CS513: THEORY & PRACTICE OF DATA CLEANING

Final Project - Phase II

Authors:

Ji Ma (Jay): jima2@illinois.edu

Yutong Wang (Andy): yutong9@illinois.edu

Minjie Fu (Jane): minjief2@illinois.edu

The purpose of this document is to propose an end-to-end data cleaning workflow, in practice of the data cleaning and provenance tools and techniques introduced in course CS513.

1. Group Members

Ji Ma (Jay): jima2@illinois.edu
Yutong Wang(Andy): yutong9@illinois.edu
Minjie Fu (Jane) : minjief2@illinois.edu

2. Database Selected

The database we used is NYPL-menus. It is a project named " What's on the Menu " NYPL (New York Public Library) labs launched in late April, 2011. The project aimed at scanning and transcribing all the 45,000 menus contained in the New York Public Library's menu collection dating from the 1840s to the present. Till now, a quarter of them have been digitized and collected in this database.

3. Database Description

The database contains 4 tables: Menu, MenuPage, MenuItem, and Dish.

a) Menu

The table "Menu" holds general information regarding the restaurants and the menus they offer. The attribute "id" is the unique identifier for each menu, and will be used as the primary key. Other information include the name of the restaurant, the sponsor, the location, the occasion, the physical description of the menu, number of menu pages, number of dishes on the menu, etc. The table's schema is shown as follows:

```
CREATE TABLE menu (
id BIGINT PRIMARY KEY NOT NULL,
name TEXT,
sponsor TEXT,
event TEXT,
venue TEXT,
place TEXT,
physical_desc TEXT,
occasion TEXT,
notes TEXT,
call_number TEXT,
date DATE,
location TEXT,
currency TEXT,
currency_symbol TEXT,
```

```
status TEXT,
page_count INTEGER,
dish_count INTEGER
);
```

b) MenuPage

The table “MenuPage” contains information regarding each page of the menu. Each page is identified by a unique id. The attribute “menu_id” shows on which menu the page appears. It is used as the foreign key to link the table “MenuPage” and the table “Menu”. Other information include the height and width of the page, the page number, the scanned image id, etc. The table’s schema is shown as follows:

```
CREATE TABLE menupage (
id BIGINT PRIMARY KEY NOT NULL,
menu_id BIGINT REFERENCES menu (id),
page_number INTEGER,
image_id INTEGER,
full_height INTEGER,
full_width INTEGER,
uuid TEXT
);
```

c) MenuItem

The table “MenuItem” contains information regarding the items shown on the menu. Each item is identified by a unique id. The attribute “menu_page_id” lists the id of the menu page on which the item appears. This attribute can be used as the foreign key to link the table “MenuItem” and the table “Menupage”. Also, the attribute “dish_id” lists the id of the dish that the item is assigned to, which can be used as the foreign key to link the table “MenuItem” and the table “Dish”.

Other information include the price of the item, the time it was created, and other meta information such as x pos, y pos, which is the position that the item is located on the scanned image. The table’s schema is shown as follows:

```
Table : menuitem
CREATE TABLE menuitem (
id BIGINT PRIMARY KEY NOT NULL,
menu_page_id BIGINT REFERENCES menupage (id),
price DOUBLE,
high_price DOUBLE,
dish_id BIGINT REFERENCES dish (id),
created_at DATE,
```

```
updated_at DATE,
xpos DOUBLE,
ypos DOUBLE
);
```

d) Dish

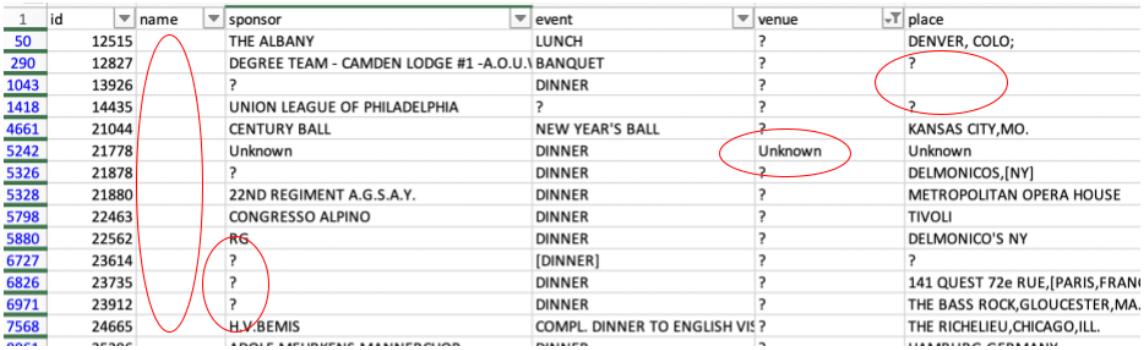
The table “Dish” contains information regarding each dish on the menu. The attribute “id” is the unique identifier for each dish, and will be used as the primary key. Other information includes the name of the dish, the number of times it appears on the menus, the first and last year it appears, its lowest and highest prices, etc. The table’s schema is shown as follows:

```
CREATE TABLE dish (
id BIGINT PRIMARY KEY NOT NULL,
name TEXT,
menus_appeared INTEGER,
times_appeared INTEGER,
first_appeared DATETIME,
last_appeared DATETIME,
lowest_price DOUBLE,
highest_price DOUBLE
);
```

4. Quality issues

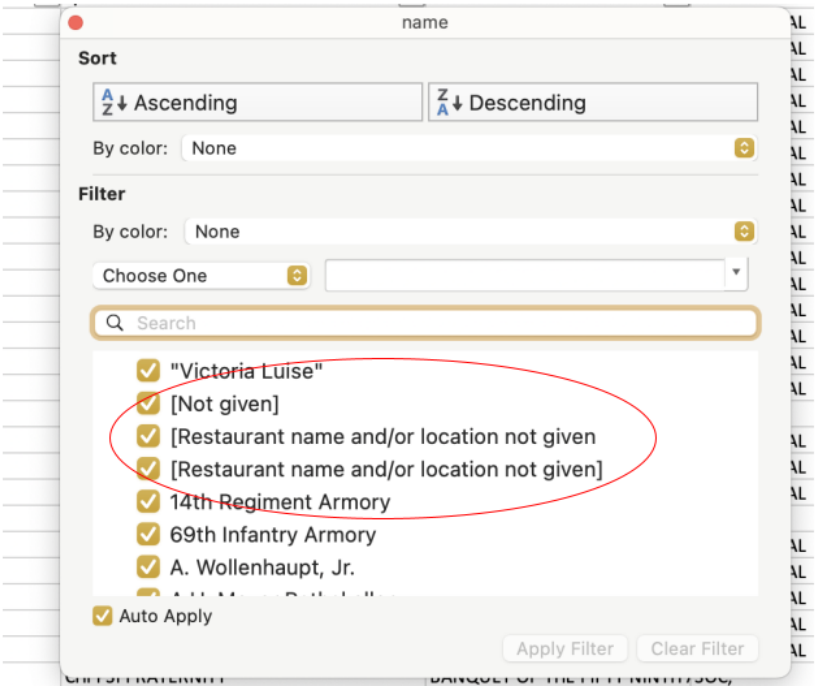
We took an initial inspection of the dataset by using the Filter Function in Excel. We had to admit that the data quality is poor. We spotted and listed some obvious data quality problems as follows:

a) Missing Values and Inconsistent Handling Methods



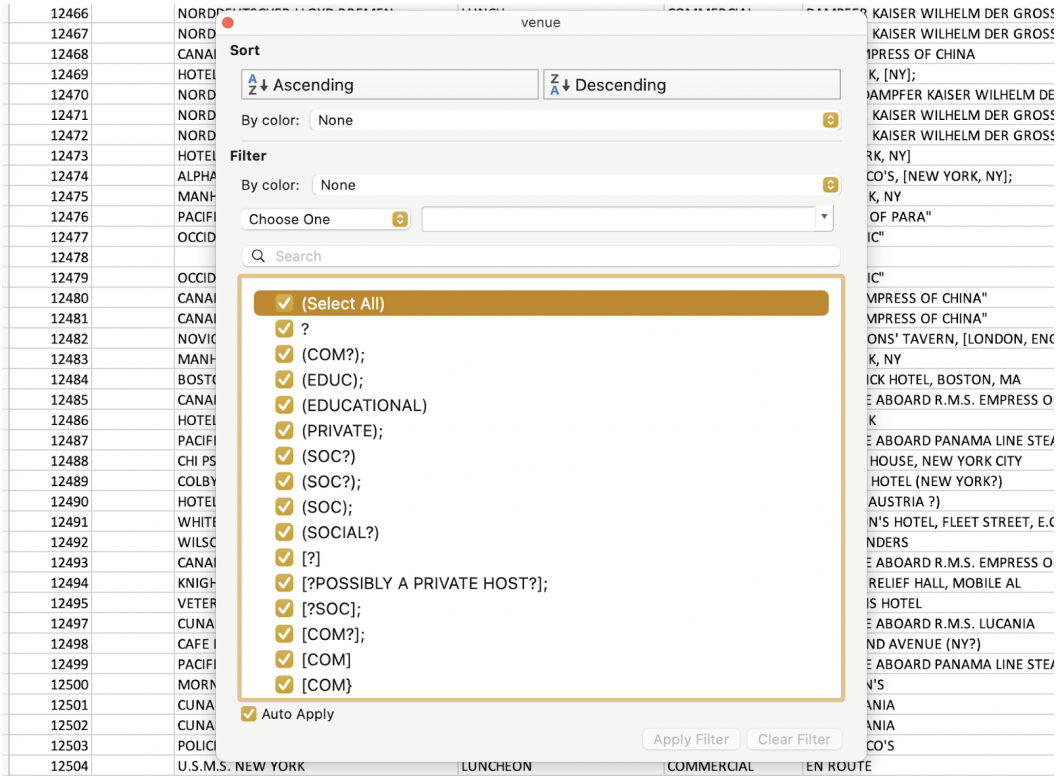
1	id	name	sponsor	event	venue	place
50	12515		THE ALBANY	LUNCH	?	DENVER, COLO;
290	12827		DEGREE TEAM - CAMDEN LODGE #1 -A.O.U.\	BANQUET	?	?
1043	13926		?	DINNER	?	
1418	14435		UNION LEAGUE OF PHILADELPHIA	?	?	?
4661	21044		CENTURY BALL	NEW YEAR'S BALL	?	KANSAS CITY,MO.
5242	21778		Unknown	DINNER	Unknown	Unknown
5326	21878		?	DINNER	?	DELMONICOS,[NY]
5328	21880		22ND REGIMENT A.G.S.A.Y.	DINNER	?	METROPOLITAN OPERA HOUSE
5798	22463		CONGRESSO ALPINO	DINNER	?	TIVOLI
5880	22562		RG	DINNER	?	DELMONICO'S NY
6727	23614		?	[DINNER]	?	?
6826	23735		?	DINNER	?	141 QUEST 72e RUE,[PARIS,FRAN
6971	23912		?	DINNER	?	THE BASS ROCK,GLOUCESTER,MA.
7568	24665		H.V.BEMIS	COMPL. DINNER TO ENGLISH VIS	?	THE RICHELIEU,CHICAGO,ILL.

There are a lot of missing values in the dataset, and different methods are used to deal with these missing values. For example, in the table “Menu”, some of the missing values are left blank. Some of them are filled in with “?” or “Unknown”. In the column “name”, it uses the description of “[Not given]” or “[Restaurant name and/or location not given]” to handle the missing values.



