



# **CS 598 Theory & Practice of Data Cleaning**

## **Final Project Report**

### **Cleaning the NYPL menus' Dataset**

Abhisek Jana (abhisek2)  
Ashish Kumar(ashishk2)  
Aruna Neervannan(arunarn2)

## Table of Contents

1	Introduction .....	4
2	Structure of Dataset.....	4
2.1	Potential Use Cases .....	6
2.2	Use Cases requiring no clean up .....	6
2.3	Will it ever be good enough? .....	6
3	OpenRefine .....	6
3.1	Environment Setup.....	6
3.2	Changes in Menu CSV.....	6
3.2.1	Transformations on “sponsor” column .....	6
3.2.2	Transformations on “event” column .....	10
3.2.3	Transformations on “venue” column .....	16
3.2.4	Transformation on “place” column .....	19
3.2.5	Transformations on “physical_description” column .....	23
3.2.6	Transformations on “occasion” column .....	23
3.2.7	Transformation on “notes” column.....	27
3.2.8	Transformation on “call_number” column.....	27
3.2.9	Transformation on “date” column.....	27
3.2.10	Transformation on “location” column.....	28
3.2.11	Columns Deleted or Merged.....	31
3.2.12	Columns Untouched .....	31
3.2.13	Operation History & Provenance.....	32
3.3	Changes in Dish CSV .....	32
3.3.1	Transformation on “name” column.....	32
3.3.2	Transformations on “first_appeared” column .....	36
3.3.3	Transformations on “last_appeared” column .....	36
3.3.4	Transformations on “menus_appeared” column.....	37
3.3.5	Transformations on “lowest_price” column .....	37
3.3.6	Transformations on “highest_price” column .....	38
3.3.7	Columns Deleted or Merged.....	38
3.3.8	Columns left Untouched .....	38
3.3.9	Operation History & Provenance.....	39

3.4	Changes in MenuPage CSV.....	39
3.4.1	Operation History & Provenance.....	39
3.5	Changes in MenuItem CSV .....	40
3.5.1	Operation History & Provenance.....	42
3.6	Columns after cleanup with OpenRefine .....	42
3.6.1	Menu csv.....	42
3.6.2	Dish csv.....	43
3.6.3	MenuItem csv .....	43
3.6.4	MenuPage csv .....	43
4	SQLite .....	43
4.1	Schema .....	44
4.1.1	Enable foreign key constraint support in SQLite .....	44
4.2	Data Definition Language .....	44
4.3	CSV Import.....	46
4.3.1	Adding Data Source.....	46
4.3.2	Import Data.....	47
4.4	Integrity Constraints.....	50
4.4.1	Row Counts .....	52
4.5	Clean Scripts .....	52
4.6	Data Extraction.....	53
4.6.1	Data Extraction Query.....	53
5	Provenance .....	53
6	Dataset Quality Analysis .....	54
6.1	Analysis of empty data in the tables .....	54
6.1.1	Percentage of empty data in menu table .....	54
6.1.2	Percentage of empty data in menu_page table .....	55
6.1.3	Percentage of empty data in dish table.....	55
6.1.4	Percentage of empty data in menu_item table.....	56
6.2	Regression Analysis Using Least Squares Method .....	56
6.2.1	Price vs Date (logScale) .....	56
6.2.2	Python code for Date Transformation and Outlier Detection.....	57
6.2.3	R Code for Linear Regression .....	58
6.2.4	Analysis .....	59

6.3	Visualization (Dirty Vs Clean Data).....	59
6.3.1	Top 30 Dishes organized by Occasion/Special Event.....	59
6.3.2	Top 30 Dishes and the venues where they appeared .....	60
6.3.3	Food Evolution of Top 30 Dishes .....	61
6.3.4	Top 30 popular Dishes of all times.....	63
7	Observations and Conclusions.....	64
7.1	Source and What We Cleaned .....	64
7.2	Integrity Constraints and Other issues in SQLite .....	64
7.3	What we learned.....	65
7.4	How long did the process take? .....	65
7.5	Use cases answers.....	65
8	Links .....	65

## 1 Introduction

We have selected the **New York Public Library's** crowd sourced **historical menus dataset** available [here](#) for our project ([our local copy](#)). This dataset is an interesting collection of historical menus compiled and curated by The New York Public Library as part of their digitizing effort. The collection includes about 45,000 menus from the 1840s to the present, and the goal of the digitization project is to transcribe each page of each menu, creating an enormous database of dishes, prices, locations, and so on. As of 07/01/2017, the transcribed database contains 423,388 dishes from 17,546 menus. Here are some interesting details on the [NYPL project](#) and the purpose of this transcription.

The dataset consists of following CSV files:

- Menu
- MenuPage
- MenuItem
- Dish

## 2 Structure of Dataset

**Menu** is the key table of this database. Each row in this table represents a menu and has following columns:

- id – An int type field used to uniquely identify the record
- name – A string field and there are lots of empty entries. When not empty, it appears to be a duplicate of the “sponsor” field
- sponsor – A string field typically representing the name of the hotel or a venue. All entries are in upper case.
- event – A string field representing the name event for the menu was created. Typical values are Breakfast, Lunch, Dinner etc.
- venue – A string field representing the type of facility – commercial, social. This column has missing entries.
- place – A string field representing the location of the sponsor.
- physical Description – This is a compound field reflecting type of the menu and the dimensions of the menu
- notes – Additional details about the menu card are stored in this string field
- call\_number – A range of numbers and its intent is not clear
- keywords – empty field
- language – empty field
- date – MM/DD/YY format. Some entries are in YYYY/MM/DD format and several are missing
- location – A string field, appears to be a duplicate of sponsor except entries are in the mixed case
- location\_type – An empty string field

- currency – Field to capture currency on the menu. The field is either empty or has Dollars value
- currency\_symbol – Field to capture currency's sign. The field is either empty or has a \$ symbol
- status – A categorical value to reflect whether entry is complete or is it under view
- page\_count – A numeric field to store number of pages in the menu
- dish\_count – A numeric field to store number of dishes in a menu

**Dish** table represents the dishes that appeared on a menu and has following columns

- id – A numeric field to store id of the entry
- name – A string field to store the name of the dish
- description – A string field, can be empty or stores description of the dish
- menus\_appeared – A numeric field, purpose unclear
- time\_appeared – a numeric field, purpose unclear
- first\_appeared – A number field storing the year when dish appeared first in a menu
- last\_appeared – A number field storing the year when dish appeared last in a menu
- lowest\_price – A floating point number to store lowest price, has no current symbol.
- highest\_price – A floating point number to store lowest price, has no current symbol.

**MenuItem** is the one that links a MenuPage to a Dish and has following columns

- id – table entry ID
- menu\_page – numeric field to identify menu page, can be used as a key for the menu page dataset
- price – floating number
- high\_price – highest price compared to original price
- dish\_id – numeric field to identify the dish, can be used to identify dish in the dish dataset
- created\_at - Timestamp field shows when entry was created
- updated\_at – Timestamp field shows when entry was last updated
- xpos - floating number representing location of item on the menu on the x-axis
- ypos - floating number representing location of item on the menu on the y-axis

Each row here represents a menu item on the menu page.

**MenuPage** represent a page on a menu (linked via menu\_id column) and has following columns

- id – table entry identifier
- menu\_id – numeric field to store menu id, can be used as a key for the menu dataset
- page\_number – numeric field to store the specific page number of a given menu
- image\_id – identifier for the image of the menu page. Images of menu pages are stored in some cases.
- full\_height – numeric field, no unit of measure provided, assumed to be in millimeters
- fullwidth – numeric field, no unit of measure provided, assumed to be in millimeters
- uuid – A composite of 5 field of alpha-numeric values separated by a dash

## 2.1 Potential Use Cases

- Create a chronicle of food evolution in the 19<sup>th</sup> and 20<sup>th</sup> century.
- What were the most popular dishes? Were entrees more popular than dessert?
- Is there a relationship between dish and the venue?
- Were certain dishes popular on certain occasions?

## 2.2 Use Cases requiring no clean up

- A deeper analysis of dish dataset can help uncover popular dishes (based on how many different menus listed these dishes) and how long they were popular by calculating the difference between first appeared and last appeared values
- MenuPage csv files for the most part require no major cleanup and can provide historical information on structure of menu in the late 19<sup>th</sup> and early 20<sup>th</sup> century.

## 2.3 Will it ever be good enough?

- The MenuItem dataset has lot of missing values in the high\_price and dish\_id column.
- Since there were too many values missing we did not fill the column with the median price of an item.
- The created\_at field and updated\_at field may be useful for the data curator, however it has a very limited value for a data analyst.
- Several columns in the Menu CSV have missing values too.

## 3 OpenRefine

### 3.1 Environment Setup

For this project, we used [OpenRefine 2.7](#). After install, we edited the refine.ini to accommodate the large transformations in the csv files and modified the following defaults to:

```
REFINE_MAX_FORM_CONTENT_SIZE=14521430  
REFINE_MEMORY=4096M
```

Launched the OpenRefine webapp using the refine command line script.

### 3.2 Changes in Menu CSV

#### 3.2.1 Transformations on “sponsor” column

1. Clustering operation with keying function as “fingerprint”. For example, this would replace the following values:

```
"RED STAR LINE - ANTWERPEN -NY",  
"RED STAR LINE - ANTWERPEN NY",
```

"RED STAR LINE -ANTWERPEN NY",  
 "RED STAR LINE - ANTWERPEN - NY",  
 "RED STAR LINE - ANTWERPEN - NY",  
 "RED STAR LINE - ANTWERPEN NY",  
 "RED STAR LINE -ANTWERPEN - NY",  
 "RED STAR LINE -ANTWERPEN -NY",  
 "RED STAR LINE- ANTWERPEN NY",  
 "RED STAR LINE- ANTWERPEN -NY"

with "RED STAR LINE - ANTWERPEN - NY"

**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 ▾ 327 clusters found

Cluster Size	Row Count	Values in Cluster	Merge? <input checked="" type="checkbox"/>	New Cell Value
10	24	<ul style="list-style-type: none"> <li>• RED STAR LINE - ANTWERPEN - NY (6 rows)</li> <li>• RED STAR LINE - ANTWERPEN -NY (5 rows)</li> <li>• RED STAR LINE - ANTWERPEN NY (5 rows)</li> <li>• RED STAR LINE -ANTWERPEN NY (2 rows)</li> <li>• RED STAR LINE - ANTWERPEN - NY (1 rows)</li> <li>• RED STAR LINE - ANTWERPEN NY (1 rows)</li> <li>• RED STAR LINE -ANTWERPEN - NY (1 rows)</li> <li>• RED STAR LINE -ANTWERPEN -NY (1 rows)</li> <li>• RED STAR LINE - ANTWERPEN NY (1 rows)</li> <li>• RED STAR LINE- ANTWERPEN -NY (1 rows)</li> </ul>	<input checked="" type="checkbox"/>	RED STAR LINE - ANTWERPEN -
9	667	<ul style="list-style-type: none"> <li>• NORDDEUTSCHER LLOYD BREMEN (476 rows)</li> <li>• Norddeutscher Lloyd Bremen (125 rows)</li> <li>• NORDDEUTSCHER LLOYD - BREMEN (30 rows)</li> <li>• NORDDEUTSCHER LLOYD BREMEN (29 rows)</li> <li>• NORDDEUTSCHER LLOYD BREMEN (2 rows)</li> <li>• NORDDEUTSCHER LLOYD, BREMEN (2 rows)</li> <li>• BREMEN NORDDEUTSCHER LLOYD (1 rows)</li> <li>• NORDDEUTSCHER LLOYD - BREMEN (1 rows)</li> <li>• NORDDEUTSCHER LLOYD -BREMEN (1 rows)</li> </ul>	<input checked="" type="checkbox"/>	NORDDEUTSCHER LLOYD BREM

# Choices in Cluster

# Rows in Cluster

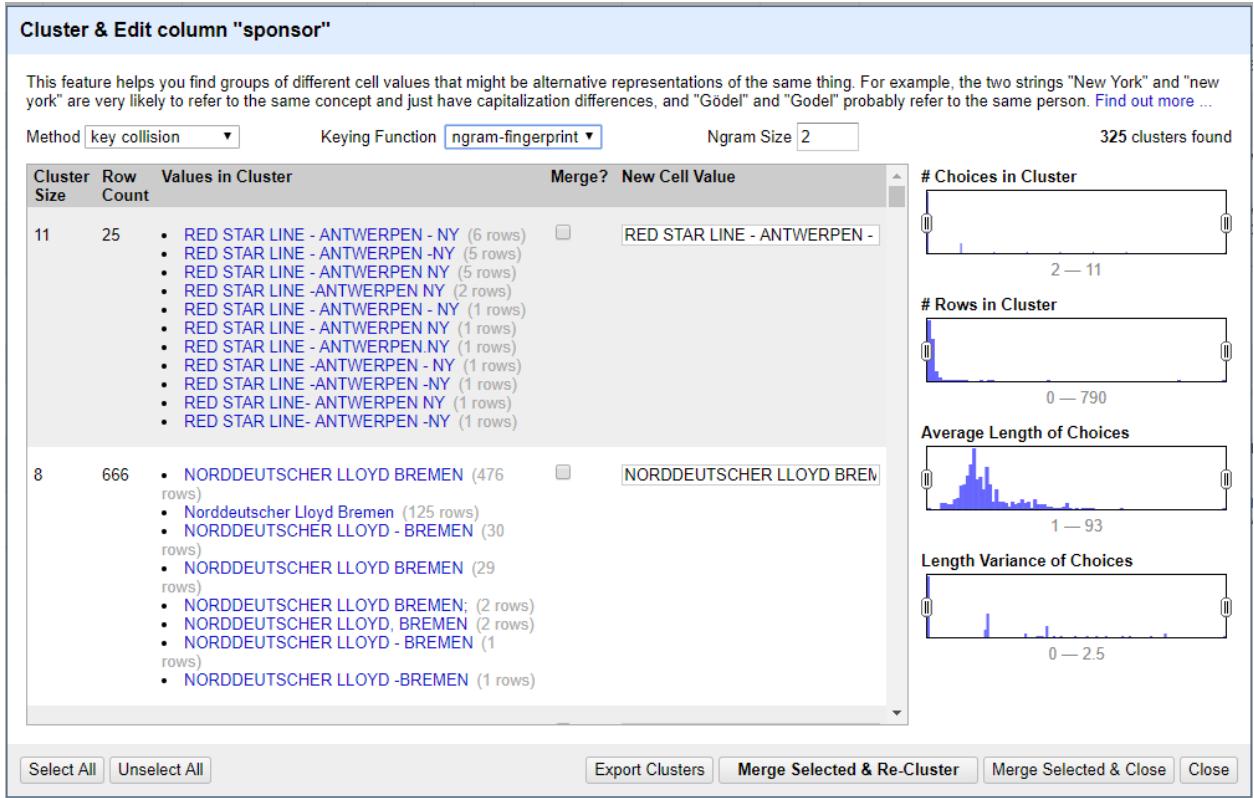
Average Length of Choices

Length Variance of Choices

2. Clustering operation with keying function as "ngram-fingerprint" with ngram size as 2. For example, this would replace the following values:

"NIPPON YUSEN KAISHA - S.S.KOBE MARU",  
 "NIPPON YUSEN KAISHA - S.S. KOBE MARU",  
 "NIPPPON YUSEN KAISHA - S.S. KOBE MARU"

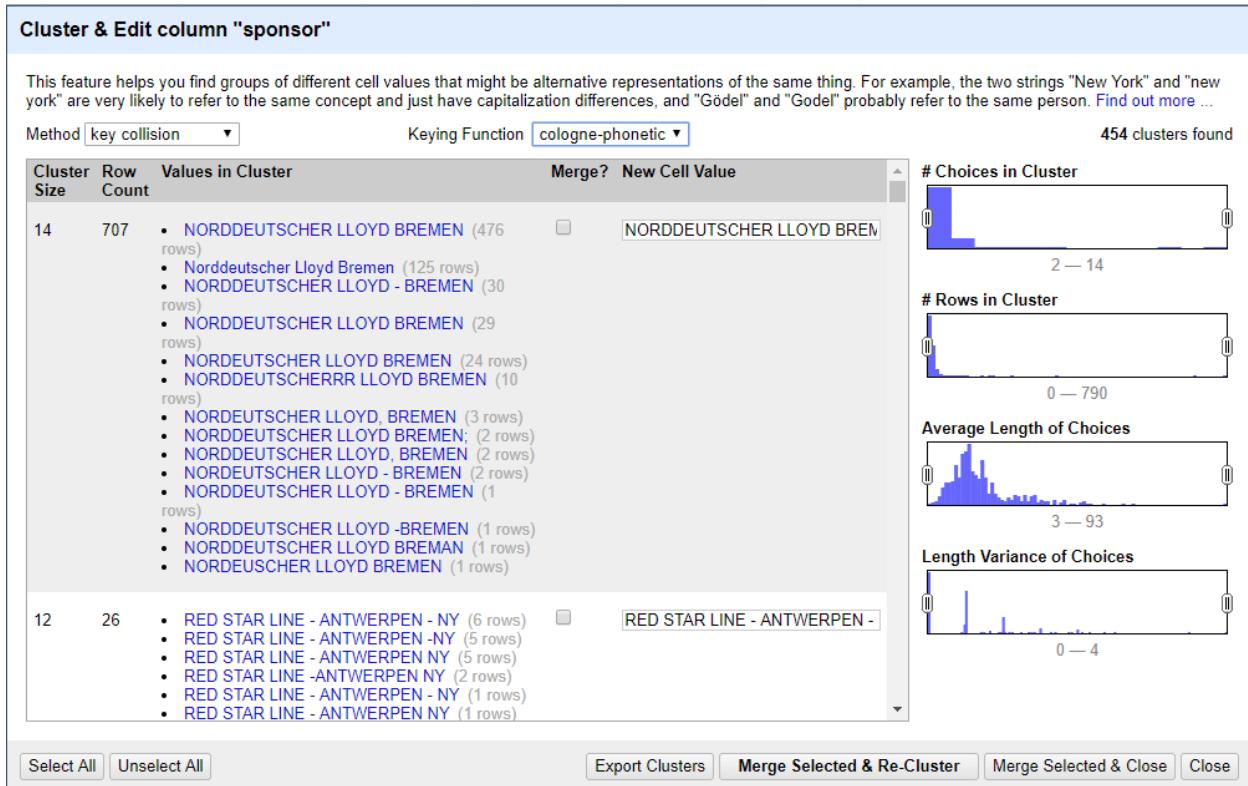
with "NIPPON YUSEN KAISHA - S.S.KOBE MARU"



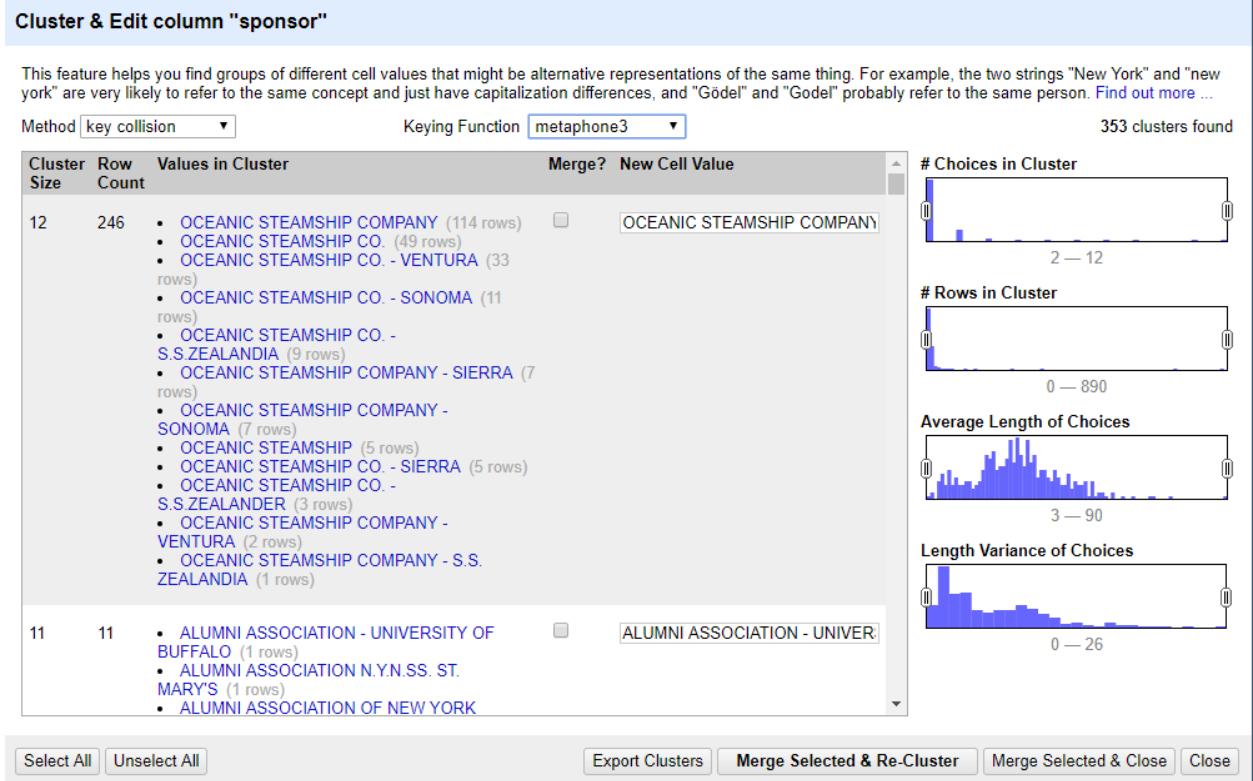
3. Clustering operation with keying function as “cologne-phonetic”. For example, this would replace the following values (using some common-sense and logic to make sure that the values made sense for the scenario):

"THE LOTOS CLUB",  
 "THE LOTUS CLUB"

with "THE LOTUS CLUB"



4. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
5. Replace special characters "[,],(,),\*,%,," with "".
6. We **did not choose** the clustering operation with keying function as “metaphone3” as it results in losing the existing organization breakdown per state or per county.

