CS513: Theory & Practice of Data Cleaning Project

Phase II - Report

University of Illinois at Urbana-Champaign Summer 2023

Team 140 - Akriti Sinha, Christopher Buja, Tianhao Luo

Team 140 information :

Members: Akriti Sinha (akritis5)
Christopher Buja (cbuja2)
Tianhao Luo (tluo3)

1. Actual data cleaning workflow

Step I: Python (stage1.ipynb)

Input file : Menu.csv, Dish.csv, MenuPage.csv, MenuItem.csv

1. In the string columns (such as occasion), delete the following characters
   ()[];?
2. Also, for the string columns, convert NaN to “Unknown”
3. Every single name is converted to title case
4. This transformation has been applied to
   a. ‘name’ column of ‘dish’ dataset
   b. 'name','sponsor','event','venue','place','occasion','notes','location' columns of ‘menu’ dataset

Output file : Menu_stage1.csv, MenuPgae_stage1.csv, MenuItem_stage1.csv, Dish_stage1.csv

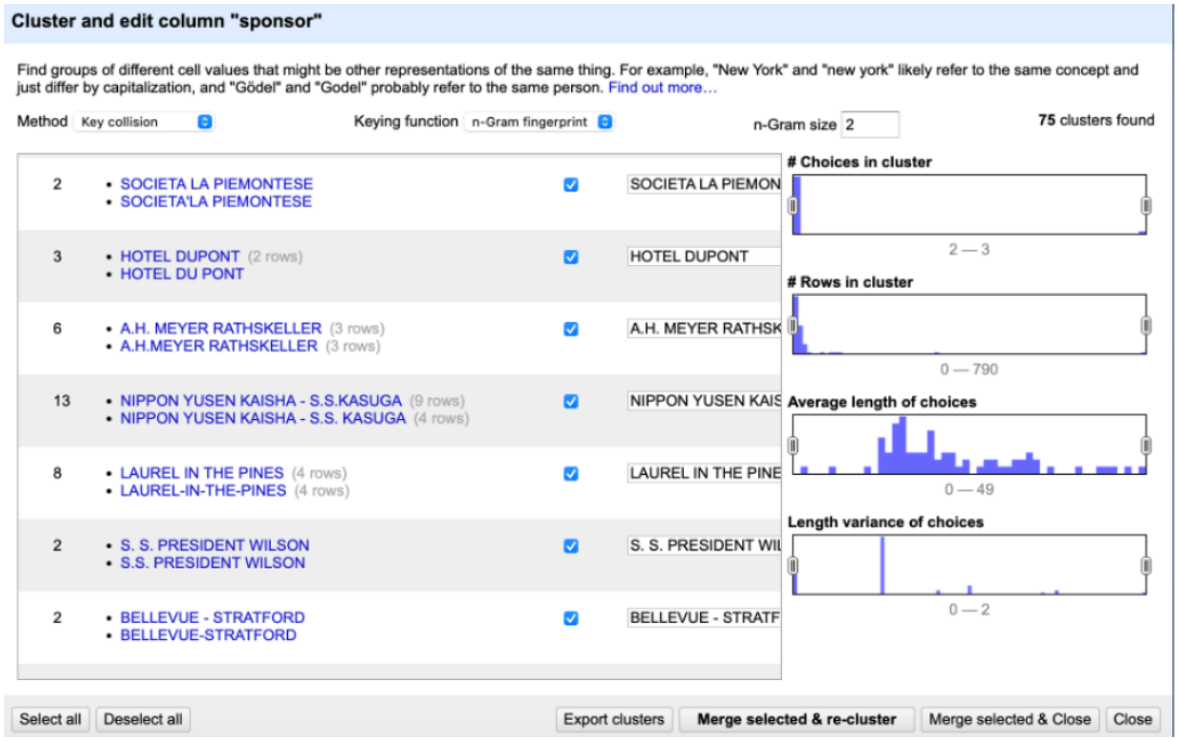
Step II: OpenRefine

Cleaning of Menu_stage1.csv

For column: sponsor and event

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

(6) Make a facet and perform the cluster operation using the key-collision method and fingerprint function. Next merge the selected clusters.
(7) Repeat the previous step with n-gram, fingerprint, meta-phone3, and cologne-phonetic methods.