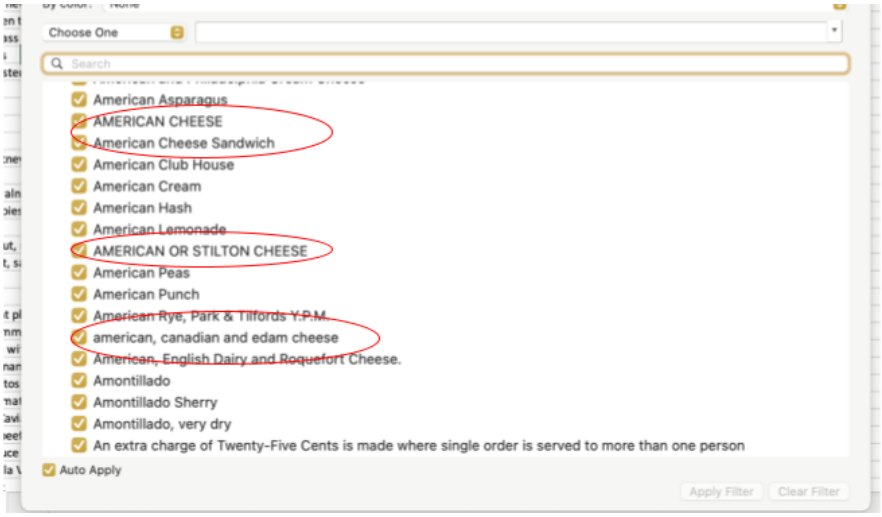
b) Leading/Trailing Special Characters

Inconsistent special characters are used at the leading and trailing positions through the whole dataset. Take the column “venue” in the table “Menu” as an example. Some fields are enclosed with special characters such as “()”, “[]”, “;” and “?”, which causes the content inconsistency.



c) Inconsistent Upper/Lower Case

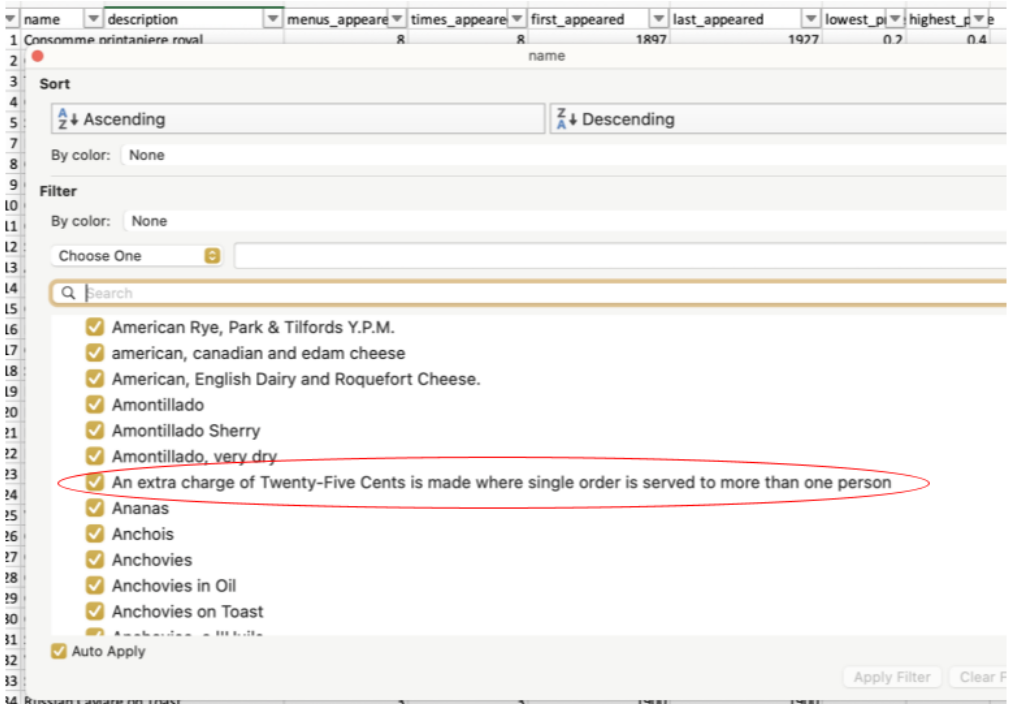
The dataset inconsistently uses uppercase and lowercase letters. Some fields capitalize the first letter, some fields capitalize all the letters, others use all lowercase letters. The following chat highlight an example spotted in the column “name” in the table “Dish”.



d) Spelling and Entry Errors

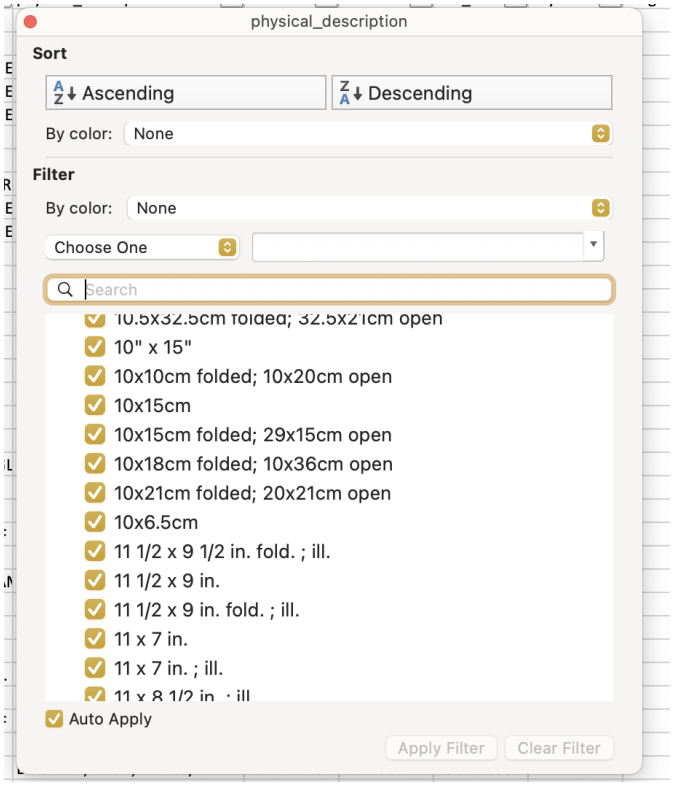
NORDDEUTSCHER LLOYD BREMEN	LUNCH	COMMERCIAL
NORDDEUTSCHER LLOYD BREMEN	[DINNER]	COMMERCIAL
HOTEL MARLBOROUGH	CAFE LUNCHEON	COMMERCIAL
ALPHA OF ZETA PSI	ANNUAL BANQUET	COMMERCIAL
MANHATTAN HOTEL	DINNER	COMMERCIAL
PACIFIC MAIL STEAMSHIP COMPANY	DINNE	COMMERCIAL
OCCIDENTAL & ORIENTAL	BREAKFAST	COMMERCIAL
OCCIDENTAL & ORIENTAL STEAMSHIP COMP.	DINNER	COMMERCIAL
CANADIAN PACIFIC RAILWAY COMPANY	BREAKFAST	COMMERCIAL

The dataset contains a lot of spelling errors. For example, “DINNER” is spelled as “DINNE”. Also, the content of some fields does not match the corresponding column’s header. For example, shown in the following chart, the highlighted content should be filled in the column “description”, not the column “name”. It is obviously an entry error.



e) Different measurement units and language

The measurement units used through the dataset are not consistent. For example, in the column “physical_description” of the table “Menu”, some fields use “cm”, while others use “in”. The inconsistent measurement units are misleading. Meanwhile, we found some dishes’ names use English, while others use Italian, French or other languages, which may also lead to data quality problems.



f) Logical mistakes:

Some data are obviously wrong in logic. For example, “0” for price, “1” for first_appeared year, “2928” for first_appeared year or last_appeared year, “0001-01-01” for the date, etc.

	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
1	Consomme printaniere royal		8	8	1897	1927	0.2	0.4
2	Chicken gumbo		111	117	1895	1960	0.1	0.8
3	Tomato aux croutons		13	13	1893	1917	0.25	0.4
4	Onion au gratin		41	41	1900	1971	0.25	1
5	St. Emilion		66	68	1881	1981	0	18
7	Radishes		3262	3346	1854	2928	0	25
8	Chicken soup with rice		48	49	1897	1961	0.1	0.6
9	Clam broth (cup)		14	16	1899	1962	0.15	0.4
10	Cream of new asparagus, croutons		2	2	1900	1900	0	0
11	Clear green turtle		157	157	1893	1937	0.25	60
12	Striped bass saute, meuniere		2	2	1900	1900	0	0
13	Anchovies		453	484	1858	1987	0	30
14	Fresh lobsters in every style		4	4	1899	1900	0	0
15	Celery		4246	4690	1	2928	0	50
16	Pim-olas		145	148	1897	1918	0.15	35
17	Caviar		505	534	1880	1987	0	75
18	Sardines		1425	1484	1856	2928	0	50
19	India chutney		16	16	1865	1901	0.1	0.2
20	Pickles		453	472	1852	1987	0	10
21	English walnuts		83	86	1851	1948	0.1	0.3
22	Pate de foies-gras		9	9	1898	1901	1	1
23	Pomard		11	11	1880	1950	0.75	5
24	Brook trout, mountain style		2	2	1900	1900	0	0
25	Whitebait, sauce tartare		4	4	1900	1901	0.3	0.75
26	Clams		170	180	1881	1970	0.1	0.9
27	Oysters		289	292	1862	1963	0	35
28	Claremont planked shad		2	2	1899	1900	1.5	2.5
29	G. H. Mumm & Co's Extra Dry		14	14	1895	1914	2	4
30	Cerealine with Milk		1	1	1901	1901	0	0
31	Sliced Bananas		220	238	1900	1987	0	15
32	Wheat Vitos		3	3	1900	1900	0	0
33	Sliced Tomatoes		1195	1256	1873	2928	0	25
34	Russian Caviare on Toast		3	3	1900	1900		
35	Smoked beef in cream		21	21	1896	1933	0.3	0.9
38	Apple Sauce		721	829	1	1987	0	20
39	Potage a la Victoria		5	5	1899	1901		

g) Inconsistent format

This issue can be seen throughout the dataset. For example, as is shown in the following screenshot for the table “Menu”, the date format is inconsistent, some use ISO date format (YYYY-MM-DD), while others use different date formats, such as MM/DD/YY, YYYY, etc. It is the same case with the address format. Some listed the street, city, and country, while others listed the restaurant name instead. Some use the abbreviations, while others use the full forms.

F	M
place	date
[66 STREET AND BROADWAY,NEW YORK,	3/2/00
[66TH STREET AND BROADWAY,NEW YO	3/2/00
(NEW YORK?)	3/14/00
(NEW YORK,NY)	4/15/00
"DREAMLAND", CONEY ISLAND, NY	9/6/07
(NEW YORK, NY?)	6/20/05
(NY)	1899-12-25
"LITTLE HUNGARY" 257 EAST HOUSTON S	12/13/06
(NY)	3/30/01
(NEW YORK, NY?)	6/19/05
(NEW YORK,NY)	1897-03-18
(NEW YORK?)	1898-10-27
(NEW YORK?)	11/21/06
(NEW YORK,NY)	1897-03-31
(NY)	1899-02-26

In general, the data quality of this dataset is messy. We listed some obvious data quality issues spotted at our first glance. Surely, there are many other formatting or data quality problems, which will be addressed in our next phase, based on our use case and data cleaning goals, to achieve the desired “fitness for use”.

5. Use Cases

a) U0: The use case for which the original dataset is clean enough, thus data cleaning is not necessary

The descriptive data created for each entity when they were cataloged is digitally searchable, including useful information like the name of the restaurant, its geographical location, the physical description of its menu, etc. Although the dataset is dirty, we can use it to run some simple queries. For example, we can sort the column “times_appeared” in the table “Dish” to know which dish appeared the most number of times on the menus. The answer is “Coffee”, which appeared 8484 times in total. It makes sense to me. We can also search the “status” of a certain menu by using the

"menu id" and "status" attributes in the table "Menu". These 2 columns seem to be clean and complete. Another possible question the uncleaned dataset can help to answer is which dishes first appeared in 1900. We can get a list of the 20 dishes that first appeared in 1900 by running a quick query on the table "Dish".

b) U2: The use case for which the dataset will never be clean enough or usable, thus data cleaning is not sufficient

Data cleaning cannot resolve missing data or mis-transcribed data problems. Thus, the dataset can not be used to correctly answer certain queries if the related information is incomplete or inaccurate. For example, there is a lot of missing data in the column "first_appeared" of the table "Dish". Thus it may be impossible to answer a query such as "when the dish Celery first appeared on the menu", regardless how much data cleaning efforts we put in. The large amount of missing or unmatched values in the "price" related columns in the table "Dish" and the table "MenuItem" may also make a rigid study on price per each dish impossible.

c) U1: The target use case that matches our data cleaning goals, for which the data cleaning is both necessary and sufficient

After data cleaning, the dataset can be used for many use cases. The one we are interested in is to find out which 10 restaurants have the most menus collected in this dataset and which 10 dishes most often appeared on their menus. Another U1 use case we considered is to study the most popular dishes for a certain event (e.g. breakfast).

