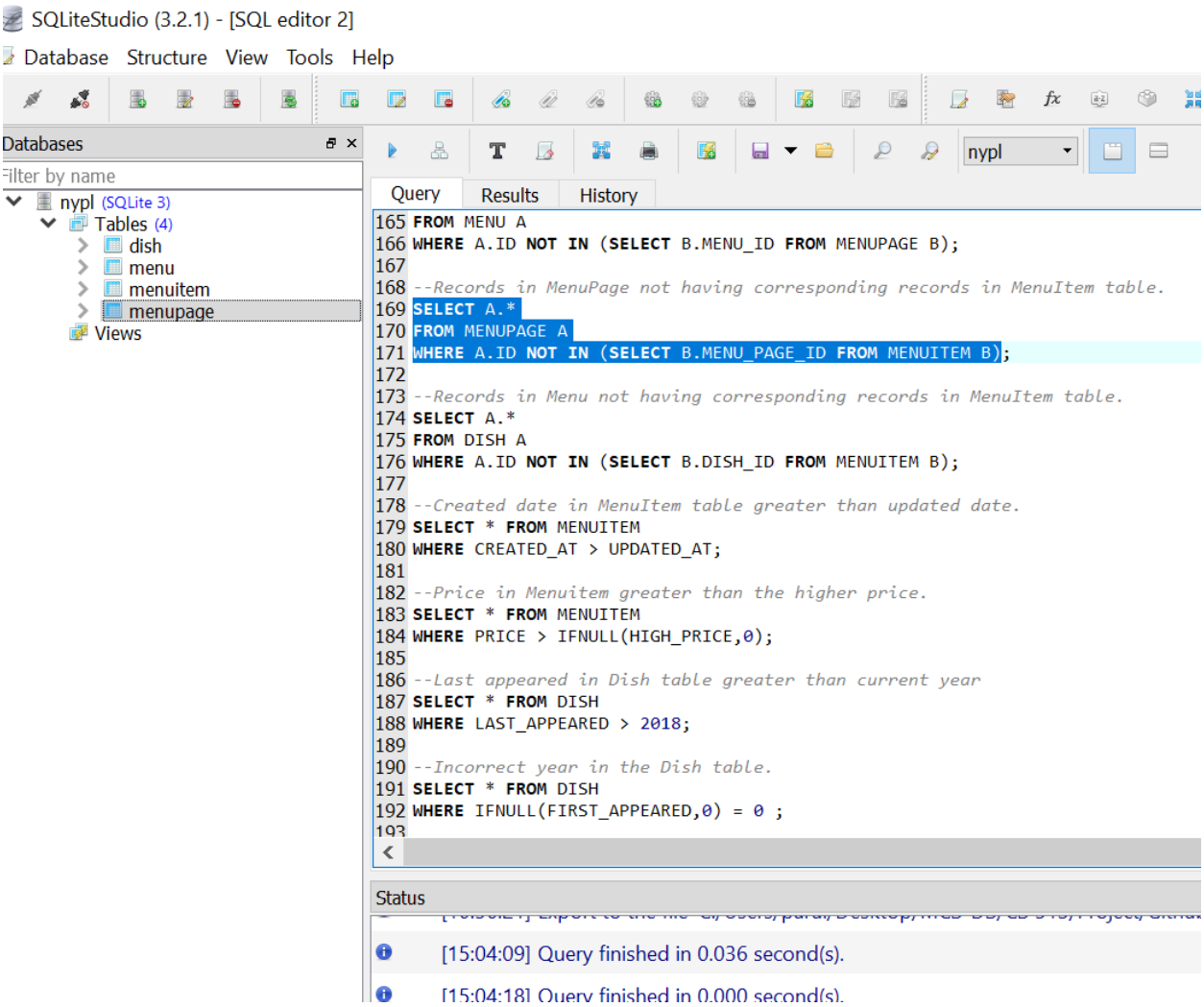
- Dish
- Menu
- Menupage
- MenuItem

Screenshot (for data import)



Integrity Constraints check

We created several queries to check the logical integrity constraints on the tables. All the scripts for the schema/table creation and integrity constraints check are listed under the 'SQL_scripts'.sql file. SQLiteStudio was used to run the queries and obtain results.



Below is a brief description of the various checks done on the data.

Key Data Duplication Check

This step checked presence of any duplicate data in the table based on ID field. Since, we applied primary key constraint on ID field during the data load, the duplicate data was filtered out already from the tables. Our final data in the tables returned no duplicate records.