For column: physical_description

(1) Split the columns using ';'
(2) Then rename the first column: physical_description_type
(3) Use GREL to join 'physical_description 1', 'physical_description 2', 'physical_description 3', and 'physical_description 4'.

(4)One column ‘physical_description 1’ renamed to physical_description_type.

(5)‘physical_description 2’, ‘physical_description 3’, and ‘physical_description 4’ are combined using ‘ - ‘ to separate the values from the different columns (Remember the space before and after the dash) to column physical_description_additional.

For column: date

(1) Convert date format to YYYY-MM-DD

For column: call_number and id

(1) Trim leading/trailing white spaces

(2) Collapse consecutive white spaces

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

Output file : Menu_clean.csv

Cleaning of MenuPage_stage1.csv

There was nothing to be refined here so the

Unchanged columns: id, menu_id, page_number, image_id, full_height, full_width, uuid

Output file : MenuPage_clean.csv

Cleaning of MenuItem_stage1.csv

For column: created_at

(1) Convert date format to YYYY-MM-DD

For column: updated_at

(1) Convert date format to YYYY-MM-DD

Unchanged columns: id, menu_page_id, price, high_price, dish_id, created_at, updated_at, xpos, ypos

Output file : MenuItem_clean.csv

Cleaning of Dish_stage1.csv

For column: name

(1) Use key-collision to cluster values.

