CS 513: Theory and Practice of Data Cleaning

Final Project: Data Cleaning NYPL Menus
(Team 52)

Data Cleaning Project Phase II

Improved Phase - 1 report

Target Use Case U1

We wanted to clean the data to find out historical information about the menu and dishes which were served. For example, what were the menu items which were served for Dinner at New York back in the 1960s? And how menu items were priced and what were the dishes served during that event? What were the locations where these dishes were served?

Data Cleaning Performed

Introduction

In this section, we have described how the data cleaning was performed using various technologies and tools. We used the dataset NYPL menus. We used Openrefine primarily to clean most of the data, we also used Python to clean up special characters from strings using Regex. And finally we used SQLlit3 database and queries to fix integrity constraint violations. The Overview Of the cleaning process section shows how the cleaning process was done for the use case mentioned above.

1) Using OpenRefine

We performed data cleaning on all the four csv files of NYPL. All the 4 csv files were uploaded into the Openrefine to do the cleaning. The cleaning operations were captured in the JSON files.

Menu.csv (Total number of records = 17,546)

Below is the list of columns for the Menu.csv file.

- ➔ Please be advised that we performed a lot of steps (100+) so it is not possible to list down all of them in this document. Many steps were repeatedly performed to clean data on various columns.
- ➔ Below are some of the updates we made on the columns of the csv -

- **Id**
  - Converted the column values to number
- **name**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
  - We used clustering to group the data values into multiple groups. The below methods were used to cluster the data (Key collision and Nearest Neighbor).

- **sponsor**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
- **Event**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
  - We used clustering to group the data values into multiple groups. The below methods were used to cluster the data (Key collision).
- **Venue**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
  - We used clustering to group the data values into multiple groups. The below methods were used to cluster the data (Key collision and Nearest Neighbor).
- **place**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.

- We used clustering to group the data values into multiple groups. The below methods were used to cluster the data (Key collision and Nearest Neighbor).

- **physical_description**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
- **Occasion**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
  - We used clustering to group the data values into multiple groups. The below methods were used to cluster the data (Key collision and Nearest Neighbor).
- **Notes**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
- **Call_number**
  - Converted the column values to number
- **keywords,language,location_type**
  - Removed the columns
- **date**
  - Converted the column values to date
- **location**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.
- **currency**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
- **currency_symbol**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.

- **status**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
- **page_count**
  - Converted the column values to number
- **dish_count**
  - Converted the column values to date

Dish.csv (Total number of records = 423397)

Data was mostly clean. Most columns were numeric and some cleanup has been done. The name column has cleanup in terms of clustering of data and cleanup of special characters.

- ➔ Id
  - Transformed 423397 cells in column id to Number
- ➔ Menus_appeared
  - Transformed 423397 cells in column menus_appeared to Number
- ➔ Times_appeared
  - Transformed 423397 cells in column times_appeared to Number
- ➔ First_appeared
  - Transformed 423397 cells in column first_appeared to Number
- ➔ Last_appeared
  - Transformed 423397 cells in column last_appeared to Number
- ➔ Lowest_price
  - Transformed 394297 cells in column lowest_price to Number
- ➔ Highest_price
  - Transformed 394297 cells in column highest_price to Number
- ➔ Name
  - Trimmed 9045 cells in column name to remove leading and trailing spaces
- ➔ Trimmed 6415 cells in column name to remove consecutive white spaces
  - Transformed 406726 cells in column name to have Upper Case
- ➔ Created a column facet name_facet and applied RegEx to update 423397 rows with below:
  - value.replace('[',' ')
    .replace(']',' ')
    .replace('{',' ')
    .replace('}',' ')
    .replace('"',' ')
    .replace(/[,;]$/, ' ')
    .replace('?',' ')
    .replace(/\'/,' ')
    .replace('*',' ')