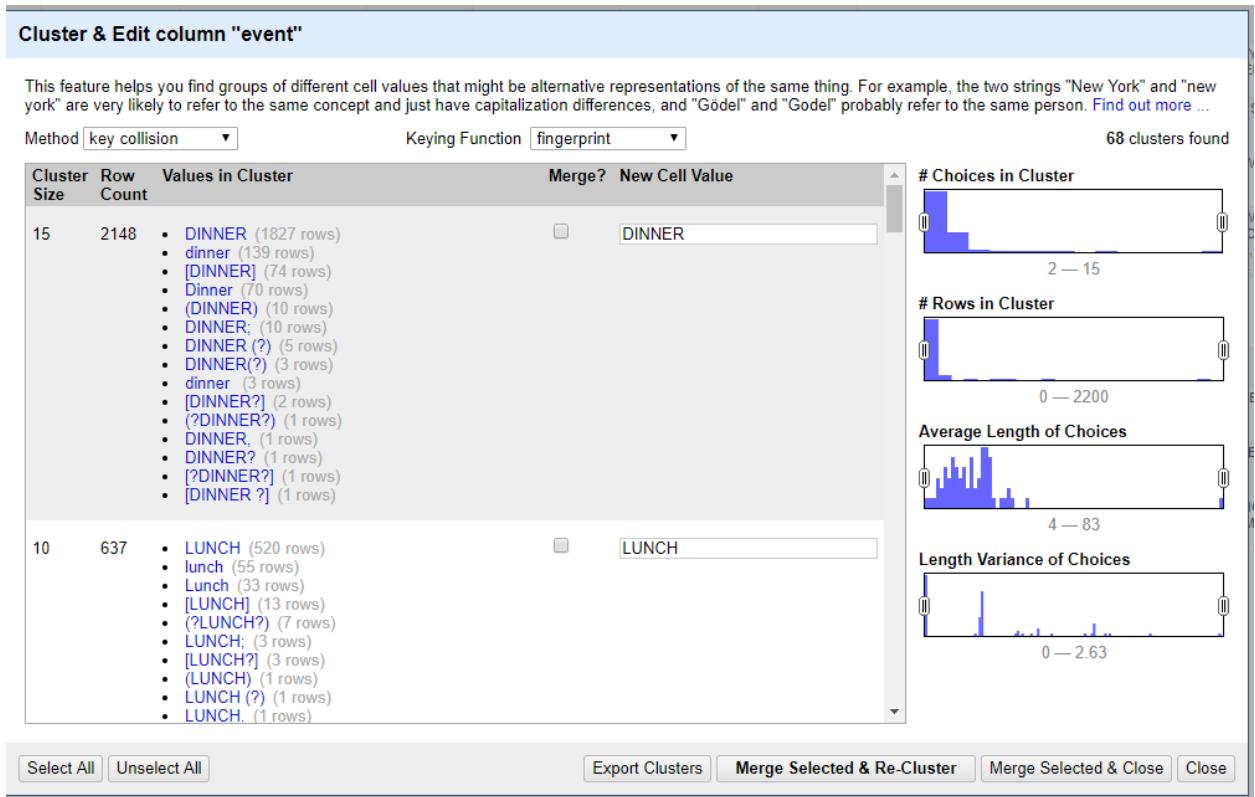


### 3.2.2 Transformations on “event” column

1. Clustering operation with keying function as “fingerprint”. For example, this would replace the following values:

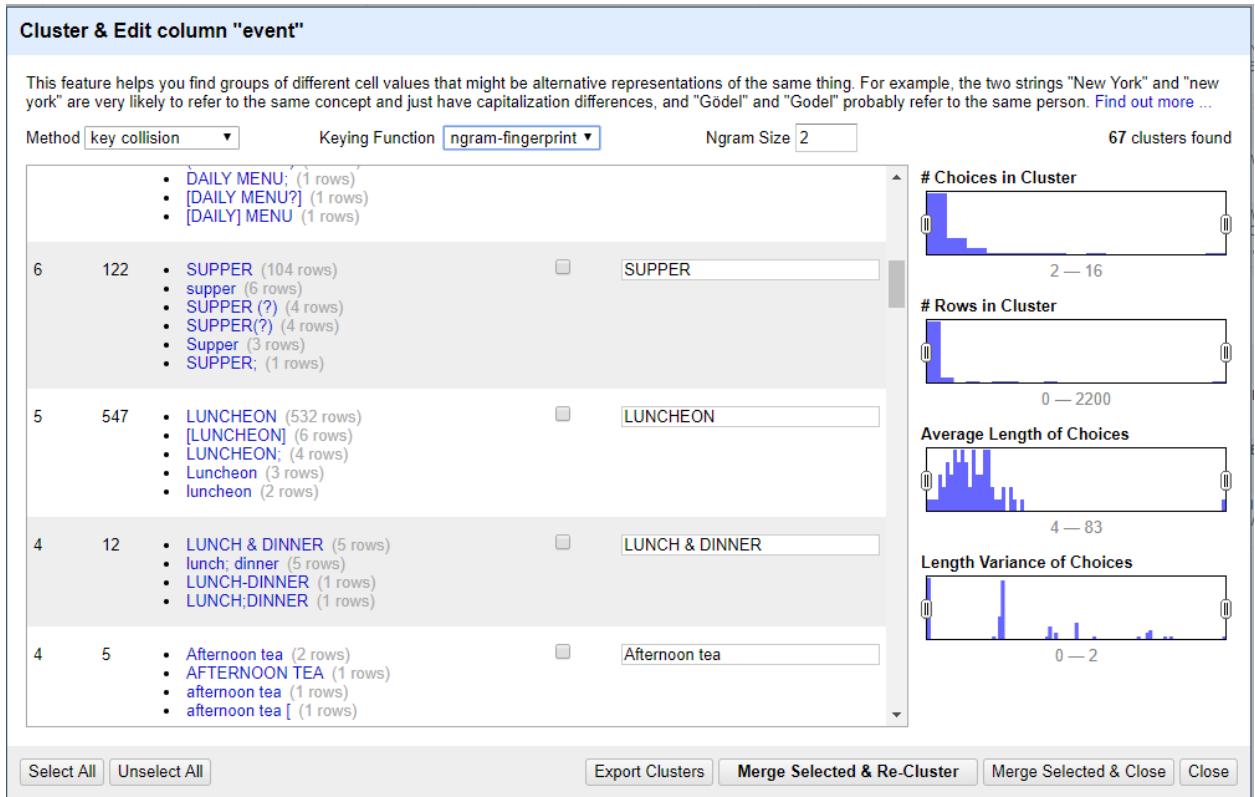
```
"BREAKFAST",
"Breakfast",
"breakfast",
"[BREAKFAST]",
"BREAKFAST;",
"BREAKFAST(?)",
"BREAKFAST(?)",
"[BREAKFAST ?]"
```

with "BREAKFAST"



- Clustering operation with keying function as "ngram-fingerprint" with ngram size as 2. For example, this would replace the following values:  
 "DINNER/DINER",  
 "DINER/DINNER",  
 "DINNER, DINER"

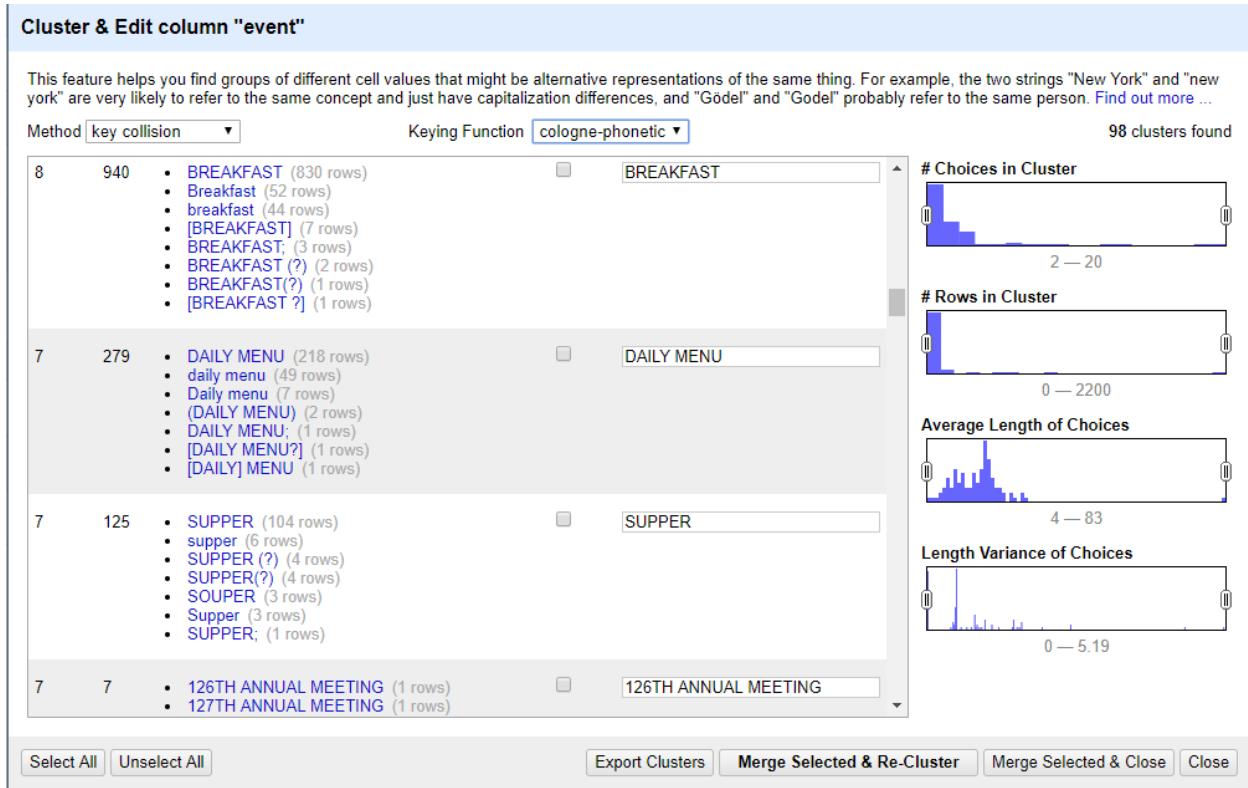
with "DINNER"



3. Clustering operation with keying function as “cologne-phonetic”. For example, this would replace the following values (using some common-sense and logic to make sure that the values made sense for the scenario):

"LUNCH AND DINNER",  
 "LUMCH AND DINNER"

with "LUNCH AND DINNER"



4. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
5. Replace special characters "[,],(,),\*,%,," with "".
6. Clustering operation with keying function as “metaphone3” to merge some of obvious ones, for example:

```

"11TH ANNUAL DINNER",
"15TH ANNUAL DINNER",
"4TH ANNUAL DINNER",
"14TH ANNUAL DINNER",
"18TH ANNUAL DINNER",
"20TH ANNUAL DINNER",
"12TH ANNUAL DINNER",
"13TH ANNUAL DINNER",
"16TH ANNUAL DINNER",
"25TH ANNUAL DINNER",
"28TH ANNUAL DINNER",
"30TH ANNUAL DINNER",
"34TH ANNUAL DINNER",
"37TH ANNUAL DINNER",
"57TH ANNUAL DINNER",
"5TH ANNUAL DINNER",
"7TH ANNUAL DINNER",

```

were replaced with "ANNUAL DINNER"

Some of the recommendations from OpenRefine were not taken as they did not make sense to our use case.

**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 metaphone3 ▾ 223 clusters found

Count	Value
21	BANQUET IN HONOR OF THE THIRD ANNUAL CONVENTION OF ABOVE (2 rows)
22	<ul style="list-style-type: none"> <li>• BANQUET IN HONOR OF ABOVE AND OTHR MEMBERS OF THE BOARD OF HEALTH (1 rows)</li> <li>• BANQUET IN HONOR OF CHARLES W. FAIRBANKS, VICE PRESIDENT OF U.S. (1 rows)</li> <li>• BANQUET IN HONOR OF CONGRESSMEN FROM GREATER NEW YORK (1 rows)</li> <li>• BANQUET IN HONOR OF COUNT FERDINAND DE LESSUPS (1 rows)</li> <li>• BANQUET IN HONOR OF FERDINAND W. PECK (1 rows)</li> <li>• BANQUET IN HONOR OF GRAND LODGE OFFICERS (1 rows)</li> <li>• BANQUET IN HONOR OF HOO HOO (1 rows)</li> <li>• BANQUET IN HONOR OF JUDGE PETER S. GROSSCUP &amp; JUDGE CHRISTIAN C. KOHLSAAT (1 rows)</li> <li>• BANQUET IN HONOR OF K. ISHII (1 rows)</li> <li>• BANQUET IN HONOR OF KAISER WILHELM II (1 rows)</li> </ul>

# Choices in Cluster  
2 — 40

# Rows in Cluster  
0 — 2200

Average Length of Choices  
0 — 120

Length Variance of Choices  
0 — 77

Select All Unselect All Export Clusters Merge Selected & Re-Cluster Merge Selected & Close Close

7. Clustering operation with Method as “nearest neighbor”, distance function as “levenshtein”, Radius=1.0, Block Chars=6

**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	nearest neighbor	Distance Function	levenshtein	Radius	1.0	Block Chars	6	6 clusters found
Cluster Size	Row Count	Values in Cluster			Merge?	New Cell Value	# Rows in Cluster	
2	3	<ul style="list-style-type: none"> <li>DINNER TO PRINCE AND PRINCESS OF WALES (2 rows)</li> <li>DINNER O PRINCE AND PRINCESS OF WALES (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	DINNER TO PRINCE AND PRINCE	 0 — 290	
2	6	<ul style="list-style-type: none"> <li>TABLE D'HOTE (5 rows)</li> <li>TABLE D'HOTER (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	TABLE D'HOTE	 10 — 38	
2	38	<ul style="list-style-type: none"> <li>MITTAGESSEN (37 rows)</li> <li>MITTAGESSES (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	MITTAGESSEN	 0 — 0.5	
2	11	<ul style="list-style-type: none"> <li>FOURTH ANNUAL DINNER (10 rows)</li> <li>FOURTH ANNUSL DINNER (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	ANNUAL DINNER		
2	283	<ul style="list-style-type: none"> <li>DAILY MENU (282 rows)</li> <li>DAILY MENUS (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	DAILY MENU		
2	51	<ul style="list-style-type: none"> <li>CHRISTMAS DINNER (50 rows)</li> <li>CHRISTMAN DINNER (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	CHRISTMAS DINNER		

Buttons: Select All | Unselect All | Export Clusters | Merge Selected & Re-Cluster | Merge Selected & Close | Close

8. Clustering operation with Method as “nearest neighbor”, distance function as” PPM”, Radius=1.0, Block Chars=6

**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	nearest neighbor	Distance Function	PPM	Radius	1.0	Block Chars	6	52 clusters found
Cluster Size	Row Count	Values in Cluster			Merge?	New Cell Value	# Choices in Cluster	
3	3	<ul style="list-style-type: none"> <li>THIRTEENTH ANNUAL REUNION AND DINNER (1 rows)</li> <li>17TH ANNUAL REUNION AND DINNER (1 rows)</li> <li>FIFTEENTH ANNUAL REUNION AND DINNER (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	ANNUAL REUNION AND DINNER	 2 — 4	
3	158	<ul style="list-style-type: none"> <li>ANNUAL BANQUET (146 rows)</li> <li>THIRD ANNUAL BANQUET (11 rows)</li> <li>3D ANNUAL BANQUET (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	ANNUAL BANQUET	 0 — 160	
3	14	<ul style="list-style-type: none"> <li>FIFTH ANNUAL BANQUET (7 rows)</li> <li>TWELFTH ANNUAL BANQUET (6 rows)</li> <li>TWENTY-FIFTH ANNUAL BANQUET (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	ANNUAL BANQUET	 11 — 60	
3	8	<ul style="list-style-type: none"> <li>TABLE D'HOTE (6 rows)</li> <li>TABLE D'HOTE MENU (1 rows)</li> <li>TABLE D'HOTE MEAL (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	TABLE D'HOTE	 0 — 5	
2	3	<ul style="list-style-type: none"> <li>LUNCHEON TO HOTEL MEN'S MUTUAL BENEFIT ASSOCIATION (2 rows)</li> <li>COLLATION TO HOTEL MEN'S MUTUAL BENEFIT ASSOCIATION (1 rows)</li> </ul>			<input checked="" type="checkbox"/>	LUNCHEON		

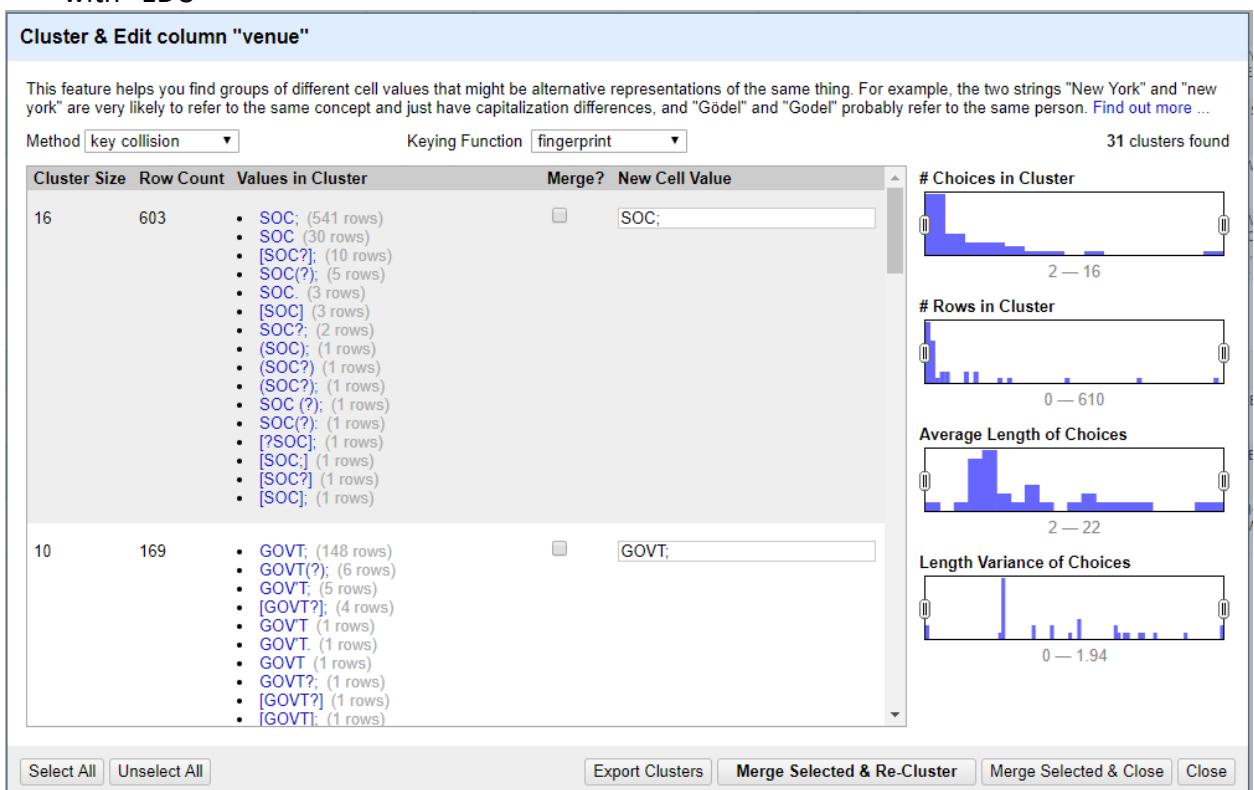
Buttons: Select All | Unselect All | Export Clusters | Merge Selected & Re-Cluster | Merge Selected & Close | Close

### 3.2.3 Transformations on “venue” column

1. Clustering operation with keying function as “fingerprint”. For example, this would replace the following values:

```
"EDUC;",
"EDUC",
"EDUC (?);",
"[EDUC];",
"(EDUC);",
"[EDUC?];"
```

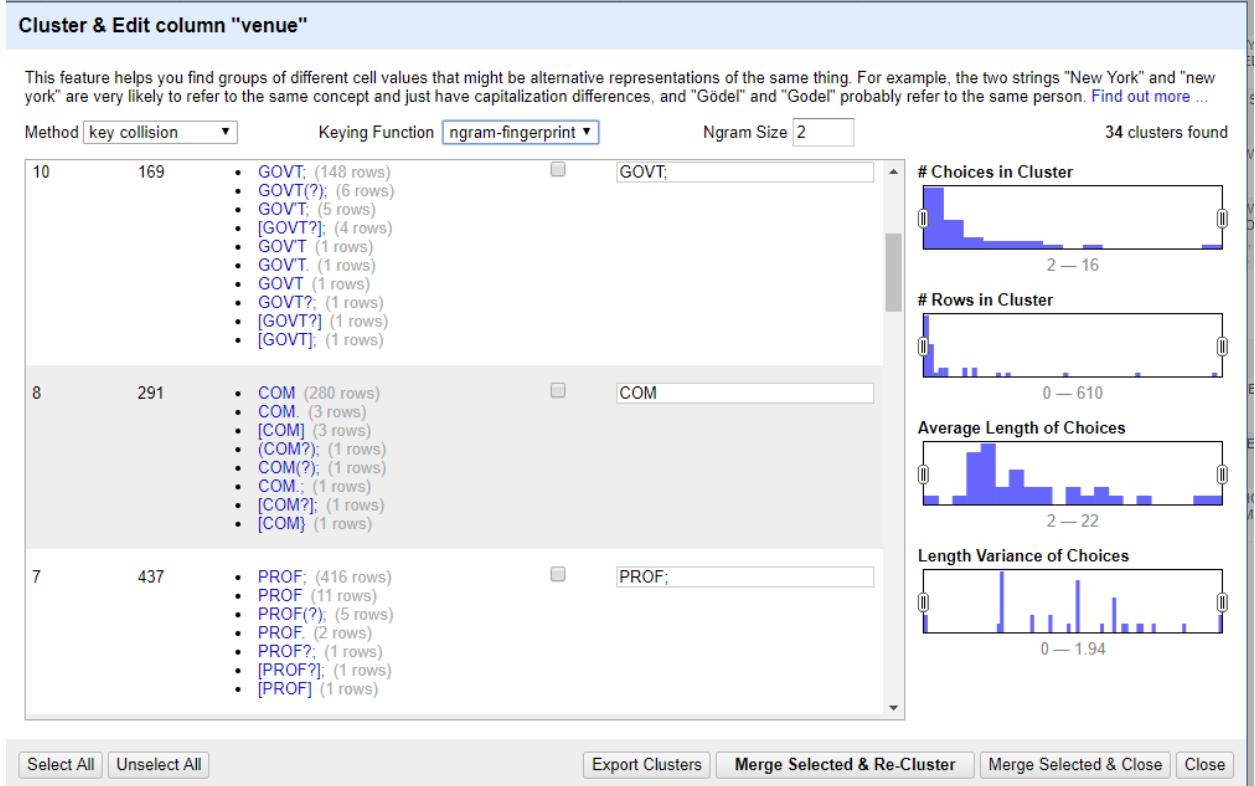
with "EDU"



2. Clustering operation with keying function as “ngram-fingerprint” with ngram size as 2. For example, this would replace the following values:

```
"SECULAR HOLIDAY;",
"SECULAR HOLIDAY",
"SECULAR HOLIDAY. HOLIDAY",
"SECULAR HOLIDAY. HOLIDAY;",
"SECULAR HOLIDAY HOLIDAY;",
"SECULAR HOLIDAY HOLIDAY"
```

with "SECULAR HOLIDAY"



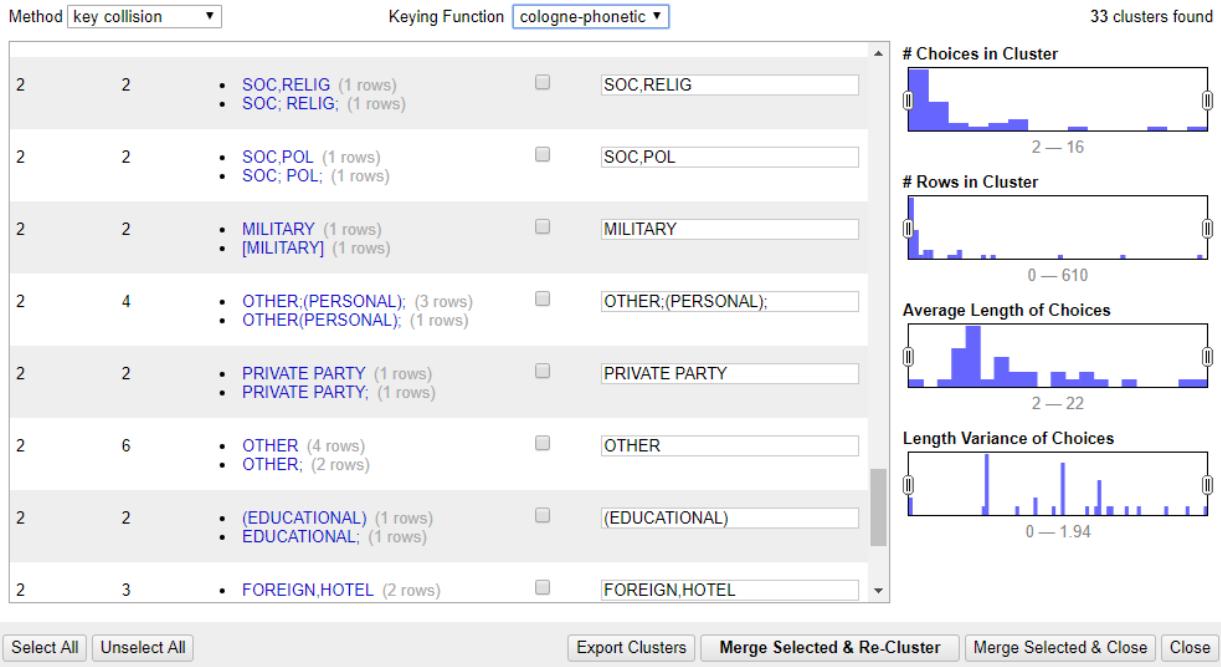
3. Clustering operation with keying function as “colgne-phonetic”. For example, this would replace the following values (using some common-sense and logic to make sure that the values made sense for the scenario):

"DINNER A LA CARTE",  
 "DINNER-A LA CARTE"

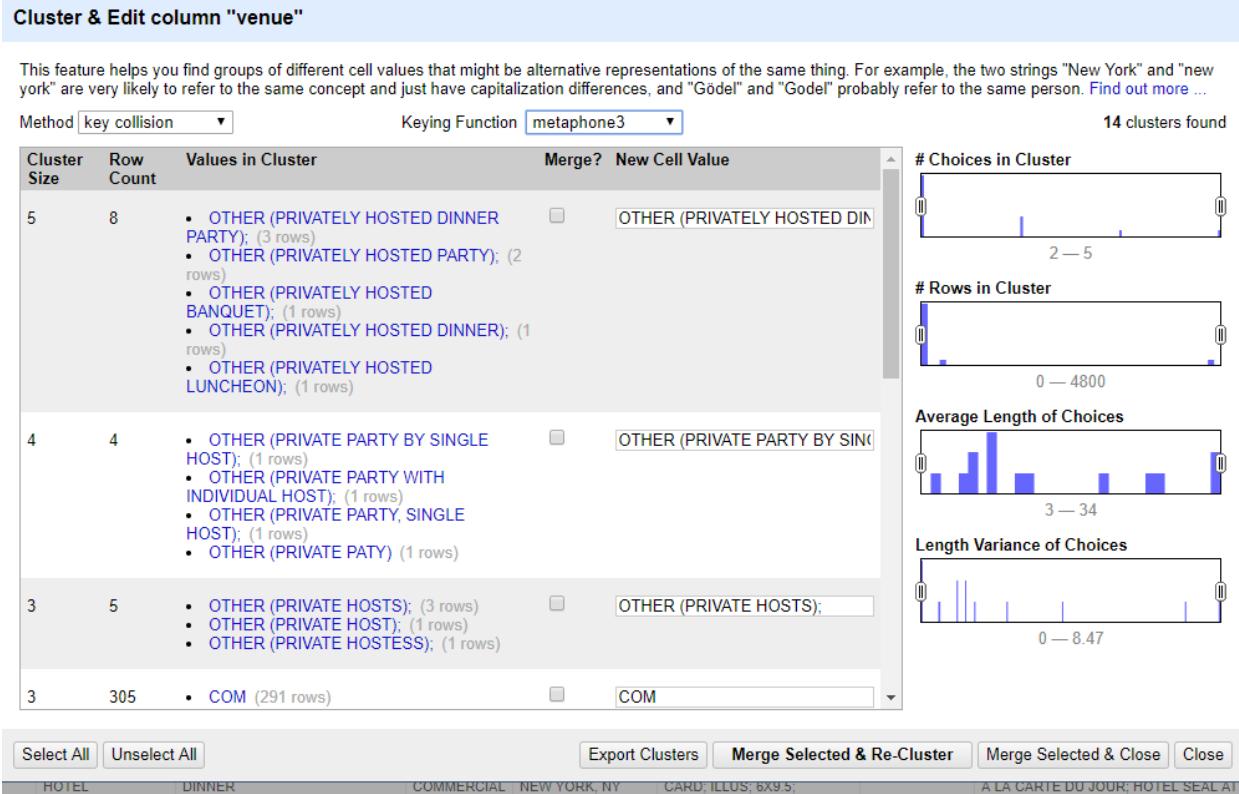
with "DINNER A LA CARTE"

### Cluster & Edit column "venue"

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 ...](#)



4. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
5. Replace special characters "[,],(,),\*,%,," with "".
6. We did not choose the clustering operation with keying function as “metaphone3” as it results in losing the existing event breakdown.