Output file : Dish_clean.csv

Step III : Develop Relational Database Schema

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

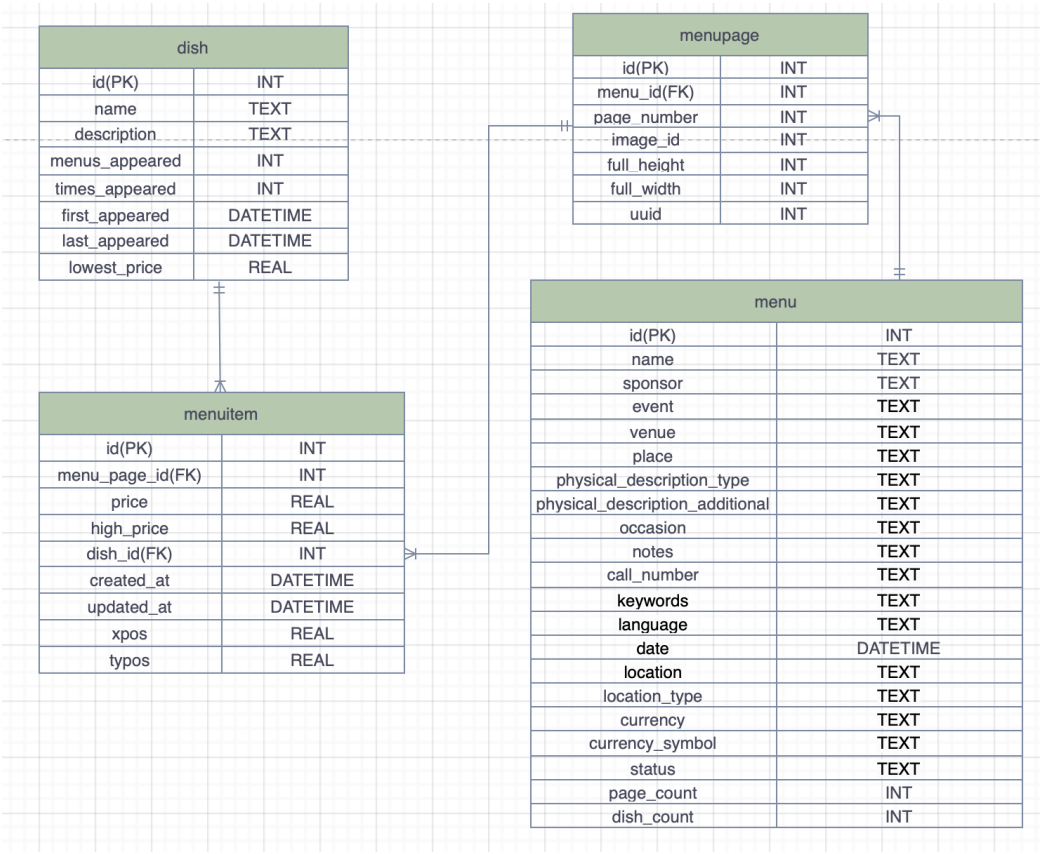


Table creation and Data load

SQL to create Database as per the Schema defined above.

```
CREATE TABLE dish(
"id" INTEGER,
"name" TEXT,
“description” TEXT,
"menus_appeared" INTEGER,
"times_appeared" INTEGER,
"first_appeaared" DATETIME,
"last_appeared" DATETIME,
"lowest_price" REAL,
"highest_price" REAL
);
CREATE TABLE menuitem(
"id" INTEGER,
"menu_page_id" INTEGER,
"price" REAL,
"high_price" REAL,
"dish_id" INTEGER,
"created_at" DATETIME,
"updated_at" DATETIME,
"xpos" REAL,
"ypos" REAL
);
CREATE TABLE menupage(
"id" INTEGER,
"menu_id" INTEGER,
"page_number" INTEGER,
"image_id" INTEGER,
"full_height" INTEGER,
"full_width" INTEGER,
"uuid" TEXT
);
```

```
CREATE TABLE menu(
"id" INTEGER,
"name" TEXT,
"sponsor" TEXT,
"event" TEXT,
"venue" TEXT,
"place" TEXT,
"physical_description_type" TEXT,
"physical_description_additional" TEXT,
"occasion" TEXT,
"notes" TEXT,
"call_number" TEXT,
"keywords" TEXT,
"language" TEXT,
"date" DATETIME,
"location" TEXT,
"location_type" TEXT,
"currency" TEXT,
"currency_symbol" TEXT,
"status" TEXT,
"page_count" INTEGER,
"dish_count" INTEGER
);
```

```
sqlite> .mode csv
sqlite> .import /uiuc/cs513/project/clean_data/Dish_clean.csv dish
sqlite> .import /uiuc/cs513/project/clean_data/Menu_clean.csv menu
sqlite> .import /cs513/project/clean_data/MenuItem_clean.csv menuitem
sqlite> .import /uiuc/cs513/project/clean_data/MenuPage_clean.csv
menupage
```

Checking for Integrity Constraints :

The following are the Integrity Constraint Violations check as follows :

Menu (menu) Table

- Id cannot be Null
- Page count should not be NULL

Dish (dish) Table

- Id should not be Null
- menus_appeared or times_appeared cannot be NULL
- Lowest price of the dish should be less than the highest price

MenuPage (menupage) Table

- Id cannot be Null
- Page number should not be NULL or “0”. No such rows were identified
- created_at date should be always greater than updated_at.

Menu Item (menuitem) Table

- Id cannot be Null
- created_at date should be always greater than updated_at.
- Xpos and ypos values should always be between 0 and 1.

These IC Violations can be checked using SQLite. Akriti will take up the IC check part.