.replace("","' ')
    - Trimmed 8550 cells in column name to remove leading and trailing spaces
    - Trimmed 11302 cells in column name to remove consecutive white spaces
- ➔ Did iterative clustering with name_facet column using "key Collision/fingerprint" to remove the clusters till there were none left. Below are the results in multiple iterations:
    - Mass edit 8284 cells in column name_facet
    - Mass edit 7666 cells in column name_facet
    - Mass edit 7152 cells in column name_facet
    - Mass edit 6848 cells in column name_facet
    - Mass edit 6741 cells in column name_facet
    - Mass edit 6343 cells in column name_facet
    - Mass edit 5959 cells in column name_facet
    - Mass edit 5957 cells in column name_facet
    - Mass edit 5993 cells in column name_facet
    - Mass edit 5984 cells in column name_facet
    - Mass edit 5995 cells in column name_facet
    - Mass edit 5983 cells in column name_facet
    - Mass edit 5671 cells in column name_facet
- ➔ Removed the original column name
- ➔ Renamed the column "name_facet" to "name"

MenuItem.csv(Total number of records = 104,586)

- ➔ Data was mostly clean. Most of the columns were numeric so not much cleaning was needed. Some generic transformations were done.
    - **id**
        - Converted the column values to number
    - **menu_page_id**
        - Converted the column values to number
    - **price**
        - Converted the column values to number
    - **high_price**
        - Converted the column values to number
    - **dish_id**
        - Converted the column values to number
    - **created_at**
        - Converted the column values to Date
    - **updated_at**
        - Converted the column values to Date

- **xpos**
  - Converted the column values to number
- **ypos**
  - Converted the column values to number

MenuPage.csv(Total number of records = 66,938)

➔ Data was mostly clean. Most of the columns were numeric so much cleaning was needed.

- **id**
  - Converted the column values to number
- **menu_id**
  - Converted the column values to number
- **page_number**
  - Converted the column values to number
- **image_id**
  - Converted the column values to number
- **full_height**
  - Converted the column values to number
- **full_width**
  - Converted the column values to number
- **uuid**
  - Removed the white spaces (trailing spaces, consecutive white spaces, etc.)
  - Converted the column values to Uppercase.
  - We used Regex on GREL to remove the special characters.

2) Using Python

We used the **pandas** library to primarily clean the Menu.csv. We used regex on the following columns using pandas dataframe -

a. name
b. sponsor
c. venue
d. event
e. place
f. Location

Also, we fixed the date column using Pandas and then we filtered out the future dates.

One challenge we faced was that there was no one size fits all regex string to work with. We had to write different regexes to remove different sets of characters. Here is a code snippet of cleaning the text columns using regex.

```
# Removing white trailing spaces
df[column_name] = df[column_name].str.strip()

# Removing double quotes
df[column_name] = df[column_name].str.replace('"', '', regex=True)
df[column_name] = df[column_name].str.replace('""', '', regex=True)

# Removing Square brackets
df[column_name] = df[column_name].str.replace('[', '', regex=True)

df[column_name] = df[column_name].str.replace(']', '', regex=True)

# cleaning text (?)
df[column_name] = df[column_name].str.replace('\(\?\)', '', regex=True)

# cleaning text (;$)
df[column_name] = df[column_name].str.replace('\;$', '', regex=True)

# cleaning text ?
df[column_name] = df[column_name].str.replace('?', '', regex=True)

# cleaning text ;
df[column_name] = df[column_name].str.replace(';', '', regex=True)

# cleaning text ()
df[column_name] = df[column_name].str.replace('\(\)', '', regex=True)

# cleaning text (
df[column_name] = df[column_name].str.replace('\(', '', regex=True)

# cleaning text )
df[column_name] = df[column_name].str.replace('\)', '', regex=True)
```

3) Using SQLite3

After the Openrefine and Python cleaning is completed, we loaded the data into a SQLlite3 database with 4 tables. In the section ER Diagram, the relationship diagram captures how the tables are connected via the foreign key relationship.

Below are the summary of changes we did -

- Table name menu
  - Verify null value in id column.
    - Result - No rows were found
  - Check duplicate ids in the id column.
    - Result - No rows were found
  - check count of menu rows where name is null and place is null.
    - Result - 6202 rows were found
    - Action - We deleted such rows.
  - check how many event rows have blank or null values
    - Result - 3387 rows were retrieved.
    - Action - We updated such rows with the value event = Unknown
  - Check how many name values are null or blank
    - Result - 7558 rows were retrieved.
    - Action - We updated such rows with the value name = Unknown
  - Check how many place values are null or blank
    - Result - 3195 rows were retrieved.
    - Action - We updated such place rows with the value = Unknown
  - Check how many location values are null or blank
    - Result - 27 rows were retrieved.
    - Action - We updated such location rows with the value = Unknown

- Table menu_item
  - Verify null value in id column.
    - Result - No rows were found
  - Check duplicate ids in the id column.
    - Result - No rows were found
  - Check if there are any Integrity Constraint violations with the dish table.
    - Result - 3 rows were found
    - Action - We deleted such rows.
  - Check if there are any Integrity Constraint violations with the menu page table.
    - Result - 0 rows were retrieved.