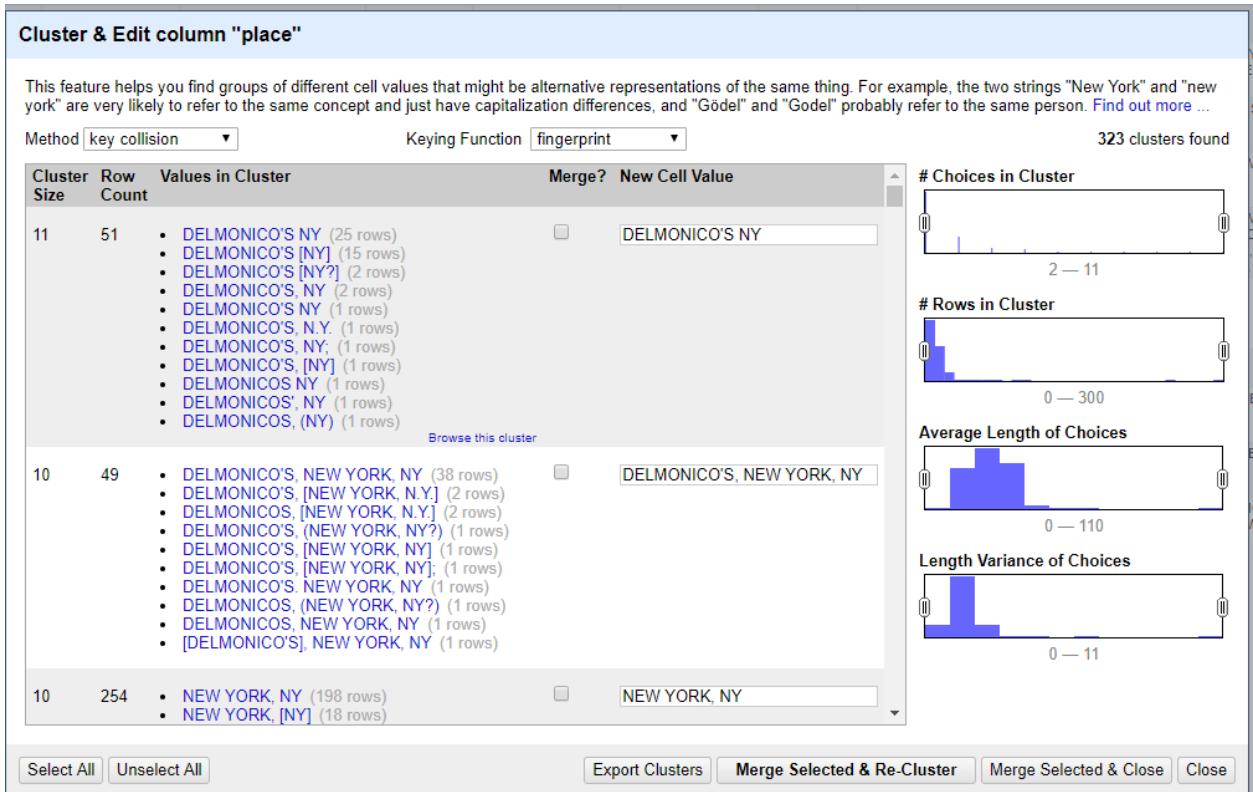


### 3.2.4 Transformation on “place” column

1. Clustering operation with keying function as “fingerprint”. For example, this would replace the following values:

"DELMONICO'S NY",  
 "DELMONICO'S [NY]",  
 "DELMONICO'S [NY?]",  
 "DELMONICO'S, NY",  
 "DELMONICO'S NY",  
 "DELMONICO'S, N.Y.",  
 "DELMONICO'S, NY;",  
 "DELMONICO'S, [NY]",  
 "DELMONICOS NY",  
 "DELMONICOS', NY",  
 "DELMONICOS, (NY)"

with "DELMONICO'S, NEW YORK, NY"

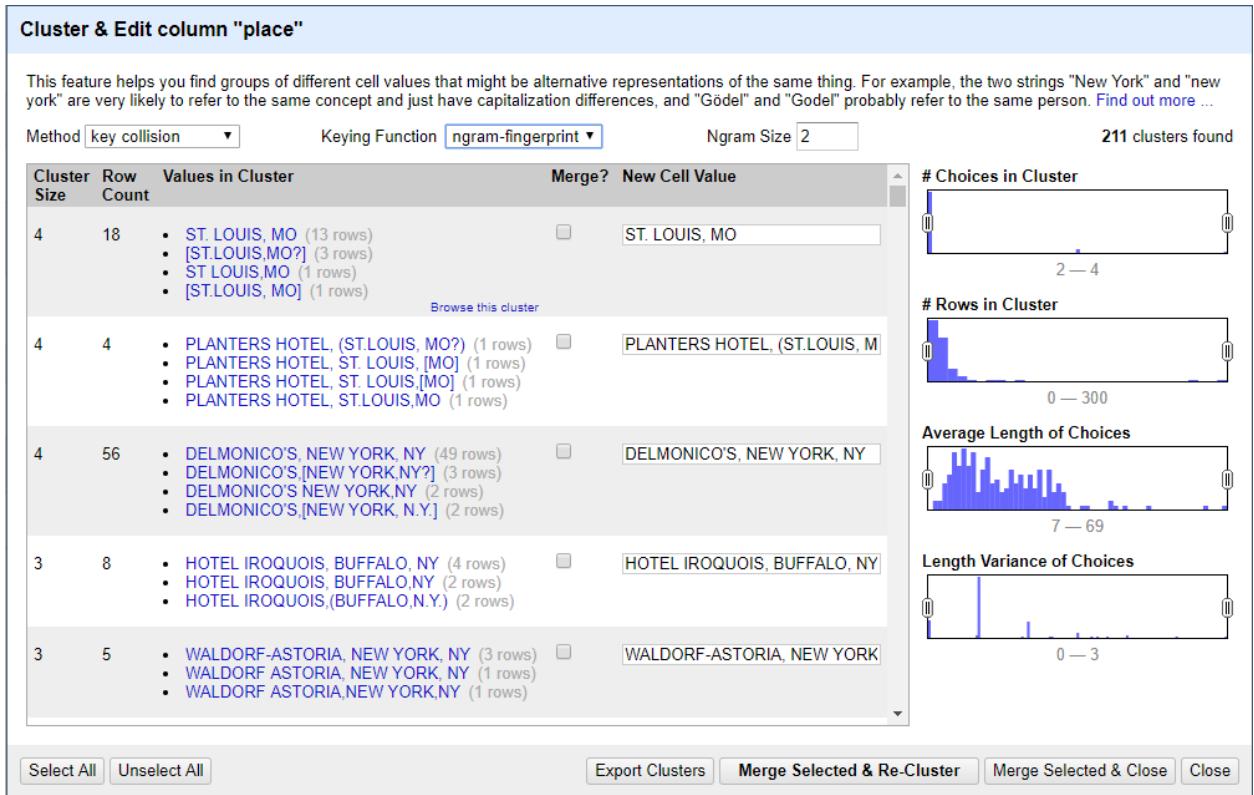


## 2. Clustering operation with keying function as “ngram-fingerprint” with ngram size as 2.

For example, this would replace the following values:

"PLANTERS HOTEL, (ST.LOUIS, MO?)",  
 "PLANTERS HOTEL, ST. LOUIS, [MO]",  
 "PLANTERS HOTEL, ST. LOUIS,[MO]",  
 "PLANTERS HOTEL, ST.LOUIS,MO"

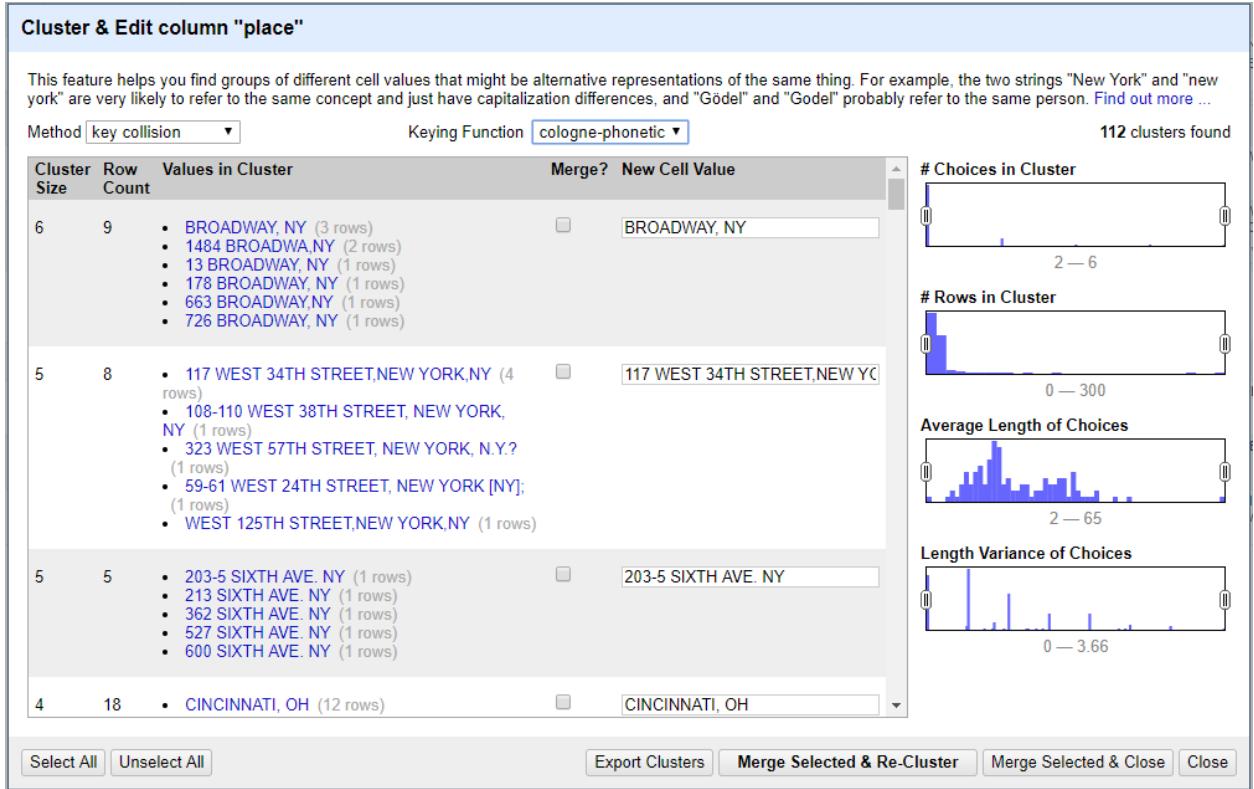
with "PLANTERS HOTEL, ST.LOUIS, MO"



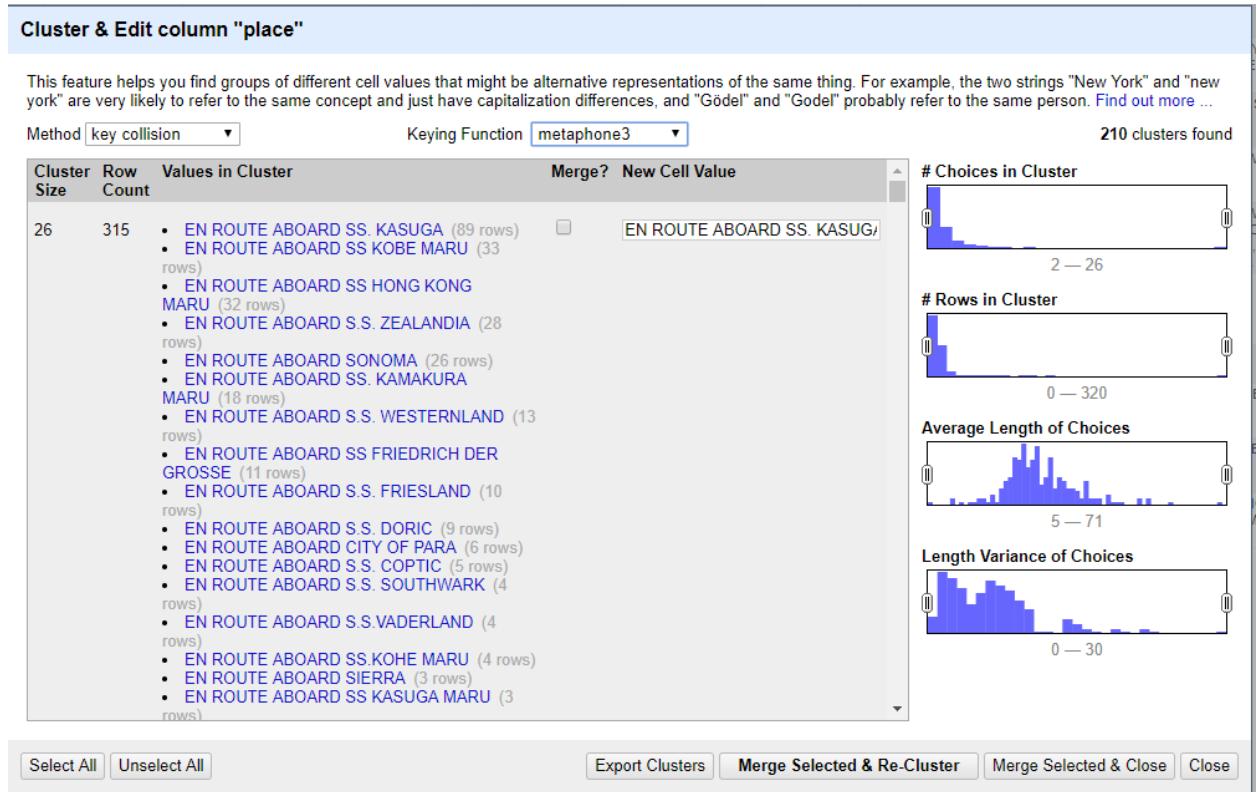
3. Clustering operation with keying function as “cologne-phonetic”. For example, this would replace the following values (using some common-sense and logic to make sure that the values made sense for the scenario):

"SUMMERTVILLE, S.C.",  
 "SOMMERTVILLE, S.C.",  
 "SUMERVILLE,S.C."

with "SUMMERTVILLE, SC"



4. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
5. Replace special characters with this transformation:  
`value.replace(/[%@#!\?\\(\)\[\]]+/,"").replace('*','').replace('','','')`
6. We did not choose the clustering operation with keying function as “metaphone3” as it results in losing the existing organization breakdown per state or per county.



### 3.2.5 Transformations on “physical\_description” column

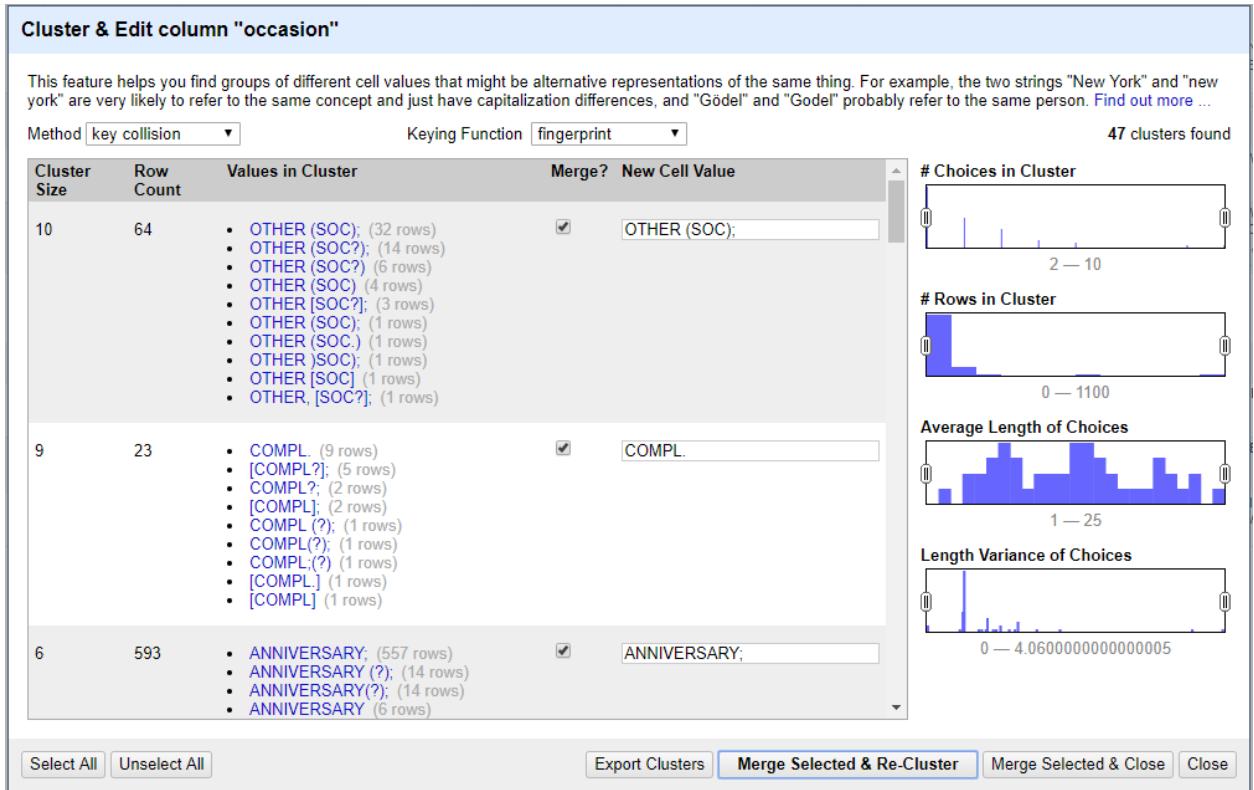
1. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.

