CS 513: Theory and Practice of Data Cleaning

Final Project: Data Cleaning
University of Illinois, Urbana-Champaign
Summer 2021
Report - II

By

Pradosh Kumar Subudhi
Email: pradosh2@illinois.edu

Thomas James Kalnik
Email: tkalnik2@illinois.edu

Data Cleaning Project Phase II

Our team chooses the following Phase-1 option:
- [X] (A) No change to Phase-1 report
- [ ] (B) Improved (or extended) Phase-1 report

1. **Data cleaning performed :** Identification and Description all data cleaning steps (Dataset Used NYPL Menu)

a. Data Cleaning : Data Cleaning with Python :

For the data cleaning portion of this assignment, we focused our efforts on using OpenRefine, SQLite and Python (with Pandas and the re Regex module). Some of the challenges we encountered with this assignment related to, extraneous character values (such as semicolons, quotation marks, square brackets and various other extraneous characters), erroneous date values (such as dates with year values that are well in the future, or almost 1000 years in the past) and multiple values contained in the same columns.

The data cleaning steps we performed were as follows:

For the event, venue, place and date columns, we used Python and Regex to sanitize the data values (removing extraneous characters and bringing the dates to within a reasonable range) in an appropriate way. The data output was then saved to a new csv file and later merged with the output of the data cleaning that was completed using OpenRefine and SQL.

For the other columns in the dataset, namely the id, name, sponsor, physical_description, occasion, notes, call_number, keywords, language, location, location_type, currency, status, page_count and dish_count columns we used OpenRefine and SQL to clean and sanitize the data.

Once most of the cleaning was completed using Python and OpenRefine, SQLite Studio was used to load the data into different tables and do additional data validation (Integrity Constraint Violation) and cleaning where possible. As most of the data were cleaned using OpenRefine, the scope for cleaning was minimal. Below is the detail of the ER Diagrams, Integrity Constraint Violation Validation and several Cleaning steps performed using SQLite.

b. Data Cleaning : Data Cleaning With OpenRefine :

All the 4 dirty datasets were loaded into the OpenRefine and cleaning operation was performed. The cleaning operation is captured in the json file that is generated by OpenRefine.

1) **Menu.csv**
   Number of records – 17, 547
   a. Please note that 90+ steps were performed on the dataset and it was the most dirty dataset of all. Some steps were repeated and data set was cleaned. So, it was disffilcult to capture all the numbers in the process.

   b. Generic : All the columns where numeric fields were present were converted to number type. Date fields were converted to date type and text types to text.
      i. ***name*** :
         1. Removed leading and trailing spaces.
         2. Collapsed consecutive white spaces.
         3. Converted the values to titlecase.

4. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
    a. Method : key collision
        i. fingerprint
        ii. ngram-fingerprint
        iii. metaphone-3
        iv. cologne-phonetic
        v. Daiktch-Mokotoff
        vi. Beider-Morse
    b. Nearest neighbor
        i. levenshtein
        ii. ppm
5. Used GREL to remove values like " , ( ) { } * [ ] '. ? ;
6. All these above steps were repeated multiple times as each clustering and and cleaning open up an opportunity to apply the transaformation again.

ii. ***sponsor*** :
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Converted the values to titlecase.
4. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
    a. Method : key collision
        i. fingerprint
        ii. ngram-fingerprint
        iii. metaphone-3
        iv. cologne-phonetic
        v. Daiktch-Mokotoff
        vi. Beider-Morse
    b. Nearest neighbor
        i. levenshtein
        ii. ppm
5. Used GREL to remove values like " , ( ) { } * [ ] '. ? \ \ ;
6. All these above steps were repeated multiple times as each clustering and and cleaning open up an opportunity to apply the transaformation again.

iii. ***occasion*** :
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Converted the values to titlecase.
4. Used GREL to remove values like " , ( ) { } * [ ] '. ? \ \ ;
5. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
    a. Method : key collision
        i. fingerprint
        ii. ngram-fingerprint
        iii. metaphone-3
        iv. cologne-phonetic
        v. Daiktch-Mokotoff
        vi. Beider-Morse

        b. Nearest neighbor
            i. levenshtein
            ii. ppm
    6. The opportuinity to clean the data based on the clustering was very helpful. Data like ‘Anniv’, "ANNIVERSARYERSARY;" etc. were clubbed together as ‘Anniversary’. This was one of the example from the list of values those were cleaned.
    7. Used GREL to remove values like “ , ( ) { } * [ ] ‘. ? \ \
    8. All these above steps were repeated multiple times as each clustering and and cleaning open up an opportunity to apply the transformation again.