- Table menu_pages
    - Verify null value in id column.
        - Result - No rows were found
    - Check duplicate ids in the id column.
        - Result - No rows were found
    - We checked for empty row values for all columns and updated them to 0. Below are the columns -
        - page_number = 1202 rows updated with 0.
        - full_height = 329 rows were updated with 0.
        - full_width = 329 rows were updated with 0.
    - We also checked for negative values in all numeric columns.
        - No rows found.

- Table dish
    - Verify null value in id column.
        - Result - No rows were found
    - Check duplicate ids in the id column.
        - Result - No rows were found
    - We searched for negative values in times_appeared column-
        - Result - 21 rows were retrieved.
        - Action - Updated them to 0.
    - Checked for the first_appeared and last_appeared columns where values are less than 100 and greater than 2021.
        - Result - 55503 rows found.
        - Action - updated those values to a blank value.
    - Checked for blank values in the lowest_price and highest_price values
        - Result - 29100 rows found.
        - Action - updated those values to 0.

Cleaned records report

Summary table changes

Below is the report of the number of records updated.

Table Name	Column Name	Change log
Menu.csv	place	● Removed white spaces ● Removed unwanted commas, extra characters like [, {, },] using regex. ● Removed clustered the similar names into one using openrefine.
	id	● Text transform on 17545 cells in column id: value.toNumber()
	name	● Text transform on 9 cells in column name: value.trim() ● Text transform on 1 cells in column name:value.replace(/\s+/,' ') ● Text transform on 3196 cells in column name: value.toUppercase() ● Create new column name_facet based on column name by filling 3197 rows with grel:value
	sponsor	● Text transform on 14 cells in column sponsor: value.trim() ● Text transform on 127 cells in column sponsor: value.replace(/\s+/,' ') ● Text transform on 8535 cells in column sponsor: value.toUppercase() ● Create new column sponsor_facet based on column sponsor by filling 15984 rows with grel:value ● Remove column sponsor ● Rename column sponsor_facet to sponsor
	venue	● Text transform on 0 cells in column venue: value.trim() ● Text transform on 0 cells in column venue: value.replace(/\s+/,' ') ● Text transform on 9 cells in column venue: value.toUppercase() ● Create new column venue_facet based on column venue by filling 8131 rows with grel:value ● Mass edit 2251 cells in column venue_facet ● Remove column venue

		• Rename column venue_facet to venue
	physical_description	• Text transform on 384 cells in column physical_description: value.trim() • Text transform on 38 cells in column physical_description: value.replace(/\s+/,' ') • Text transform on 7371 cells in column physical_description: value.toUppercase() • Create new column physical_description_facet based on column physical_description by filling 14768 rows with grel:value
	notes	• Text transform on 125 cells in column notes: value.trim() • Text transform on 195 cells in column notes: value.replace(/\s+/,' ') • Text transform on 3873 cells in column notes: value.toUppercase() • Create new column notes_facet based on column notes by filling 10613 rows with grel:value
	call_number	• Text transform on 1 cells in column call_number: value.toNumber()
	keywords	• Text transform on 0 cells in column keywords: value.trim() • Text transform on 0 cells in column keywords: value.replace(/\s+/,' ') • Text transform on 0 cells in column keywords: value.toUppercase()
	language	• Text transform on 0 cells in column language: value.trim() • Text transform on 0 cells in column language: value.replace(/\s+/,' ') • Text transform on 0 cells in column language: value.toUppercase()
	date	• Text transform on 16959 cells in column date: value.toDate()
	location	• Text transform on 14 cells in column location: value.trim() • Text transform on 555 cells in column location: value.replace(/\s+/,' ') • Text transform on 17434 cells in column location: value.toUppercase()