### 3.2.6 Transformations on “occasion” column

1. Clustering operation with keying function as “fingerprint”. For example, this would replace the following values:

```
"ANNIVERSARY;",
"ANNIVERSARY (?);",
"ANNIVERSARY(?)",
"ANNIVERSARY",
"ANNIVERSARY.",
"ANNIVERSARY?"
```

with "ANNIVERSARY"



2. Clustering operation with keying function as “ngram-fingerprint” with ngram size as 2.

For example, this would replace the following values:

"RELIG. HOLIDAY",  
 "RELIG. HOLIDAY;",  
 "RELIG HOLIDAY;"

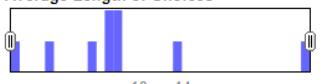
with "RELIGIOUS HOLIDAY"

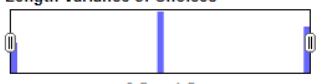
**Cluster & Edit column "occasion"**

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	9 clusters found
Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value		
2	2	<ul style="list-style-type: none"> <li>(COMPL;FOR MAHARAJA SCINDIYA (1 rows)</li> <li>COMPL; FOR MAHARAJA SCINDIYA (1 rows)</li> </ul>	<input type="checkbox"/>	(COMPL;FOR MAHARAJA SCINDI		
2	6	<ul style="list-style-type: none"> <li>OTHER (ANNUAL EVENT) (4 rows)</li> <li>OTHER,ANNUAL EVENT (2 rows)</li> </ul>	<input type="checkbox"/>	OTHER (ANNUAL EVENT)		
2	243	<ul style="list-style-type: none"> <li>DAILY MENU; (242 rows)</li> <li>DAILYMENU (1 rows)</li> </ul>	<input type="checkbox"/>	DAILY MENU;		
2	2	<ul style="list-style-type: none"> <li>OTHER (COMMEMORATION); (1 rows)</li> <li>OTHER,Commemoration (1 rows)</li> </ul>	<input type="checkbox"/>	OTHER (COMMEMORATION);		
2	2	<ul style="list-style-type: none"> <li>OTHER (ANNUAL BANQUET); (1 rows)</li> <li>OTHER,Annual Banquet (1 rows)</li> </ul>	<input type="checkbox"/>	OTHER (ANNUAL BANQUET);		Browse this cluster
2	3	<ul style="list-style-type: none"> <li>OTHER (ANNUAL MEETING); (2 rows)</li> <li>OTHER,Annual Meeting (1 rows)</li> </ul>	<input type="checkbox"/>	OTHER (ANNUAL MEETING);		
2	10	<ul style="list-style-type: none"> <li>RELIG. HOLIDAY (9 rows)</li> <li>RELIG HOLIDAY (4 rows)</li> </ul>	<input type="checkbox"/>	RELIG. HOLIDAY		

# Rows in Cluster  
  
 0 — 250

Average Length of Choices  
  
 10 — 44

Length Variance of Choices  
  
 0.5 — 1.5

Select All | Unselect All | Export Clusters | Merge Selected & Re-Cluster | Merge Selected & Close | Close

3. Clustering operation with keying function as “cologne-phonetic”. For example, this would replace the following values (using some common-sense and logic to make sure that the values made sense for the scenario):

"OTHER (COMMEMORATIVE)",  
 "OTHER [COMMEMORTIVE?]"

with "OTHER (COMMEMORATIVE)"

### Cluster & Edit column "occasion"

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 cologne-phonetic ▾

7 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
3	66	<ul style="list-style-type: none"> <li>• OTHER (SOC); (64 rows)</li> <li>• OTHER (SOC?); (1 rows)</li> <li>• OTHR (SOC); (1 rows)</li> </ul>	<input type="checkbox"/>	OTHER (SOC);
3	595	<ul style="list-style-type: none"> <li>• ANNIVERSARY; (593 rows)</li> <li>• 113 ANNIVERSARY; (1 rows)</li> <li>• ANIVERSARY; (1 rows)</li> </ul>	<input type="checkbox"/>	ANNIVERSARY;
2	31	<ul style="list-style-type: none"> <li>• [ANNIV?]; (26 rows)</li> <li>• ANIV; (5 rows)</li> </ul>	<input type="checkbox"/>	[ANNIV?];
2	7	<ul style="list-style-type: none"> <li>• OTHER (COMMEMORATIVE); (6 rows)</li> <li>• OTHER [COMMEMORTIVE?]; (1 rows)</li> </ul>	<input type="checkbox"/>	OTHER (COMMEMORATIVE);
2	98	<ul style="list-style-type: none"> <li>• ANNUAL; (97 rows)</li> <li>• AMNNUAL; (1 rows)</li> </ul>	<input type="checkbox"/>	ANNUAL
2	7	<ul style="list-style-type: none"> <li>• OTHER (DAILY HOTEL MENU); (6 rows)</li> <li>• OTHEER (DAILY HOTEL MENU); (1 rows)</li> </ul>	<input type="checkbox"/>	OTHER (DAILY HOTEL MENU);

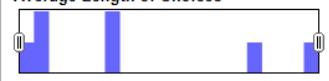
# Choices in Cluster



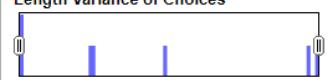
# Rows in Cluster



Average Length of Choices



Length Variance of Choices

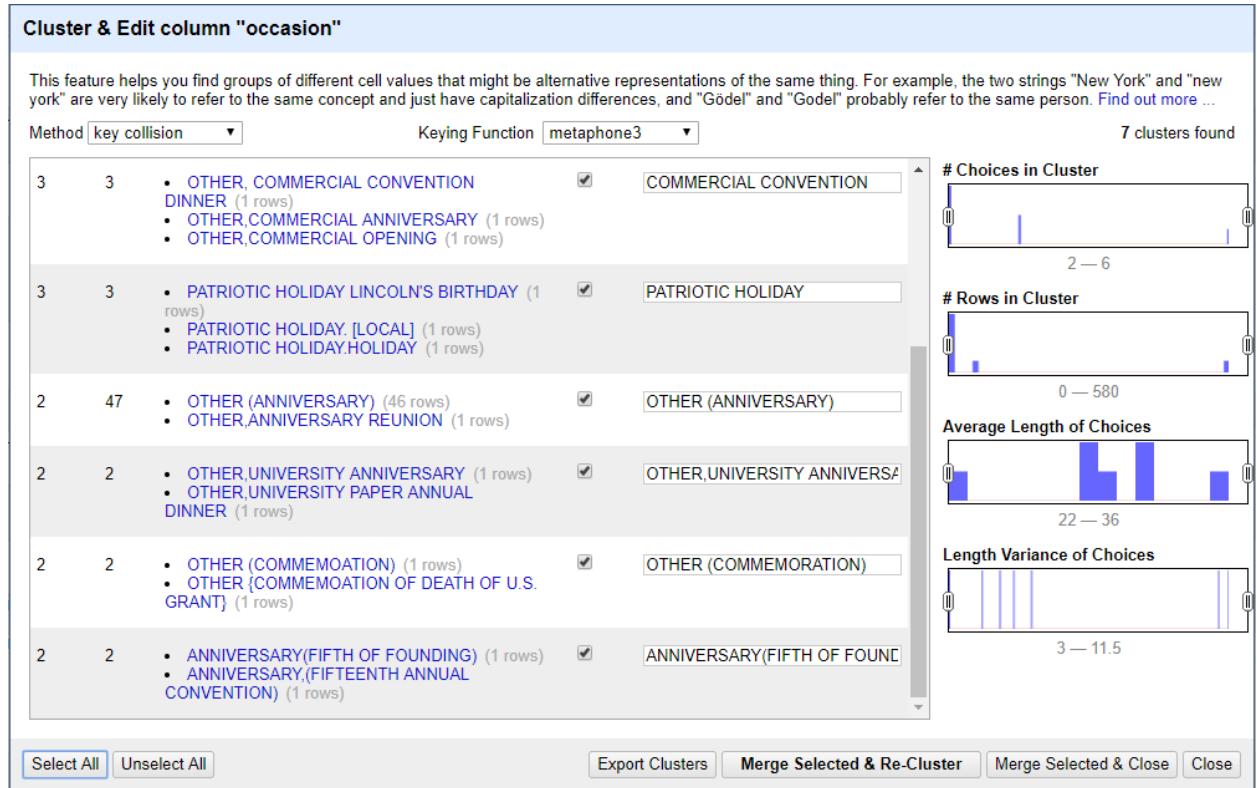


Select All Unselect All

Export Clusters Merge Selected & Re-Cluster Merge Selected & Close Close

4. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
5. Replace special characters "[,],(,),\*,%,," with "".

## 6. Clustering operation with keying function as “metaphone3”



### 3.2.7 Transformation on “notes” column

1. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
2. Replace special characters "[,),(,\*%,," with "".

### 3.2.8 Transformation on “call\_number” column

1. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.

### 3.2.9 Transformation on “date” column

1. Convert column type to “date”.
2. Use transformation “value.toString("MM/dd/yyyy")” to keep MM/DD/YYYY standard on the date column.
3. Convert column type to “text”.

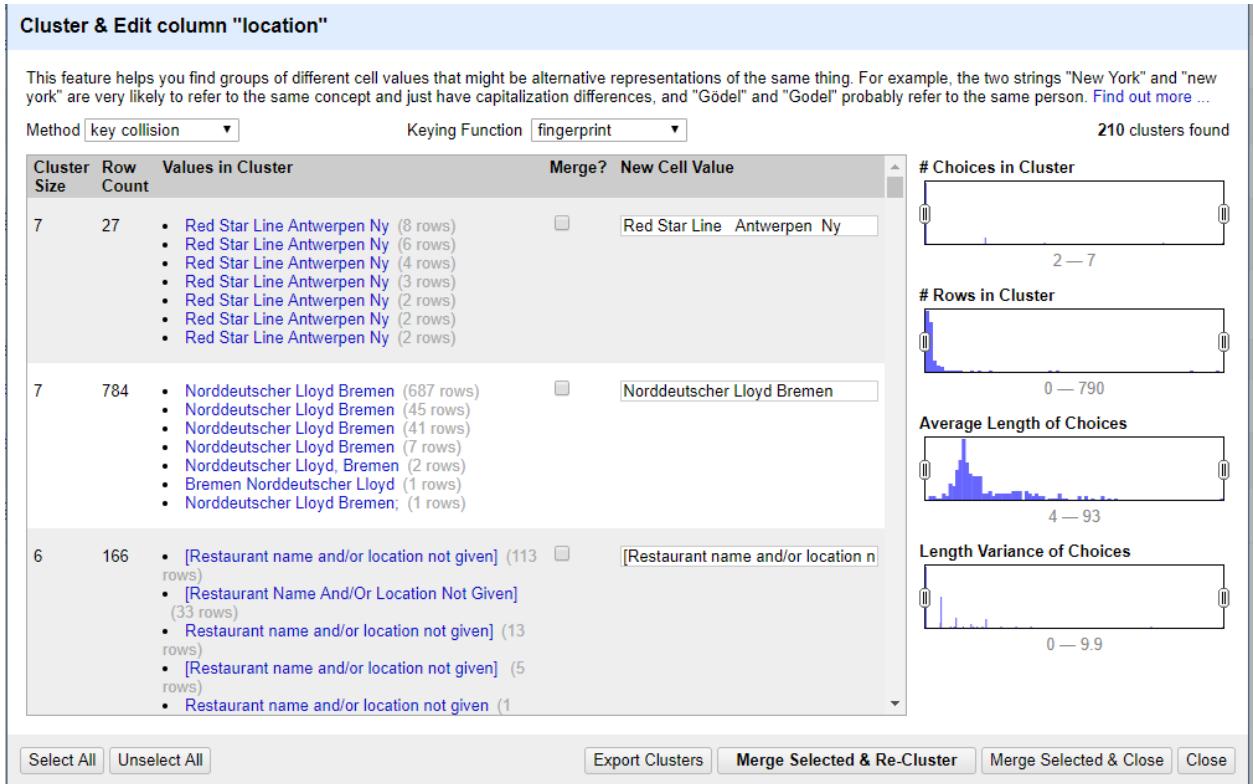
17545 rows														Extensions:	
Show as: rows	records	Show: 5 10 25 50 rows												< first < previous 1 - 50 next > last >	
event	venue	place	physical_description	occasion	notes	call_number	keywords	language	date	location	location_type	currency	currency_symbol		
BREAKFAST	COMMERCIAL	HOT SPRINGS, AR	CARD: 4.75X7.5;	EASTER;		1900-2822			04/15/1900	Hotel Eastman					
[DINNER]	COMMERCIAL	MILWAUKEE, WI	CARD: ILLUS: COL: 7.0X9.0;	EASTER;	WEDGWOOD BLUE CARD, WHITE EMBOSSED GREEK KEY BORDER, "EASTER SUNDAY" EMBOSSED IN WHITE, VIOLET COLORED SPRAY OF FLOWERS IN UPPER LEFT CORNER;	1900-2825			04/15/1900	Republican House					
FRUHSTUCK/BREAKFAST;	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE	CARD: ILLU: COL: 5.5X8.0;		MENU IN GERMAN AND ENGLISH, ILLUS: STEAMSHIP AND SAILING VESSEL.	1900-2827			04/16/1900	Norddeutscher Lloyd Bremen					
LUNCH:	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE	CARD: ILLU: COL: 5.5X8.0;		MENU IN GERMAN AND ENGLISH, ILLUS: HARBOR SCENE WITH SAILING VESSEL;	1900-2828			04/16/1900	Norddeutscher Lloyd Bremen					
DINNER;	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE;	FOLDER: ILLU: COL: 5.5X7.5;		MENU IN GERMAN AND ENGLISH, ILLUS: HARBOR SCENE WITH ROCKS AND LIGHTHOUSE, STEAMSHIP AND SAILING VESSEL, CONCERT PROGRAM; NOTES: ON GERMAN SIDE OF MENU: "MONTAG, DEN 16 APRIL 1900"; ON ENGLISH SIDE OF MENU: "MONDAY, APRIL 15TH, 1900".	1900-2829			04/16/1900	Norddeutscher Lloyd Bremen					
[DINNER]	COMMERCIAL	R.M.S. "EMPEROR OF CHINA"	CARD: ILLUS: COL: 4.75X7.25;		ILLUS: RED AND WHITE CHECKERED FLAG;	1900-2831			04/16/1900	Canadian Pacific Railway Company					
SUPPER	COMMERCIAL	NEW YORK, NY	CARD: ILLUS: COL: 6.0X8.75;		HOTEL CREST IN BLUE; PRICED MENU;	1900-2838			04/16/1900	Hotel Newland		Dollars	\$		
FRUHSTUCK/BREAKFAST	COMMERCIAL	SCHNELL DAMPFER KAISER WILHELM DER GROSSE;	BROADSIDE: ILLUS: COL: 5.5X3.50;		MENU IN GERMAN AND ENGLISH, ILLUS: LIGHTHOUSE, STEAMSHIP, FISHING DORY;	1900-2839			04/17/1900	Norddeutscher Lloyd Bremen					
LUNCH	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE;	BROADSIDE: ILLUS: COL: 5.5X3.50;		MENU IN GERMAN AND ENGLISH, ILLUS: SAILING SHIPS/SHEETS FURLED; STEAMSHIP;	1900-2840			04/17/1900	Norddeutscher Lloyd Bremen					
[DINNER]	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE;	FOLDER: ILLU: COL: 5.5X7.5;		MENU IN GERMAN AND ENGLISH, ILLUS: HARBOR, LIGHTHOUSE, ROCKS, STEAMSHIP, SAILING SHIPS; CONCERT PROGRAM.	1900-2841			04/17/1900	Norddeutscher Lloyd Bremen					
CAFE LUNCHEON	COMMERCIAL	NEW YORK, NY	CARD: ILLUS: COL: 4.25X5.5;		HOTEL CREST IN BLUE;	1900-2843			04/17/1900	Hotel Marlborough					
ANNUAL BANQUET	COMMERCIAL	DELMONICO'S, NEW YORK, NY	BOOKLET: ILLUS: COL: 5.5X7.0;		VELLUM COVER: CREST OF ZETA PSI TIED WITH BLUE SATIN RIBBON, PRICED WINE LIST ALPHA CHAPTER OFFICERS; DECORATIVE BORDER; A LA CARTE DU JOUR; POEM: "TO ZETA PSI WE PLEDGE TO-NIGHT"; A POEM, "THE BADGE OF ZETA PSI"; DECORATIVE BORDER; NEATH THE ARCH OF ZETA'S TEMPLE, ANCHORS;	1900-2844			04/17/1900	Alpha of Zeta Psi		Dollars	\$		
DINNER	COMMERCIAL	NEW YORK, NY	CARD: ILLUS: 6X9.5;		A LA CARTE DU JOUR; HOTEL SEAL AT TOP OF MENU;	1900-2847			04/18/1900	Hotel Manhattan					
DINNE	COMMERCIAL	SS CITY OF PARA	FOLDER: ILLUS: 6X9.25;		DECORATIVE BORDER;	1900-2849			04/18/1900	Pacific Mail Steamship Company					
BREAKFAST	COMMERCIAL	SS DORIC	BROADSIDE: ILLUS: 5.5X8.5;		HANDWRITTEN: STEAMSHIP COMPANY	1900-2851			04/18/1900	Occidental & Oriental					

### 3.2.10 Transformation on "location" column

1. Clustering operation with keying function as "fingerprint". For example, this would replace the following values:

"Red Star Line Antwerpen Ny",  
 "Red Star Line Antwerpen Ny"

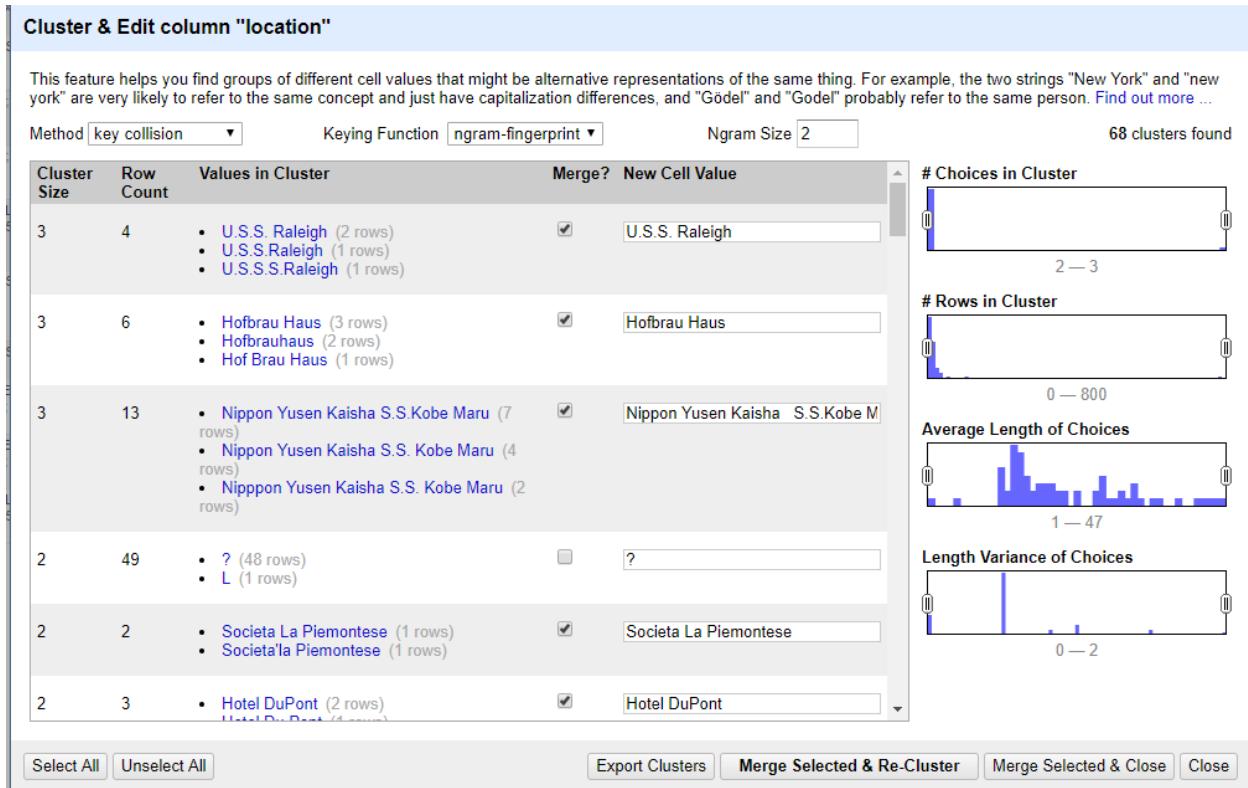
with "Red Star Line Antwerpen Ny"



4. Clustering operation with keying function as “ngram-fingerprint” with ngram size as 2.
- For example, this would replace the following values:

"U.S.S. Raleigh",  
 "U.S.S.Raleigh",  
 "U.S.S.S.Raleigh"

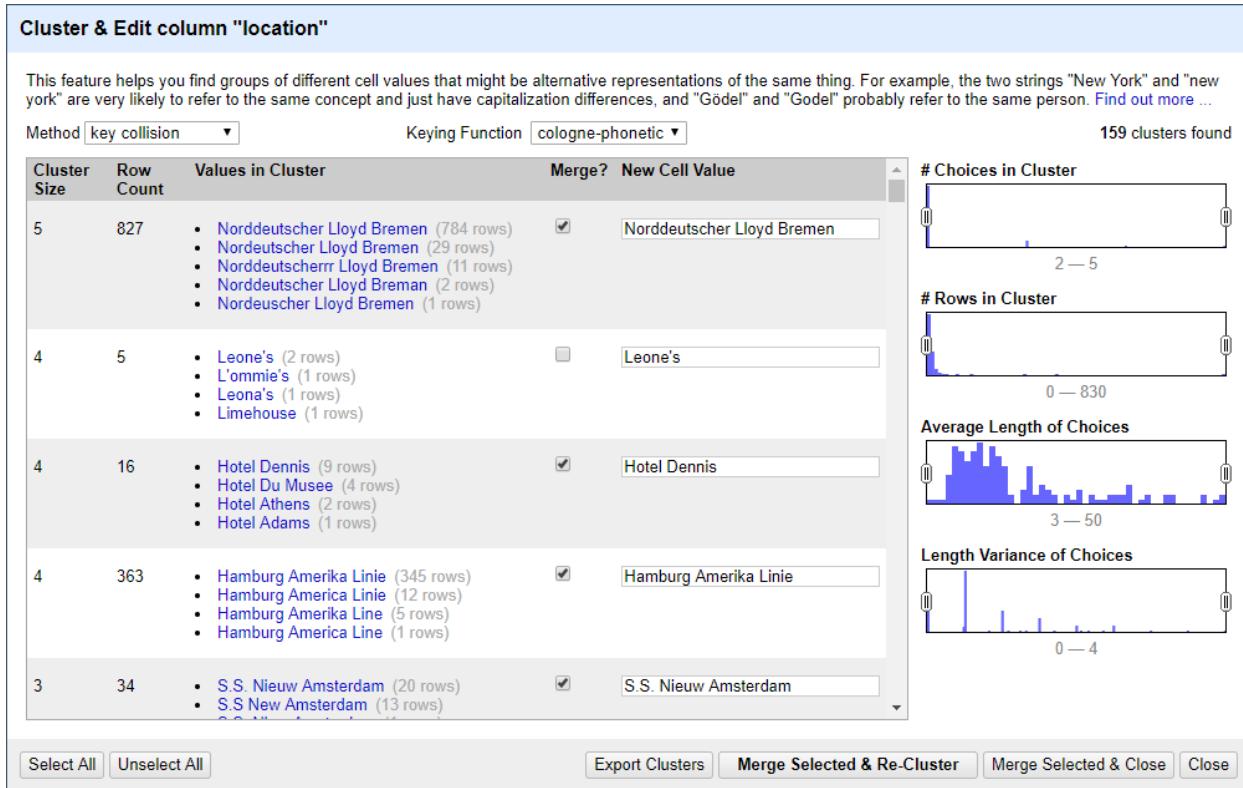
with "U.S.S. Raleigh"



5. Clustering operation with keying function as “cologne-phonetic”. For example, this would replace the following values (using some common-sense and logic to make sure that the values made sense for the scenario):

"Norddeutscher Lloyd Bremen",  
 "Nordeutscher Lloyd Bremen",  
 "Norddeutscherr Lloyd Bremen",  
 "Norddeutscher Lloyd Breman",  
 "Nordeuscher Lloyd Bremen"

with "Norddeutscher Lloyd Bremen"



6. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.
7. Replace special characters "[,],(,),\*,%,," with "".

### 3.2.11 Columns Deleted or Merged

1. The following columns were eliminated as they had 100% missing values:
  - a. name
  - b. language
  - c. keyword
  - d. location\_type
  - e. currency\_symbol
2. Location column was a duplicate of sponsor column and was empty in several cases. We merged sponsor and location with this transformation:  
`if(isNull(cells["sponsor"].value), cells["sponsor"].value, cells["location"].value)`
3. Deleted location column
4. Renamed sponsor to name.