Foreign Key Constraint Check

This step checked the foreign key constraints violation on the tables. That is if any table that used primary key of another table, had data that not matched with the primary key value in the source table. We applied this constraint during the data load itself for Menu, Dish and MenuItem tables. We had to disable the foreign key constraints only on the MenuItem table during the data load as it would have otherwise resulted in loss of too many rows of data and we wouldn't get enough data for further analysis.

Not Null Check

We applied not null constraint on certain table columns to ensure that only those records were loaded into the table that had values in these columns. This resulted in discarding irrelevant records which would not be useful for further analysis. This constraint was applied on all the tables during the data load itself.

Other Checks

We applied following integrity checks to identify and correct other integrity constraint violations:

- Created date in MenuItem table greater than updated date.
- Price in Menuitem greater than the higher price.

We found that 1091 records were in violation of these constraints hence got deleted.

We checked for further integrity violations as well but for these cases we did not perform any corrective action. Corrective action is often context dependent and can be different for different use cases. Thus, we refrained from taking corrective actions. Rather, we have just highlighted the violations via sql queries. Here are the constraint descriptions along with record count returned after running the queries:

- Last appeared in Dish table greater than current year (2018)
  Result: 179 records returned.
- Incorrect year in the Dish table.
  Result: 55287 records returned.
- First_appeared greater than last_appeared in Dish table.
  Result: 6 records returned.
- Lowest price greater than highest price in Dish table.
  Result: 0 records returned.
- Count of menus_appeared in dish table not matching count obtained from menuitem and menupage tables.
  Result: 2375 records returned.
- Mismatch of page count between menu and menupage tables.
  Result: 147 records returned.

Process Workflow (Provenance via YesWorkflow)

We created overall process workflow using online YesWorkflow tool (link in appendix section) for provenance. This workflow provides a birds-eye view of the overall process used for this data cleaning exercise.

We also created two separate detailed workflows for SQLite and Data Cleanup Operations. Each of these operations involved multiple steps and these workflows provide the details about these operations.

Summary Workflow

Key Inputs/Outputs

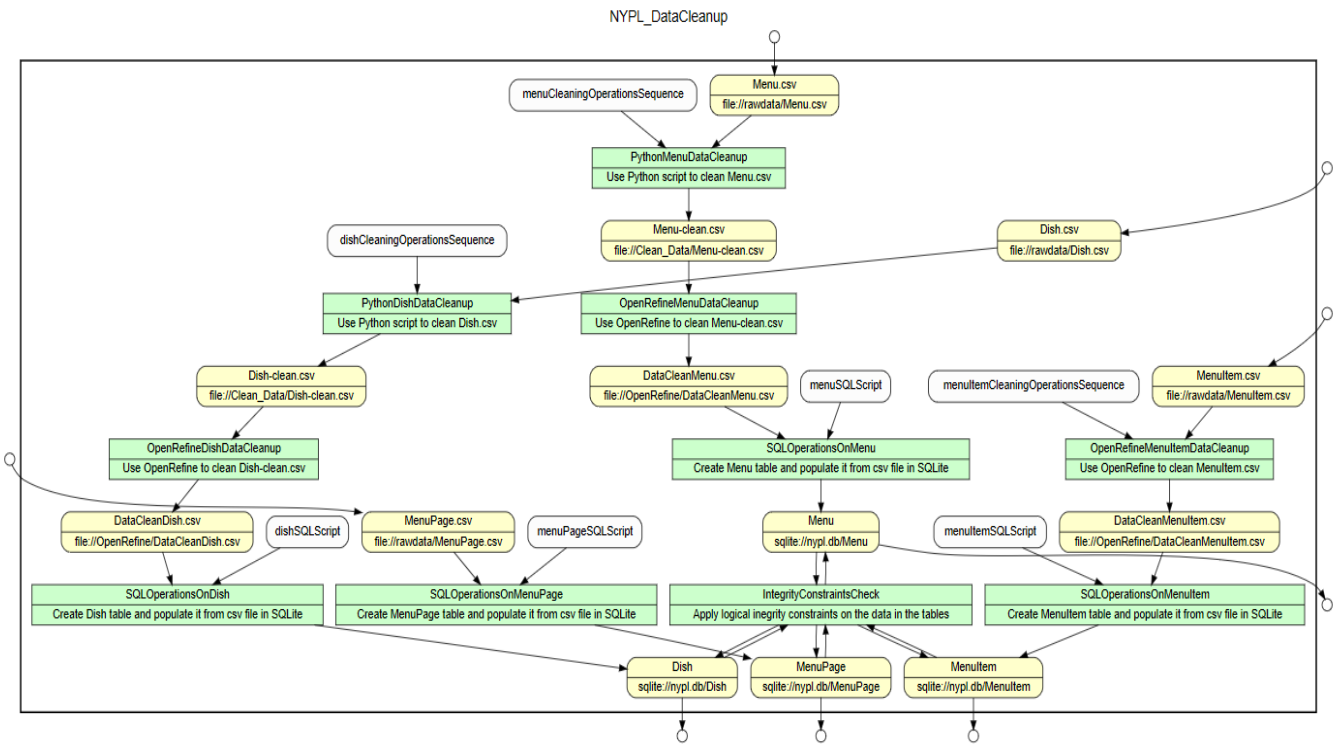
The key inputs to the summary workflow process are the four input data files namely, menu.csv, dish.csv, menuitem.csv, menupage.csv along with the data cleaning operations comprising of OpenRefine step, Python script step, and SQLite step.