		• Create new column location_facet based on column location by filling 17545 rows with grel:value
	location_type	• Remove column location_type
	currency	• Text transform on 0 cells in column currency: value.trim() • Text transform on 0 cells in column currency: value.replace(/\s+/,' ') • Text transform on 6456 cells in column currency: value.toUppercase() • Create new column currency_facet based on column currency by filling 6456 rows with grel:value • Remove column currency • Rename column currency_facet to currency
	currency_symbol	• Text transform on 4 cells in column currency_symbol: value.trim() • Text transform on 0 cells in column currency_symbol: value.replace(/\s+/,' ') • Text transform on 169 cells in column currency_symbol: value.toUppercase()
	status	• Text transform on 0 cells in column status: value.trim() • Text transform on 0 cells in column status: value.replace(/\s+/,' ') • Text transform on 17545 cells in column status: value.toUppercase()
	page_count	• Text transform on 17545 cells in column page_count: value.toNumber()
	dish_count	• Text transform on 17545 cells in column dish_count: value.toNumber()
	occasion	• Text transform on 0 cells in column occasion: value.trim() • Text transform on 3 cells in column occasion: value.replace(/\s+/,' ') • Text transform on 17 cells in column occasion: value.toUppercase() • Create new column occasion_facet based on column occasion by filling 3803 rows with grel:value • Mass edit 2779 cells in column occasion_facet • Mass edit 608 cells in column occasion_facet

		• Mass edit 8 cells in column occasion_facet • Remove column occasion • Rename column occasion_facet to occasion
	event	• Text transform on 3 cells in column event: value.trim() • Text transform on 6 cells in column event: value.replace(/\s+/,' ') • Text transform on 729 cells in column event: value.toUppercase() • Mass edit 5314 cells in column event - Cluster using “Key Collision” • Text transform on 0 cells in column event: value.trim() • Text transform on 0 cells in column event: value.replace(/\s+/,' ') • Create new column event_facet based on column event by 8154 rows with grel:value: grel:value.replace('[',' ')\n.replace(']',' ')\n.replace('{',' ')\n.replace('}',' ')\n.replace('\"',' ')\n.replace(/[,;]$/, ' ')\n.replace('?',' ')\n.replace(/\\'/,' ')\n.replace(')',' ')\n.replace('(',' ')\n.replace('*',' ')\n.replace('\"',' ') • Text transform on 124 cells in column event_facet: value.trim() • Text transform on 89 cells in column event_facet : value.replace(/\s+/,' ') • Text transform on 0 cells in column event: value.toUppercase() • Remove column event • Rename column event_facet to event
Dish.csv	id	Transformed 423397 cells in column id to Number
	menus_appeared	Transformed 423397 cells in column menus_appeared to Number
	times_appeared	Transformed 423397 cells in column times_appeared to Number
	first_appeared	Transformed 423397 cells in column first_appeared to Number
	last_appeared	Transformed 423397 cells in column last_appeared to Number
	lowest_price	Transformed 394297 cells in column lowest_price to Number

	highest_price	Transformed 394297 cells in column highest_price to Number
	name	• Trimmed 9045 cells in column name to remove leading and trailing spaces • Trimmed 6415 cells in column name to remove consecutive white spaces • Transformed 406726 cells in column name to have Upper Case • Created a column facet name_facet and applied RegEx to update 423397 rows with below: • value.replace('[',' ') .replace(']',' ') .replace('{',' ') .replace('}',' ') .replace('"',' ') .replace(/[,;]$/, ' ') .replace('?',' ') .replace(/\'/,' ') .replace('*',' ') .replace('"',' ') • Trimmed 8550 cells in column name to remove leading and trailing spaces • Trimmed 11302 cells in column name to remove consecutive white spaces • Did iterative clustering with name_facet column using "key Collision/fingerprint" to remove the clusters till there were none left. Below are the results in multiple iterations: ○ Mass edit 8284 cells in column name_facet ○ Mass edit 7666 cells in column name_facet ○ Mass edit 7152 cells in column name_facet ○ Mass edit 6848 cells in column name_facet ○ Mass edit 6741 cells in column name_facet ○ Mass edit 6343 cells in column name_facet ○ Mass edit 5959 cells in column name_facet ○ Mass edit 5957 cells in column name_facet ○ Mass edit 5993 cells in column name_facet ○ Mass edit 5984 cells in column name_facet ○ Mass edit 5995 cells in column name_facet ○ Mass edit 5983 cells in column name_facet ○ Mass edit 5671 cells in column name_facet • Removed the original column name • Renamed the column "name_facet" to "name"
MenuItem.csv		• Not much to clean. • Most of the data was clean and columns were mostly numeric. • Text transform on 1332726 cells in column menu_page_id: value.toNumber()