Comparison with Phase I

i. One dish can be mapped to several IDs

515676	Claret: Chateau Larose, Cruse et Fils Freres
515677	Claret: Chateau Lafite, Cruse et Fils Freres

On a closer look, although these two dishes bear striking resemblance, on a closer look they can be two distinct dishes, ‘Larose’ and ‘Lafite’. We should not attempt to merge them without 100% confidence that they are exactly the same dish.

ii. A dish has more ‘menus_appeared’ than ‘times_appeared’

id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
208	Luncheon	NaN	19	18	1900	1993	0.65	0.65
825	Rice, Semolina	NaN	1	0	1900	1900	0.00	0.00
1082	Caviare	NaN	86	85	1888	1906	0.40	0.50
1136	Carta blanca	NaN	8	7	1900	1981	0.00	0.00
1346	Hackley's Sour Mash	NaN	5	4	1900	1900	0.15	0.15
...	...	...	...	...	...	...	...	...
15598	Apricot Cômpote	NaN	1	0	0	0	0.00	0.00
15599	Pear Cômpote	NaN	1	0	0	0	0.00	0.00
15600	Guava Cômpote	NaN	1	0	0	0	0.00	0.00
15601	Bilberies Cômpote	NaN	1	0	0	0	0.00	0.00
15602	Fig Cômpote	NaN	1	0	0	0	0.00	0.00

While this is definitely some data quality problem, we decide to shelve it because we are unable to deduce a dish’s a) menus appeared b) times appeared from the dataset.

iii. A same occasion can have different names

‘DAILY;’, ‘DAILY’

'ANNIVERSARY(?);', 'ANNUAL','OTHER (ANNUAL DINNER);'

This has been taken care of in Python (by getting rid of characters such as ()?;[], and converting cases) the OpenRefine part (using ABC distance metric)

iv. (Not directly related to the use case) Some menus do not have continuous pages

2. Narrative that ties all steps together

For the main use case U1, we want

- What are the most popular (in terms of time of appearances on all menus in the dataset) dish under each occasion (e.g. Easter, Thanksgiving, Christmas etc.)?

- What are the most popular breakfast/lunch/dinner dishes?

The data cleaning steps described above can help us

- Combine all variations of a same occasion under the same category
  e.g. Anniversaries can have variations such as 'ANNIVERSARY(?);', 'ANNUAL','OTHER (ANNUAL DINNER);'
- Make the occasions more human readable (getting rid of special characters etc.)

3. Documentation

a. Python

	id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
0	1	Consomme Printaniere Royal	NaN	8	8	1897	1927	0.20	0.4
1	2	Chicken Gumbo	NaN	111	117	1895	1960	0.10	0.8
2	3	Tomato Aux Croutons	NaN	13	13	1893	1917	0.25	0.4
3	4	Onion Au Gratin	NaN	41	41	1900	1971	0.25	1.0
4	5	St. Emilion	NaN	66	68	1881	1981	0.00	18.0
5	7	Radishes	NaN	3262	3346	1854	2928	0.00	25.0
6	8	Chicken Soup With Rice	NaN	48	49	1897	1961	0.10	0.6
7	9	Clam Broth Cup	NaN	14	16	1899	1962	0.15	0.4
8	10	Cream Of New Asparagus, Croutons	NaN	2	2	1900	1900	0.00	0.0
9	11	Clear Green Turtle	NaN	157	157	1893	1937	0.25	60.0
10	12	Striped Bass Saute, Meuniere	NaN	2	2	1900	1900	0.00	0.0
11	13	Anchovies	NaN	453	484	1858	1987	0.00	30.0
12	14	Fresh Lobsters In Every Style	NaN	4	4	1899	1900	0.00	0.0
13	15	Celery	NaN	4246	4690	1	2928	0.00	50.0
14	16	Pim-Olas	NaN	145	148	1897	1918	0.15	35.0
15	17	Caviar	NaN	505	534	1880	1987	0.00	75.0
16	18	Sardines	NaN	1425	1484	1856	2928	0.00	50.0
17	19	India Chutney	NaN	16	16	1865	1901	0.10	0.2
18	20	Pickles	NaN	453	472	1852	1987	0.00	10.0
19	21	English Walnuts	NaN	83	86	1851	1948	0.10	0.3