iv. ***event, venue*** :
    1. Removed leading and trailing spaces.
    2. Collapsed consecutive white spaces.
    3. Converted the values to titlecase.
    4. Used GREL to remove values like “ , ( ) { } * [ ] ‘. ? \ \
    5. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
        a. Method : key collision
            i. fingerprint
            ii. ngram-fingerprint
            iii. metaphone-3
            iv. cologne-phonetic
            v. Daiktch-Mokotoff
            vi. Beider-Morse
        b. Nearest neighbor
            i. levenshtein
            ii. ppm
    6. Used GREL to remove values like “ , ( ) { } * [ ] ‘. ? \ \
    7. All these above steps were repeated multiple times as each clustering and and cleaning open up an opportunity to apply the transformation again.

v. ***place***
    1. Removed leading and trailing spaces.
    2. Collapsed consecutive white spaces.
    3. Used GREL to remove values like “ , ( ) { } * [ ] ‘. ? \ \
    4. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
        a. Method : key collision
            i. fingerprint
            ii. ngram-fingerprint
            iii. metaphone-3
            iv. cologne-phonetic
            v. Daiktch-Mokotoff
            vi. Beider-Morse
        b. Nearest neighbor
            i. levenshtein
            ii. ppm
    5. Used GREL to remove values like “ , ( ) { } * [ ] ‘. ? \ \

6. All these above steps were repeated multiple times as each clustering and and cleaning open up an opportunity to apply the transformation again.

vi. ***physical_descrription***
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Used GREL to remove values like " , ( ) { } * [ ] '. ? \ \
4. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
5. The opportunity to clean the data was minimal.
6. The cleaning was not necessary and could have been left as it is.

vii. ***occasion***
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Used GREL to remove values like " , ( ) { } * [ ] '. ? \ \
4. Clustering was used to group different data values and appropriate action were taken on the values. All these below methods were used to cluster the data.
    a. Method : key collision
        i. fingerprint
        ii. ngram-fingerprint
        iii. metaphone-3
        iv. cologne-phonetic
        v. Daiktch-Mokotoff
        vi. Beider-Morse
    b. Nearest neighbor
        i. levenshtein
        ii. ppm
5. Used GREL to remove values like " , ( ) { } * [ ] '. ? \ \
6. All these above steps were repeated multiple times as each clustering and and cleaning open up an opportunity to apply the transformation again

viii. ***notes***
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Used GREL to remove values like " , ( ) { } * [ ] '. ? \ \
4. Few clusters were available for cleaning considering the nature of data avalable.
5. Opportunity was less.

ix. ***call_number***
1. Few clusters were available for cleaning considering the nature of data avalable.
2. Opportunity was less.

x. ***keywords, language, location, location_type***
1. The columns were removed because of lack of data in them.

xi. ***date***
1. Converted the column to Date type.

2. Formatted the data to ISO standard - toString(toDate(value),"yyyy-MM-dd")

xii. ***currency***
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Converted the values to titlecase.

xiii. ***status***
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Converted the values to titlecase.
4.

xiv. ***page_count, dish_count***
1. No scope for transformation other than changing the column type to number type.

xv. currency_symbol
1. No Cleaning.

2) Dish.csv

Number of records – 428,082

a. Please note that because of the size of the dataset, the OpenRefine tool was not able to apply any Facets. Because of this, the mess in the data was not possible to identify.
b. With careful observation, many dirty data were cleaned.
c. Generic : All the columns where numeric fields were present were converted to number type. Date fields were converted to date type and text types to text.

i. ***name*** :
1. Removed leading and trailing spaces. – Rows Affected - 6582
2. Collapsed consecutive white spaces. - Rows Affected - 9288
3. Converted the values to titlecase.
4. Clustering was not possible.
5. Used GREL to remove values like " , ( ) { } * [ ] '. ? -
a. Rows Affected – 7009, 2568, 5 1801, 18, 309, 33015 (different types of characters)

ii. id, menus_appeared, times_appeared, lowest_price, highest_price :
1. Removed leading and trailing spaces.
2. Collapsed consecutive white spaces.
3. Converted the values to numbers.

iii. ***first_appeared, last_appeared***
1. Date transformations.