Dependencies

The complete workflow includes several steps that are interdependent on each other. The data cleaning operation done with OpenRefine and Python script is dependent on the raw data files. OpenRefine step is dependent on python script output for dish.csv and menu.csv. The SQL operations of schema creation and integrity constraints is dependent on the files cleansed through open refine for dish.csv, menu.csv and menuitem.csv.

Diagram

Below is a snapshot of the Overall Workflow Diagram:

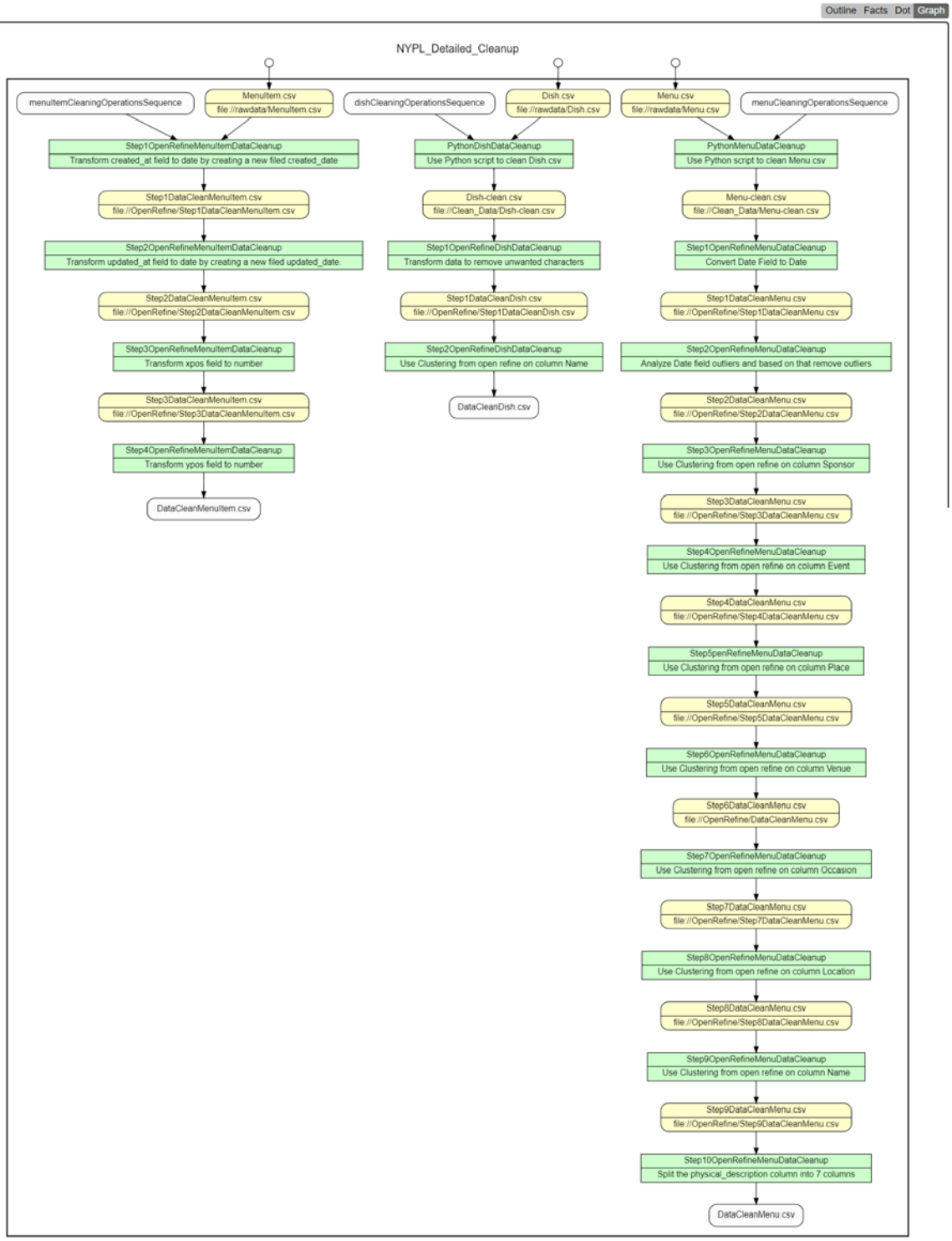


Data Cleanup Detailed Workflow

The input here are the raw data files and the output are the cleansed Menu, MenuItem and Dish data files. Since, Menupage.csv was already clean, it was not part of this step. The multiple sub steps created partially cleaned files which have been named accordingly in the diagram.