6. Data Cleaning Goals (matching the Target Use Case U1)

The goal of our data cleaning work is to "fit-for-purpose". We will clean the dataset to the extent that it can match our use case. We will demo our use case, analyze and visualize the results to show the difference brought by the data cleaning.

Based on our Target Use Case U1, we plans to take the following steps to clean the dataset:

(1) Which 10 restaurants have the most menus collected in this dataset?

To match this use case, we need to clean the table "menu", especially focusing on the column "name".

(2) Which 10 dishes most often appeared on the menus of the top restaurants?

To match this use case, we need to link or join the four tables. We need to focus on cleaning the column "name" in the table "dish"."

(3) Which 10 dishes are most popular for a certain event (e.g. breakfast)?

To match this use case, we need to link or join the four tables. We need to focus on cleaning the column “event” in the table “menu”, in addition to the two columns we cleaned for the previous use cases.

Besides, in order to successfully import the data into the schema we created in SQL, we need to convert the data type of the columns for all 4 tables. All the numeric fields will be converted to number type, and all the date fields will be converted to date type.

Lastly, We need to clean the special characters, like “ and ‘, because the SQL will report the errors when importing the data if the quotes are not paired.

7. Data Cleaning with OpenRefine

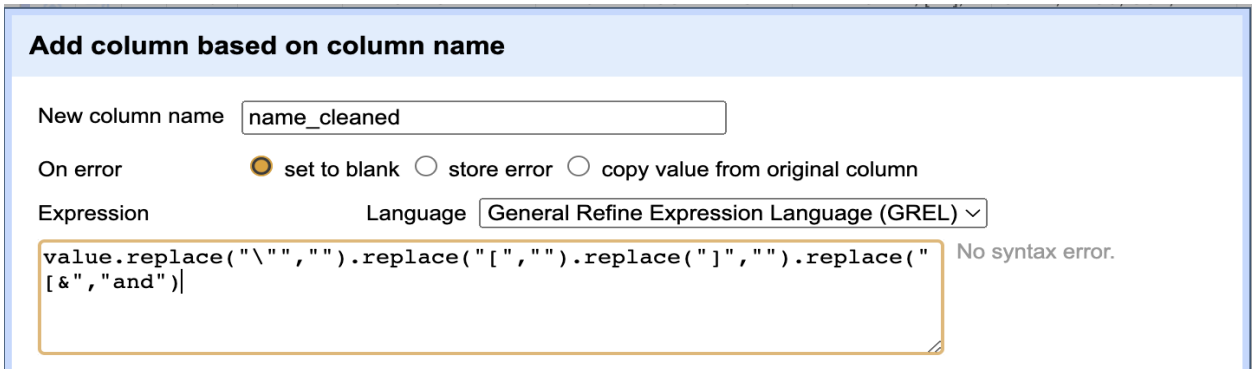
(1) Narrative of Data Cleaning Workflow

a) Menu.csv

17550 rows imported

Column “name”:

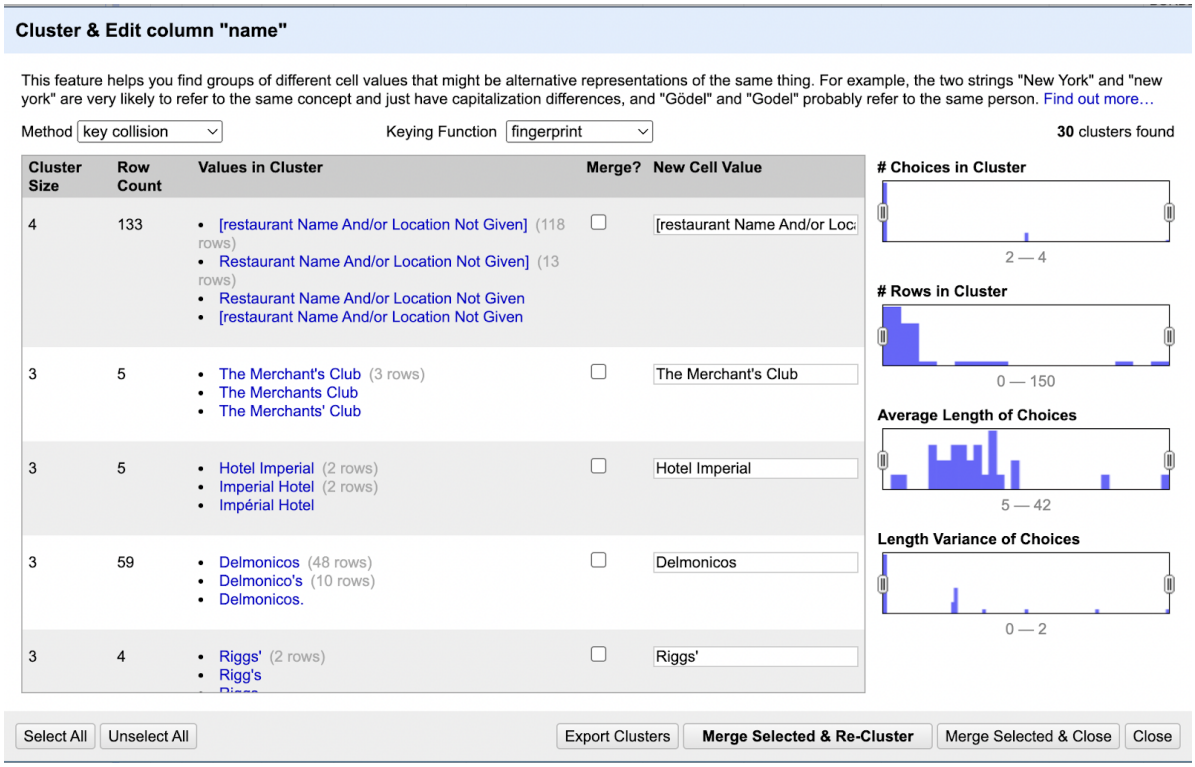
- Trim leading and trailing whitespace
- Collapse consecutive whitespace
- Transforms to title case
- GREL remove special characters, such as “ [ ] , and replace “&” with “and”



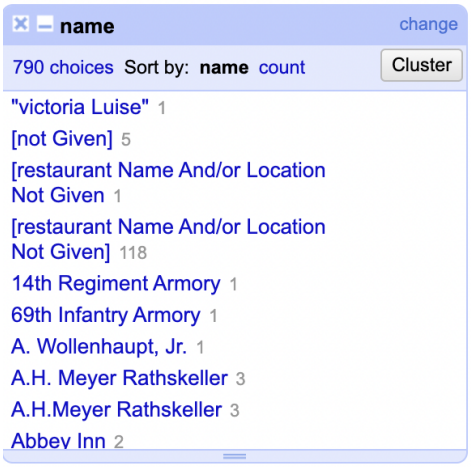
- Apply “Clustering”; Try the following methods and keying functions one one; Choose the ones that make sense.
    - Key-collision

i. fingerprint
ii. ngram-fingerprint
iii. metaphone-3
iv. cologne-phonetic
v. Daitch-Mokotoff
vi. Beider-Morse

- Nearest neighbor
  i. levenshtein
  ii. ppm



After cleaning, the text facet choices reduced from 790 to 681, which is a large improvement in the data quality.

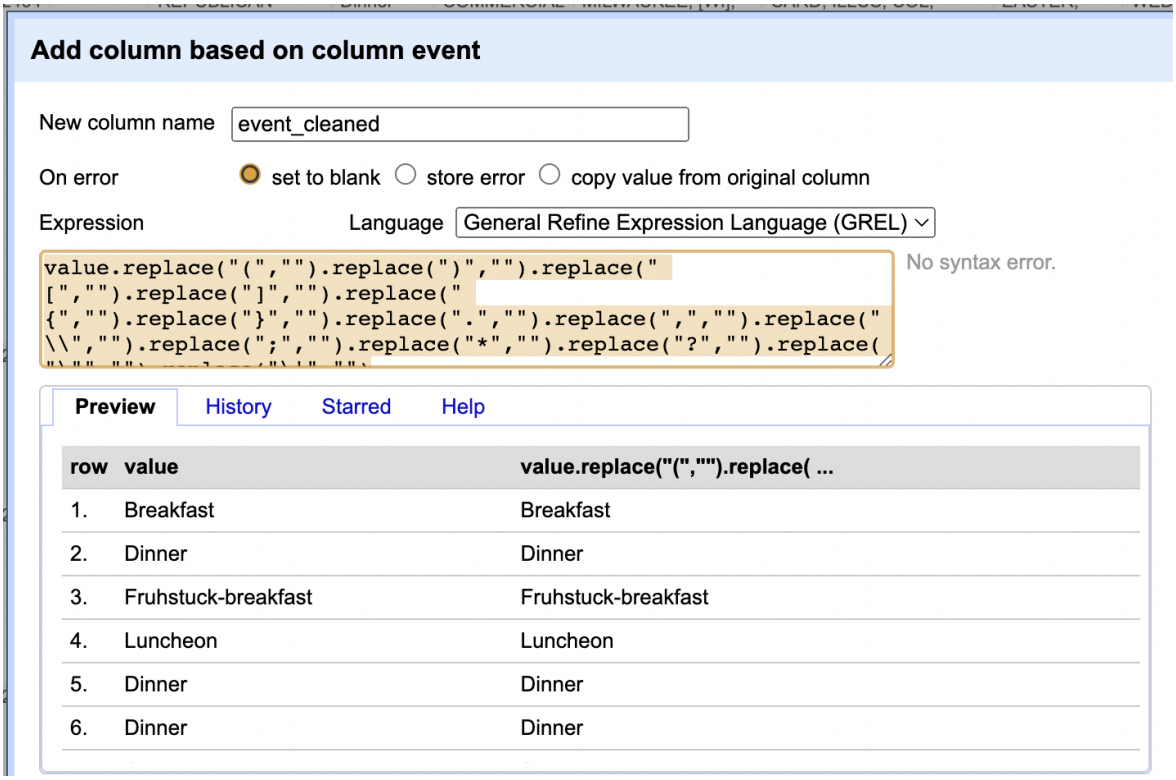


Column “event”:

- Trim leading and trailing whitespace
- Collapse consecutive whitespace
- Transforms to title case
- Apply “Clustering”; Try the following methods and keying functions one one; Choose the ones that make sense.
    - Key-collision
      i. fingerprint
      ii. ngram-fingerprint
      iii. metaphone-3
      iv. cologne-phonetic
      v. Daitch-Mokotoff
      vi. Beider-Morse
    - Nearest neighbor
      i. levenshtein
      ii. ppm

- GREL remove values () [] {} . , ; \ * ? ‘ “

```
value.replace("(","").replace(")","").replace("[","").replace("]","").replace("{","").replace("}","").replace(".","").replace(",","").replace("\\","").replace(";","").replace("*","").replace("?","").replace("\"","").replace("\'","")
```



The above cleaning steps largely improved the data quality. The text facet choices were reduced from 1734 to 1604.