Data: Dish (before Python cleaning)

	id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
0	1	Consomme Printaniere Royal	NaN	8	8	1897	1927	0.20	0.4
1	2	Chicken Gumbo	NaN	111	117	1895	1960	0.10	0.8
2	3	Tomato Aux Croutons	NaN	13	13	1893	1917	0.25	0.4
3	4	Onion Au Gratin	NaN	41	41	1900	1971	0.25	1.0
4	5	St. Emilion	NaN	66	68	1881	1981	0.00	18.0
5	7	Radishes	NaN	3262	3346	1854	2928	0.00	25.0
6	8	Chicken Soup With Rice	NaN	48	49	1897	1961	0.10	0.6
7	9	Clam Broth Cup	NaN	14	16	1899	1962	0.15	0.4
8	10	Cream Of New Asparagus, Croutons	NaN	2	2	1900	1900	0.00	0.0
9	11	Clear Green Turtle	NaN	157	157	1893	1937	0.25	60.0
10	12	Striped Bass Saute, Meuniere	NaN	2	2	1900	1900	0.00	0.0
11	13	Anchovies	NaN	453	484	1858	1987	0.00	30.0
12	14	Fresh Lobsters In Every Style	NaN	4	4	1899	1900	0.00	0.0
13	15	Celery	NaN	4246	4690	1	2928	0.00	50.0
14	16	Pim-Olas	NaN	145	148	1897	1918	0.15	35.0
15	17	Caviar	NaN	505	534	1880	1987	0.00	75.0
16	18	Sardines	NaN	1425	1484	1856	2928	0.00	50.0
17	19	India Chutney	NaN	16	16	1865	1901	0.10	0.2
18	20	Pickles	NaN	453	472	1852	1987	0.00	10.0
19	21	English Walnuts	NaN	83	86	1851	1948	0.10	0.3

Data: Dish (after Python cleaning)

	id	name	sponsor	event	venue	place	physical_description	occasion	notes	call_number	keywords
0	12463	NaN	HOTEL EASTMAN	BREAKFAST	COMMERCIAL	HOT SPRINGS, AR	CARD; 4.75X7.5;	EASTER;	NaN	1900-2822	NaN
1	12464	NaN	REPUBLICAN HOUSE	[DINNER]	COMMERCIAL	MILWAUKEE, [WI];	CARD; ILLUS; COL; 7.0X9.0;	EASTER;	WEDGEWOOD BLUE CARD; WHITE EMBOSSED GREEK KEY ...	1900-2825	NaN
2	12465	NaN	NORDDEUTSCHER LLOYD BREMEN	FRUHSTUCK/BREAKFAST;	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE;	CARD; ILLU; COL; 5.5X8.0;	NaN	MENU IN GERMAN AND ENGLISH; ILLUS, STEAMSHIP A...	1900-2827	NaN
3	12466	NaN	NORDDEUTSCHER LLOYD BREMEN	LUNCH;	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE;	CARD; ILLU; COL; 5.5X8.0;	NaN	MENU IN GERMAN AND ENGLISH; ILLUS, HARBOR SCEN...	1900-2828	NaN
4	12467	NaN	NORDDEUTSCHER LLOYD BREMEN	DINNER;	COMMERCIAL	DAMPFER KAISER WILHELM DER GROSSE;	FOLDER; ILLU; COL; 5.5X7.5;	NaN	MENU IN GERMAN AND ENGLISH; ILLUS, HARBOR SCEN...	1900-2829	NaN