Hello

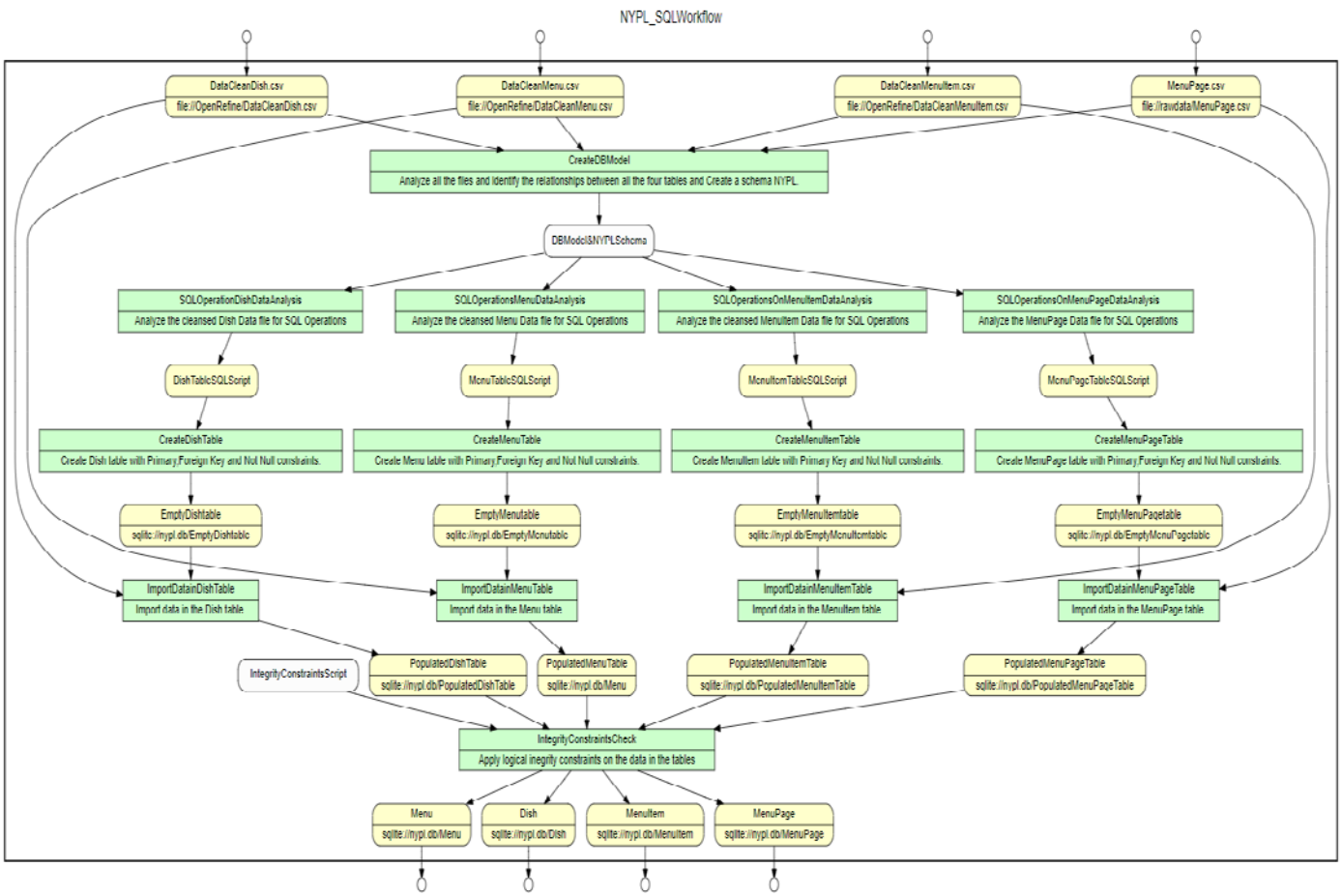
Outline Facts Dot Graph



Python Ace 12 Light Dark Steps Data Params Ports Box Title Down Across 75%

SQLite Detailed Workflow

The input here is the MenuPage file and other data files cleaned via OpenRefine and Python and the output are the 4 tables with clean data after all the integrity constraints violation checks.