### 3.2.12 Columns Untouched

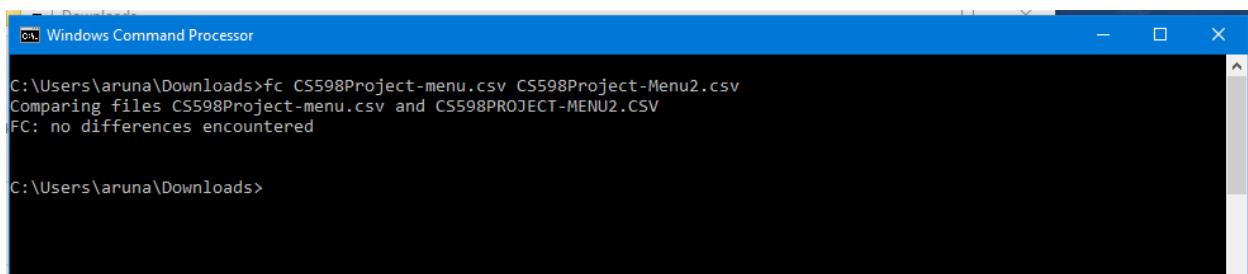
1. The following columns were not edited at all as they were deemed to contain valid values:

- a. id
- b. currency
- c. status
- d. page\_count
- e. dish\_count

Export CS598Project-menu.csv from OpenRefine

### 3.2.13 Operation History & Provenance

The entire operation history has been exported from OpenRefine as a json file (**menuOperationHis.json**, attached with project files). To check the provenance, we created a new project in OpenRefine with the original menu.csv and applied this menuOperationHis.json. This automatically executes all the transformations on all the columns. We exported the new csv and compared to the CS598Project-menu.csv that was exported previously using the windows file comparator and the two csv files were identical proving that the transformation are 100% replicable.



```
C:\Users\aruna\Downloads>fc CS598Project-menu.csv CS598Project-Menu2.csv
Comparing files CS598Project-menu.csv and CS598PROJECT-MENU2.CSV
FC: no differences encountered

C:\Users\aruna\Downloads>
```

## 3.3 Changes in Dish CSV

### 3.3.1 Transformation on “name” column

1. Clustering operation with keying function as “fingerprint”. For example, this would replace the following values:

```
" French Fried Potatoes",
" Potatoes French Fried",
"FRENCH FRIED POTATOES",
"French fried potatoes",
"French Fried (potatoes)",
"French Fried Potatoes",
"French Fried Potatoes.",
"French Fried [Potatoes]",
"French Fried potatoes",
With " French Fried Potatoes"
```

**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 "Godel" and "Godel" probably refer to the same person. [Find out more ...](#)

Method **key collision** Keying Function **fingerprint** 41334 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
48	48	<ul style="list-style-type: none"> <li>• FRIED SWEET POTATOES (1 rows)</li> <li>• Fried [Sweet Potatoes] (1 rows)</li> <li>• Fried [sweet potatoes] (1 rows)</li> <li>• Fried sweet Potatoes (1 rows)</li> <li>• Fried sweet [potatoes] (1 rows)</li> <li>• Fried sweet potatoes (1 rows)</li> <li>• POTATOES: Fried sweet (1 rows)</li> <li>• Potatoes SWEET Fried (1 rows)</li> <li>• Potatoes Fried Sweet (1 rows)</li> <li>• Potatoes Sweet Fried (1 rows)</li> <li>• Potatoes sweet_fried (1 rows)</li> <li>• Potatoes, Fried Sweet (1 rows)</li> <li>• Potatoes, Fried sweet (1 rows)</li> <li>• Potatoes, Sweet, Fried (1 rows)</li> <li>• Potatoes, fried sweet (1 rows)</li> <li>• Potatoes, sweet fried (1 rows)</li> <li>• SWEET POTATOES - Fried (1 rows)</li> <li>• SWEET POTATOES - fried (1 rows)</li> <li>• Sweet Potatoes, fried (1 rows)</li> <li>• Sweet (potatoes), fried (1 rows)</li> <li>• Sweet Fried Potatoes (1 rows)</li> <li>• Sweet Fried potatoes (1 rows)</li> <li>• Sweet Potatoes - Fried (1 rows)</li> <li>• Sweet Potatoes - fried (1 rows)</li> </ul>	<input type="checkbox"/>	FRIED SWEET POTATOES

# Choices in Cluster

2 — 48

# Rows in Cluster

2 — 48

Average Length of Choices

0 — 850

Length Variance of Choices

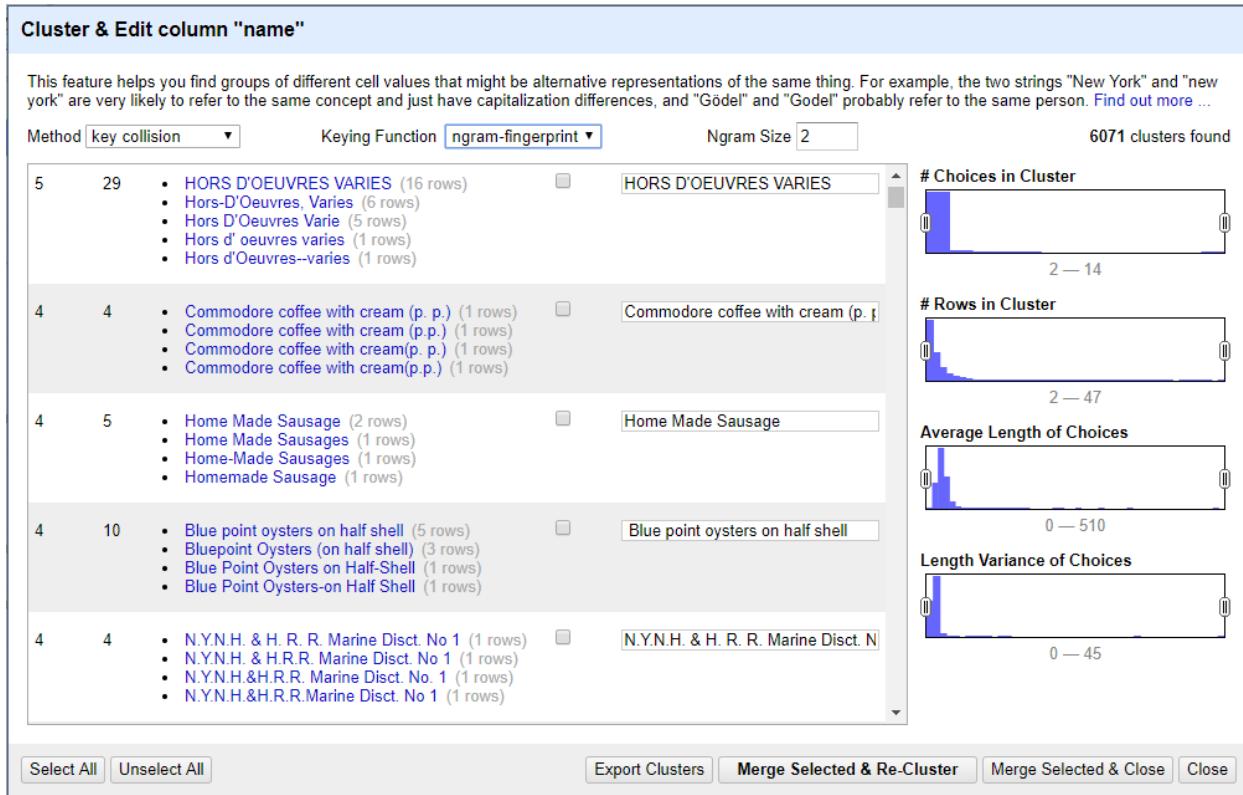
0 — 33

2. Clustering operation with keying function as “ngram-fingerprint” with ngram size as 2.

For example, this would replace the following values:

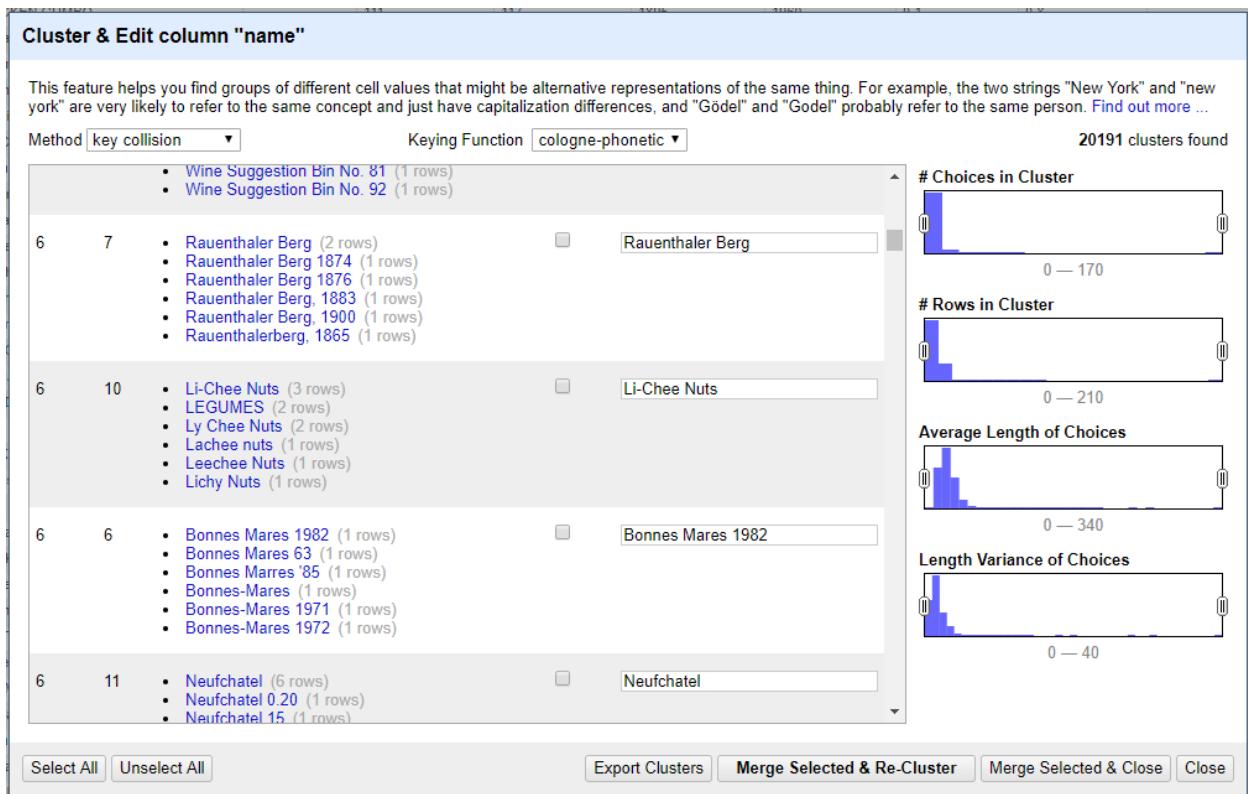
"Cream Potatoes",  
 "Cream potatoes",  
 "Cream, Potatoes",  
 "Potatoes & Cream"

with "Cream Potatoes"

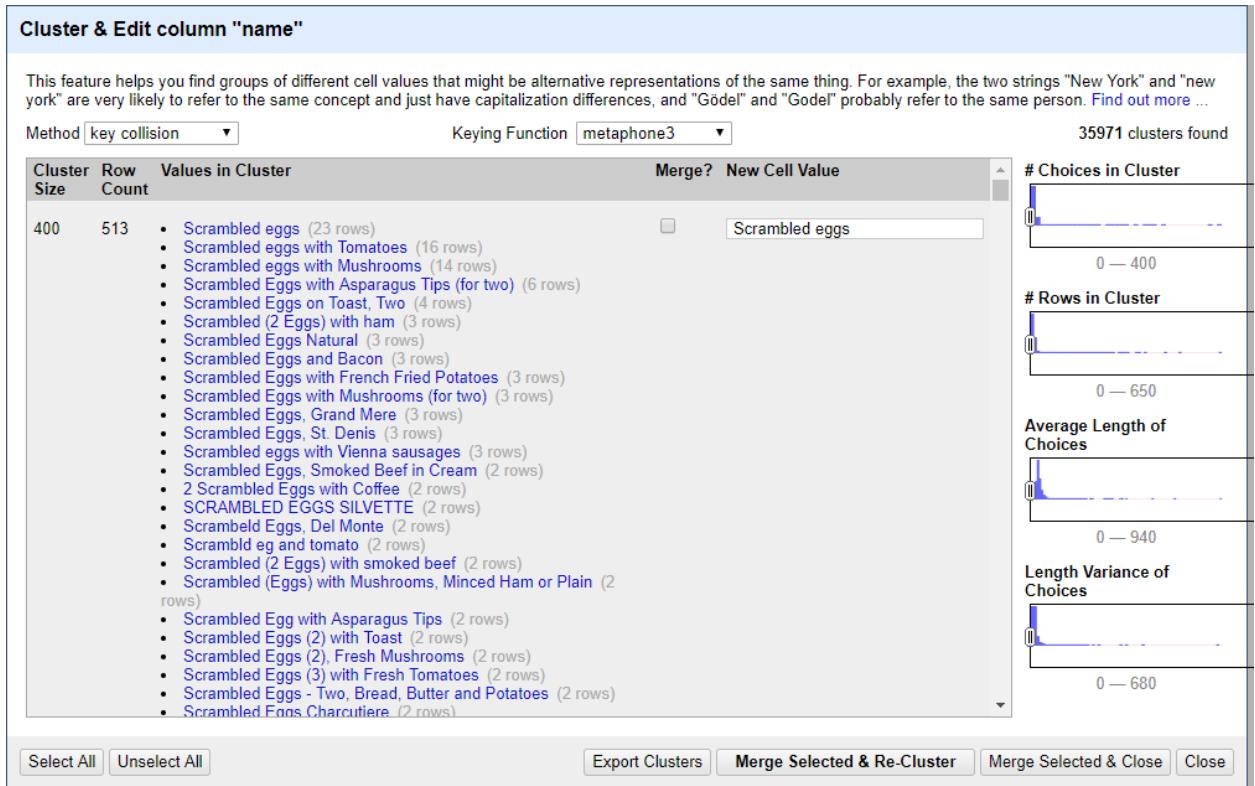


3. Clustering operation with keying function as “cologne-phonetic”. For example, this would replace the following values (a common sense based selective replacement was done):

"Coffee (Pot)",  
 "Coffe Pot",  
 "Coffee (pot - 1)",  
 "Cofee pot",  
 "Coffee (pot) .30",  
 "Coffee, Pota"  
 with "Coffee (Pot)"



4. We chose **not** to perform the clustering operation with keying function as "metaphone3" because the operation was resulting in a loss of feature selection in a dish. The screen shot shows an example where "Scrambled Eggs with Tomatoes" should not be listed as the same as "Scrambled Eggs with Mushrooms"



5. Replace special characters with this transformation:  
`value.replace(/[%@#!\?\\(\\\\|\|]+/, "").replace('*', '').replace('\"', '')`
6. Trim leading and trailing space, collapse consecutive whitespaces and convert all to UPPERCASE.

### 3.3.2 Transformations on “first\_appeared” column

1. Some of the columns in the date were entered as “2928” etc. These were fixed using the GREL transformation:  
`value.replace(/^29/, "19").replace(/^28/, "18")`
2. Set column to a number and value to 0 if null or blank with the following GREL transformation:  
`if(or(isBlank(value),isNull(value)), 0.0, value)`
3. Convert the column type to number.

### 3.3.3 Transformations on “last\_appeared” column

1. Some of the columns in the date were entered as “2928” etc. These were fixed using the GREL transformation:  
`value.replace(/^29/, "19").replace(/^28/, "18")`
2. Set column to a number and value to 0 if null or blank with the following GREL transformation:

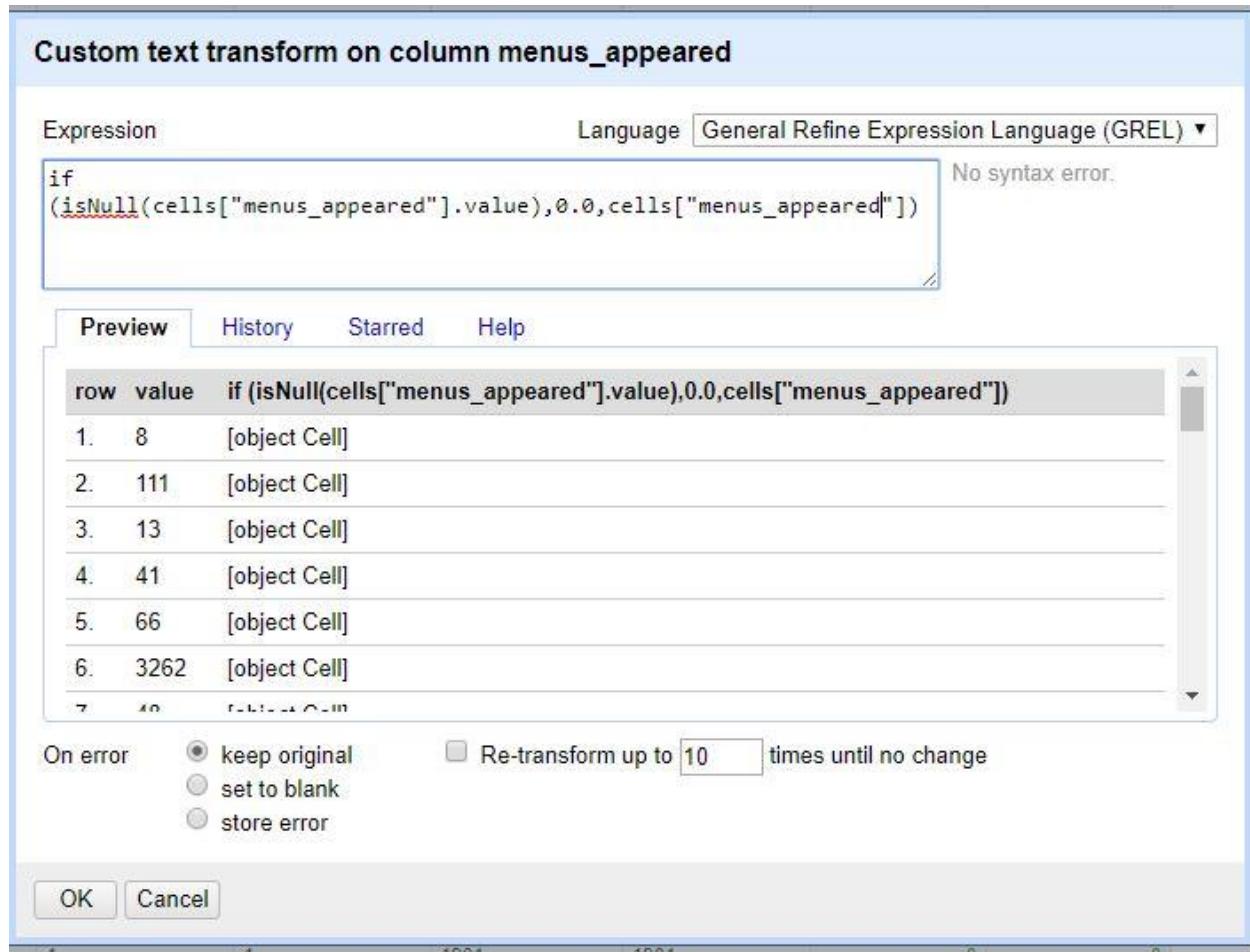
```
if(or(isBlank(value),isNull(value)), 0.0, value)
```

3. Convert the column type to number.

#### 3.3.4 Transformations on “menus\_appeared” column

1. Set column to a number and value to 0 if null or blank with the following GREL transformation:  

```
if(or(isBlank(value),isNull(value)), 0.0, value)
```
2. Convert the column type to number.



#### 3.3.5 Transformations on “lowest\_price” column

1. Set column to a number and value to 0 if null or blank with the following GREL transformation:  

```
if(or(isBlank(value),isNull(value)), 0.0, value)
```
2. Convert the column type to number.

Custom text transform on column highest\_price

row	value	if(or(isBlank(value),isNull(value)), 0.0, value)
1.	0.4	0.4
2.	0.8	0.8
3.	0.4	0.4
4.	1.0	1.0
5.	18.0	18.0
6.	25.0	25.0
7.	∞	∞

On error  keep original  Re-transform up to 10 times until no change  
 set to blank  
 store error

OK Cancel

### 3.3.6 Transformations on “highest\_price” column

- Set column to a number and value to 0 if null or blank with the following GREL transformation:  
 $\text{if(or(isBlank(value),isNull(value)), 0.0, value)}$
- Convert the column type to number.

### 3.3.7 Columns Deleted or Merged

The following columns were eliminated as they had 100% missing values:

- name

### 3.3.8 Columns left Untouched

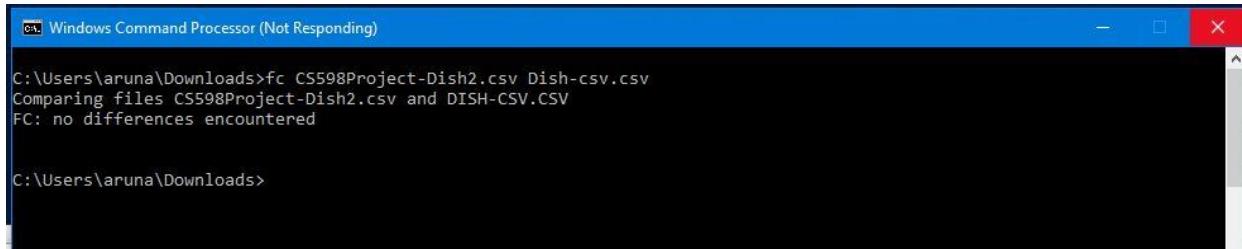
The following columns left untouched:

- id

Export the CS598Project-dish.csv from OpenRefine.

### 3.3.9 Operation History & Provenance

The operation history has been exported from OpenRefine as a json file (**dishOperationHis.json**, attached with project files). To check the provenance, we created a new project in OpenRefine with the original dish.csv and applied this dishOperationHis.json. This automatically performs all the transformations on all the columns. We exported the new csv and compared to the CS598Project-dish.csv using the windows file comparator and the two csv files were identical proving that the transformation are 100% replicable.



```
Windows Command Processor (Not Responding)

C:\Users\aruna\Downloads>fc CS598Project-Dish2.csv Dish-csv.csv
Comparing files CS598Project-Dish2.csv and DISH-CSV.CSV
FC: no differences encountered

C:\Users\aruna\Downloads>
```

## 3.4 Changes in MenuPage CSV

Created a project in OpenRefine for the MenuPage.csv data and determined that all numeric columns need no further cleaning. The csv contained only numeric fields.

1. Set the page\_number to 0 if null or blank using the GREL transformation:  
$$\text{if(or(isBlank(value), isNull(value)), 0.0, value)}$$
and converted column type to number.
2. The image\_id column for the most contained entirely numbers (almost 67K rows) except for 30 which contained text followed by numerals. We performed a GREL transformation to remove the alphabets and store it as a number:

```
value.replace('psnypl_rbk_','').replace('ps_rbk_','')
```

It is possible that these 30 entries were a typo and should contain numbers. Since this column is likely to be a foreign key to a table which the OCR scanned images of the menu (not included in this dataset). **We did not remove these 30 entries because these items are referenced in the MenuPage table via the menu\_page\_id and removing them will result in an integrity constraint violation.**

3. Null and blank entries in full\_height and full\_width columns where set to 0 using the GREL transformation:  
$$\text{if(or(isBlank(value),isNull(value)), 0, value)}$$
4. The remaining columns were left untouched.
5. CS598Project-MenuPage.csv was exported from the project.

### 3.4.1 Operation History & Provenance

The operation history has been exported from OpenRefine as a json file (**menuPageOperationHis.json**, attached with project files). To check the provenance, we created

a new project in OpenRefine with the original dish.csv and applied menuPageOperationHis.json. This automatically performs all the transformations on all the columns. We exported the new csv and compared to the CS598Project-menuPage.csv using the windows file comparator and the two csv files were identical proving that the transformation are 100% replicable.

The screenshot shows the OpenRefine interface with a project named "CS598Project-MenuPage". The main workspace displays a table with 66937 rows. The columns are labeled: id, menu\_id, page\_number, image\_id, full\_height, full\_width, and wuid. The data consists of approximately 66,937 rows of menu item information. The interface includes a header with "Open", "Export", and "Help" buttons, and a toolbar with "Facet / Filter", "Undo / Redo", "Extract", and "Apply". There are also buttons for "Show as: rows records" and "Show: 5 10 25 50 rows". A sidebar on the left shows a "Create project" section with a note about transforming 66608 cells in the "full\_width" column to numbers. The bottom right corner of the table area has a message: "first < previous 1 - 50 next > last".

### 3.5 Changes in MenuItem CSV

Created a project in OpenRefine for the MenuItem.csv data and determined that all numeric columns need no further cleaning.

1. Used a combination of python and GREL transformations for the created\_at and updated\_at fields to convert them to the standard MM/DD/YYYY format:

```
Python Transformation: a=value.split(" ")
return a[0]
```

```
GREL Transformation:
value.toDate('yyy-MM-dd','yyy-MM-dd').toString('MM/dd/yyyy')
```

2. Set blank and null entries in the high\_price column to 0 and changed the column type to number
3. Set blank and null entries in the price column to 0 and changed the column type to number

Custom text transform on column price

Expression      Language: General Refine Expression Language (GREL) ▾

```
if(or(isBlank(value),isNull(value)), 0.0, value)
```

No syntax error.

Preview    History    Starred    Help

row	value	if(or(isBlank(value),isNull(value)), 0.0, value)
1.	0.4	0.4
2.	0.6	0.6
3.	0.4	0.4
4.	0.5	0.5
5.	0.5	0.5
6.	0.1	0.1

On error     keep original     Re-transform up to  times until no change

4. **dish\_id column has blank entries we did not fill in the missing values as this is an identified integrity constraint violation.**
5. Final export of CS598Project-menuitem.csv from OpenRefine.

Refine CS598Project-MenuItem Permalink

1332734 rows

Show as: rows records Show: 5 10 25 50 rows

Extract... Apply...

