

Points received for initial Phase-1 submission : 44 (out of 100) 2. Our team chooses the following Phase-1 option:

- ☐ (A) No change to Phase-1 report
- ☒ (B) Improved (or extended) Phase-1 report

Phase 1 Change Report

[-8] Conceptual modeling/relational schema of the data set is missing;	See ER Diagram page 6
[-7] No enough narrative on each column in the dataset;	See Data Cleaning Process for each entity, Page 10 - 14
[-10] No distinguishing U1, U0, U2. Questions or queries are missing.	See Use cases on Page 8-9
[-5] The narrative of the target use case (U1) is not sufficient.	See Use cases on Page 8-9
[-10] No answer for two minor use cases.;	See Use cases on Page 8-9
[-2] use bullet points instead of sentences	See Data Cleaning Process for each entity, Page 10-14
[-3] only list quality issues without explanation	See Initial Quality and Problems page 7
[-2] lack description of data quality issues that existed in the dataset (e.g., inconsistency, missing values...)	See Initial Quality and Problems page 7
[-5] did not discuss the connection between data quality issues and use cases (how these data quality issues affect your use case);	See Page 9
[-2] Did not specify methods for checking whether the original dataset has been cleaned for your use case after cleaning (e.g. using integrity constraints).	See Page 15 - 22
[-2] Did not specify how you will document changes made to the original dataset after cleaning (e.g. using a summary table).	See Menu, Dish Cleaning Details page 10-13

CS 513 Theory & Practice of Data Cleaning

Final Project

Phase 2 Report

Contributors:

Roy Alda - ralda2@illinois.edu

Atif Malik - atifqm2@illinois.edu

Dan Crosson - dfc3@illinois.edu

Jason Lundquist - jasondl3@illinois.edu

Abstract	4
Introduction	5
Data Source	6
Data Set Description	6
Initial Quality and Problems	7
Initial Plan	8
Target Use Case	8
Impact of Data Cleaning on U1	9
Technology	9
Data Cleaning Process	10
Menu	10
Sponsor_Clean - Detailed	11
Dish	13
MenuPage	14
MenuItem	14
Integrity Constraint Checks	15
Dish	15
Menu	18
MenuPage	20
MenuItem	21
Results of Data Cleaning for Purposes of U1	23
Summary Table - Quality Changes	23
Summary of Quality - ICV Queries	24
Workflow Models	26
Outer Model	26
Inner Models	26
Menu	26
Dish	27
Analytics	27
Summary / Conclusions	30
Contribution Summary	30

Abstract

Report containing proposed data cleaning to be conducted on New York Public Library's (NYPL) crowd-sourced historical menus dataset with a goal to curate the datasets for data analytics use cases. NYPL Menu data set is refreshed on a monthly basis. NYPL processes the menus using Optical Character Recognition (OCR) on raw images of the menu. The variations in menu graphics results in the majority of the quality issues found in the data. Data provenance best practices are explored and applied to each dataset provided. We compared the cleaned and uncleaned data through regression for our most affected dataset. We demonstrate the impact of applying data cleaning techniques through residuals.

Introduction

This document is the project report for a data cleaning process performed on the New York Public Library's historical menus dataset. The report outlines the proposed data cleaning, the details of the steps to perform and the methods for measuring the data cleaning outcome. The New York Public Library's restaurant menu collection holds data about menus and dishes from 1840 to present. This is a crowdsourced dataset collected through spreadsheets and APIs. Since the data is crowdsourced and collected via various means, the data quality is very poor. The data contains errors due to limitations in quality achieved with Optical Character Recognition software. There are a wide variety of menus with various levels of readability.

The objective of this project is to apply various data cleaning techniques and analyze their effectiveness on the data set. After performing data cleaning, it is expected that the data will be "fit for use" in the data analytics use cases outlined in this report.

NYPL Labs

What's on the menu?

Est. 2011

Menus

Dishes

Data

Blog

Help The New York Public Library improve a unique collection!

We need you!

Help review


It's easy! No registration required

So far: 1,334,792 dishes transcribed from


Connect: menus@nypl.org | [Twitter](#) | [Facebook](#)

The search function is being worked on!


Unfortunately, the search is not working properly on the site, but we are working on it and appreciate your patience!