		• Text transform on 886810 cells in column price: value.toNumber() • Text transform on 91905 cells in column high_price: value.toNumber() • Text transform on 1332485 cells in column dish_id: value.toNumber() • Text transform on 0 cells in column created_at: value.toDate() • Text transform on 1332726 cells in column xpos: 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 66936 cells in column uuid: value.toUppercase() • Text transform on 0 cells in column uuid: value.trim() • Text transform on 0 cells in column uuid: value.replace(/\s+/,' ')

Rationale For design

Each cleaning step above played an important role for the U1 use case. Let's revisit the U1

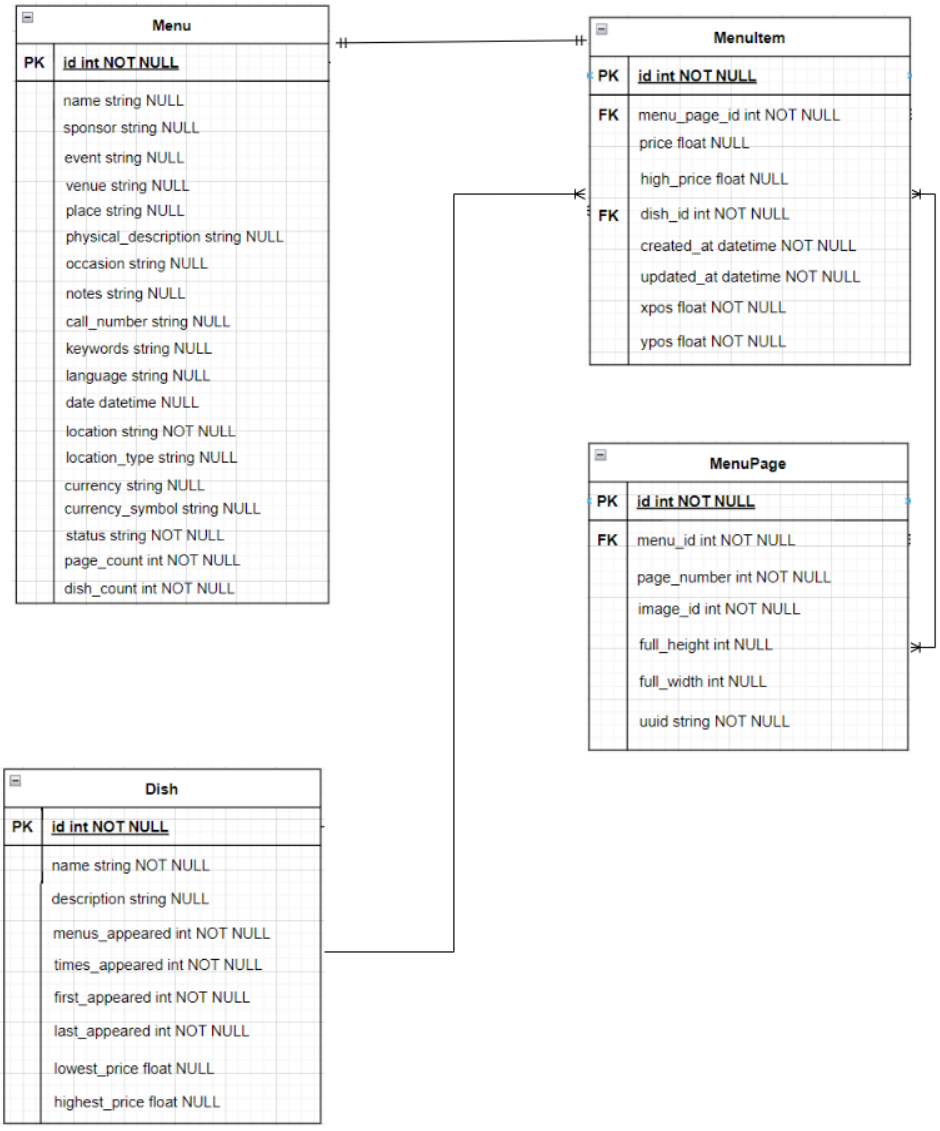
- "*What were the menu items which were served for Dinner at New York between any certain time-frame(ex - 1950 - 1960)? And how menu items were priced and what were the dishes served during that event? What were the locations where these dishes were served?* "