Data: menu (before Python cleaning)

	id	name	sponsor	event	venue	place	physical_description	occasion	notes	call_number	keywords	language	d
0	12463	Unknown	Hotel Eastman	Breakfast	Commercial	Hot Springs, Ar	CARD; 4.75X7.5;	Easter	Unknown	1900-2822	NaN	NaN	19 04
1	12464	Unknown	Republican House	Dinner	Commercial	Milwaukee, Wi	CARD; ILLUS; COL; 7.0X9.0;	Easter	Wedgewood Blue Card White Embossed Greek Key B...	1900-2825	NaN	NaN	19 04
2	12465	Unknown	Norddeutscher Lloyd Bremen	Fruhstuck/Breakfast	Commercial	Dampfer Kaiser Wilhelm Der Grosse	CARD; ILLU; COL; 5.5X8.0;	Unknown	Menu In German And English Illus, Steamship An...	1900-2827	NaN	NaN	19 04
3	12466	Unknown	Norddeutscher Lloyd Bremen	Lunch	Commercial	Dampfer Kaiser Wilhelm Der Grosse	CARD; ILLU; COL; 5.5X8.0;	Unknown	Menu In German And English Illus, Harbor Scene...	1900-2828	NaN	NaN	19 04
4	12467	Unknown	Norddeutscher Lloyd Bremen	Dinner	Commercial	Dampfer Kaiser Wilhelm Der Grosse	FOLDER; ILLU; COL; 5.5X7.5;	Unknown	Menu In German And English Illus, Harbor Scene...	1900-2829	NaN	NaN	19 04

Data: menu (after Python cleaning)

```
temp = menupage.groupby('menu_id').agg({'page_number':set}).reset_index()
temp['missingPage'] = temp['page_number'].apply(lambda x:missingPage(x))
```

```
temp[temp['missingPage'] == True]
```

	menu_id	page_number	missingPage
5810	21467	{1.0, 3.0}	True
6068	21725	{32.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0,...	True
6250	21907	{1.0, 2.0, 4.0, 5.0, 6.0}	True
6605	22265	{2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0,...	True
8267	23936	{1.0, 2.0, 4.0, 5.0}	True
...	...	...	...
18243	33953	{1.0, 4.0}	True
18311	34021	{2.0, 3.0, 4.0, 5.0, 6.0}	True
19186	34897	{1.0, 7.0}	True
19256	34967	{1.0, 2.0, 3.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0,...	True
19761	35472	{1.0, 3.0}	True

1070 rows × 3 columns

Certain pages do not exist in certain menus (such as page 2 in menu_id 21467).

OpenRefine

After Cleaning :

sponsor	event	venue	place	physical_description_type	physical_description_additional
HOTEL EASTMAN	BREAKFAST	Commercial	Hot Springs, Ar	CARD	4.75X7.5 - -
REPUBLICAN HOUSE	DINNER	Commercial	Milwaukee, Wi	CARD	ILLUS - COL - 7.0X9.0;

SQLite

Before Cleaning :

```
sqlite> select * from dish where lowest_price > highest_price;
id      name    "description"  menus_appeared  times_appeared  first_appeaared  last_appeared  lowest_price  highest_price
------  ------  -------------  --------------  --------------  ---------------  -------------  ------------  -------------
137514  g (>(   >2>                            1               1                1897           1897.0        0.0
                 (

                @(B

137519  h (>(>  @ 6@0@(B                       1               1                1897           1897.0        0.0
        ?2G
```

After Cleaning :

```
sqlite> select * from dish where lowest_price > highest_price;
```

No records found.

4. A summary of data changes

- Got rid of special characters and unnecessary leading/trailing whitespaces
- cluster the strings with key-collision, fingerprint, n-gram, meta-phone3 and cologne-phonetic methods
- More detailed steps available in the section Actual Data Cleaning Workflow

For Menu.csv

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

For MenuPage.csv

There was nothing to be refined here so the

Unchanged columns: id, menu_id, page_number, image_id, full_height, full_width, uuid

For MenuItem.csv

Unchanged columns: id, menu_page_id, price, high_price, dish_id, created_at, updated_at, xpos, ypos