Column “notes”:

- GREL remove values ‘ “

Column “date”:

- Transform data type to date

  Column “page_count”:

- Transform data type to number

  Column “dish_count”:

- Transform data type to number

  Column “keywords”:

- Remove the column

  Column “language”:

- Remove the column

  Column “location_type”:

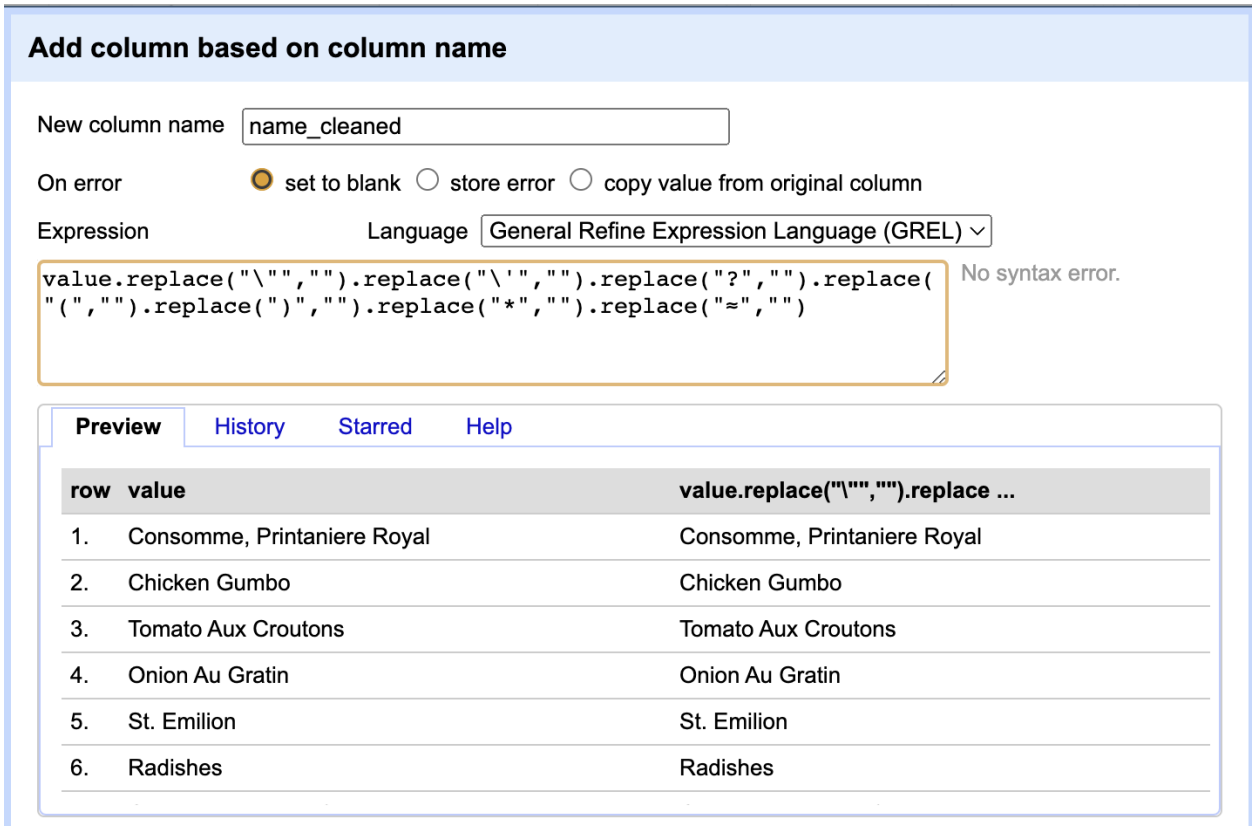
- Remove the column

b) Dish.csv

429956 rows imported

Column “name”:

- Trim leading and trailing whitespace
- Collapse consecutive whitespace
- Transforms to title case
- GREL remove special characters, such as “ ? () * ≈



- Apply “Clustering”:
  Because of the huge data size, we cannot try all the available methods and keying functions. Also, we cannot manually confirm whether each clustering makes sense or not. So, we only used the following methods and keying functions, and merge / reclustering automatically.

    a. Key-collision
       i. fingerprint
       ii. Ngram-fingerprint

Column “first_appeared”:

- Transform data type to number

Column “last_appeared”:

- Transform data type to number

Column “menus_appeared”:

- Transform data type to number

Column “times_appeared”:

- Transform data type to number

Column “lowest_price”:

- Transform data type to number

Column “highest_price”:

- Transform data type to number

c) MenuPage.csv

66937 rows imported

Column “page_number”:

- Transform data type to number

Column “full_height”:

- Transform data type to number

Column “full_width”:

- Transform data type to number

Column “uuid”:

- Transform to lowercase

d) MenuItem.csv

1334779 rows imported

Column “price”:

- Transform data type to number

Column “high_price”:

- Transform data type to number

Column “created_at”:

- Transform data type to date

Column “updated_At”:

- Transform data type to date

Column “xpos”:

- Transform data type to number

Column “ypos”:

- Transform data type to number

(2) Plan and Actual Comparison

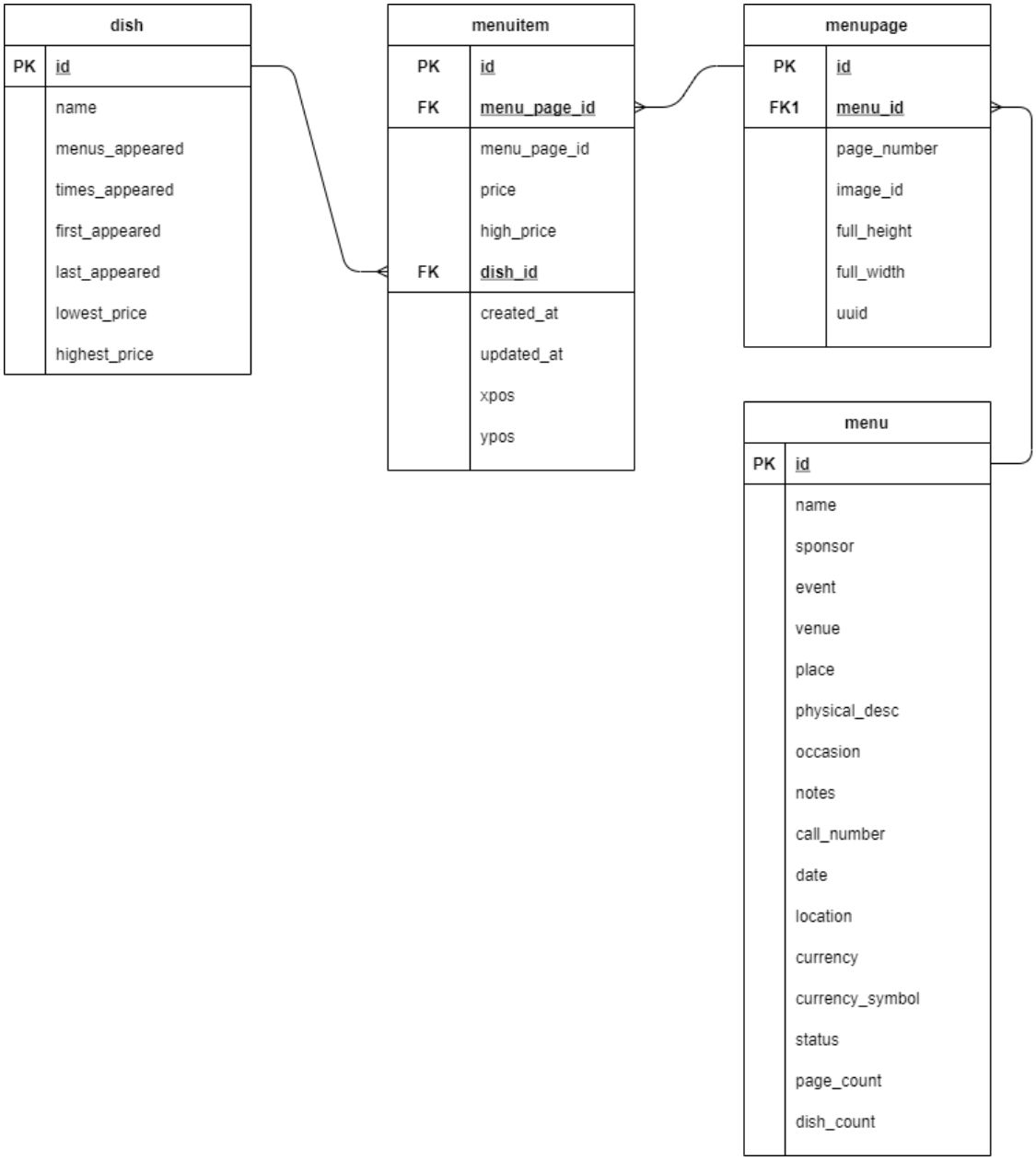
When creating the plan, we realized the huge size of the database, as well as the “messy” data quality. So we tried to make our target use cases specific and achievable. For example, we only focus on the top 10 restaurants or top 10 dishes. The results are still achievable even if we ignore some ambiguous information.

Our choice is proved to be correct. Basically, we did not change the plan. We successfully implemented our plan step by step. Although the file Dish.csv and the file MenuItem.csv are so large that they took openRefine a long time to process, it finally did the job.

8. Developing a Relational Schema with SQL

(1) ER Diagram

The database schema and the relationships between tables are shown in the following ER diagram.



(2) Creating Tables

Based on the ER diagram, we used the PostgreSQL run on pgAdmin to create tables. Listed below are scripts of creating tables for the 4 .csv files.

Dish.csv

```
CREATE TABLE dish (
id VARCHAR(255) PRIMARY KEY NOT NULL,
name VARCHAR(255),
description VARCHAR(255),
menus_appeared INTEGER,
times_appeared INTEGER,
first_appeared INTEGER,
last_appeared INTEGER,
lowest_price NUMERIC,
highest_price NUMERIC
);
```

Changed to VARCHAR(1500) due to unexpected large size of the values

Menu.csv

```
CREATE TABLE Menu (
id VARCHAR(255) PRIMARY KEY NOT NULL,
name VARCHAR(255),
sponsor VARCHAR(255),
event VARCHAR(255),
venue VARCHAR(255),
place VARCHAR(255),
physical_desc VARCHAR(255),
occasion VARCHAR(255),
notes VARCHAR(255),
call_number VARCHAR(255),
date DATE,
location VARCHAR(255),
currency VARCHAR(255),
currency_symbol VARCHAR(255),
status VARCHAR(255),
page_count INTEGER,
dish_count INTEGER
);
```

MenuItem.csv

```
CREATE TABLE menuitem (
id VARCHAR(255) PRIMARY KEY NOT NULL,
menu_page_id VARCHAR(255),
price NUMERIC,
high_price NUMERIC,
dish_id VARCHAR(255),
created_at DATE,
updated_at DATE,
xpos NUMERIC,
ypos NUMERIC
);
```

MenuPage.csv

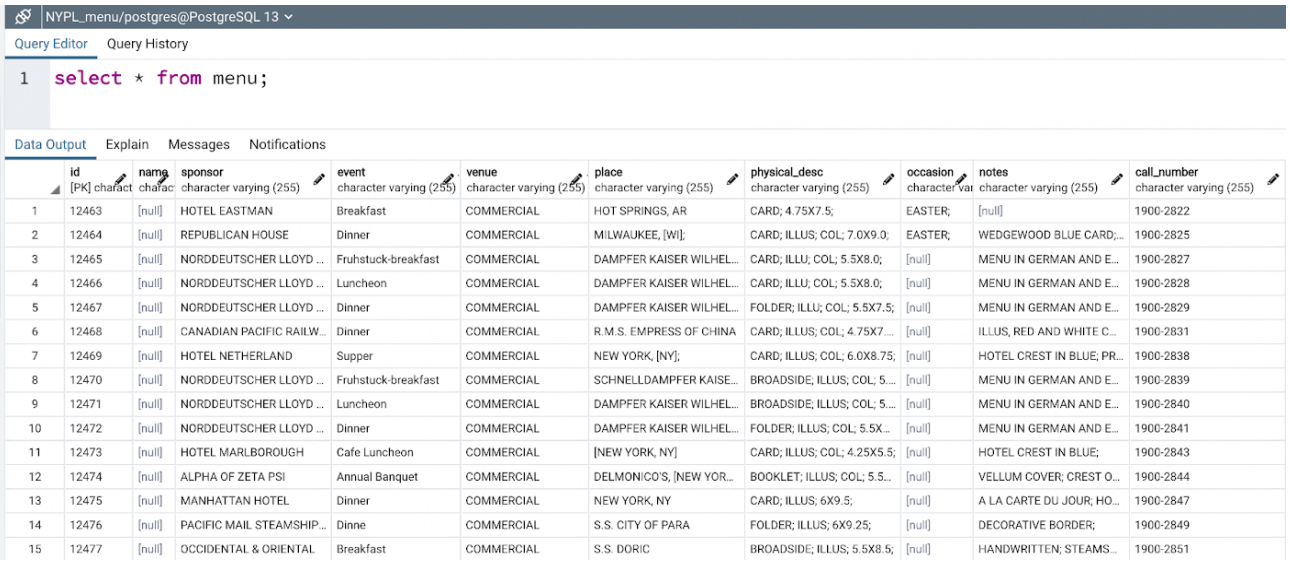
```
CREATE TABLE menupage (
id VARCHAR(255) PRIMARY KEY NOT NULL,
menu_id VARCHAR(255),
page_number INTEGER,
image_id VARCHAR(255),
full_height INTEGER,
full_width INTEGER,
uuid VARCHAR(255)
);
```

(3) Importing Data

After creating the tables, we imported the data into the tables. We did meet some problems when importing the data, like “unquoted carriage return”, “unterminated CSV quoted field”, and “extra data after the last expected column”. We went back to the data cleaning process and finally solved all the problems.

The following charts show the 4 tables imported with data.