Extensions

« first < previous 1 - 50 next > last »

**Filter:**

- o Create project
1. Text transform on 0 cells in column `created_at` valueToDate()
2. Text transform on 1332734 cells in column `updated_at` `python:a.value.split(" ") return a[0]`
3. Text transform on 1332734 cells in column `updated_at` `getvalueToDate('yy-MM-dd'+'yy-MM-dd')toString('MMddyy'yyyy')`
4. Text transform on 1332734 cells in column `created_at` `python:a.value.split(" ") return a[0]`
5. Text transform on 1332734 cells in column `created_at` `getvalueToDate('yy-MM-dd'+'yy-MM-dd')toString('MMddyy'yyyy')`

**Operations:**

id	menu_page_id	price	high_price	dish_id	created_at	updated_at	xpos	ypos	
1	1389	6.4		1	03/20/2011	04/19/2011	0.11429	0.25475	
2	2	1389	6.4	2	03/20/2011	04/19/2011	0.435571	0.25475	
3	3	1389	6.4	3	03/20/2011	04/19/2011	0.14	0.261922	
4	4	1389	0.5	4	03/20/2011	04/19/2011	0.377143	0.26272	
5	3079	0.6	1.0	5	03/20/2011	04/13/2011	0.168571	0.313178	
6	6	1389	0.1	7	03/20/2011	04/19/2011	0.11429	0.30105	
7	8	1389	0.25	9	03/20/2011	04/19/2011	0.167143	0.273101	
8	9	1389	0.75	10	03/20/2011	04/19/2011	0.555571	0.265116	
9	10	1389	0.75	11	03/20/2011	04/19/2011	0.657143	0.274698	
10	11	1389	0.6	8	03/20/2011	04/19/2011	0.68	0.253936	
11	12	1389	0.7	12	03/20/2011	04/19/2011	0.161429	0.308941	
12	13	1389	0.3	13	03/20/2011	04/19/2011	0.673571	0.302647	
13	15	1389		14	03/20/2011	03/20/2011	0.142557	0.389668	
14	16	1389	0.4	15	03/20/2011	04/19/2011	0.327143	0.30105	
15	17	1389	0.25	16	03/20/2011	04/19/2011	0.435571	0.30105	
16	18	1389	0.35	17	03/20/2011	04/19/2011	0.567143	0.302647	
17	19	1389	0.3	18	03/20/2011	04/19/2011	0.864268	0.305043	
18	20	1389	0.2	19	03/20/2011	04/19/2011	0.161429	0.313827	
19	21	1389	0.2	20	03/20/2011	04/19/2011	0.332857	0.310633	
20	22	1389	0.25	21	03/20/2011	04/19/2011	0.454268	0.313028	
21	23	1389	1.0	22	03/20/2011	04/19/2011	0.673571	0.309035	
22	24	3079	1.5	3.0	23	03/20/2011	04/19/2011	0.081429	0.470795
23	25	1389	1.0		24	03/20/2011	04/19/2011	0.142557	0.368128
24	26	1389	0.75		25	03/20/2011	04/19/2011	0.367143	0.361739
25	27	1389	0.25		26	03/20/2011	04/19/2011	0.263857	0.229181
26	28	1389	0.25		27	03/20/2011	04/19/2011	0.574268	0.230778
27	29	1388	4.0		35733	03/1/2011	04/20/2011	0.1	0.211111
28	31	12947			30	03/1/2011	03/1/2011	0.168571	0.204044
29	33	12947			7	03/1/2011	03/1/2011	0.765714	0.261516
30	34	130			31	03/1/2011	03/1/2011	0.17	0.285441
31	35	130			32	03/1/2011	03/1/2011	0.687143	0.311731
32	36	130			33	03/1/2011	03/1/2011	0.555571	0.341778
33	37	142			34	03/1/2011	03/1/2011	0.352857	0.303289
34	38	1575			35	03/1/2011	03/1/2011	0.081429	0.625622
35	39	1575			8618	03/1/2011	03/1/2011	0.367143	0.361739
36	40	1575			37	03/1/2011	04/19/2011	0.511429	0.440575
37	41	1529			38	03/1/2011	03/1/2011	0.686571	0.372708
38	42	142			39	03/1/2011	03/1/2011	0.161429	0.380227
41					40	04/8/2011	04/9/2011	0.462857	0.268862

### 3.5.1 Operation History & Provenance

The operation history has been exported from OpenRefine as a json file (**menuItemOperationHis.json**, attached with project files). To check the provenance, we created a new project in OpenRefine with the original menuItem.csv and applied this menuItemOperationHis.json. This automatically performs all the transformations on all the columns. We exported the new csv and compared it to the CS598Project-menuitem.csv using the windows file comparator and the two csv files were identical proving that the transformation are 100% replicable.

## 3.6 Columns after cleanup with OpenRefine

### 3.6.1 Menu csv

- id – unique identifier for each menu
- name – name of the menu with sponsor details
- event – event for which the menu was prepared
- venue – venue where the event was held
- place – location/address of event
- physical\_description – characteristics of menu card; type of paper, size
- occasion – special occasion for which the menu was prepared.
- notes – details on menu, language used, interesting characteristics of menu
- call\_number – 4-year digit followed by a hyphen by a 4-digit id
- date – date when the menu was created in the format MM/DD/YYYY
- currency – currency used for the menu item pricing
- status – indicates whether the menu entry is complete
- page\_count – number of pages on the menu
- dish\_count – number of dishes on the menu

### 3.6.2 Dish csv

- id – unique identifier for each dish
- name – name of the dish
- menus\_appeared – menu on which the dish has appeared on
- times\_appeared – number of times the dish has appeared on menus
- first\_appeared – 4-digit year when the menu first appeared
- last\_appeared – 4-digit year when the menu last appeared
- lowest\_price – lowest prices for the dish
- highest\_price – highest prices for the dish

### 3.6.3 MenuItem csv

- id – unique identifier for the menu item
- menu\_page\_id – page id on which menu appears (foreign key to menu page id)
- price – price of menu item
- high\_price – highest price of men item
- dish\_id – dish id for the menu item (foreign key to dish id)
- created\_at – date on which the menu item was created
- updated\_at – date on which the menu item was updated
- xpos – x coordinate of where the menu item appears on the scanned image
- ypos – y coordinate of where the menu item appears on the scanned image

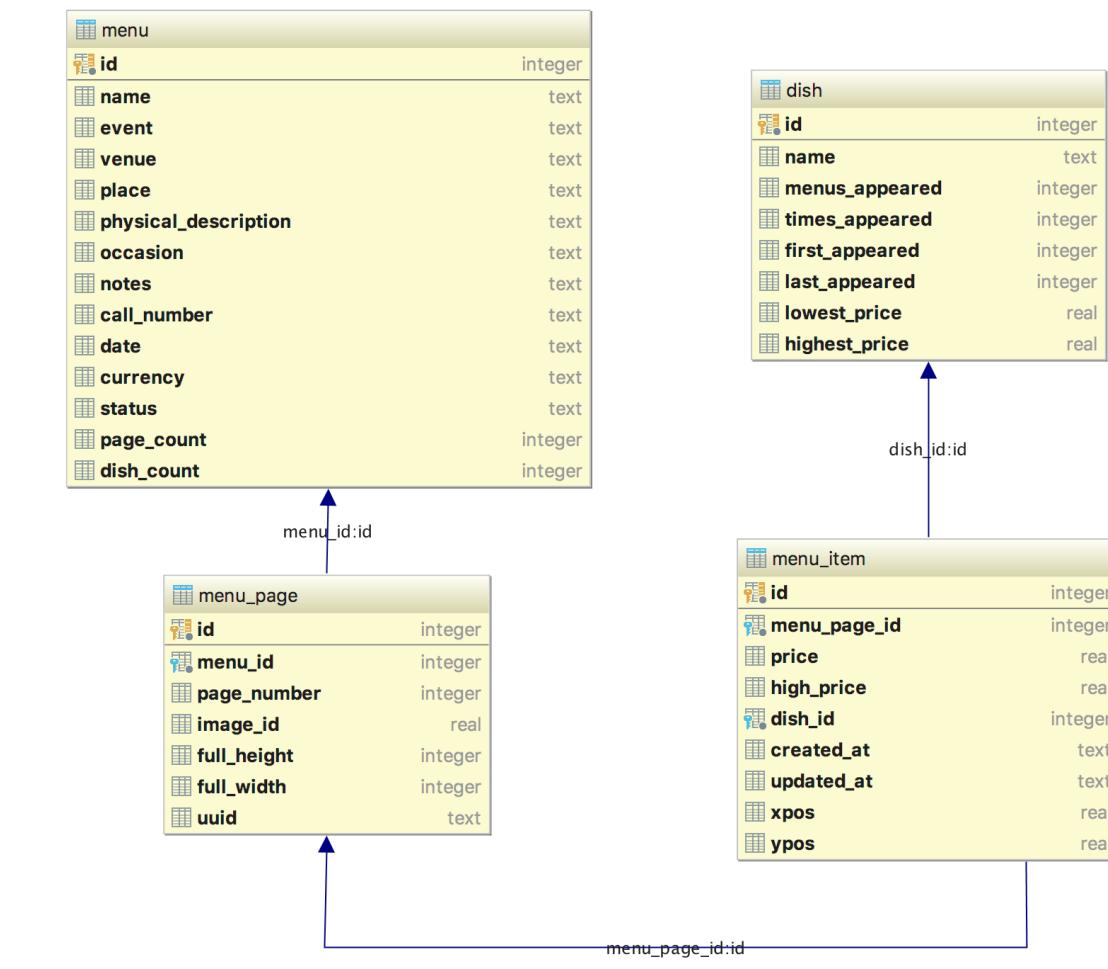
### 3.6.4 MenuPage csv

- id – unique identifier for a page on each menu
- menu\_id – identifier to menu (foreign key to menu:id)
- page\_number – page number for each menu
- image\_id – id for OCR scanned image of the page
- full\_height – height of the original page
- full\_width – width of the original page
- uuid – unique alpha numeric identifier to the image

## 4 SQLite

We were successful in importing data into SQLite. We have reproduced the schema in the section 4.1. Section 4.2 contains data definition language used to create tables and constraints are introduced for the all four tables.

## 4.1 Schema



Files

### 4.1.1 Enable foreign key constraint support in SQLite

Before importing the data into "SQLite", we had to turn on the support for foreign key restrictions.

**PRAGMA foreign\_keys = ON;**

## 4.2 Data Definition Language

Created the schema for all 4 tables with the foreign key relationship.

- The menu\_page table has the **menu\_id** as the foreign key of the menu table's primary key ( **id** )
- The menu\_item table has two foreign key relationship:
  - menu\_page\_id** with menu\_page table's primary key ( **id** )
  - dish\_id** with dish table's primary key ( **id** )

```

CREATE TABLE menu
(
    id INTEGER PRIMARY KEY NOT NULL,
    name TEXT,
    event TEXT,
    venue TEXT,
    place TEXT,
    physical_description TEXT,
    occasion TEXT,
    notes TEXT,
    call_number TEXT,
    date TEXT,
    currency TEXT,
    status TEXT,
    page_count INTEGER,
    dish_count INTEGER
)

CREATE TABLE menu_page
(
    id INTEGER PRIMARY KEY NOT NULL,
    menu_id INTEGER NOT NULL,
    page_number INTEGER,
    image_id REAL,
    full_height INTEGER,
    full_width INTEGER,
    uuid TEXT,
    CONSTRAINT menu_page_menu_id_fk FOREIGN KEY (menu_id) REFERENCES menu
(id) ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE UNIQUE INDEX menu_page_id_uindex ON menu_page (id);

CREATE TABLE dish
(
    id INTEGER PRIMARY KEY NOT NULL,
    name TEXT,
    menus_appeared INTEGER,
    times_appeared INTEGER,
    first_appeared INTEGER,
)

```

```

last_appeared INTEGER,
lowest_price REAL,
highest_price REAL
);
CREATE UNIQUE INDEX dish_id_uindex ON dish (id);

CREATE TABLE menu_item
(
    id INTEGER PRIMARY KEY NOT NULL,
    menu_page_id INTEGER NOT NULL,
    price REAL,
    high_price REAL,
    dish_id INTEGER NOT NULL,
    created_at TEXT,
    updated_at TEXT,
    xpos REAL,
    ypos REAL,
    CONSTRAINT menu_item_menu_page_id_fk FOREIGN KEY (menu_page_id)
    REFERENCES menu_page (id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT menu_item_dish_id_fk FOREIGN KEY (dish_id) REFERENCES dish (id)
    ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE UNIQUE INDEX menu_item_id_uindex ON menu_item (id);

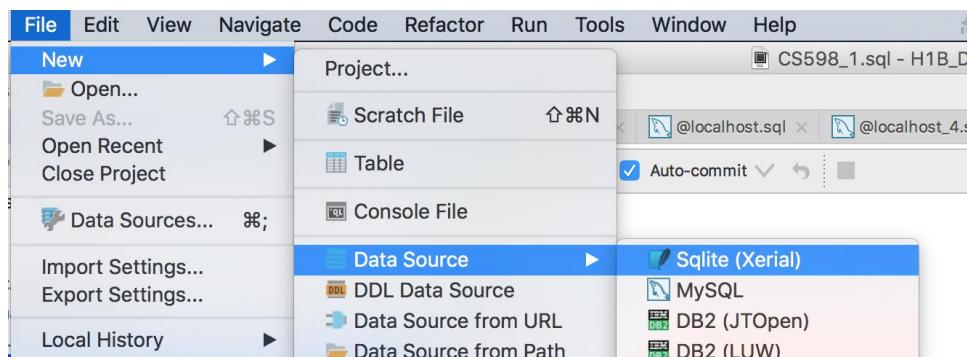
```

### 4.3 CSV Import

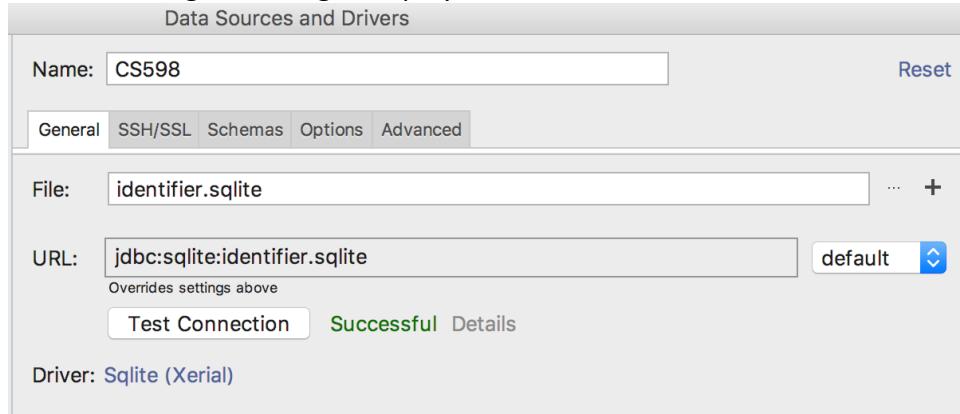
SQLite command line interface is not very robust to load complex CSV files. We used "DataGrip" to import the data. We created a data source and then imported the csv files.

#### 4.3.1 Adding Data Source

- Click on File->New->Data Source->SQLite, as shown below.



- Then in the dialog enter the name as "CS598" and click on "Test Connection". The "Successful" message should get displayed.

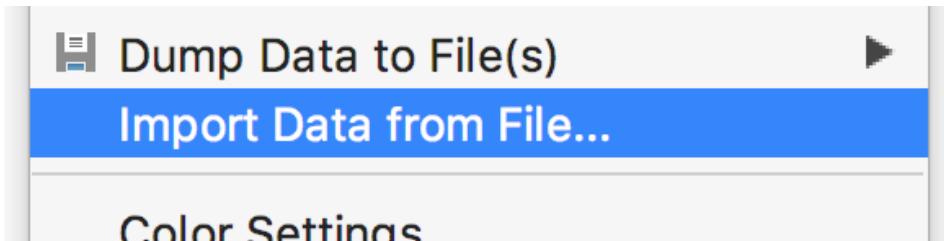


- Now click on apply to add the Data Source.

#### 4.3.2 Import Data

##### 4.3.2.1 Import menu.csv

- Right click on the data source (CS598) and click on "Import Data from File..."



- Then select the menu.csv to load the file.
- Following window will be displayed.

Import "menu.csv" File

Formats: Tab-separated (TSV) Comma-separated (CSV)\*

Value separator: Comma Row separator: Newline Null value text:

Add row prefix/suffix Quotation: " " Escape: duplicate ' ' Escape: duplicate Quote values: When needed

Trim whitespaces  First row is header  First column is header

Table: menu

Columns (14) Keys Indices Foreign Keys

C1 TEXT mapped to C1  
C2 TEXT mapped to C2  
C3 TEXT mapped to C3  
C4 TEXT mapped to C4  
C5 TEXT mapped to C5  
C6 TEXT mapped to C6  
C7 TEXT mapped to C7  
C8 TEXT mapped to C8  
C9 TEXT mapped to C9

Data Preview DDL Preview

	C1	C2	C3	C4
1	id	name	event	venue
2	12463	"HOTEL EASTMAN"	BREAKFAST	COMMIE
3	12464	"REPUBLICAN HOU...	DINNER	COMMIE
4	12465	"NORDDEUTSCHER ...	FRUHSTUCK-BREAK...	COMMIE
5	12466	"NORDDEUTSCHER ...	LUNCH	COMMIE
6	12467	"NORDDEUTSCHER ...	DINNER	COMMIE
7	12468	"CANADIAN PACIF...	DINNER	COMMIE
8	12469	"HOTEL NETHERLA...	SUPPER	COMMIE
9	12470	"NORDDEUTSCHER ...	FRUHSTUCK-BREAK...	COMMIE
10	12471	"NORDDEUTSCHER ...	LUNCH	COMMIE

Encoding: UTF-8

Write errors to file: 598/Final Project/menu\_1\_2017-07-21\_01\_28\_22.txt

Insert invertible values as null

- Select "Comma-separated (CSV)\*" in the Format section, and check on "Trim whitespaces" and "First row is header"

Import "CS598Project-Menu.csv" File

Formats: Tab-separated (TSV) Comma-separated (CSV)\* Comma-separated (CSV)\_1\*

Value separator: Comma Row separator: Newline Null value text:

Add row prefix/suffix Quotation: " " Escape: duplicate ' ' Escape: duplicate Quote values: Always

Trim whitespaces  First row is header  First column is header

Header Format

Value separator: Comma Row separator: Newline Null value text:

Add row prefix/suffix Quotation: " " Escape: duplicate ' ' Escape: duplicate Quote values: Always

Table: menu

Columns (14) Keys Indices Foreign Keys

id INTEGER mapped to id  
name TEXT mapped to name  
event TEXT mapped to event  
venue TEXT mapped to venue  
place TEXT mapped to place  
physical\_description TEXT mapped to physical\_description  
occasion TEXT mapped to occasion  
notes TEXT mapped to notes  
call\_number TEXT mapped to call\_number

Data Preview DDL Preview

	id	name
1	12463	"HOTEL EASTMAN"
2	12464	"REPUBLICAN HOUSE"
3	12465	"NORDDEUTSCHER LLOYD BREMEN"
4	12466	"NORDDEUTSCHER LLOYD BREMEN"
5	12467	"NORDDEUTSCHER LLOYD BREMEN"
6	12468	"CANADIAN PACIFIC RAILWAY COMPANY"
7	12469	"HOTEL NETHERLAND"
8	12470	"NORDDEUTSCHER LLOYD BREMEN"
9	12471	"NORDDEUTSCHER LLOYD BREMEN"
10	12472	"NORDDEUTSCHER LLOYD BREMEN"

Encoding: UTF-8

Write errors to file: /Users/abhisekjana/Desktop/MS/assignments/CS 5

Insert invertible values as null

- We will define the Primary Key of the table now. Double-click on the id row in the Columns(14) tab. Select **Not Null**, **Unique** and **Primary key** check box.

Columns (14)				Keys (1)	Indices (1)	Foreign Keys	
Name:	Type:	Default:	Mapped to:				
id	INTEGER		id	<input checked="" type="checkbox"/> Not null	<input type="checkbox"/> Auto inc	<input checked="" type="checkbox"/> Unique	<input checked="" type="checkbox"/> Primary key
name	TEXT	<i>mapped to name</i>					
event	TEXT	<i>mapped to event</i>					
venue	TEXT	<i>mapped to venue</i>					
place	TEXT	<i>mapped to place</i>					

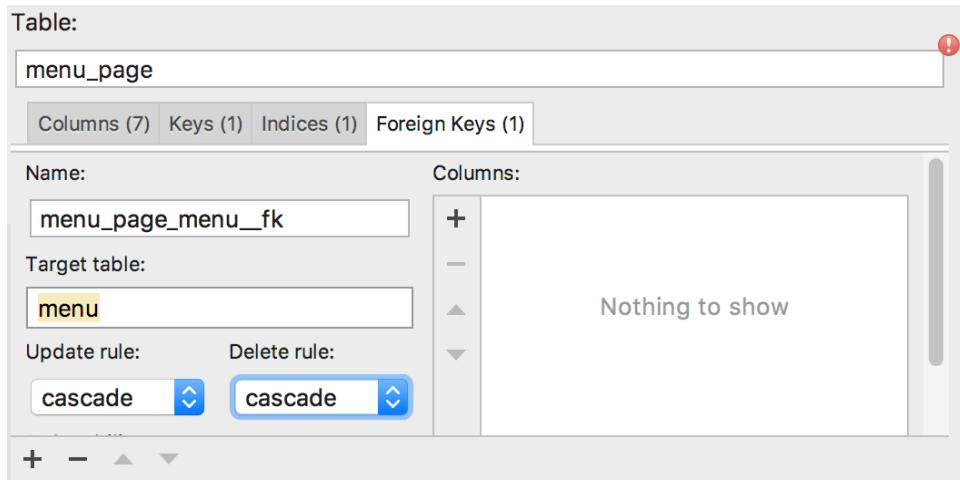
- Click on "OK" to import the data from menu.csv.

#### 4.3.2.2 Import menu\_page.csv

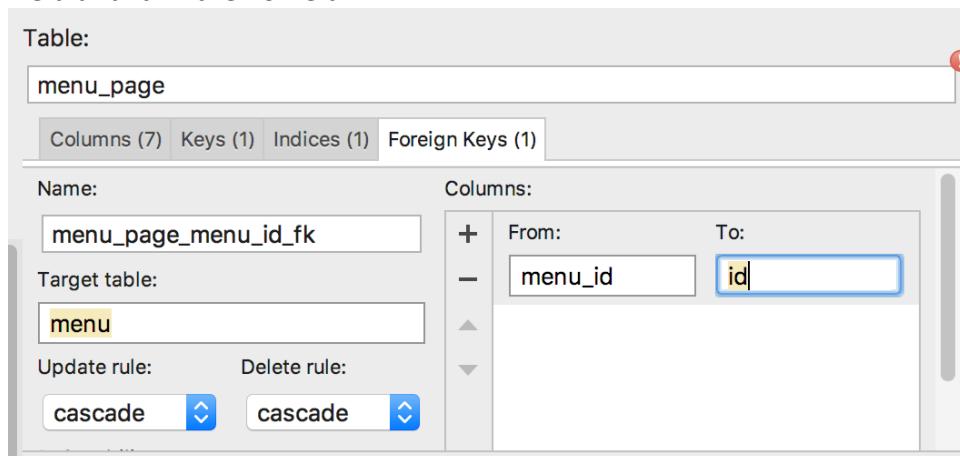
- We have imported the menu\_page.csv the same way as the menu.csv.
- Configure the Table Name and the Primary Key.
- Since menu\_page has foreign key constraint with the menu table, we need to define that.
- Double click on menu\_id row and select Not Null check box.

Table:			
menu_page			
Columns (7) Keys (1) Indices (1) Foreign Keys (1)			
<i>id INTEGER -- part of primary key mapped to id</i>			
Name:	Type:	Default:	Mapped to:
menu_id	INTEGER		menu_id
<input checked="" type="checkbox"/> Not null	<input type="checkbox"/> Auto inc	<input type="checkbox"/> Unique	<input type="checkbox"/> Primary key
page_number	INTEGER	<i>mapped to page_number</i>	
image_id	REAL	<i>mapped to image_id</i>	
full_height	INTEGER	<i>mapped to full_height</i>	
<b>+</b>	<b>-</b>	<b>▲</b>	<b>▼</b>

- Click on the Foreign Keys tab and click on the + button at the left bottom.
- Enter "menu" in the Target table field. Then select both the Update & Delete rule as cascade.