For Dish.csv

Changed Column : name

data	column	cells_changed
Menu	id	0
Menu	name	15170
Menu	sponsor	11046
Menu	event	10594
Menu	venue	17544
Menu	place	16932
Menu	occasion	17545
Menu	notes	16464
Menu	call_number	1571
Menu	keywords	17545
Menu	language	17545
Menu	date	17545
Menu	location	1962
Menu	location_type	17545
Menu	currency	11090
Menu	currency_symbol	11097
Menu	status	0
Menu	page_count	0
Menu	dish_count	0

cells changed by column: Menu

	data	column	cells_changed
0	MenuItem	id	1
1	MenuItem	menu_page_id	1
2	MenuItem	price	1
3	MenuItem	high_price	1
4	MenuItem	dish_id	1
5	MenuItem	created_at	1
6	MenuItem	updated_at	1
7	MenuItem	xpos	1
8	MenuItem	ypos	1

cells changed by column: MenuItem

	data	column	cells_changed
0	MenuPage	id	0
1	MenuPage	menu_id	0
2	MenuPage	page_number	1202
3	MenuPage	image_id	0
4	MenuPage	full_height	329
5	MenuPage	full_width	329
6	MenuPage	uuid	0

cells changed by column: MenuPage

	data	column	cells_changed
0	Dish	id	1
1	Dish	name	1
2	Dish	description	1
3	Dish	menus_appeared	1
4	Dish	times_appeared	1
5	Dish	first_appeared	1
6	Dish	last_appeared	1
7	Dish	lowest_price	1
8	Dish	highest_price	1

cells changed by column: Dish

IC constraint solved: created_at in MenuItem are now all <= updated_at

Other data quality improvement:

Reduced number of unique values in columns, meaning that we were able to merge similar strings to the same category

	data	column	before	after	reduction
0	Menu	name	797	786	1%
1	Menu	sponsor	6370	5897	7%
2	Menu	event	1770	1628	8%
3	Menu	venue	233	155	33%
4	Menu	place	3714	3486	6%
5	Menu	occasion	423	340	19%
6	Menu	notes	6969	6887	1%
7	Menu	call_number	15936	15936	0%
8	Menu	date	6599	6599	0%
9	Menu	location	6283	6251	0%
10	Menu	currency	42	42	0%
11	Menu	currency_symbol	34	34	0%
12	Menu	status	2	2	0%
13	Menu	page_count	46	46	0%
14	Menu	dish_count	555	555	0%

Reduction of number of unique values by column: Menu

	data	column	before	after	reduction
0	MenuItem	price	1336	581	56%
1	MenuItem	high_price	671	670	0%
2	MenuItem	created_at	1291090	1040	99%
3	MenuItem	updated_at	1295796	1132	99%
4	MenuItem	xpos	1323	1181	10%
5	MenuItem	ypos	616305	80975	86%

Reduction of number of unique values by column: MenuItem

	data	column	before	after	reduction
0	MenuPage	page_number	74	74	0%
1	MenuPage	full_height	5612	5612	0%

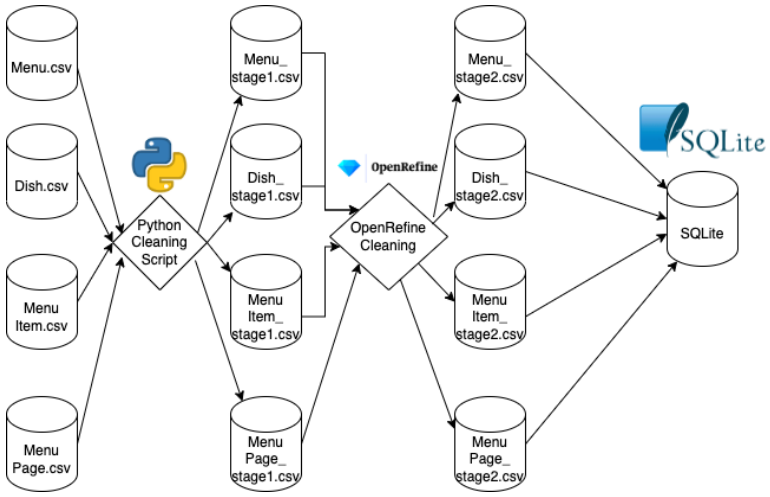
Reduction of number of unique values by column: MenuPage