Table Menu



NYPL_menu/postgres@PostgreSQL 13

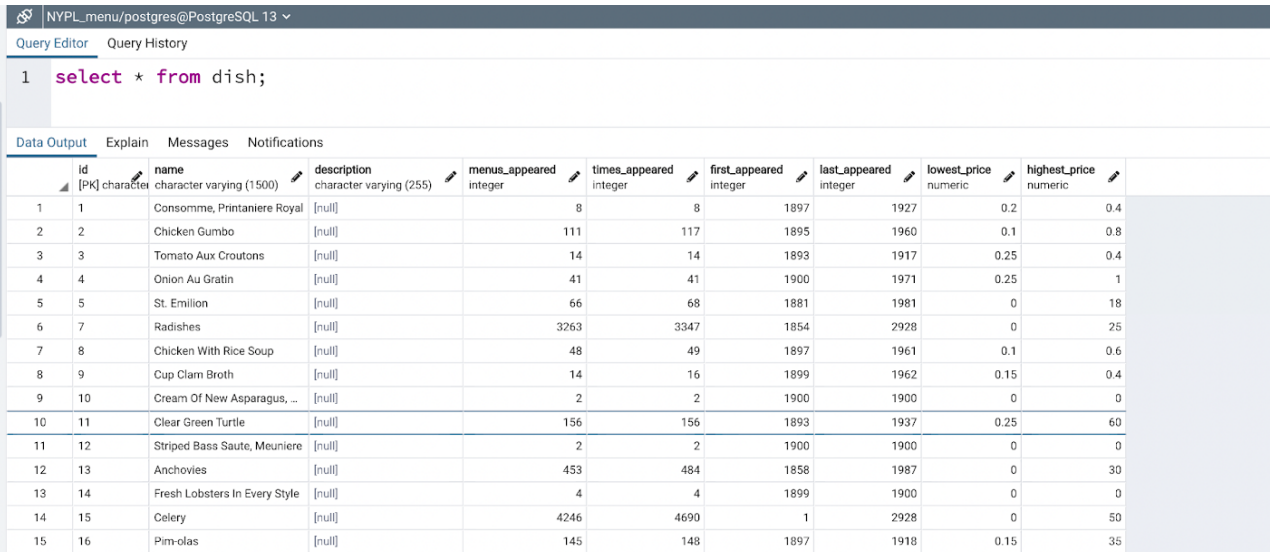
Query Editor | Query History

```
1  select * from menu;
```

Data Output | Explain | Messages | Notifications

	id [PK] charact	name charac	sponsor character varying (255)	event character varying (255)	venue character varying (255)	place character varying (255)	physical_desc character varying (255)	occasion character var	notes character varying (255)	call_number character varying (255)
1	12463	[null]	HOTEL EASTMAN	Breakfast	COMMERCIAL	HOT SPRINGS, AR	CARD; 4.75X7.5;	EASTER;	[null]	1900-2822
2	12464	[null]	REPUBLICAN HOUSE	Dinner	COMMERCIAL	MILWAUKEE, [WI];	CARD; ILLUS; COL; 7.0X9.0;	EASTER;	WEDGEWOOD BLUE CARD;...	1900-2825
3	12465	[null]	NORDDEUTSCHER LLOYD ...	Fruhstuck-breakfast	COMMERCIAL	DAMPFER KAISER WILHEL...	CARD; ILLU; COL; 5.5X8.0;	[null]	MENU IN GERMAN AND E...	1900-2827
4	12466	[null]	NORDDEUTSCHER LLOYD ...	Luncheon	COMMERCIAL	DAMPFER KAISER WILHEL...	CARD; ILLU; COL; 5.5X8.0;	[null]	MENU IN GERMAN AND E...	1900-2828
5	12467	[null]	NORDDEUTSCHER LLOYD ...	Dinner	COMMERCIAL	DAMPFER KAISER WILHEL...	FOLDER; ILLU; COL; 5.5X7.5;	[null]	MENU IN GERMAN AND E...	1900-2829
6	12468	[null]	CANADIAN PACIFIC RAILW...	Dinner	COMMERCIAL	R.M.S. EMPRESS OF CHINA	CARD; ILLUS; COL; 4.75X7....	[null]	ILLUS, RED AND WHITE C...	1900-2831
7	12469	[null]	HOTEL NETHERLAND	Supper	COMMERCIAL	NEW YORK, [NY];	CARD; ILLUS; COL; 6.0X8.75;	[null]	HOTEL CREST IN BLUE; PR...	1900-2838
8	12470	[null]	NORDDEUTSCHER LLOYD ...	Fruhstuck-breakfast	COMMERCIAL	SCHNELLDAMPFER KAISE...	BROADSIDE; ILLUS; COL; 5....	[null]	MENU IN GERMAN AND E...	1900-2839
9	12471	[null]	NORDDEUTSCHER LLOYD ...	Luncheon	COMMERCIAL	DAMPFER KAISER WILHEL...	BROADSIDE; ILLUS; COL; 5....	[null]	MENU IN GERMAN AND E...	1900-2840
10	12472	[null]	NORDDEUTSCHER LLOYD ...	Dinner	COMMERCIAL	DAMPFER KAISER WILHEL...	FOLDER; ILLUS; COL; 5.5X...	[null]	MENU IN GERMAN AND E...	1900-2841
11	12473	[null]	HOTEL MARLBOROUGH	Cafe Luncheon	COMMERCIAL	[NEW YORK, NY]	CARD; ILLUS; COL; 4.25X5.5;	[null]	HOTEL CREST IN BLUE;	1900-2843
12	12474	[null]	ALPHA OF ZETA PSI	Annual Banquet	COMMERCIAL	DELMONICO'S, [NEW YOR...	BOOKLET; ILLUS; COL; 5.5...	[null]	VELLUM COVER; CREST O...	1900-2844
13	12475	[null]	MANHATTAN HOTEL	Dinner	COMMERCIAL	NEW YORK, NY	CARD; ILLUS; 6X9.5;	[null]	A LA CARTE DU JOUR; HO...	1900-2847
14	12476	[null]	PACIFIC MAIL STEAMSHIP...	Dinne	COMMERCIAL	S.S. CITY OF PARA	FOLDER; ILLUS; 6X9.25;	[null]	DECORATIVE BORDER;	1900-2849
15	12477	[null]	OCCIDENTAL & ORIENTAL	Breakfast	COMMERCIAL	S.S. DORIC	BROADSIDE; ILLUS; 5.5X8.5;	[null]	HANDWRITTEN; STEAMS...	1900-2851

Table Dish



NYPL_menu/postgres@PostgreSQL 13

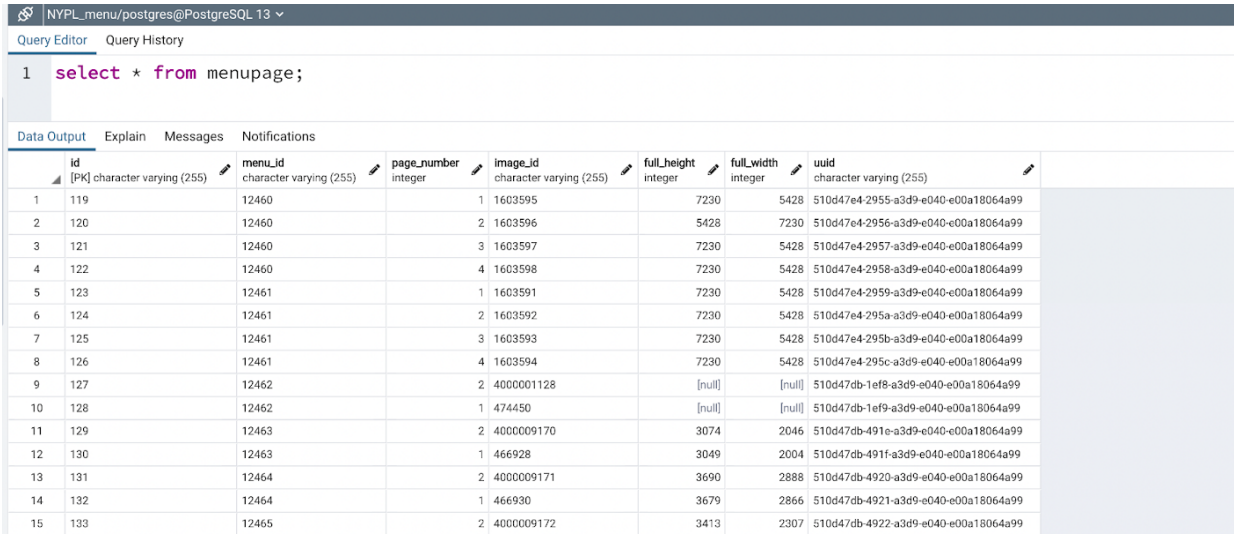
Query Editor | Query History

```
1  select * from dish;
```

Data Output | Explain | Messages | Notifications

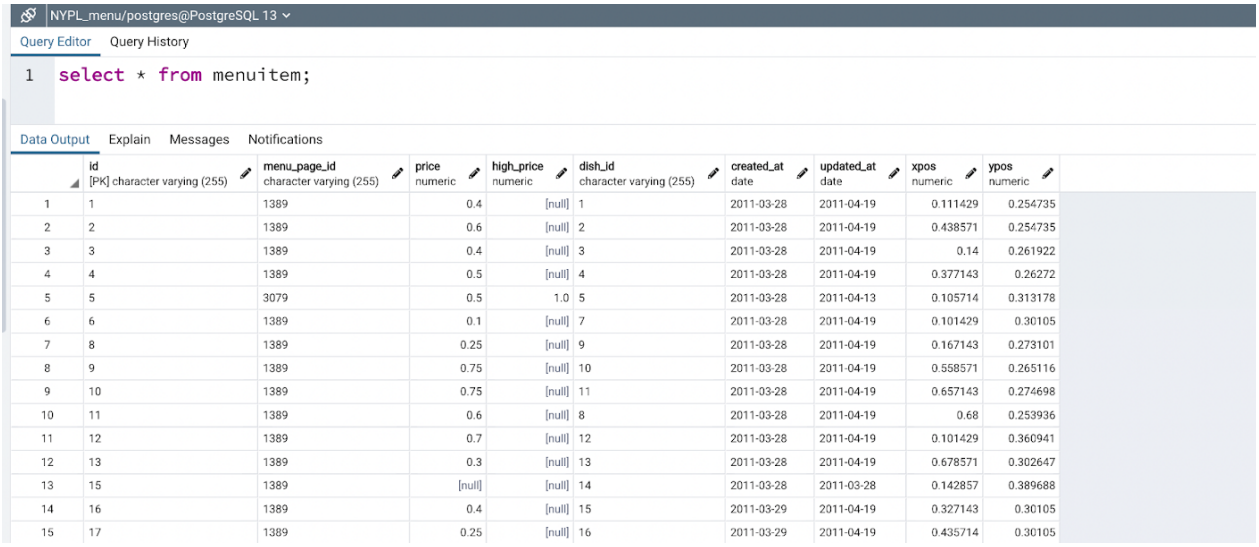
	id [PK] character	name character varying (1500)	description character varying (255)	menus_appeared integer	times_appeared integer	first_appeared integer	last_appeared integer	lowest_price numeric	highest_price numeric
1	1	Consomme, Printaniere Royal	[null]	8	8	1897	1927	0.2	0.4
2	2	Chicken Gumbo	[null]	111	117	1895	1960	0.1	0.8
3	3	Tomato Aux Croutons	[null]	14	14	1893	1917	0.25	0.4
4	4	Onion Au Gratin	[null]	41	41	1900	1971	0.25	1
5	5	St. Emilion	[null]	66	68	1881	1981	0	18
6	7	Radishes	[null]	3263	3347	1854	2928	0	25
7	8	Chicken With Rice Soup	[null]	48	49	1897	1961	0.1	0.6
8	9	Cup Clam Broth	[null]	14	16	1899	1962	0.15	0.4
9	10	Cream Of New Asparagus, ...	[null]	2	2	1900	1900	0	0
10	11	Clear Green Turtle	[null]	156	156	1893	1937	0.25	60
11	12	Striped Bass Saute, Meuniere	[null]	2	2	1900	1900	0	0
12	13	Anchovies	[null]	453	484	1858	1987	0	30
13	14	Fresh Lobsters In Every Style	[null]	4	4	1899	1900	0	0
14	15	Celery	[null]	4246	4690	1	2928	0	50
15	16	Pim-olas	[null]	145	148	1897	1918	0.15	35