- Now click on the + button in the top middle ( Columns ) section. Enter menu\_id in the From field and id in the To field.



- Click on Ok to import the data.

#### 4.3.2.3 Import dish.csv

- Importing dish.csv is same as the menu.csv.

#### 4.3.2.4 Import menu\_item.csv

- Importing menu\_item.csv is same as the menu\_page.csv.
- However, this time we need to create 2 foreign key constraints (menu\_page & dish) instead of one.

## 4.4 Integrity Constraints

We imported the data from csv after enabling the foreign key support.

- When loading the menu\_item table we found **241 integrity constraint violations** with the dish table. There are 241 rows in the menu\_item csv where the dish id is empty. We could not load the data into the menu item table.
- The log of **241 integrity constraint violation errors** can be found in the following [file](#).

- All 241 rows with empty dish\_id from the menu\_item table has been captured in a separate [csv](#).
- The menu\_page table has some menu\_id which is not present in the menu table. The following query provided the list of **5803** rows from menu\_page table which are violating the integrity constraint (There are no null in the name column of the menu table).
  - **SELECT** menu\_page.\* **from** menu\_page  
**LEFT OUTER JOIN** menu  
**on** menu\_page.menu\_id=menu.id  
**where** menu.name IS null
  - The **5803** rows from menu\_page table has been extracted to a separate csv file, which can be viewed [here](#).
- Deleted the **5803** rows from the menu\_page table using following query.
  - **delete from** menu\_page **where id in** (  
**SELECT** menu\_page.id  
**FROM** menu\_page  
**LEFT OUTER JOIN** menu  
**ON** menu\_page.menu\_id = menu.id  
**WHERE** menu.name IS NULL  
 )
- Now menu\_page has **61134** rows. Which matches with the count of the following query.
  - **SELECT** count(\*) **from** menu  
**LEFT OUTER JOIN** menu\_page  
**on** menu.id = menu\_page.menu\_id
- The menu\_item table has **86347** rows where the dish id is violating the integrity constraint (there is no reference in the dish table). There are no null entries in the name column of the dish table)
  - **select** \* **from** menu\_item  
**LEFT OUTER JOIN** dish **on** menu\_item.dish\_id = dish.id  
**where** dish.name is null
  - The **86347** rows from menu\_item table has been extracted to a separate csv file, which can be viewed [here](#).
- Deleted the **86347** rows from the menu\_item table using following query.
  - **DELETE FROM** menu\_item **where id in** (  
**SELECT** menu\_item.id  
**FROM** menu\_item  
**LEFT OUTER JOIN** dish **ON** menu\_item.dish\_id = dish.id

```

WHERE dish.name IS NULL
)

```

- Now the menu\_item table has **4744** rows where the menu\_page\_id is violating the integrity constraint (there is no reference in the menu\_page table).
  - select \*from** menu\_item  
**LEFT OUTER JOIN** menu\_page **on** menu\_item.menu\_page\_id =  
**menu\_page.id**  
**where** menu\_page.menu\_id is null
  - The **4744** rows from menu\_item table has been extracted to a separate csv file, which can be viewed [here](#).
- Deleted the **4744** rows from the menu\_item table using following query.
  - DELETE FROM** menu\_item **where id in** (  
**SELECT** menu\_item.id  
**FROM** menu\_item  
**LEFT OUTER JOIN** menu\_page **ON** menu\_item.menu\_page\_id =  
**menu\_page.id**  
**WHERE** menu\_page.menu\_id IS NULL  
)

#### 4.4.1 Row Counts

Table Name	Row Count in Original CSV	Row Count after OpenRefine cleanup	Current Row Count in SQLite
menu	17,545	17,545	17,545
menu_page	66,937	66,937	61,134
dish	423,397	369,317	369,317
menu_item	1,332,726	1,332,726	1,241,394

#### 4.5 Clean Scripts

Following three SQL commands are used to further clean up name and notes columns in menu dataset and name column in dish dataset. After cleaning the data set in Open Refine 3 columns had "" at the beginning and end. We had to clean up additional " (quote) using SQL query.

update menu **set name=SUBSTR(name, 2, LENGTH(name)-2)**

```
update menu set notes=SUBSTR(notes, 2,LENGTH(notes)-2)
```

```
update dish set name=SUBSTR(name, 2,LENGTH(name)-2)
```

## 4.6 Data Extraction

We have extracted the data based on the relevant columns available in the 4 data sets which can be used in further study. Extraction query that we used can be seen below.

### 4.6.1 Data Extraction Query

We have joined all 4 tables and included menu\_item.price, dish.name, menu.event, menu.venue, menu.currency, menu.place, menu.date, menu.page\_count, menu.dish\_count in the select query when price is not equal to 0 and date is also not empty. This data will help us to perform time series analysis on Price and Date field.

#### select

```
menu_item.price,dish.name,menu.event,menu.venue,menu.currency,menu.place,  
menu.date,menu.page_count,menu.dish_count  
from menu_item,dish,menu_page,menu  
where menu_item.dish_id=dish.id and menu_item.price<>0  
and menu_page.id=menu_item.menu_page_id  
and menu.id=menu_page.menu_id  
and trim(menu.date) <> ''  
order by menu_item.price desc
```

The extracted csv file can be found [here](#).

## 5 Provenance

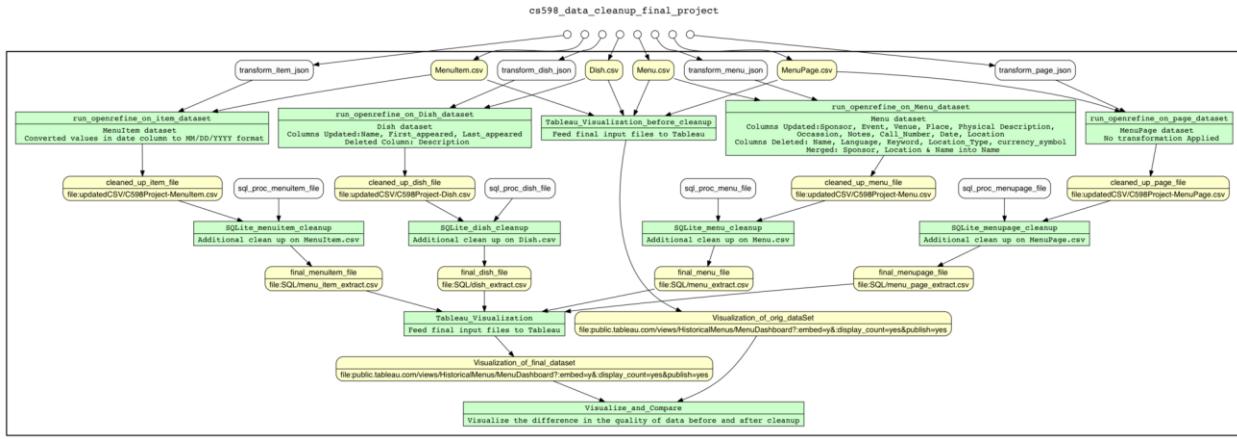
The [end to end workflow](#) was created using the YesWorkflow tool. The chart was created using offline version of YesWorkflow tool. We had to install dot tool from graphviz.com website to postprocess the gv file.

Following command was used to generate the image file of the workflow.

```
yw graph -c graph.view=COMBINED -c graph.layout=TB workflow1.py | dot -  
Tpng -o workflow1.png
```

The first level of diagram shows the OpenRefine's json file and corresponding csv file as input to the workflow. See section 3 for more details.

The run operation produces a cleaned-up version of the csv files. These files are than imported into SQLite and addition SQL procs are run to create profile and do further cleanup of the data. See section 4 for additional details.

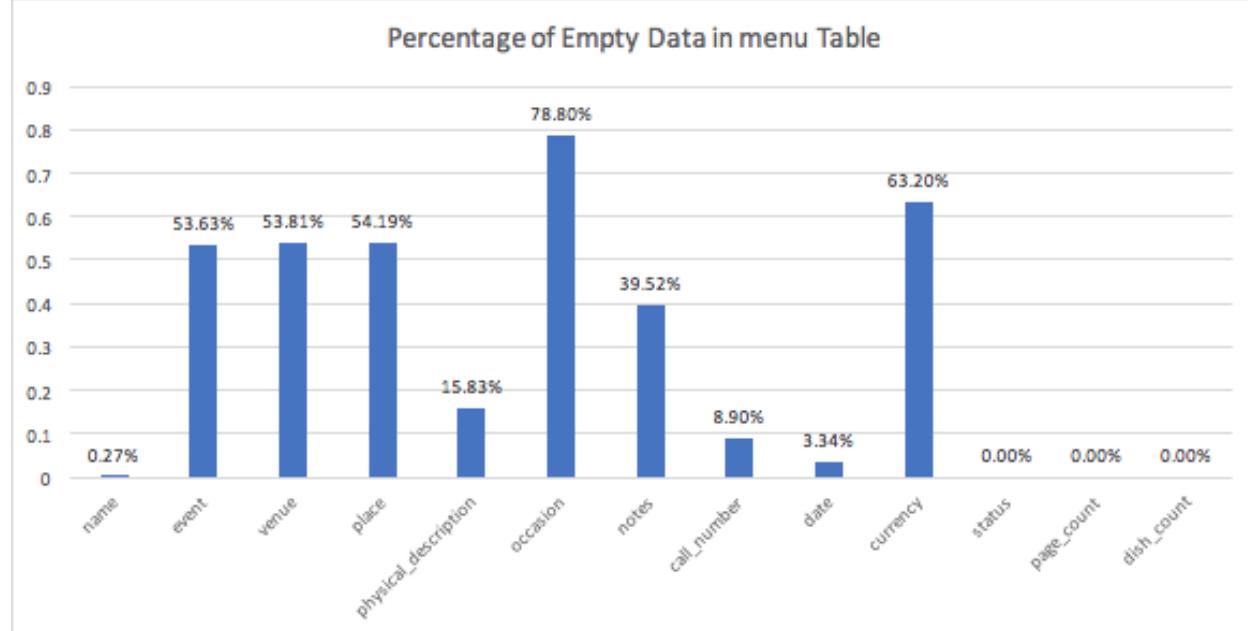


## 6 Dataset Quality Analysis

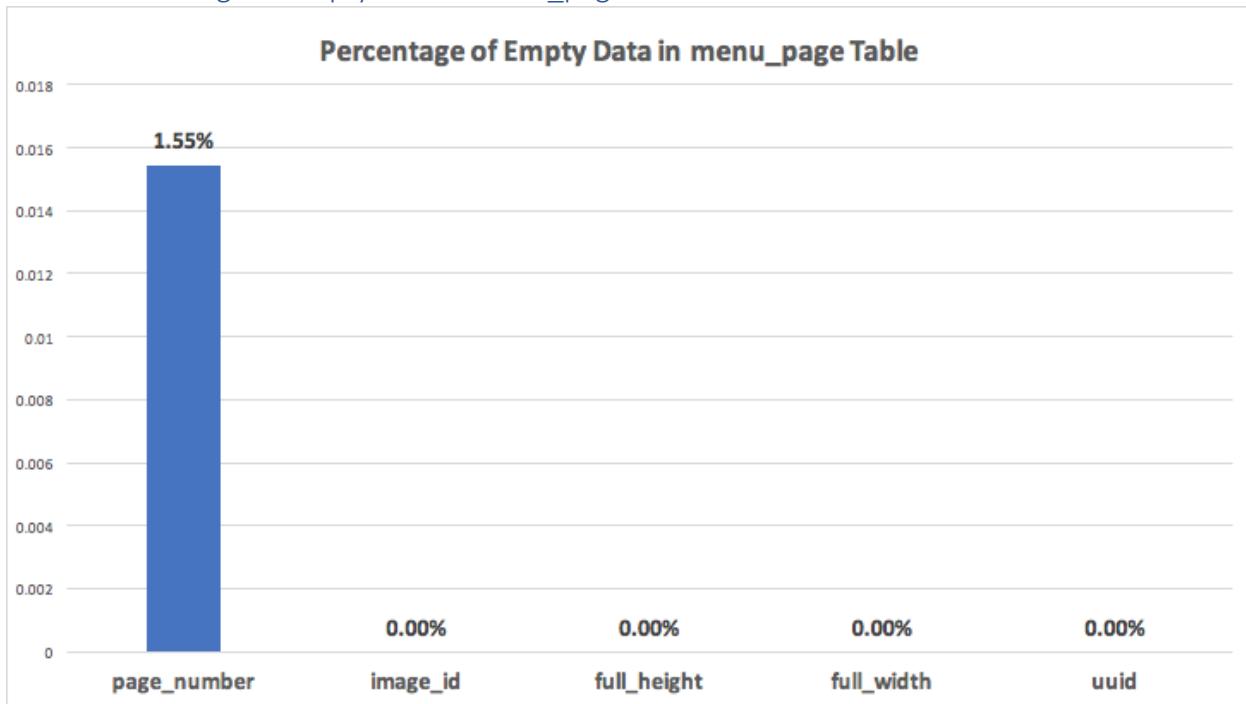
### 6.1 Analysis of empty data in the tables

The following 4 charts show the percentage of nulls and zeros in each of the columns in the tables after OpenRefine and SQLite cleanup.

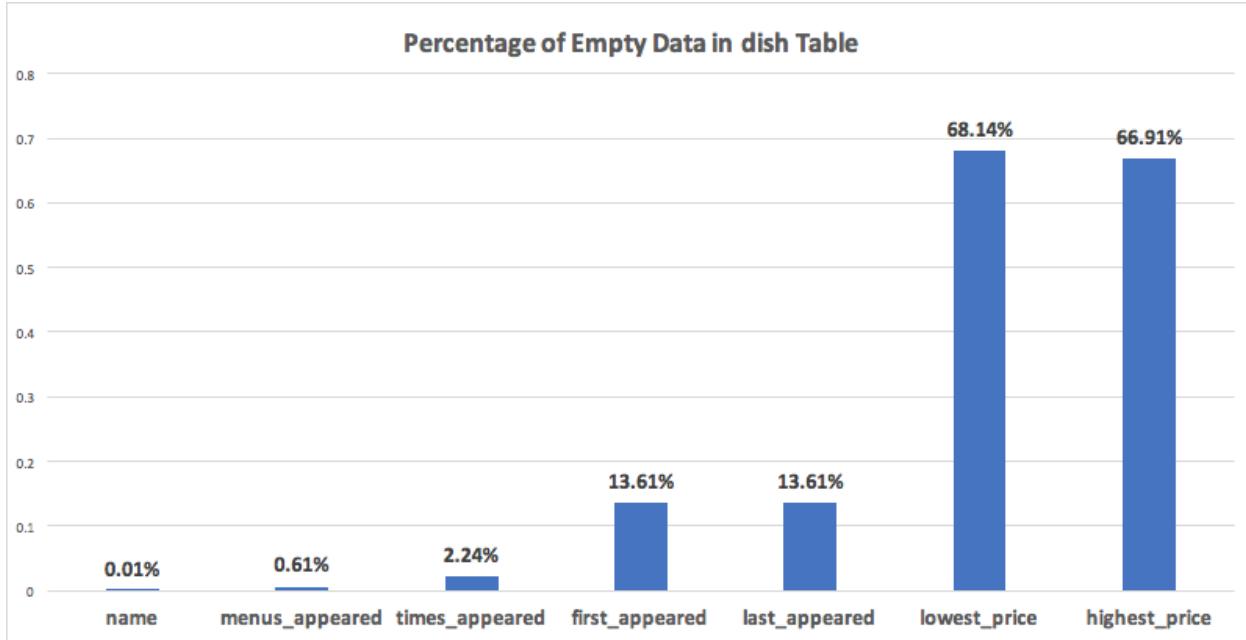
#### 6.1.1 Percentage of empty data in menu table



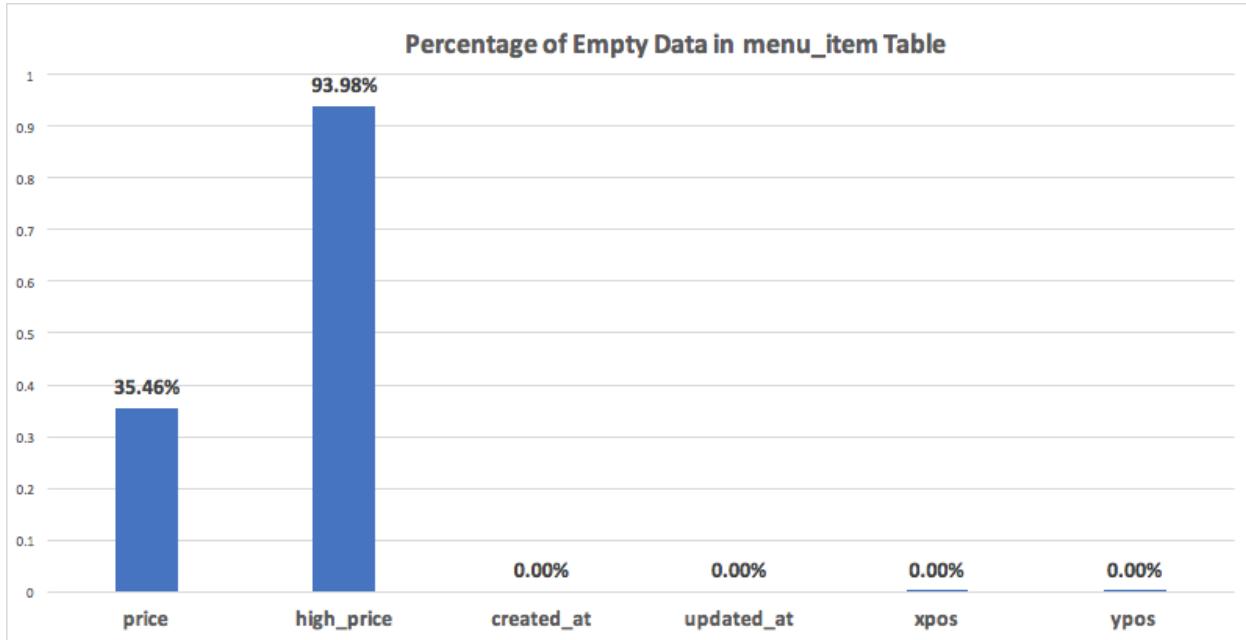
### 6.1.2 Percentage of empty data in menu\_page table



### 6.1.3 Percentage of empty data in dish table



#### 6.1.4 Percentage of empty data in menu\_item table



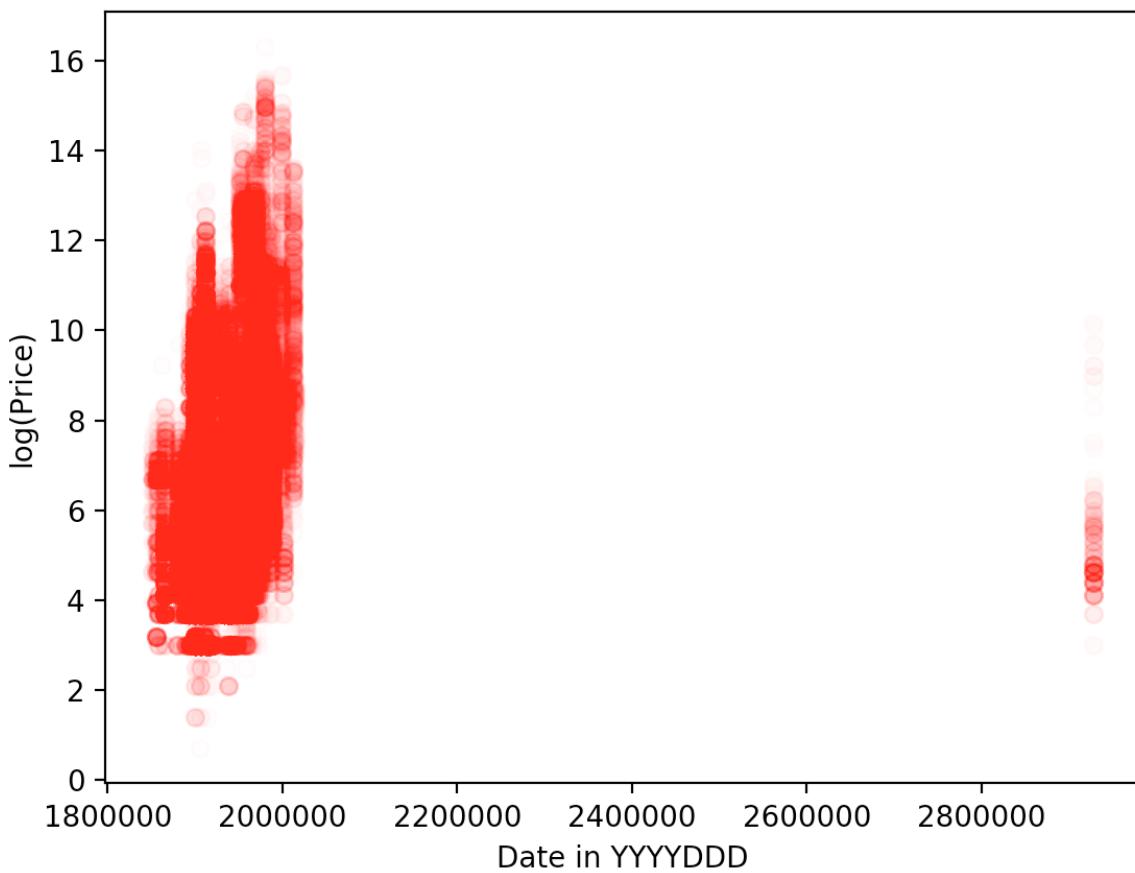
## 6.2 Regression Analysis Using Least Squares Method

We performed regression analysis to determine if any of the continuous variables (dish\_count, page\_count, data) can help predict the price (response variable).

We have transformed the date field to a continuous numeric variable (the transformation can be seen in the python code segment below). The input data file `menu_price_date_regression.csv` used for the regression can be downloaded [here](#).

### 6.2.1 Price vs Date (logScale)

The converted date was plotted against the log (price). We used a logScale because of the very high concentration of values in the low-price range. The plot clearly shows outliers for dates greater than 2800000 (clearly indicating user data entry errors).