3) MenuItem.csv

Number of records – 1,334,779

a. Please note that because of the size of the dataset 118MB, the OpenRefine tool was not able to apply any Facets.
b. Data was mostly clean.
c. Some generic transformations applied.
d. Text transform on 1334779 cells in column menu_page_id: value.toNumber()
e. Text transform on 888520 cells in column price: value.toNumber()
f. Text transform on 91979 cells in column high_price: value.toNumber()
g. Text transform on 1334538 cells in column dish_id: value.toNumber()
h. Text transform on 1334779 cells in column xpos: value.toNumber()
i. Text transform on 1334779 cells in column ypos: value.toNumber()

4) **MenuPage.csv**
Number of records – 66,937
   a. Data was mostly clean.
   b. Few generic transfortmation was applied.
   c. Text transform on 66937 cells in column id: value.toNumber()
   d. Text transform on 66937 cells in column menu_id: value.toNumber()
   e. Text transform on 65735 cells in column page_number: value.toNumber()
   f. Text transform on 66914 cells in column image_id: value.toNumber()
   g. Text transform on 66608 cells in column full_height: value.toNumber()
   h. Text transform on 66608 cells in column full_width: value.toNumber()
   i. Text transform on 1 cells in column uuid: value.toLowercase()

c. Rationale :

Each data cleaning step had an important role in the overall process of correcting the data. Referring to our original use case described in the Final Project Proposal, the U1 use case we described was:

U1 - Data is Fit-for-Purpose:

A consulting group is evaluating menu options and offerings as a service to a new fine dining restaurant set to open in New York City. The consulting group needs a clean dataset with no duplicates to conduct their research, but some minor discrepancies in the data are acceptable. Data Cleaning is valuable in this context because in the present form (raw data, i.e., D) the dataset contains blank values, inconsistency in naming convention and data types and duplicate values that can be merged. Completing these steps will improve the quality of the data enough that it is a valuable resource to a consulting firm to make recommendations to a new fine dining establishment on what type of menus they should offer.

The data cleaning steps applied were relevant or not relevant to this use case in the following ways:

- Clustering was used to group different data values, which allows for the removal of duplicate values in the dataset. As described in the U1 use case, this is a prerequisite for the U1 use case (namely. A consulting group evaluating menu options and providing recommendations to a fine dining establishment client.

- Removing trailing and leading white spaces has little relevance to the U1 use case, a consulting group could still use the data without this transformation step, however, readability and usability are improved by this step.

- Removing extraneous character values like “ , ( ) { } * [ ] ‘. ? ; has some relevance to the U1 use case. While the consulting group could theoretically make use of data containing these characters, readability is improved through the removal and this transformation step also allows for better clustering of the data values, which removes additional duplicates and improves the quality and usability of the dataset.

- Data Type conversion had some relevance to the U1 use case. While generally not directly applicable to a consultant evaluating menu options to make recommendations to a restaurant business, data type is sometimes relevant to the readability and accuracy of the data. For example, dates and number values can contain higher precision when stored with the correct data type and this precision can allow for more accurate data value storage in addition to better interpretation and utilization of the data.

2. Data Quality Improvements :

a. **Quanifying the transformations:** Summary Table Change

File	Column	Quality Change Measure
Menu.csv	id	Text transform on 676 cells in column sponsor: grel:value.replace(/\?/,'').replace(/\(|\)/,'').replace(/\"/,'')
		Text transform on 474 cells in column sponsor: grel:value.replace(/\\|\//,'').replace(/\"/,'').replace(/\[|\]/,'')
		Text transform on 8275 cells in column sponsor: value.toTitlecase()
		Clustering : key collision - fingerprint Mass edit 3937 cells in column sponsor ngram-fingerprint Mass edit 1787 cells in column sponsor metaphone-3 Mass edit 524 cells in column sponsor cologne-phonetic Mass edit 1676 cells in column sponsor Daitch-Mokotof Mass edit 163 cells in column sponsor Beider-Morse Mass edit 23 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 1617 cells in column sponsor ppm 0
	name	Text transform on 28 cells in column name: grel:value.replace(/\?/,'').replace(/\(|\)/,'').replace(/\"/,'') Text transform on 139 cells in column name: grel:value.replace(/\\|\//,'').replace(/\"/,'').replace(/\[|\]/,'').replace(/*/,''))
		Clustering : key collision - fingerprint Mass edit 556 cells in column sponsor ngram-fingerprint Mass edit 622 cells in column sponsor metaphone-3 Mass edit 762 cells in column sponsor cologne-phonetic Mass edit 4 cells in column sponsor Daitch-Mokotof Mass edit 12 cells in column sponsor Beider-Morse Mass edit 0 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 7 cells in column sponsor