	data	column	before	after	reduction
0	Dish	name	423363	343888	18%
1	Dish	menus_appeared	552	552	0%
2	Dish	times_appeared	568	568	0%
3	Dish	first_appeared	144	144	0%
4	Dish	last_appeared	144	144	0%
5	Dish	lowest_price	647	647	0%
6	Dish	highest_price	711	711	0%

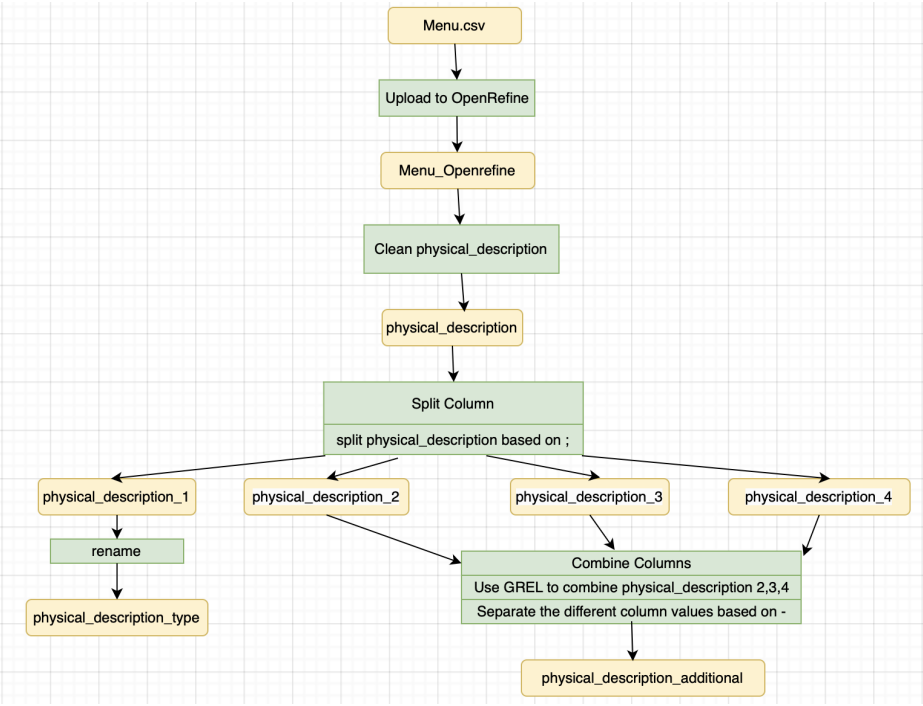
Reduction of number of unique values by column: Dish

Workflow

Figure I. In our high-level workflow document, we start with the 4 csv files. Then, we clean the data twice with Python and OpenRefine. We stood up a SQLite db with the schema files described in Phase I. Finally, we imported the cleaned data into the database.



Below is the inner workflow for cleaning physical_description that have been implemented using openrefine to :



5. A summary of findings, problems encountered and lessons learned

a. data can be very messy (there might be 100+ phrases that is related to different sort of anniversaries
b. Sometimes we need to make a judgement call. It's ambiguous that whether 'anniversary celebration' and 'anniversary reunion' should be merged to the same category
c. Sometimes Integrity Constraints are not straightforward to fix (such as missing pages in a menu)

Contributions :

Tianhao: Python script, Python version of IC constraints, starting the shared report document, taking care of deliverables

Christopher: Python script, starting the Github Repo, Outer Workflow

Akriti: OpenRefine, Inner Workflow, E-R Diagram and schema , Table Creation and Data Load into SQLite and IC Violations check using SQLite