The data cleaning processes applied were relevant because -

1. We used clustering to clean up the name, location, places, events column to serve the purpose of the above question.
2. We removed white spaces to make the data consistent across columns.
3. We removed the unwanted special characters to clean up the text values.
4. We used the data type conversion to convert columns (date) so that date queries can be performed easily.
5. The Sqlite3 database helped us remove the integrity constraints violations because then we were able to join two tables without worrying about missing or unwanted data. For example, we cleaned the Menu_item table by checking if any dish_id exists which are not valid. See here.

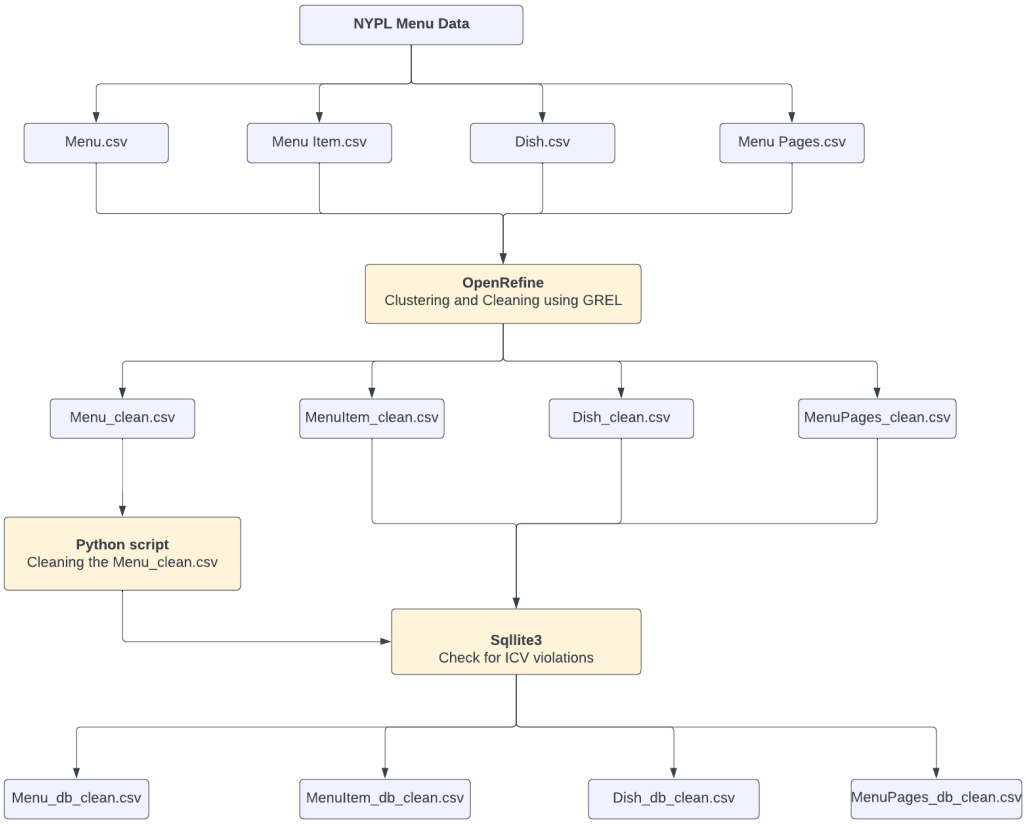
ER Diagram

Here is the ER diagram from the SQLLite3 database. Available under the folder **/sqllit3/Docs & Images/**



Overview Of the cleaning process

Below is a diagram of how the overview of the cleaning process looks like



Workflow Diagrams

Once all the csv files were cleaned we used the YesWorkflow to generate the diagrams.

We have provided the YesWorfklow diagrams for the Openrefine logs and python scripts.

A Python based OR2YWTool application was used to generate the YW and PDF files for the Workflow.

Commands used for YesWorkflow artifact generation:

- set PYTHONUTF8=1 (before each of below step to account for special UTF-8 characters)
- or2yw -i menu.json -o Serial_menu.yw
- or2yw -i menu.json -o Parellel_menu.yw -t parallel
- or2yw -i menu.json -o menu.pdf -ot pdf

We also used LucidChart and Draw.io to generate the above charts.