		ppm 60
	event	Text transform on 9393 cells in column event: value.toString()
		Text transform on 99 cells in column event: grel:value.replace(/\?/,'').replace(/\(|\)/,'').replace(/\"/,'')
		Text transform on 90 cells in column event: grel:value.replace(/\\|\//,'').replace(/\"/,'').replace(/\[|\]/,'')
		Text transform on 7826 cells in column event: value.toTitlecase()
		Clustering : key collision - fingerprint Mass edit 4027cells in column sponsor ngram-fingerprint Mass edit 2325 cells in column sponsor metaphone-3 Mass edit 3057 cells in column sponsor cologne-phonetic Mass edit 2297 cells in column sponsor Daitch-Mokotof Mass edit 98 cells in column sponsor Beider-Morse Mass edit 2 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 407 cells in column sponsor ppm 0
	venue	Text transform on 9449 cells in column venue: value.toString()
		Text transform on 8096 cells in column venue: value.toTitlecase
		Clustering: fingerprint Mass edit 2126 cells in column sponsor ngram-fingerprint Mass edit 21 cells in column sponsor metaphone-3 Mass edit 5044 cells in column sponsor cologne-phonetic Mass edit 0 cells in column sponsor Daitch-Mokotof Mass edit 505 cells in column sponsor Beider-Morse Mass edit 0 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 0 cells in column sponsor ppm Mass edit 7 cells in column sponsor
	physical_description	Text transform on 2777 cells in column physical_description: value.toString()
		Text transform on 38 cells in column physical_description: value.replace(/\s+/,' ')
		Clustering: fingerprint Mass edit 1514 cells in column sponsor

		ngram-fingerprint Mass edit 4575 cells in column sponsor metaphone-3 Mass edit 0 cells in column sponsor cologne-phonetic Mass edit 2 cells in column sponsor Daitch-Mokotof Mass edit 0 cells in column sponsor Beider-Morse Mass edit 0 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 0 cells in column sponsor ppm Mass edit 0 cells in column sponsor
	Occasion	Text transform on 13744 cells in column occasion: value.toString()
		Text transform on 508 cells in column sponsor: grel:value.replace(/\?/,'').replace(/\(|\)/,'').replace(/\"/,'')
		Text transform on 474 cells in column sponsor: grel:value.replace(/\\|\//,'').replace(/\"/,'').replace(/\[|\]/,'')
		Text transform on 2954 cells in column occasion: grel:value.replace(/\\|\//,'').replace(/\"/,'').replace(/\[|\]/,'').replace(/;/,''). replace(/*/,'')
		Clustering ; fingerprint Mass edit 969 cells in column sponsor ngram-fingerprint Mass edit 272 cells in column sponsor metaphone-3 Mass edit 1393 cells in column sponsor cologne-phonetic Mass edit 693 cells in column sponsor Daitch-Mokotof Mass edit 215 cells in column sponsor Beider-Morse Mass edit 0 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 281 cells in column sponsor ppm Mass edit 716 cells in column sponsor Text transform on 3752 cells in column occasion: value.toTitlecase()
	notes	Text transform on 195 cells in column notes: value.replace(/\s+/,' ') Text transform on 9460 cells in column notes: value.toTitlecase()
		Clustering: fingerprint Mass edit 1355 cells in column sponsor ngram-fingerprint Mass edit 320 cells in column sponsor metaphone-3 Mass edit 0 cells in column sponsor cologne-phonetic Mass edit 268 cells in column sponsor Daitch-Mokotof Mass edit 0 cells in column sponsor Beider-Morse