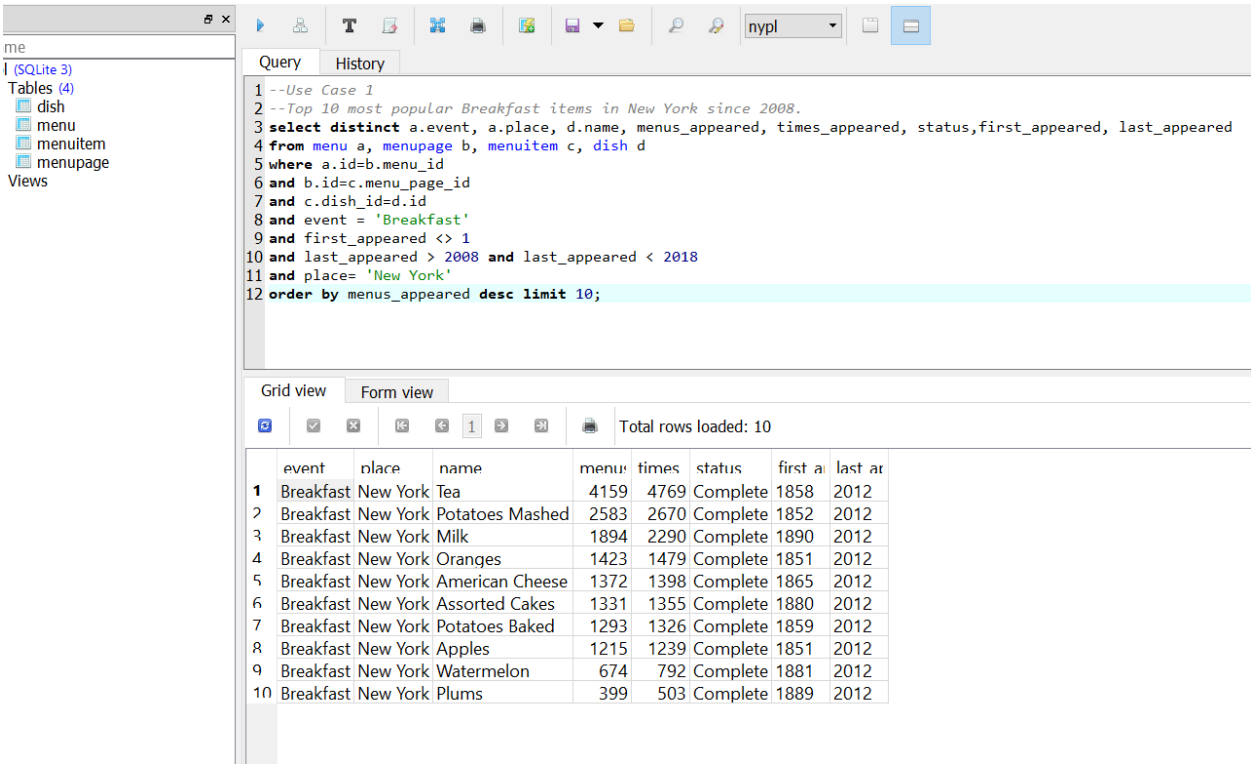


Use Case Implementation

After we were done with all the above steps, we implemented three use cases that provide meaningful data analysis to the users. This step was to check the usability of finally cleaned data. Here are the details of the use cases conducted on the final data available in the database:

Use Case 1: Top 10 Breakfast Dishes

The goal here is to identify 10 most popular Breakfast items in New York since 2008 based on the data available. Here is the query and the output:



	event	place	name	menus	times	status	first a	last ar
1	Breakfast	New York	Tea	4159	4769	Complete	1858	2012
2	Breakfast	New York	Potatoes Mashed	2583	2670	Complete	1852	2012
3	Breakfast	New York	Milk	1894	2290	Complete	1890	2012
4	Breakfast	New York	Oranges	1423	1479	Complete	1851	2012
5	Breakfast	New York	American Cheese	1372	1398	Complete	1865	2012
6	Breakfast	New York	Assorted Cakes	1331	1355	Complete	1880	2012
7	Breakfast	New York	Potatoes Baked	1293	1326	Complete	1859	2012
8	Breakfast	New York	Apples	1215	1239	Complete	1851	2012
9	Breakfast	New York	Watermelon	674	792	Complete	1881	2012
10	Breakfast	New York	Plums	399	503	Complete	1889	2012

The output indicates that tea and mashed potatoes are the most popular dishes for breakfast since 2008 which seems reasonable.

Use Case 2: Dishes Served in Hotel Eastman

The goal here is to output to the user the dishes served on Hotel Eastman's Lunch menu. Following is the query and output:

Query History

```
--Use Case 2
--Dishes served on Hotel Eastman's lunch menu.
select d.name, page_number, c.*
from menu a, menupage b, menuitem c, dish d
where a.id=b.menu_id
and b.id=c.menu_page_id
and c.dish_id=d.id
and event = 'Luncheon'
and sponsor = 'Hotel Eastman'
order by d.id desc
;
```

Grid view Form view

Total rows loaded: 48

	name	page	id	menu	price	high r	dish id	created at	updated at	creat
1	American Cheese	1	9755	2239			197085	2011-04-20 22:40:21 UTC	2011-04-20T22:40:21Z	2011
2	Lunch Biscuits	1	9940	2239			6153	2011-04-20 22:48:27 UTC	2011-04-20T22:48:27Z	2011
3	Spiced Beets	1	9935	2239			6149	2011-04-20 22:49:25 UTC	2011-04-20T22:49:25Z	2011
4	Oyster Plant	1	9930	2239			6147	2011-04-20 22:48:02 UTC	2011-04-20T22:48:02Z	2011
5	Potato Salad German Style	1	9922	2239			6143	2011-04-20 22:47:37 UTC	2011-04-20T22:47:37Z	2011
6	Celery Fritters	1	9912	2239			6139	2011-04-20 22:48:19 UTC	2011-04-20T22:48:19Z	2011
7	Easter Salad A La Russe	1	9901	2239			6133	2011-04-20 22:47:53 UTC	2011-04-20T22:47:53Z	2011
8	Strawberry Tartlets	1	9892	2239			6125	2011-04-20 22:47:35 UTC	2011-04-20T22:47:35Z	2011
9	Uneeda Biscuits	1	9883	2239			6119	2011-04-20 22:47:18 UTC	2011-04-20T22:47:18Z	2011
10	Hot Ginger Bread	1	9876	2239			6115	2011-04-20 22:45:50 UTC	2011-04-20T22:45:50Z	2011
11	Meringue Baskets Chantilly	1	9862	2239			6107	2011-04-20 22:46:23 UTC	2011-04-20T22:46:23Z	2011
12	Spiced Lamb Tongue	1	9818	2239			6082	2011-04-20 22:44:20 UTC	2011-04-20T22:44:20Z	2011
13	Marinerte Fett Herring	1	9810	2239			6075	2011-04-20 22:42:49 UTC	2011-04-20T22:42:49Z	2011
14	St Bernard Pudding Wine Sauce	1	9806	2239			6072	2011-04-20 22:42:30 UTC	2011-04-20T22:42:30Z	2011
15	Crushed Violet Sherbet	1	9779	2239			6058	2011-04-20 22:41:18 UTC	2011-04-20T22:41:18Z	2011
16	Cervelat Sausage	1	9760	2239			6047	2011-04-20 22:41:44 UTC	2011-04-20T22:41:44Z	2011
17	Cold Ham	1	9731	2239			6035	2011-04-20 22:39:03 UTC	2011-04-20T22:39:03Z	2011
18	Southern Corn Bread	1	9725	2239			6033	2011-04-20 22:39:59 UTC	2011-04-20T22:39:59Z	2011
19	Baked Beans Boston Style	1	9720	2239			6030	2011-04-20 22:38:33 UTC	2011-04-20T22:38:33Z	2011
20	Eggs With Kippered Herring	1	9711	2239			6024	2011-04-20 22:38:12 UTC	2011-04-20T22:38:12Z	2011

The dish names seem reasonable for hotel Eastman's lunch menu.