Table MenuPage



	id [PK] character varying (255)	menu_id character varying (255)	page_number integer	image_id character varying (255)	full_height integer	full_width integer	uuid character varying (255)
1	119	12460	1	1603595	7230	5428	510d47e4-2955-a3d9-e040-e00a18064a99
2	120	12460	2	1603596	5428	7230	510d47e4-2956-a3d9-e040-e00a18064a99
3	121	12460	3	1603597	7230	5428	510d47e4-2957-a3d9-e040-e00a18064a99
4	122	12460	4	1603598	7230	5428	510d47e4-2958-a3d9-e040-e00a18064a99
5	123	12461	1	1603591	7230	5428	510d47e4-2959-a3d9-e040-e00a18064a99
6	124	12461	2	1603592	7230	5428	510d47e4-295a-a3d9-e040-e00a18064a99
7	125	12461	3	1603593	7230	5428	510d47e4-295b-a3d9-e040-e00a18064a99
8	126	12461	4	1603594	7230	5428	510d47e4-295c-a3d9-e040-e00a18064a99
9	127	12462	2	4000001128	[null]	[null]	510d47db-1ef8-a3d9-e040-e00a18064a99
10	128	12462	1	474450	[null]	[null]	510d47db-1ef9-a3d9-e040-e00a18064a99
11	129	12463	2	4000009170	3074	2046	510d47db-491e-a3d9-e040-e00a18064a99
12	130	12463	1	466928	3049	2004	510d47db-491f-a3d9-e040-e00a18064a99
13	131	12464	2	4000009171	3690	2888	510d47db-4920-a3d9-e040-e00a18064a99
14	132	12464	1	466930	3679	2866	510d47db-4921-a3d9-e040-e00a18064a99
15	133	12465	2	4000009172	3413	2307	510d47db-4922-a3d9-e040-e00a18064a99

Table MenuItem



	id [PK] character varying (255)	menu_page_id character varying (255)	price numeric	high_price numeric	dish_id character varying (255)	created_at date	updated_at date	xpos numeric	ypos numeric
1	1	1389	0.4	[null]	1	2011-03-28	2011-04-19	0.111429	0.254735
2	2	1389	0.6	[null]	2	2011-03-28	2011-04-19	0.438571	0.254735
3	3	1389	0.4	[null]	3	2011-03-28	2011-04-19	0.14	0.261922
4	4	1389	0.5	[null]	4	2011-03-28	2011-04-19	0.377143	0.26272
5	5	3079	0.5	1.0	5	2011-03-28	2011-04-13	0.105714	0.313178
6	6	1389	0.1	[null]	7	2011-03-28	2011-04-19	0.101429	0.30105
7	8	1389	0.25	[null]	9	2011-03-28	2011-04-19	0.167143	0.273101
8	9	1389	0.75	[null]	10	2011-03-28	2011-04-19	0.558571	0.265116
9	10	1389	0.75	[null]	11	2011-03-28	2011-04-19	0.657143	0.274698
10	11	1389	0.6	[null]	8	2011-03-28	2011-04-19	0.68	0.253936
11	12	1389	0.7	[null]	12	2011-03-28	2011-04-19	0.101429	0.360941
12	13	1389	0.3	[null]	13	2011-03-28	2011-04-19	0.678571	0.302647
13	15	1389	[null]	[null]	14	2011-03-28	2011-03-28	0.142857	0.389688
14	16	1389	0.4	[null]	15	2011-03-29	2011-04-19	0.327143	0.30105
15	17	1389	0.25	[null]	16	2011-03-29	2011-04-19	0.435714	0.30105

(4) Integrity Constraints Check

After the data import, we related the 4 tables to check the integrity constraints. Listed below are the integrity constraint rules we tested.

a) “Menu can’t have 0 page count”. The screenshot below shows violators of this integrity constraint rule. We can see no record violating this rule.

```
1  select count(*)
2  from menu
3  where page_count=0;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	0

b) “As for dishes, the number of menus_appeared must be equal or smaller than the number of times_appeared”. The screenshot below shows violators of this integrity constraint rule. We can see 13305 records violating this rule.

```
1  select count(*)
2  from dish
3  where menus_appeared > times_appeared;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	13305

c) “As for dishes, the first_appeared time must be earlier than or the same as the last_appeared time”. The screenshot below shows violators of this integrity constraint rule. We can see 6 records violating this rule.

```
1  select count(*)
2  from dish
3  where first_appeared > last_appeared;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	6

d) “As for menu items, the created_at date must be earlier than or the same as the updated_at date”. The screenshot below shows violators of this integrity constraint rule. We can see 3 records violating this rule.

```
1  select count(*)
2  from menuitem
3  where created_at > updated_at;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	3