		Mass edit 0 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 0 cells in column sponsor ppm Mass edit 0 cells in column sponsor
		Text transform on 851 cells in column sponsor: grel:value.replace(/\?/,'').replace(/\(|\)/,'').replace(/\"/,'')
		Text transform on 267 cells in column notes: grel:value.replace(/\\|\//,'').replace(/\"/,'').replace(/\[|\]/,'')
		Text transform on 6034 cells in column notes: grel:value.replace(/\;$/,'')
	call_number	Clustering : Mass edit 0 cells in column sponsor ngram-fingerprint Mass edit 0 cells in column sponsor metaphone-3 Mass edit 0 cells in column sponsor cologne-phonetic Mass edit 0 cells in column sponsor Daitch-Mokotof Mass edit 0 cells in column sponsor Beider-Morse Mass edit 0 cells in column sponsor nearest-neighbor - lavenshtein Mass edit 0 cells in column sponsor ppm Mass edit 0 cells in column sponsor
	date	Text transform on 16958 cells in column date: value.toDate() Text transform on 16958 cells in column date: grel:toString(toDate(value),"yyyy-MM-dd")
	status	Text transform on 17547 cells in column status: value.toTitlecase()
	page_count, dish_count	Text transform on 17547 value.toNumber()
Menupage.csv	id	Text transform on 66937 cells in column id: value.toNumber()
	menu_id	Text transform on 66937 cells in column menu_id: value.toNumber()
	page_number	Text transform on 65735 cells in column page_number: value.toNumber()
	image_id	Text transform on 66914 cells in column image_id: value.toNumber()
	full_height	Text transform on 66608 cells in column full_height: value.toNumber()
	full_width	Text transform on 66608 cells in column full_width: value.toNumber()
	uuid	Text transform on 1 cells in column uuid: value.toLowercase()
Menuitem.csv	menu_page_id	Text transform on 1334779 cells in column menu_page_id: value.toNumber()
	price	Text transform on 888520 cells in column price: value.toNumber()
	high_price	Text transform on 91979 cells in column high_price: value.toNumber()
	dish_id	Text transform on 1334538 cells in column dish_id: value.toNumber()
	xpos	Text transform on 1334779 cells in column xpos: value.toNumber()
	ypos	Text transform on 1334779 cells in column ypos: value.toNumber()

Dish.csv	name	Removed leading and trailing spaces. – Rows Affected - 6582 Collapsed consecutive white spaces. - Rows Affected - 9288 Converted the values to titlecase. Clustering was not possible. Used GREL to remove values like " , ( ) { } * [ ] '. ? - Rows Affected – 7009, 2568, 5 1801, 18, 309, 33015 (different types of characters)
	id, menus_appeared, times_appeared, lowest_price, highest_price	Removed leading and trailing spaces. Collapsed consecutive white spaces. Converted the values to numbers.
	first_appeared, last_appeared	Date transformations. (mostly all records)
SQL		Quality improvements by SQL is given below.

b. Data Cleaning With SQL & Quality Improvements:

Once most of the cleaning was completed using Pyhton and OpenRefine, SQLite Stuido was used to load the data into different tables and do additional data validation (Integrity Constraint Violation) and cleaning where possible. As most of the data were cleaned using OpenRefine, the scope for cleaning was miminlal. Below is the detail of the ER Diagrams, Integrity Constraint Violation Vaidation and few Cleaning steps perfomed using SQLite.

The tables were designed using SQLite and the ER Diagram and Schema has been placed in appropriate folder.

Please see the ICV queries and summary statistics provided below. Please also see the provided queries.txt file in the attached zip archive.

Integrity Constraint Check & Additonal Cleaning:

1) **Criteria:** id shouldn't have dupliacte or NULL Values on menu table.

   a) ***Table : menu***
   Query:
   SELECT id, COUNT(id) AS cnt_id
   FROM menu
   GROUP BY id
   HAVING COUNT(id) > 1;

   ***Resut** : 0 Records*
   ***Action** : No action.*

   b) ***NULL Value Check:***
   Query:
   *SELECT * FROM menu*
   WHERE id IS NULL;

   ***Result:** 0 such records.*
   ***Action** : No action.*

2) **Criteria:** id shouldn't have dupliacte or NULL Values on menuitem table.

a) ***Table : menuitem***
Query:
SELECT id, COUNT(id) AS cnt_id
FROM menu
GROUP BY id
HAVING COUNT(id) > 1;

***Resut** : 0 Records*
***Action** : No action.*

b) ***NULL Value Check:***
Query:
*SELECT * FROM menuitem*
WHERE id IS NULL;