Use Case 3: Five Oldest Dishes

The goal here is to output to the user 5 of the oldest dishes on the menu. Following is the query and output:

Query History

```
--Use Case 3
--5 of the oldest dishes on the Menu (first appeared in 1851)
select * from dish where
first_appeared= (select min(first_appeared) from dish where first_appeared not in ( 0,1)) order by id desc limit 5
```

Grid view Form view

Total rows loaded: 5

	id	name	descri	menus	times	first a	last ar	lowest	highes
1	493867	Claret Chateau Lafitte		1	1	1851	1851	1	2
2	493866	Claret Chateau La Rose		1	1	1851	1851	0.75	1.5
3	493865	Porter And Ale Guinness'S Xx Dublin Brown Stout		1	1	1851	1851	0.25	0.5
4	493864	Porter And Ale Byass'S London Porter		1	1	1851	1851	0.25	0.5
5	493863	Porter And Ale Tennant'S Scotch Ale		1	1	1851	1851	0.25	0.5

The result shows us few oldest wines which makes sense.

Challenges

Ambiguity

One of the key challenges that we faced was the ambiguity in the data. We did not intend to remove the ambiguity in the data as that would have required us to make assumptions. For example, if we found an item as omelette 1 and omelette 2 then we did not try to merge it to “omelette”. The reason behind it was that these two could be a variant of omelette with different ingredient and sometimes different size of servings. Assuming them to represent one dish could hamper the data integrity. Thus, there was a tradeoff between ambiguity and integrity. So, we left those cases as it is

OpenRefine Efficiency

Open refine was not able to load large datasets such as dish.csv. We had to change following configurations in refine.ini

REFINE_MAX_FORM_CONTENT_SIZE=99999999

REFINE_MEMORY=4096M

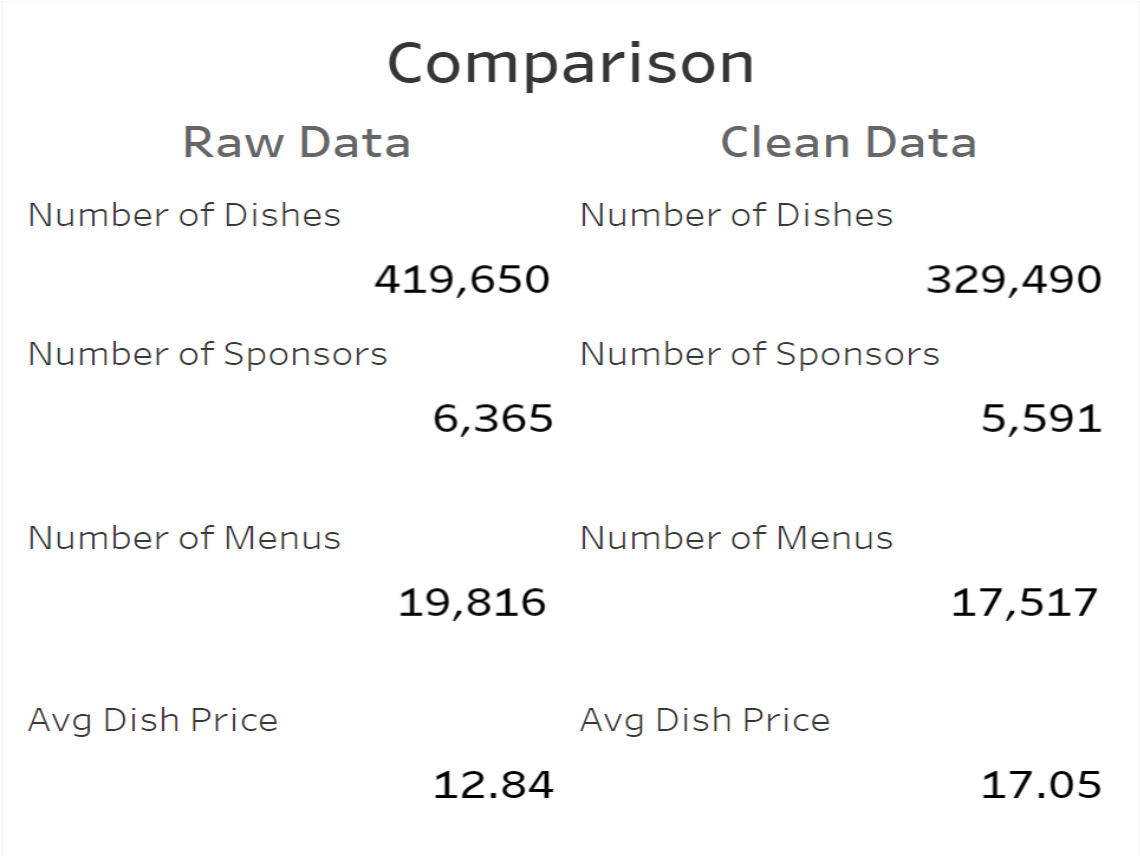
Same can be achieved by following command: openrefine /m 4096M.