e) “As for dishes, the lowest_price must be lower than or the same as the highest_price”. The screenshot below shows violators of this integrity constraint rule. We can see 0 records violating this rule.

```
1  select count(*)
2  from dish
3  where lowest_price > highest_price;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	0

We decided not to delete the records violating the integrity constraint rules. Firstly, they do not impact the results of our target use cases. Secondly, although some fields violate the integrity constraint rules, the information in other fields of the records are still useful for future use in other studies.

9. Creating the Workflow Models with OR2YWTool and yesWorkflow

We used the yesWorkflow web application to create the overall workflow model. As for the OpenRefine workflow model, we used OR2YWTool to generate the .yw files.

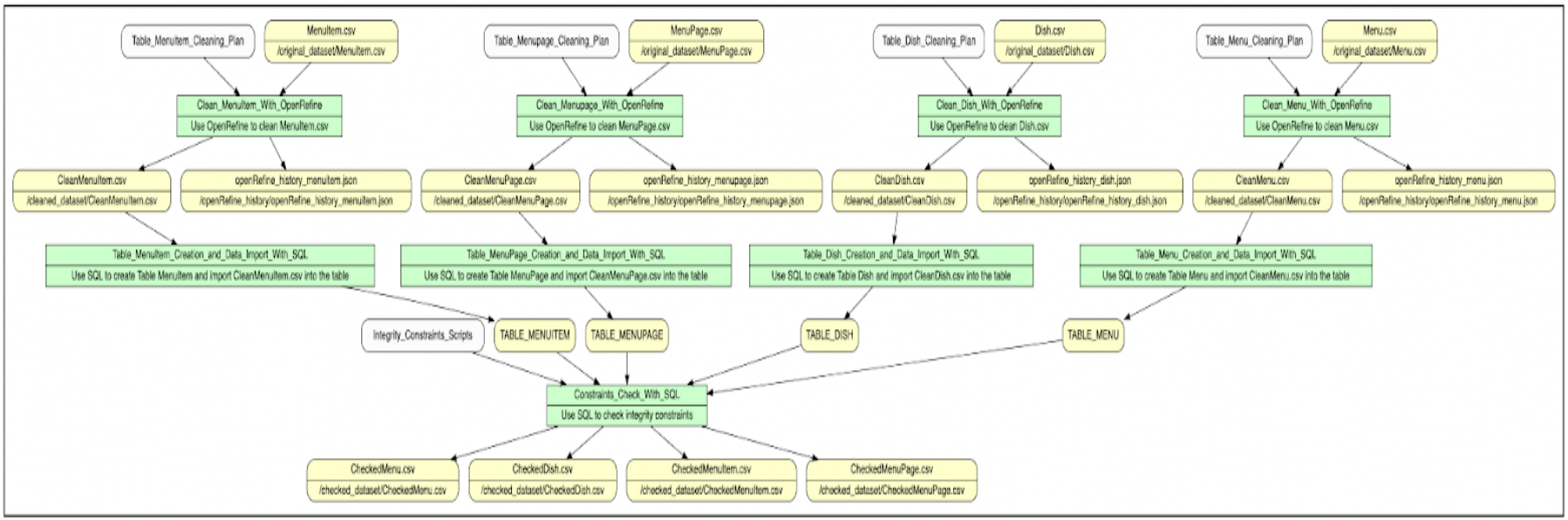
（1）Overall Workflow

The overall workflow provides a bird's eye view of the overall process of this project. It ties all steps together. The inputs are the 4 original .csv files, that is, Menu.csv, Dish.csv, MenuItem.csv, and MenuPage.csv. We imported each of them into openRefine, and cleaned them based on our data cleaning plan. The outputs are 4 openRefine cleaned .csv files and 4 openRefine history .json files. The 4 openRefine cleaned .csv files would be used as inputs of PostgreSQL to check the integrity constraints. The 4 openRefine history .json files would be used as inputs of OR2YWTool to generate the 4 openRefine overflow .yw files.

Then we created schema and tables using PostgreSQL, and imported these 4 openRefine cleaned .csv files into the tables we created. We related these 4 tables and did integrity constraints checks based on our integrity constraints check script (.txt file). The final outputs are 4 SQL checked .csv files.

The chart below shows the overall workflow model we created.

Overall_Workflow



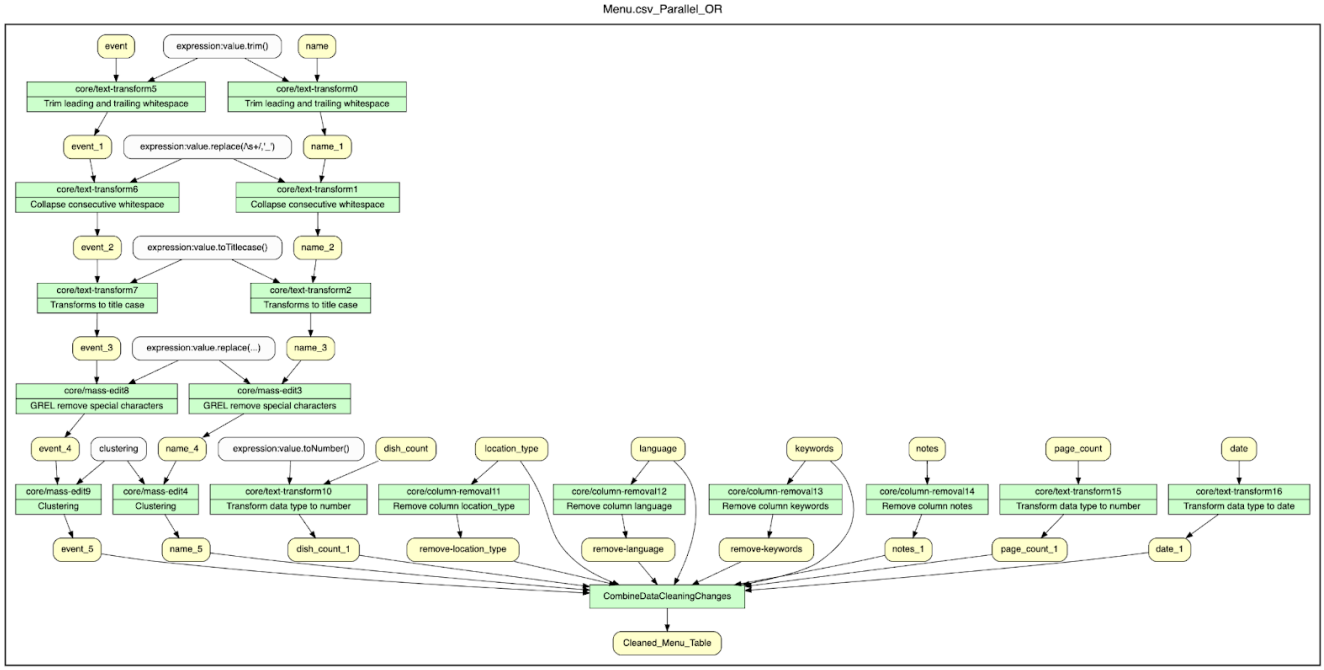
(2) OpenRefine Data Cleaning Workflow

The OpenRefine workflow was created using OR2YWTool, which provides an auto-parsing method to transfer the openRefine history .json file to the yesWorkflow .yw file.

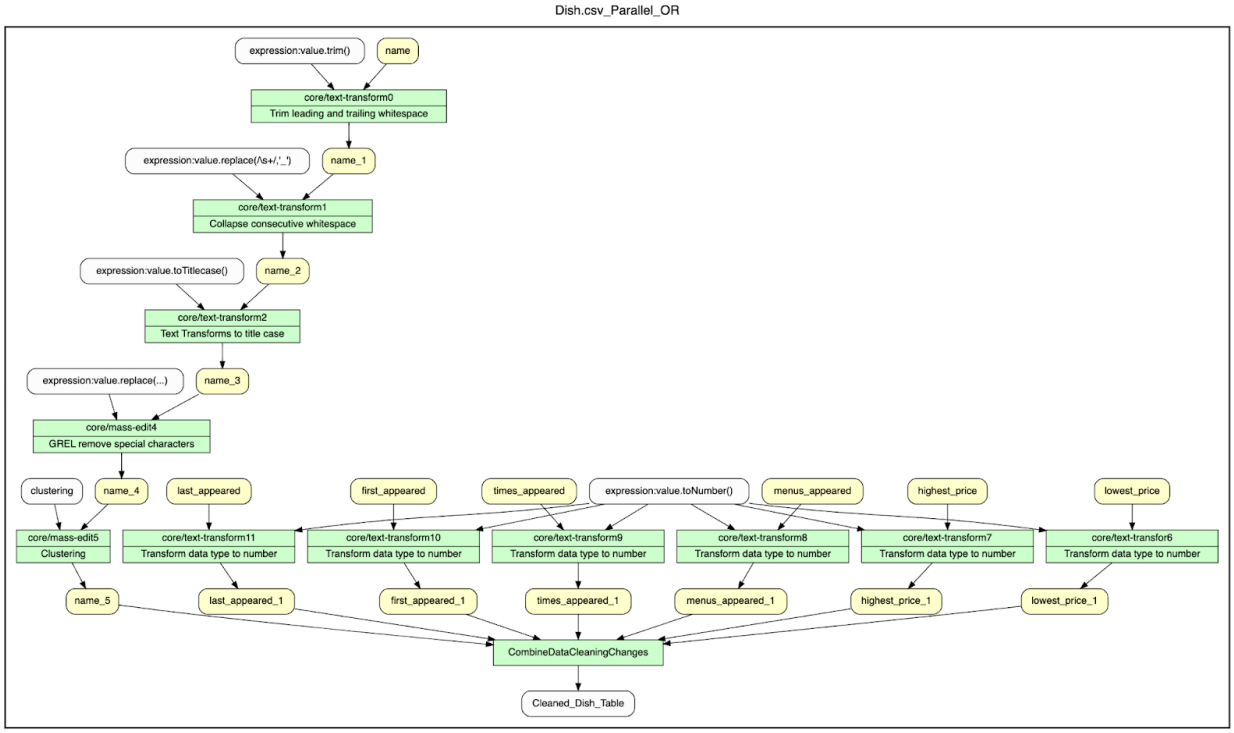
In order to make the graphs and annotations more clear and concise, we did some revisions based on the .yw files generated. We also compacted similar operations (such as clustering using different methods and functions).

The 4 charts below show the openRefine workflow models we created for each .csv file.

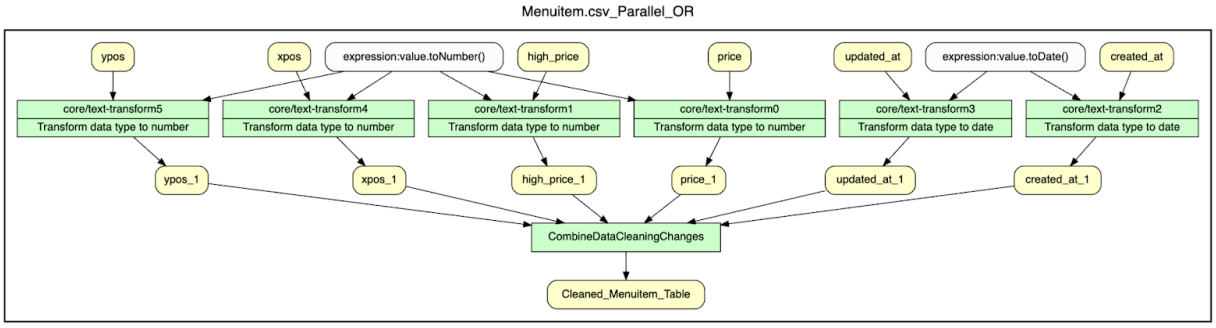
a) Menu.csv OpenRefine Workflow Model



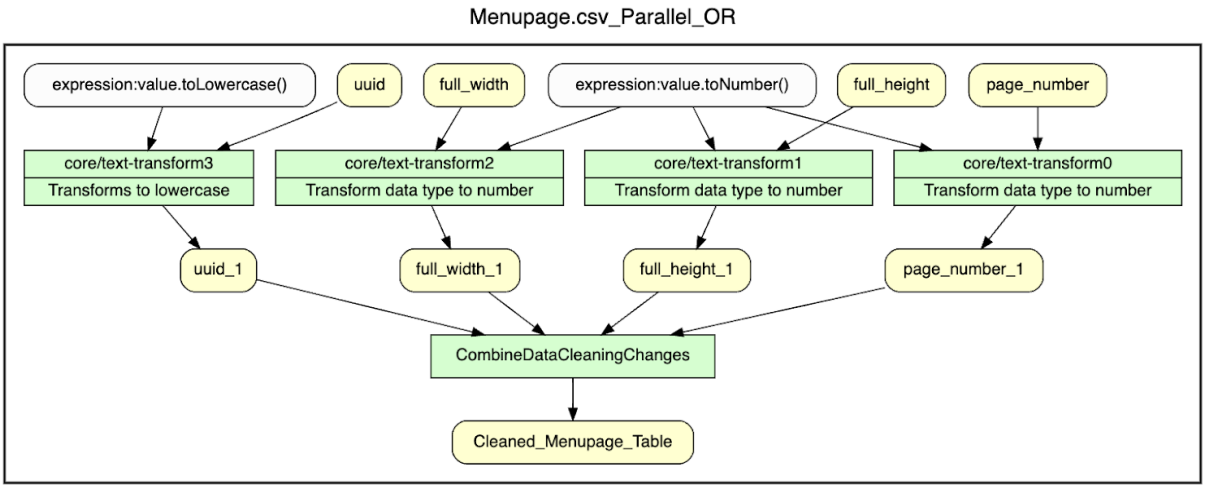
b) Dish.csv OpenRefine Workflow Model



c) MenuItem.csv OpenRefine Workflow Model



d) MenuPage.csv OpenRefine Workflow Model



10. Summary of the Data Changes resulting from the Overall Workflow

The following tables summarize the data changes as the result of the overall workflow.

1) Table Menu

Column	Action	Cells Affected
name	Trim leading and trailing whitespace	9
name	Collapse consecutive whitespace	1
name	Transforms to title case	629
name	Clustering	2079
name	GREL remove special characters	3202
event	Trim leading and trailing whitespace	3

event	Collapse consecutive whitespace	6
event	Transforms to title case	7829
event	Clustering	9513
event	GREL remove special characters	8154
date	Transforms to date	16964
page_count	Transforms to number	17550
dish_count	Transforms to number	17550
notes	GREL remove ‘ and ‘’	10618

2) Table Dish

Column	Action	Cells Affected
name	Trim leading and trailing whitespace	9359
name	Collapse consecutive whitespace	6663
name	Transforms to title case	286342
name	Clustering	126113
name	GREL remove special characters	429956
menus_appeared	Transforms to number	429956
times_appeared	Transforms to number	429956
first_appeared	Transforms to number	429956
last_appeared	Transforms to number	429956
lowest_price	Transforms to number	400855
highest_price	Transforms to number	400855

3) Tabal MenuPage

Column	Action	Cells Affected
page_number	Transforms to number	65735
full_height	Transforms to number	66608
full_width	Transforms to number	66608
uuid	Transforms to lowercase	1

4) Table ManuItem

Column	Action	Cells Affected
price	Transforms to number	888938
high_price	Transforms to number	91973
created_at	Transforms to date	0
updated_at	Transforms to date	0
xpos	Transforms to number	1335284
ypos	Transforms to number	1335284

11. Quality Improvements (Use Case Demo)

To prove the quality improvements of the “after-cleaning” dataset compared with the “original” dataset, we use the Target Use Case as a demo, and compare the answers of “before query” and “after query”.

（1）Which 10 restaurants have the most menus collected in this dataset?

- Queries

NYPL_menu/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select name, count(name) from menu
2  group by name
3  order by count(name) desc
4  limit 10;
```

- Before Queries Answers

Data Output | Explain | Messages | Notifications

	name character varying (255)	count bigint
1	Waldorf Astoria	476
2	Hotel Astor	143
3	[Restaurant name and/or lo...	113
4	The Biltmore	96
5	Healy's Forty-second Street ...	93
6	Waldorf-Astoria	84
7	Wm. F. Healy's 42nd St. Grill	82
8	Hotel McAlpin	62
9	Childs	51
10	The Mouquin Restaurant an...	50

- After Queries Answers

Data Output | Explain | Messages | Notifications

	name character varying (255)	count bigint
1	Waldorf Astoria	566
2	Hotel Astor	148
3	Healy's Forty-second Stree...	136
4	restaurant Name And/or L...	133
5	The Biltmore	96
6	Wm. F. Healy's 42nd St. Grill	82
7	Hotel Mcalpin	62
8	Delmonicos	59
9	Pennsylvania Railroad	56
10	Childs	52

(2) Which 10 dishes most often appeared on the menus of the top restaurants?

- Queries

```sql
select d.name,count(d.name)
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 a.name='Waldorf Astoria'
group by d.name
order by count(d.name) desc
limit 10;
```

- Before Queries Answers

Data Output | Explain | Messages | Notifications

	name character varying (1500)	count bigint
1	Celery	477
2	Olives	477
3	Assorted Cakes	362
4	Apollinaris	325
5	Turkish Coffee	315
6	Caviar, Special Importation	310
7	Crab Flake Cocktail	307
8	Cafe Parfait	305
9	Plover	302
10	Ices in Souvenir	299

- After Queries Answers

Data Output | Explain | Messages | Notifications

	name character varying (1500)	count bigint
1	Olives	598
2	Celery	594
3	Assorted Cakes	482
4	Apollinaris	456
5	Cafe Parfait	451
6	Turkish Coffee	417
7	Caviar, Special Importation	407
8	Ices In Souvenir	406
9	French Coffee	395
10	Crab Flake Cocktail	394

(3) Which 10 dishes are most popular for a certain event (e.g. breakfast)?

- Queries

```
1  select distinct d.name,count(d.name), a.event
2  from menu a, menupage b, menuitem c, dish d
3  where a.id=b.menu_id
4  and b.id=c.menu_page_id
5  and c.dish_id=d.id
6  and a.event='Breakfast'
7  group by d.name, a.event
8  order by count(d.name) desc
9  limit 10;
```

- Before Queries Answers

Data Output | Explain | Messages | Notifications

	name character varying (1500)	count bigint	event character varying (255)
1	Hominy	55	Breakfast
2	Coffee	53	Breakfast
3	Tea	53	Breakfast
4	Kaffee	44	Breakfast
5	Marmalade	39	Breakfast
6	Marmelade	37	Breakfast
7	Oatmeal	36	Breakfast
8	Hafergrutze	35	Breakfast
9	Boiled potatoes	29	Breakfast
10	Tee	29	Breakfast

- After Queries Answers

Data Output | Explain | Messages | Notifications

	name character varying (1500)	count bigint	event character varying (255)
1	Coffee	834	Breakfast
2	Tea	678	Breakfast
3	Cocoa	517	Breakfast
4	Chocolate	409	Breakfast
5	Marmalade	362	Breakfast
6	Oatmeal	325	Breakfast
7	Boiled Potatoes	313	Breakfast
8	Oranges	293	Breakfast
9	Fried Potatoes	290	Breakfast
10	Boiled Eggs	289	Breakfast

As we can see from the comparison, we got much more counts for each item after cleaning. The data quality is largely improved.

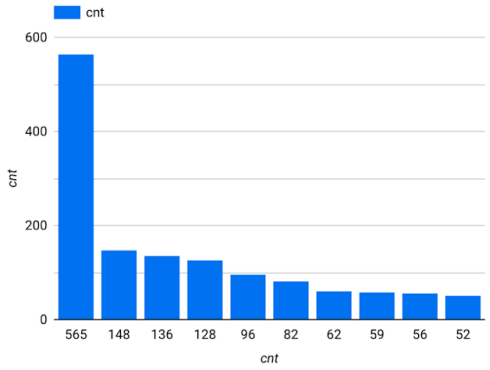
12. Use Case Visualization

After SQL queries, we used the data studio to visualize the results of our target use cases.