Anthony's Fish Grotto
184 dishes




Fisherman's Grotto
9 dishes




The Famous Anchor Sea Food House
1962
339 dishes



Anchor Inn Seafood Restaurant
1968
175 dishes



Legal Sea Foods
1998
103 dishes




The Great American Fish Company
1987
99 dishes

Help review

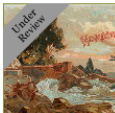
Transcribed menus that need a second pair of eyes. Help fix misspellings, fill in missing data...

Explore


Browse the collection. 17,547 menus digitized and counting...




Sweets Restaurant
1960
290 dishes




Norddeutscher Lloyd Bremen
1901
72 dishes




Claremont Hotel
1900
221 dishes



Norddeutscher Lloyd Bremen
1901
22 dishes



Alamo Hotel(?)
1900
32 dishes



St. Louis Club
1901
11 dishes

Figure 1: New York Public Library supplies a dataset that publishes historic menu information.
Source: <http://menus.nypl.org/> (retrieved 2021-07-31)

5

Data Source

NYPL_DataSet (New York Public Library's crowd-sourced historical menus dataset)

<https://drive.google.com/file/d/1Abm7P1Moe1050BH-R5sCTL2TPU3oC4oB/view?usp=sharing>

Cleaned data is available

<https://drive.google.com/file/d/1WHyf-UHslqPZvmLa1PFT5ApF51PUJTPc/view?usp=sharing>

Data Set Description

A crowd-sourced dataset of restaurant menus over time. The dataset contains 4 csv files: Menu, Dish, MenuPage, and MenuItem. Dish, MenuPage, and MenuItem contain statistics and ancillary information about the menus in the Menu file. The Menu file contains information about all the menus in the dataset including the restaurant name, occasion, meal, location, length, and physical description.