We will provide the paths of those diagrams belows.

Summary / Lesson Learned

We enjoyed working on this project a lot and data cleaning is fun to be a part of. We ran the query on the dirty dataset and on the cleaned dataset.

Here is the query -

```
SELECT distinct  d.name, m.event, m.date, m.place, m.location, mi.price, mp.image_id
FROM dish d, menu_item mi, menu m, menu_page mp
WHERE d.id = mi.dish_id
AND mi.menu_page_id = mp.id
AND mp.menu_id = m.id
AND m.place = 'NEW YORK, NY'
AND m.date > '1900-01-01'
order by date desc;
```

Here are the results -

Before -We retrieved 28,302 rows from the dataset.

After - Once the cleaning was done, 53,946 rows.

As we cleaned the dataset, all places named “New York, NY “ were cleaned and we replaced different places with this above string.

We also learned a few things along the way -

1. Openrefine gets very slow when we load more than 100K+ records, which is not too many in industry standard. I would prefer to use Pandas at that point. However, we then have to write our own code.
2. Openrefine is good for clustering using different Algorithms. But I prefer using the nearest neighbor algorithm much better.
3. Trying to figure out a common Regex for all columns was very difficult. There were so many unwanted characters in so many different forms. For example, some columns had brackets like this “(...)” and some had “[....]”.
4. There were so many erroneous values while dealing with Date columns, e.g. - Future dates.
5. We learnt that cleaning improves the data quality. This was observed by the different number of rows retrieved for the same query on querying the uncleaned dataset and cleaned dataset.

Acknowledgements

- Our sincere thanks to Prof. Bertram Ludaescher and all the TAs in CS 513 - Theory and Practice Data Cleaning course for making the course engaging and helping us with all the queries.
- Also special thanks to the open source community of Python to show us endless Regex examples.
- Thanks to our project team members for working late night despite being in different Timezones and making this project successful.
- A heartfelt appreciation to University of Illinois - Urbana Champaign for providing students with endless tools to become successful in their courses. Here are some of the tool examples such as box, sharepoint, Google Drive, zoom, slack, CampusWire and many more.

User Stories and Contributions

1. **Epic - Cleaning the files using OpenRefine**
    a. Sanitize Menu.csv - Kavithaa Suresh Kumar
    b. Sanitize Dish.csv - Sudipto Sarkar
    c. Sanitize MenItem.csv - Arnab Kar Sarkar
    d. Sanitize MenuPages.csv - Kavithaa Suresh Kumar

2. **Epic - Cleaning the files using PythonScript**
    a. Clean all text columns using Pandas - Arnab Kar Sarkar

3. **Epic - SQLlite3**
    a. Sanitize Menu.csv - Kavithaa Suresh Kumar
    b. Sanitize Dish.csv - Sudipto Sarkar
    c. Sanitize MenItem.csv - Arnab Kar Sarkar
    d. Sanitize MenuPages.csv - Kavithaa Suresh Kumar

4. **Epic - Documentation and Github setup**
    a. Write different sections - Kavithaa Suresh Kumar , Arnab Kar Sarkar Sudipto Sarkar

5. **Epic - Yes Workflow and Other Diagrams**
    a. OpenRefine - Sudipto Sarkar

b. FlowChart - Arnab Kar Sarkar
c. ER Diagram - Kavithaa Suresh Kumar

Project Team members

Name	NetId	Email
Kavithaa Suresh Kumar	ks64	ks64@illinois.edu
Arnab KarSarkar	arnabk2	arnabk2@illinois.edu
Sudipto Sarkar	sudipto2	sudipto2@illinois.edu

Github Details

Github link - https://github.com/arnabksarkar/CS513-DataCleaning-Project

FilePath Details

- OpenRefine details
  - Logs - **/OpenRefine/Logs/json**
  - YesWorkflow docs - **/OpenRefine/YWDocs**
- Python details
  - Code - **/Python/CodeFiles**
  - YesWorkflow - **/Python/YWDocs**
- SQLite3 details
  - Queries - **/sqllite3/Queries.sql**
  - SQL Workflow - **/sqllite3/Docs & Images**
- Documentations
  - Final Project PDF document - **/Docs/Final Project**
  - First Draft PDF Document - **/Docs/First Draft**