***Result:** 0 such records.*
***Action** : No action.*

3) **Criteria:** id shouldn't have dupliacte or NULL Values on dish table.
a) ***Table : menuitem***
Query:
SELECT id, COUNT(id) AS cnt_id
FROM dish
GROUP BY id
HAVING COUNT(id) > 1;

***Resut** : 0 Records*
Action** : No **action.

b) ***NULL Value Check:***
Query:
*SELECT * FROM dish*
WHERE id IS NULL;

***Result:** 0 such records.*
***Action** : No action.*

4) **Criteria:** id shouldn't have dupliacte or NULL Values on menupage table.
a) ***Table : menuitem***
Query:
SELECT id, COUNT(id) AS cnt_id
FROM menupage
GROUP BY id
HAVING COUNT(id) > 1;

***Resut** : 0 Records*
***Action** : No action.*

b) NULL Value Check:

Query:

*SELECT * FROM menupage*
WHERE id IS NULL;

Result: *0 such records.*
Action *: No action.*

Data Cleaning Using SQL:

1) **Table** : ***menu***
   a. Identify and delete irrelevant records.

Query:

SELECT COUNT(*)
FROM menu
WHERE name = ‘’
AND sponsor= ‘’;
Result*:* 1619

Obervation : 1680 records where both name and sponsor are blank don’t have most of its other columns not populated. So, all such recors were deleted since they don’t hold enough data for those records to be relevant.
Action : Delete such records.
Query:
DELETE FROM menu
WHERE name = ‘’
AND sponsor= ‘’;

Result*:* 1619 records were deleted.

   b. Default columns where most of the data are missing so that they can be ised for categorizing.

Query:

SELECT COUNT(*)
FROM MENU
WHERE NAME='';
Result*:* 12729
Action : Default such records with value as ‘Unavailable’.

Query:

UPDATE menu
SET name = 'Unavialble'
WHERE name = '';
Result*:* 12729 records were updated.

c. Default name and sponsor coumn where most of the data are like (Restaurant Andor Location Not Given./ Restaurant Name Andor Location Not Given/ Not Given) so that they can be used for categorizing
Query:
SELECT COUNT(*) FROM menu
WHERE name LIKE '%NOT GIVEN%'
Result*:* 138
Action : Default such records with value as 'Unavailable'.

Query:
UPDATE menu
SET name = 'Unavialble'
WHERE name LIKE '%NOT GIVEN%';
Result*:* 138 records were updated.

Query:
SELECT COUNT(*) FROM menu
WHERE sponsor LIKE '%NOT GIVEN%';
Result*:* 216

Query:
UPDATE menu
SET sponsor = 'Unavialble'
WHERE sponsor LIKE '%NOT GIVEN%';
Result*:* 216 records were updated.

2) **Table** : ***dish***
a. Identify and delete irrelevant records.
Query:
SELECT count(*) FROM dish
WHERE times_appeared = '';
Result*:* 0

Query:
SELECT count(*) FROM dish
WHERE menus_appeared = '';
Result*: 0*

Action : None. No such records were found.

b. Last appeared_date had years beoyond 2021.
Query:
SELECT COUT+NT(*) FROM dish
WHERE first_appeared > '2021';
Result*:* 11

Action : Delete such records.

DELETE FROM dish
WHERE first_appeared > '2021';
Result*:* 11 records were deleted.

c. Last appeared_date had years beoyond 2021.
Query:
SELECT COUT+NT(*) FROM dish
WHERE last_appeared > '2021';
Result*:* 180
Action : Delete such records.
DELETE FROM dish
WHERE last_appeared > '2021';
Result*:* 180 records were deleted.

d. highest_price column contains a lot spaces. Update them to 0.
Query :
*SELECT * FROM dish*
WHERE highest_price = ' '
Result: 29,098
Action : Default the spaces to 0.

Query :
UPDATE dish
SET highest_price = 0
WHERE highest_price = '';
Result: 29,098 records updated.

e. Check the same for lowest_price.
Query :
*SELECT * FROM dish*
WHERE lowest_price = ' '
Result: 0
Action : None.

Query :
UPDATE dish
SET lowest_price = 0
WHERE highest_price = '';
Result: 446,259 records updated.

3) **Table** : ***menuitem***

a. high_price column contains a lot spaces. Update them to 0.
Query :
SELECT * FROM menuitem
WHERE high_price = ' '
Result: 1,242,800
Action : Default the spaces to 0.