## 6.2.2 Python code for Date Transformation and Outlier Detection

```

1. import numpy as np
2. import pandas as pd
3. import datetime
4. import matplotlib.pyplot as plt
5.
6. def convertDate(old_dt):
7.     fmt = '%m/%d/%Y'
8.     dt = datetime.datetime.strptime(old_dt, fmt)
9.     tt = dt.timetuple()
10.    jd = tt.tm_year * 1000 + tt.tm_yday
11.    return jd
12.
13. df = pd.read_csv('menu_price_date_regression.csv')
14. df['n_date'] = df['date'].apply(convertDate)
15. df['n_price'] = np.log(df.price) + 6
16.
17. # Plot log(Price) Vs. Date
18. plt.plot(df.n_date, df.n_price, 'ro', alpha=.01)
19. plt.ylabel('log(Price)')
20. plt.xlabel('Date in YYYYDDD')
21. plt.show()
22.
23. # Remove outliers

```

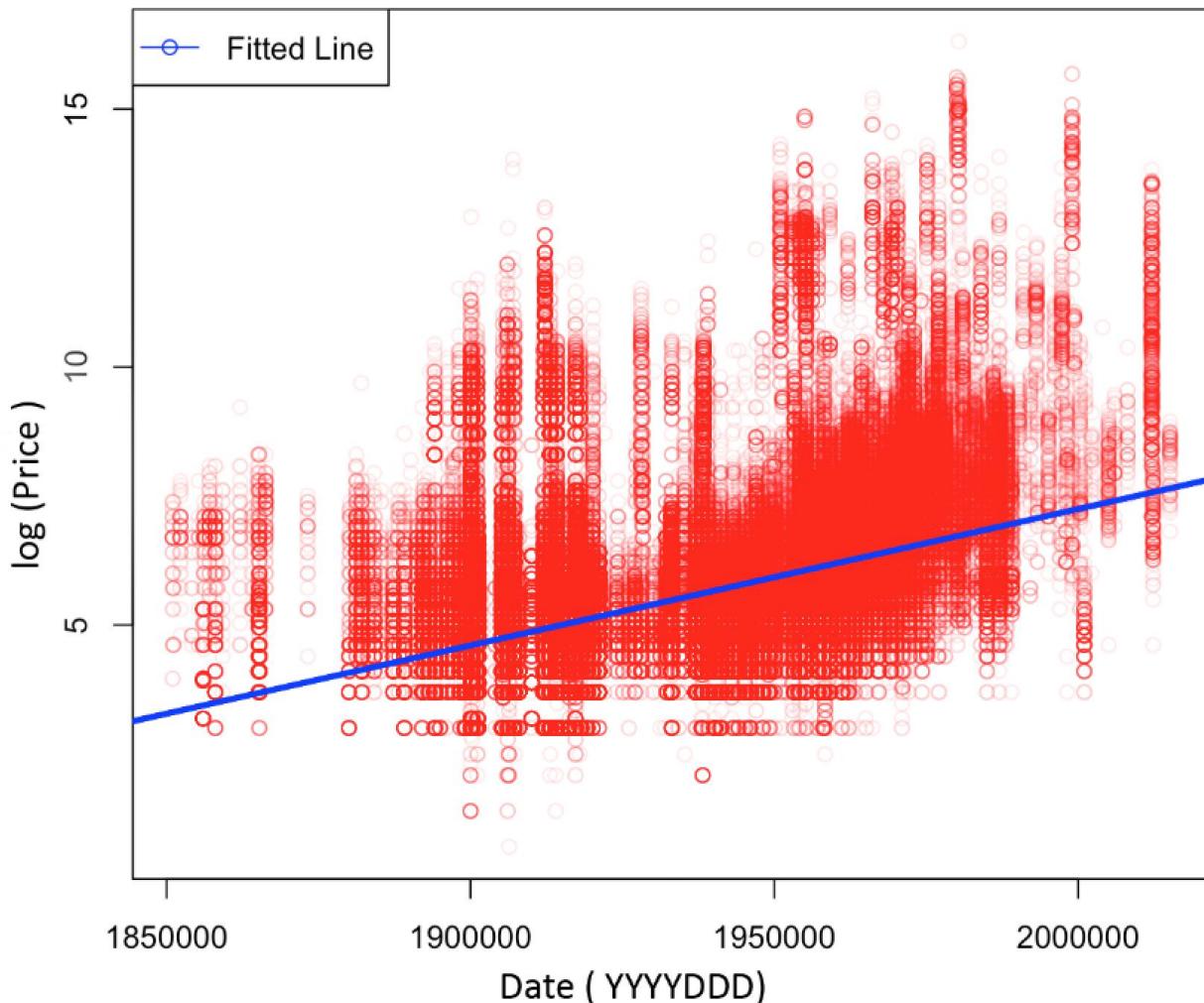
```
24. df = df[df.n_date < 2200000]
25. df.to_csv("menu_price_date_regression_final.csv")
```

We have removed the outliers and performed a regression analysis.

### 6.2.3 R Code for Linear Regression

We used R to perform the regression analysis and built a linear model regressing the log(Price) against the date. The input data file `menu_price_date_regression_final.csv` used for the regression can be downloaded [here](#).

- `data=read.csv(file='menu_price_date_regression_final.csv')`
- `model=lm(n_price~n_date,data=data)`
- `plot(n_price~n_date,data=data,col=rgb(1,0,0,.1),xlab='Date',`
- `ylab='Price',)`
- `abline(model,col=rgb(0,0,1),lwd=3)`



#### 6.2.4 Analysis

- We can say from the plotted data that there is a relationship between the price and date.
- The fitted regression line shows the estimated line to explain the change of price with date.

### 6.3 Visualization (Dirty Vs Clean Data)

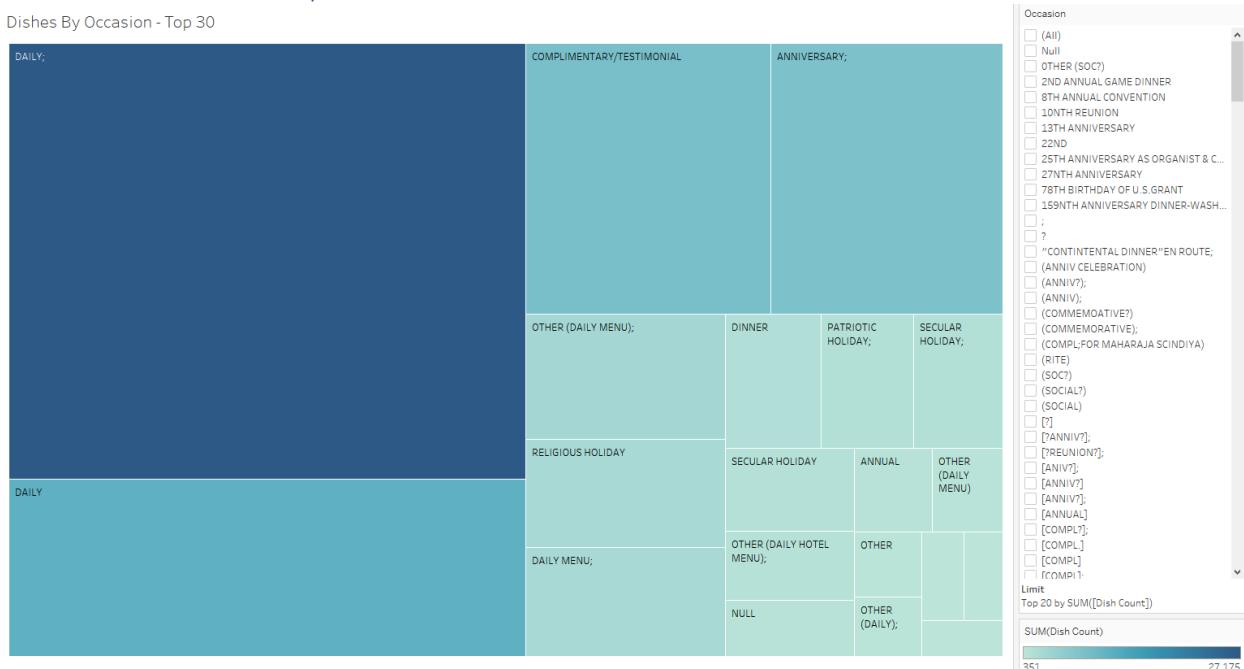
We created visualizations in **Tableau** using the data extracted after the SQLite cleanup to draw a complete picture of how food evolved during the late 19<sup>th</sup> and early 20<sup>th</sup> century.

We have attempted to model the price, however there are still several items where the price was not valid. We addressed some of our initial use cases/questions, though not all could be addressed due to incomplete nature of the missing data. These complete visualization can be viewed on these links: [before cleanup](#) and [after cleanup](#).

#### 6.3.1 Top 30 Dishes organized by Occasion/Special Event

##### 6.3.1.1 Prior to Cleanup

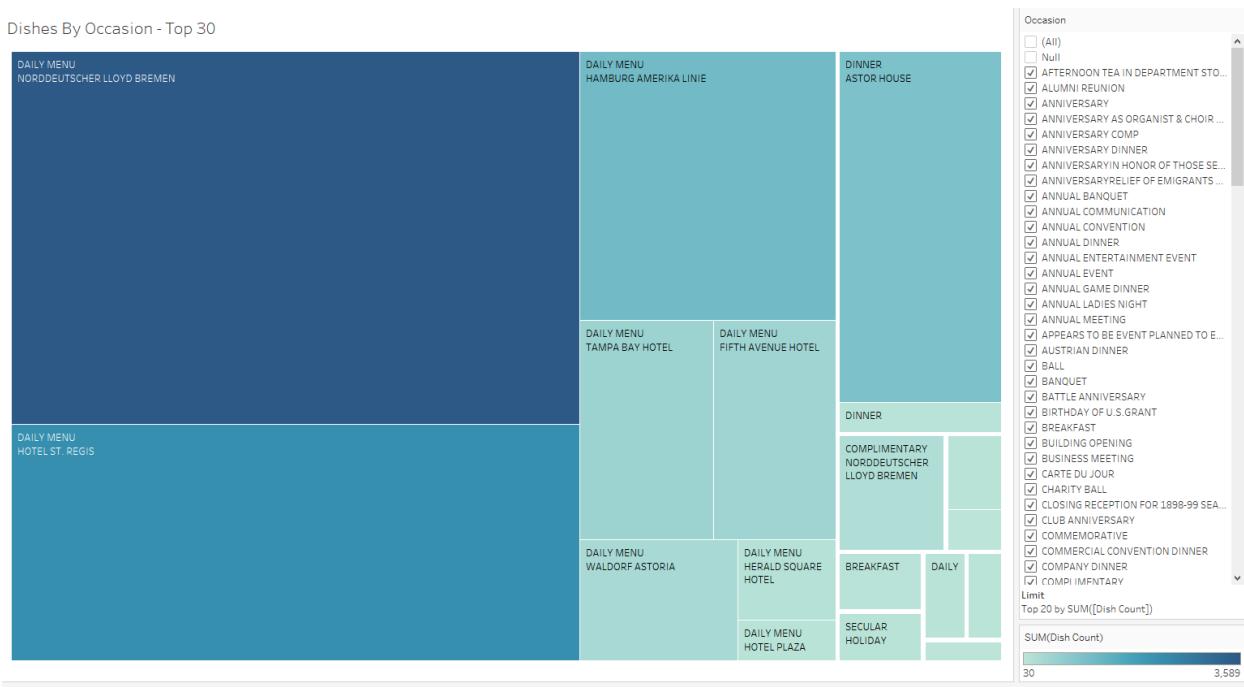
Dishes By Occasion - Top 30



Has lot of null entries and the left-hand panel invalid and special characters that are not meaningful for extracting any results.

##### 6.3.1.2 After Cleanup

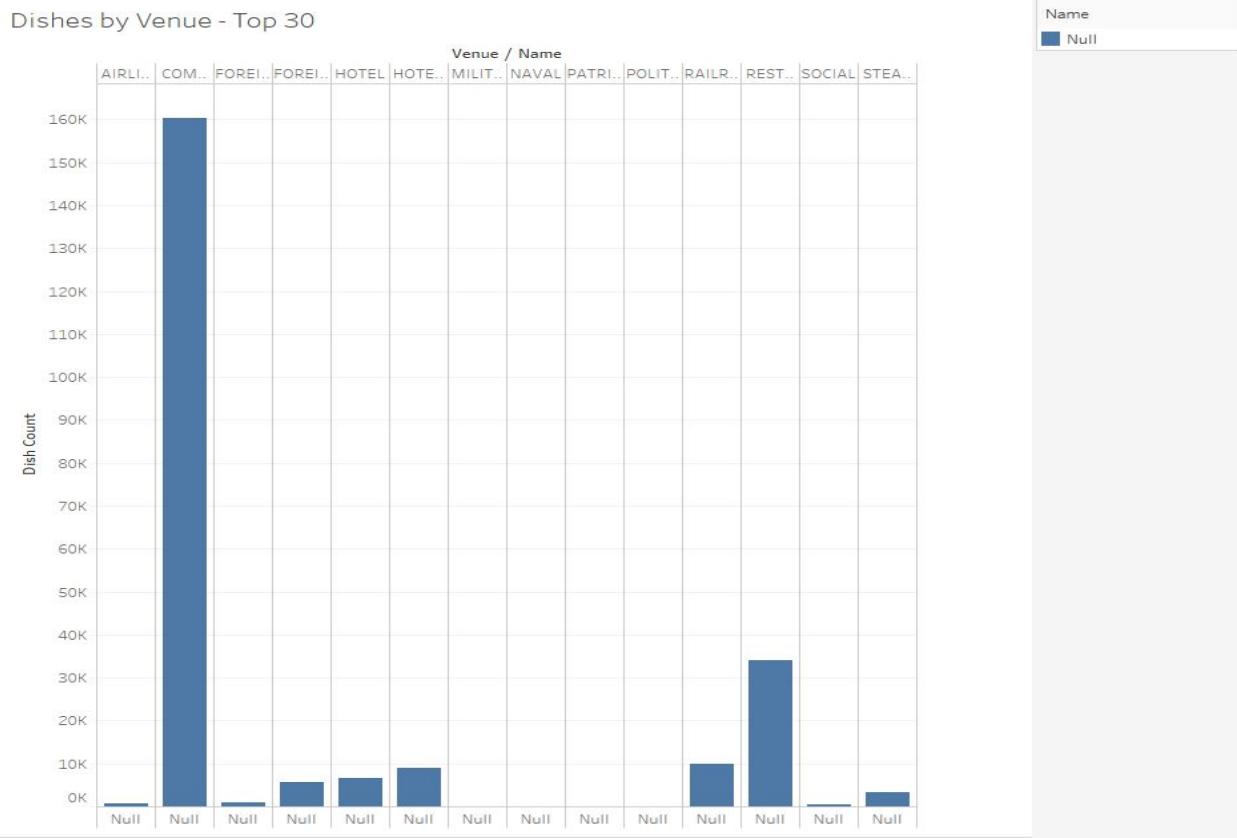
Dishes By Occasion - Top 30



### 6.3.2 Top 30 Dishes and the venues where they appeared

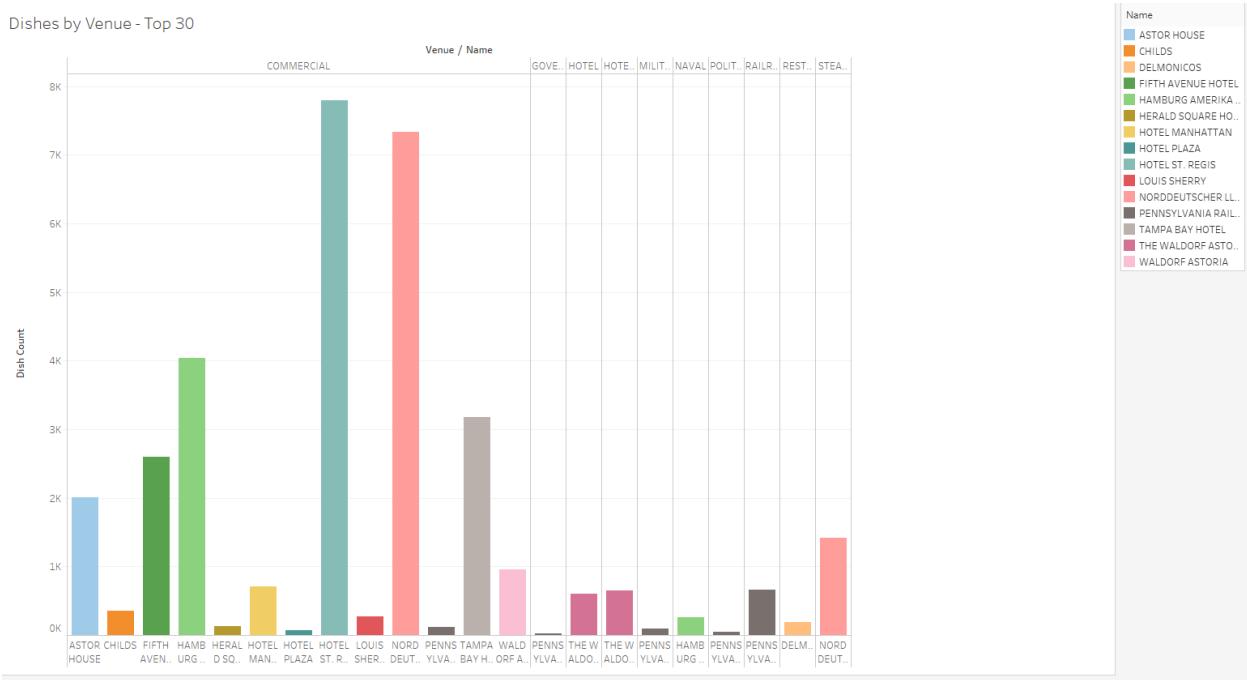
#### 6.3.2.1 Prior to Cleanup

Dishes by Venue - Top 30



### 6.3.2.2 After Cleanup

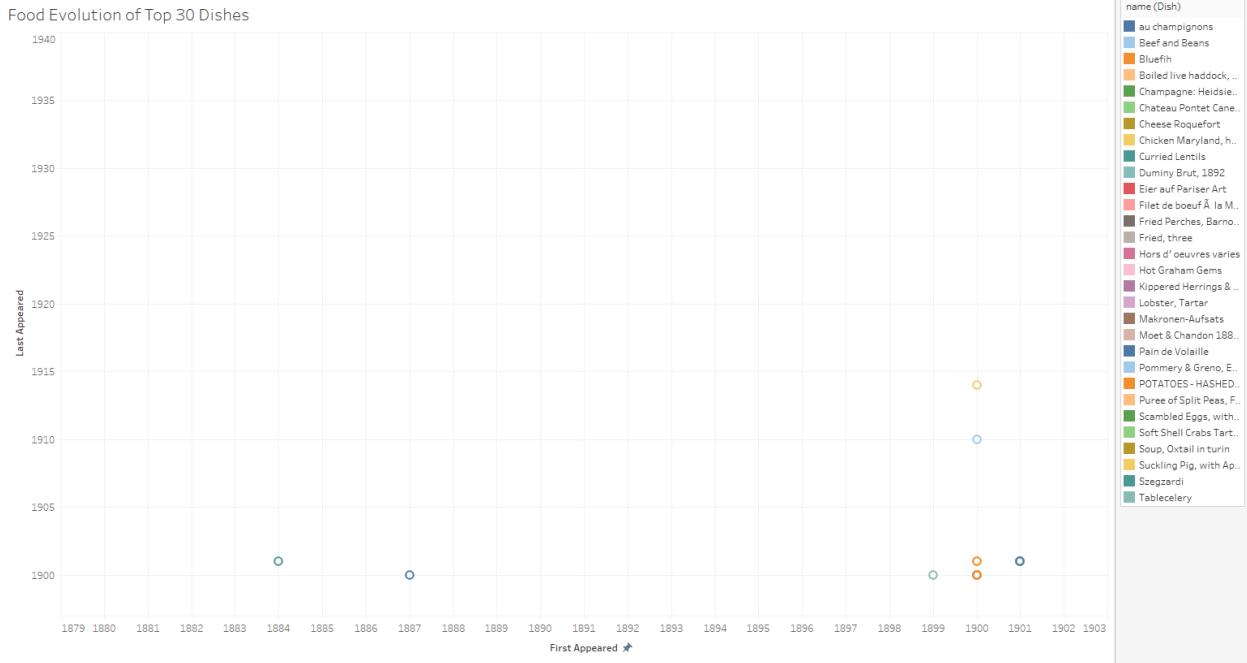
Dishes by Venue - Top 30



### 6.3.3 Food Evolution of Top 30 Dishes

#### 6.3.3.1 Prior to Cleanup

Food Evolution of Top 30 Dishes



#### 6.3.3.2 After Cleanup



### 6.3.4 Top 30 popular Dishes of all times

#### 6.3.4.1 Prior to Cleanup

How popular was a Dish? Top 30



name (visn)	
French Pastry	659
Gorgonzola Cheese	435
FRENCH DRESSING	373
Candied Sweet Potatoes	365
Stuffed celery	355
Green Turtle Soup	337
Cape Cod Oysters	300
Stuffed Tomato	261
Cream of tomato soup	257
Chocolate Parfait	239
Pistachio Ice Cream	236
Raspberry sherbet	218
Fancy Ices	215
Dutch cheese	214
Rolls and butter	204
Cheddar Cheese	201
Heart of Lettuce Salad	178
Figs in syrup	169
Broiled Squab	166
Assorted Fresh Fruit	159
Chocolate Sundae	155
Capon	145
French Sardines	139
Pineapple Cheese	132
Grilled sweet potatoes	128
Force	127
Beets in Butter	116
Two Fried Eggs	115
Kakao	114
Kraftbruhe	113

This is the only visualization that did not have any major missing entries mainly because we are viewing only the top 30 dishes. There were several missing but did not just fall under the top 30 visualized here. The cleanup helped fix all these gaps.

### 6.3.4.2 After Cleanup

How popular was a Dish? Top 30



Click on these links to view the entire visualization [before cleanup](#) and [after cleanup](#).

## 7 Observations and Conclusions

### 7.1 Source and What We Cleaned

The quality of the originally provided dataset was very poor. It is not a surprise because the data was crowd sourced and in some cases the source is at least a few decades old. We used OpenRefine to introduce basic but essential transformation such as consistent formatting for dates, regular expressions to remove punctuation/non-alphanumeric characters and various clustering features to merge similar words and phrases. We took opportunity to eliminate columns with no values and merge duplicate columns. It helped reduced the size of the dataset without impacting the structure of the dataset.

### 7.2 Integrity Constraints and Other issues in SQLite

Despite the deep cleaning of various columns in OpenRefine, we encountered integrity issues after importing it into SQLite. The empty, null and zero value columns made it harder to import all rows of the dataset.

### 7.3 What we learned

This project made us appreciate the challenges and nuances of data cleaning. Data cleaning involves multiple steps and typically requires multiple iterations before data is clean enough to be used for further analytics.

The semantic understanding is just as important as the syntactic understanding of the dataset. While we can easily infer the intent of the date column and why it is part of the dataset, we had tough time figuring out the purpose and intent of other columns such as – call\_number. There is no universal definition of clean data and it will vary from one user to another. We believe that we have improved quality of the dataset by a significant margin and it is ready to be used for deeper analytics.

### 7.4 How long did the process take?

The data cleaning process is a time-consuming process. It took us 16 hours to go through multiple iterations to clean up the data in OpenRefine. It took us another 8 hours to import data into SQLite, apply additional transformations and integrity constraints. The YesWorkflow portion took about 4 hours and tableau visualization process took another 4 hours.

### 7.5 Use cases answers

- The desserts and appetizers were more popular than the actual entrees.
- French Pastry was the most popular dish of all time.
- Hotel St. Regis served most dishes amongst all venues.
- Daily menus were more popular than meal only menus.
- Desserts and appetizers have shown remarkable resilience and continued to be a popular choice over multiple decades. For instance, Raspberry Sherbet and Pistachio ice cream lasted almost 90 years.
- Cream of tomato soup lasted 95 years.
- Lettuce Salad continued to be popular all the way through 2012.
- French dressing always accompanied lettuce salad. The likely explanation is that it was the only dressing offered on the menu.
- Many of the most popular dishes in the data set are single ingredients or beverages, rather than full entrées, something that is indicative of the food tastes of the upper class as most of the data is from high end hotels and cruise lines.
- We performed a basic regression analysis using least squares method and found a linear trend between the price on a log scale and the date.

## 8 Links

- Original dataset <https://uofi.box.com/s/8i7ydcraf6s6h7ihypat4d7169g1h72h4>
- OpenRefine's operation history can be downloaded from here:
  - [menuOperationHis.json](#)– history of changes for Menu CSV
  - [dishOperationHis.json](#)– history of changes for Dish CSV
  - [menuItemOperationHis.json](#)– history of changes for MenuItem CSV
  - [menuPageOperationHis.json](#)– history of changes for MenuPage CSV
- The output CSV files after OpenRefine transformation can be downloaded here:

- [CS598Project-Menu.csv](#)– updated Menu CSV output from OpenRefine
  - [CS598Project-Dish.csv](#)– updated Dish CSV output from OpenRefine
  - [CS598Project-MenuItem.csv](#)– updated MenuItem CSV output from OpenRefine
  - [CS598Project-MenuPage.csv](#) – updated MenuPage CSV output from OpenRefine
- 
- SQLite DB dump can be found [here](#).
  - Data Extract can be found here:
    - [menu\\_extract.csv](#)– Database extract for menu table
    - [dish\\_extract.csv](#)– Database extract for dish table
    - [menu\\_item\\_extract.csv](#)– Database extract for menu\_item table
    - [menu\\_page\\_extract.csv](#)– Database extract for menu\_page table
  - YesWorkflow [script](#).
  - YesWorkflow [diagram](#).
  - Tableau visualization of the dataset
    - [Before cleanup](#)
    - [After cleanup](#)