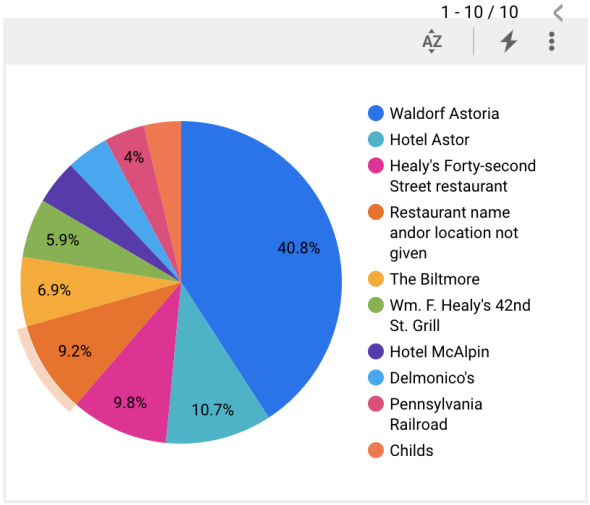
（1）Which 10 restaurants have the most menus collected in this dataset?

	name	cnt ▾
1.	Waldorf Astoria	565
2.	Hotel Astor	148
3.	Healy's Forty-second Street restau...	136
4.	Restaurant name andor location n...	128
5.	The Biltmore	96
6.	Wm. F. Healy's 42nd St. Grill	82
7.	Hotel McAlpin	62
8.	Delmonico's	59
9.	Pennsylvania Railroad	56
10.	Childs	52

1 - 10 / 10



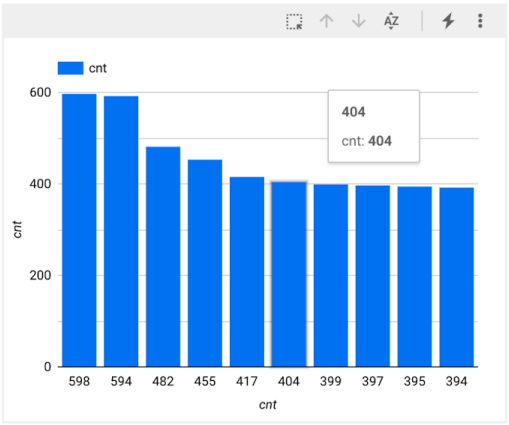
1 - 10 / 10

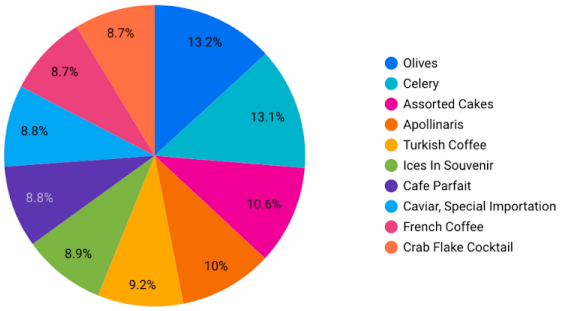


(2) Which 10 dishes most often appeared on the menus of the top restaurants?

	name	cnt
1.	Olives	598
2.	Celery	594
3.	Assorted Cakes	482
4.	Apollinaris	455
5.	Turkish Coffee	417
6.	Ices In Souvenir	404
7.	Cafe Parfait	399
8.	Caviar, Special Importation	397
9.	French Coffee	395
10.	Crab Flake Cocktail	394

1 - 10 / 10

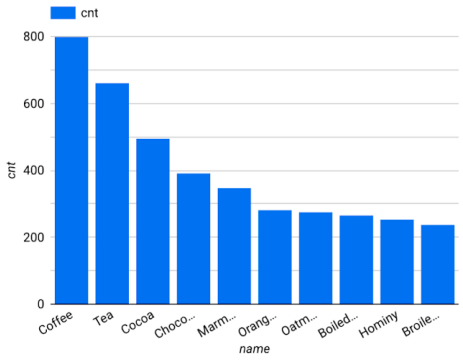


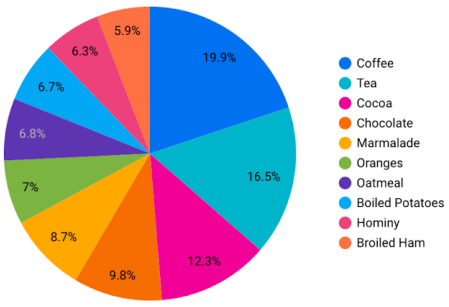


(3) Which 10 dishes are most popular for a certain event (e.g. breakfast)?

	name	cnt
1.	Coffee	800
2.	Tea	661
3.	Cocoa	495
4.	Chocolate	393
5.	Marmalade	349
6.	Oranges	283
7.	Oatmeal	275
8.	Boiled Potatoes	268
9.	Hominy	255
10.	Broiled Ham	237

1 - 10 / 10





13. Challenges, Findings, Lessons Learned, and Possible Next Steps.

(1) Challenges and Findings

One major challenge we had is working with big data. Actually the challenge has two layers, one is the challenge big data brings to the data cleaning related tools that we use, such as OpenRefine, and the other layer is the difficulty big data brings for human confirmation and involvement in the process of data cleaning.

Using the Dish file as an example. To perform the cleaning process, multiple configurations had to be done on OpenRefine and we had to find a computer with big RAM to run them. Also, when clustering the name column of the dish.csv file, due to the huge amount of clustering options, it is impossible for us to check and confirm every cluster manually. We had to trust OpenRefine's judgment to select, merge, and re-clustered automatically.

Another challenge we had is dealing with ambiguity. We have to rely on personal experiences and judgments to make decisions during clustering or cleaning. For example, whether the "Bankers' Club of Chicago" is the same as "Bankers Club". What does "high_price" mean? Is "high_price" the same as "highest_price"? Everyone has different answers and different methods of dealing with ambiguity. That's why establishing consistent rules and documenting those rules are important. On the other hand, we realized it is really time consuming to document them case by case because of the size of the data. We had a small taste of the Dilemma of Provenance and Reproducibility learned in this class.

The third challenge we had was balancing between "cleaning the data" and "not deleting useful information". We had a lot of discussions on whether to merge all kinds of dinner, or whether to merge all departments of a sponsor. Finally, we decided not to do so. We want to keep the details. Keeping these ambiguities will not largely impact the results of our target use cases. On the contrary, if a user wants to dig into our cleaned dataset, we believe he would like to see more detailed information, such as "the sponsor is the Class of 1858 of Harvard College" , rather than "the sponsor is the Harvard College".

(2) Lessons Learned:

The first lesson we learned is that tools are important. A good tool can save us a lot of time cleaning the data, processing the data, and documenting the workflow. In this class, we learned several useful tools, such as OpenRefine, YesWorkflow, OR2YWTool, and SQL, which are all super helpful in our future job.

The second lesson we learned is that every tool has its limitations, for example, OpenRefine is slow in processing big data, and OR2YWTool is not good at slicing a big image into smaller ones to fit them into screen or A4 paper. Human involvements and judgements are still necessary at the current stage.

Lastly, during this practice in data cleaning and provenance, we started to understand why companies invest so much human and other resources in these 2 areas. They are extremely time consuming!!! We spent many more hours on this project than we expected. However, we have to admit these 2 areas are extremely important in any data related projects. This project gave us a real-life taste of this. It is a great learning experience. We all enjoyed this course.

(3) Possible Next Steps

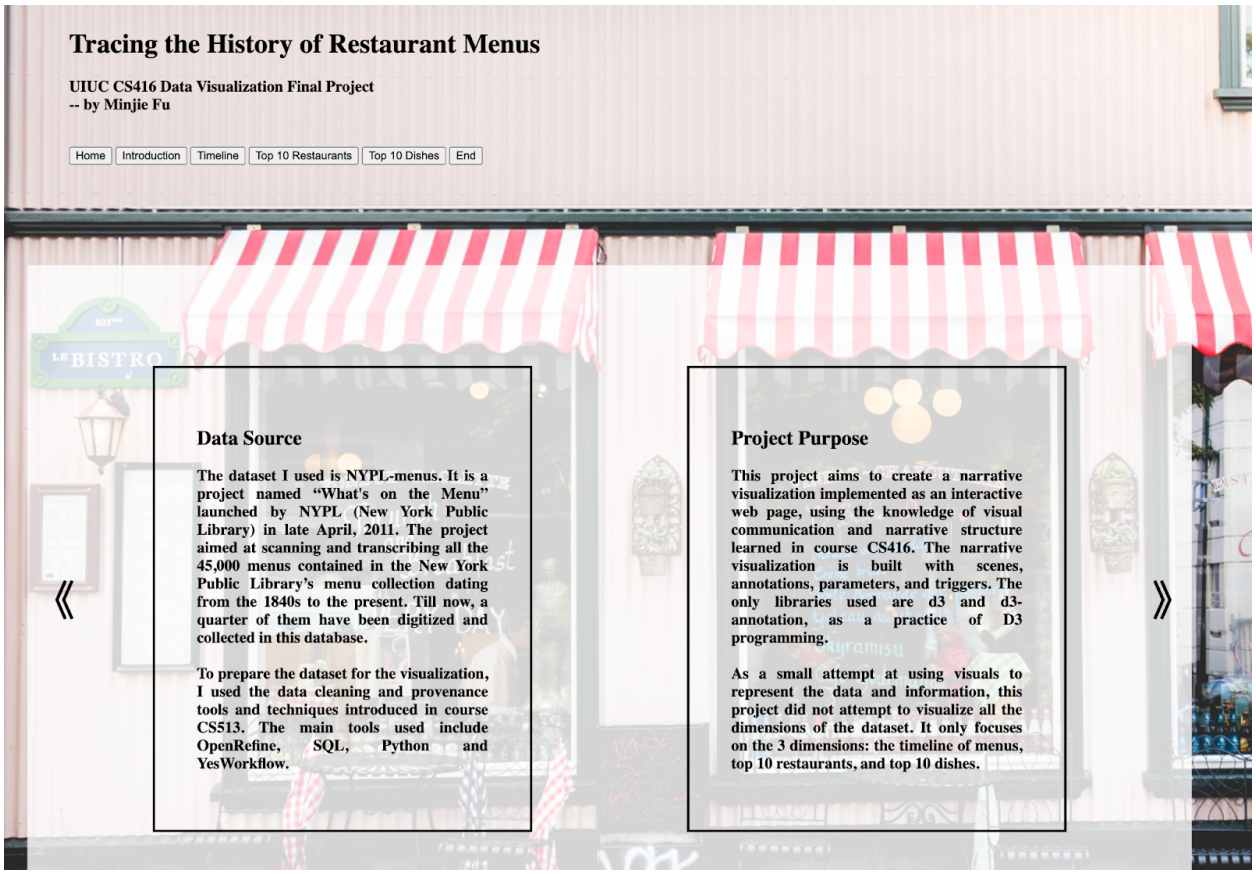
We plan to use the cleaned NYPL-menus dataset to create an interactive website using the D3 visualization library. It is also a project for the course CS416.

The website is still under construction.
Pls visit https://minjiefu.github.io/cs416_final-project_minjie-fu/index.html for updates.

Tracing the History of Restaurant Menus

UIUC CS416 Data Visualization Final Project
-- by Minjie Fu

Home | Introduction | Timeline | Top 10 Restaurants | Top 10 Dishes | End



Data Source

The dataset I used is NYPL-menus. It is a project named "What's on the Menu" launched by NYPL (New York Public Library) in late April, 2011. The project aimed at scanning and transcribing all the 45,000 menus contained in the New York Public Library's menu collection dating from the 1840s to the present. Till now, a quarter of them have been digitized and collected in this database.

To prepare the dataset for the visualization, I used the data cleaning and provenance tools and techniques introduced in course CS513. The main tools used include OpenRefine, SQL, Python and YesWorkflow.

Project Purpose

This project aims to create a narrative visualization implemented as an interactive web page, using the knowledge of visual communication and narrative structure learned in course CS416. The narrative visualization is built with scenes, annotations, parameters, and triggers. The only libraries used are d3 and d3-annotation, as a practice of D3 programming.

As a small attempt at using visuals to represent the data and information, this project did not attempt to visualize all the dimensions of the dataset. It only focuses on the 3 dimensions: the timeline of menus, top 10 restaurants, and top 10 dishes.