Query :
UPDATE menuitem
SET high_price = 0
WHERE high_price = '';

Result: 1,242,800 records updated.

b. price column contains a lot spaces. Update them to 0.
Query :
SELECT * FROM menuitem
WHERE price = ' '
Result: 446,259
Action : Default the spaces to 0.

Query :
UPDATE menuitem
SET price = 0
WHERE price = '';
Result: 446,259 records updated.

4) **Table** : ***menupage***
No relevant data found to be cleaned as such.

Quality Improvements:

There has been a lot of quality improvement in the data. The data we started with was very messy and unreadable. It was not possible for any analyst to perform any analysis to gain insight of the data to answer the basic questions. The data couldn't be used in any tables or any tools to start with the analysis/ exploration.

After the data cleaning steps were performed, we clearly see from the statistics above that the with cleaning/ transformations operation, we were able to fix recrods ranging from few 100 to even more that 100K. These includes removing bad data, cleaning records which don't contain any data, integrity constraint check (which ensures the data can be used in table), formatting the data, grouoing/ clusetring etc. which resulted in a dataset which is clean, nice and in a more usable format for next step as analysis/ exploration.

3. WorkFlow Models :

Once the cleaning operation were performed, we worked on creating the WorkFlow diagram to provide an overall picture of what operations were performed, the data flow and even the granular level of details of each and every step involved in the cleaning.

The overal flow was created using draw.io.

For Python, YesWorkflow tool was used and the necessary outputs are provided in the appropriate folders.

For OpenRefine, or2ywtool was used to create the flow diagrams. The yw files and diagrams (both parallel and linear) have been provided. The tool didn't generate the .gv files and hence are not supplied.

Please see the provided Flow Diagrams and YW Files in the Flow Diagrams & YW Files and Python Code & python YW diagrams folders of the zip attachment.

Please see the provided Flow Diagrams and YW Files in the Flow Diagrams & YW Files and Python Code & python YW diagrams folders of the zip attachment.

4. Conclusion & Summary :

This project involved the use of OpenRefine, SQL and Python in cleaning the NY Public Library Menus dataset.

This project was beneficial in a way that it allowed us to gain deeper experience with the tools and workflows presented in the class, specifically, OpenRefine, SQL and Python Pandas. These tools are powerful assistants in the Data Analyst's arsenal and allow for fast and efficient manipulation of large, messy datasets that contain problems such as missing features, extraneous characters, and issues related to data type and dates. Proficiency in these tools is an invaluable skill set for any aspiring or seasoned Data Analyst, Data Engineer, Data Scientist or Software Developer.

The contributions of the teammates were as follows:

Pradosh led the data cleaning efforts with OpenRefine, SQLite, SQL Database Schema Diagram, OverallFlow diagram, SQLFlow Diagram, , or2ywtool, contributed to writing the report.

Thomas was involved in the generation of the SQL Database Schema Diagram, YesWorkflow Diagrams wrote the python code used for the data cleaning component of the project and took the lead in compiling and writing the report.

5. Supplemental Materials :

Conceptual Model/ Database Schema :

\CS513 - Data Cleaning Final Project - Submission\Conceptual model - Database Schema

OpeRefine Operation History : (Folder contains a consolidated recipe as well separate recipe)

****CS513 - Data Cleaning Final Project - Submission\OpenRefine History

OpeRefine Operation History :

\CS513 - Data Cleaning Final Project - Submission\OpenRefine History

Outer and Inner Workflow Models:

OpenRefine\ YesWorkFlow Diagram\ SQL Workflow : (Outer & Inner)

\CS513 - Data Cleaning Final Project - Submission\OpenRefina & SQL - Flow Diagrams & YW Files (Folder contains both parallel and linear flow diagrams and .yw files.)

Note : Overall workflow was prepared using draw.io. Hence .yw and .gv files are not applicable. Similarly, or2ywflow didn't produce any .gv files and hence not supplied.

Pyhton Script\ YesWorlFlow Diagram: (Inner)

\CS513 - Data Cleaning Final Project - Submission\Python Code & python YW diagrams

Queries:

\CS513 - Data Cleaning Final Project - Submission\Queries

Raw & Cleaned Datasets: (File contains the link to UIUC box drive)

\CS513 - Data Cleaning Final Project - Submission\Data\DataLinks.txt