Further, for some of the operations, especially facet creation on large dataset, open refine was not able to perform optimally. We had to reduce the data size by applying a filter and then we were able to

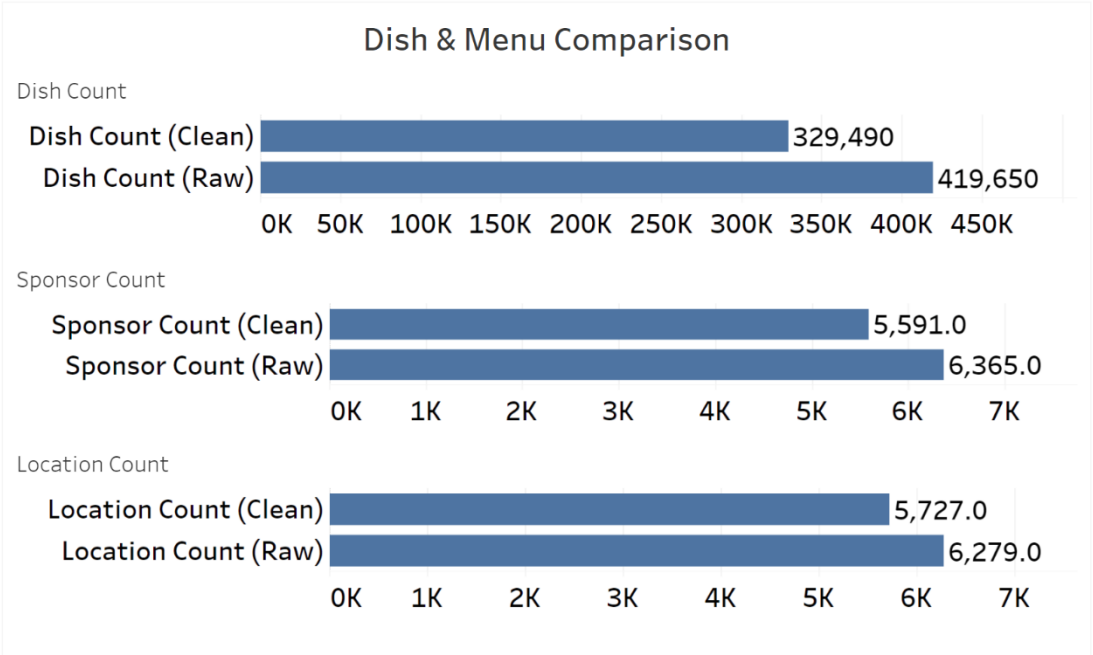
create the facet. We also found that it is more time-efficient to apply clustering directly instead of first creating facet and then applying the clustering.

Results

After cleaning the data, we compared the raw data with the cleaned data. We used tableau to visualize the comparison. The comparison can be accessed here.



We also compared some other fields in the dish and menu files. The comparison can be accessed here.



Also, from the same link user can access both the raw data and the clean data via tableau dashboards. These tableau dashboards are available on public.tableau.com. This way clean data can be accessed for further data analytics purpose.

We can see that a there is a huge impact of the cleaning exercise. There were several fields with dirty data and were cleaned as part of this exercise. For dish names alone almost 90K dish names were found to be either duplicate or had some sort of spelling mistakes!!

Appendix

Dataset

New York Public Library Menus data

Tools/Software Used

Software	Version/Url	Usage
Python	3.6.0	Scripting Language used for Data Cleaning
OpenRefine	2.6-rc.2	Data Cleaning Tool
SQLiteStudio	3.2.1	UI tool for working with SQLite
YesWorkflow	http://try.yesworkflow.org/	Provenance Tool
Tableau	2018.1.2	Visualization Representation
Vertabelo	https://my.vertabelo.com/	Database Modeler
GitHub	https://github.com/rohitbansal83/513FinalProject	Source control repository
Slack	https://uiuc-mcsds.slack.com/	Communication channel used for project work
zoom	https://www.zoom.us/	Video Conferencing tool for coordination

Project Deliverables

All the projects deliverables including the raw dataset, sql scripts, cleansed table data, Yesworkflow scripts can be accessed at this link.