Source: http://curatingmenus.org/data_dictionary/

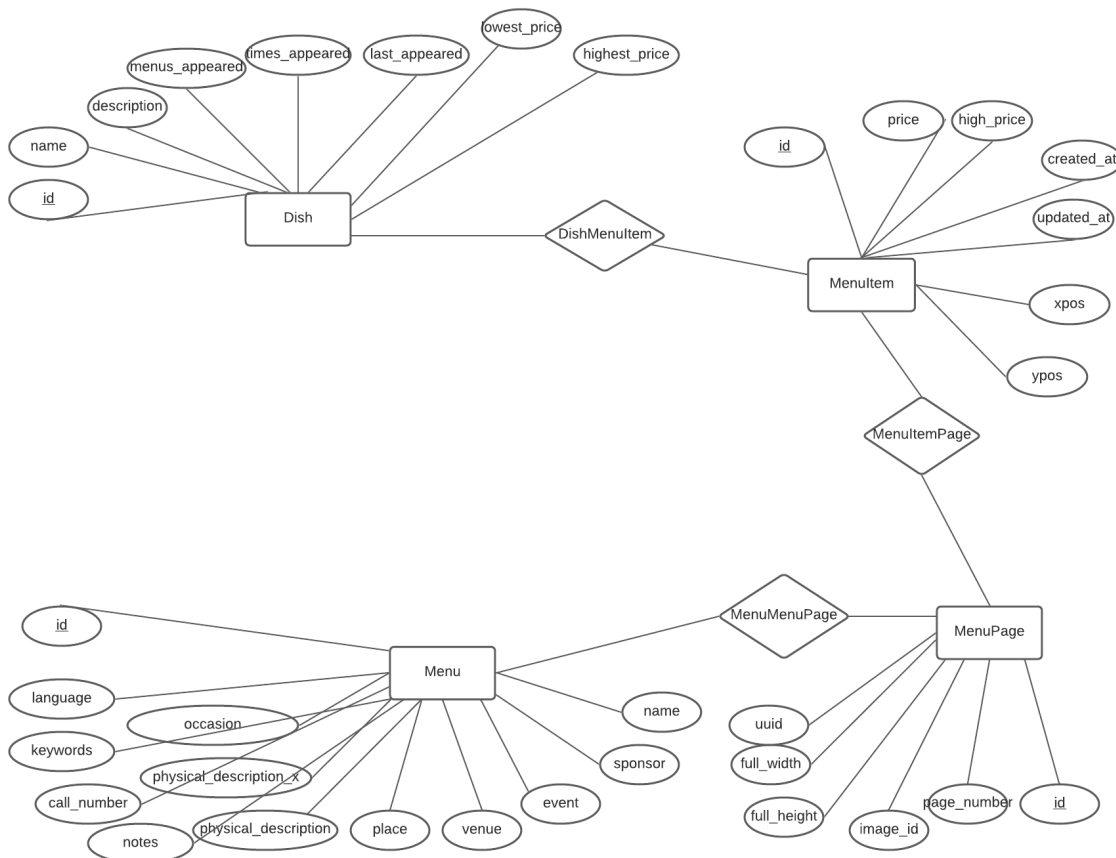


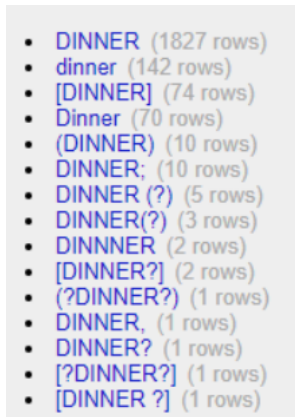
Figure 2: E-R Diagram

Initial Quality and Problems

Focused on the following key areas for data quality assessment:

- Leading/trailing spaces
- Collapse consecutive spaces
- Inconsistent date format for date columns
- Blank values for columns - (Currency, Currency_symbol)
- Inconsistent representation or missing information
- Special Characters like #%\()\[\], Carriage Return and Line Feed, and TAB characters
- Spelling
- Inconsistent value format
- Double and single quote usage
- Integrity key constraint checks between tables.
- Negative or incorrect values for numeric data such as price

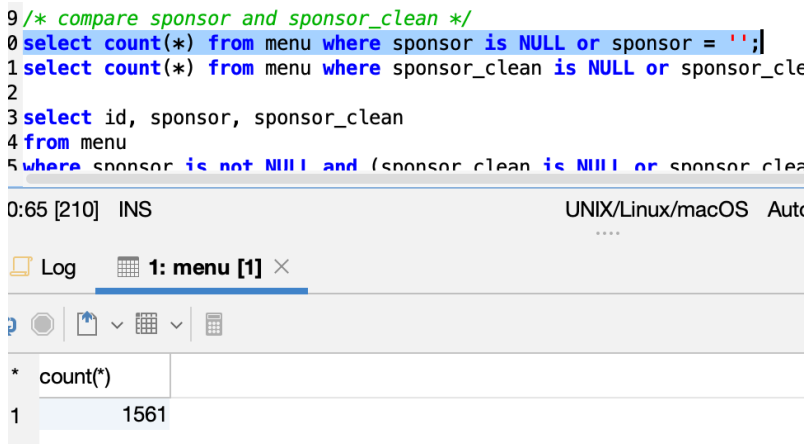
Below is a screenshot illustrating some of the above data quality issues with the event column



- DINNER (1827 rows)
- dinner (142 rows)
- [DINNER] (74 rows)
- Dinner (70 rows)
- (DINNER) (10 rows)
- DINNER; (10 rows)
- DINNER (?) (5 rows)
- DINNER(?) (3 rows)
- DINNNER (2 rows)
- [DINNER?] (2 rows)
- (?DINNER?) (1 rows)
- DINNER, (1 rows)
- DINNER? (1 rows)
- [?DINNER?] (1 rows)
- [DINNER ?] (1 rows)

As this screenshot shows, there are special characters, and inconsistent formats.

Many columns of our dataset had null values to begin with. Below is an example of how many null values were in the sponsor column of the Menu table:



```
9 /* compare sponsor and sponsor_clean */
0 select count(*) from menu where sponsor is NULL or sponsor = '';
1 select count(*) from menu where sponsor_clean is NULL or sponsor_clean = '';
2
3 select id, sponsor, sponsor_clean
4 from menu
5 where sponsor is not NULL and (sponsor_clean is NULL or sponsor_clean = '');
0:65 [210] INS UNIX/Linux/macOS Auto
Log 1: menu [1] x
* count(*)
1 1561
```

As shown by this query, the sponsor column of Menu had over 1500 nulls in its raw format.

Initial Plan

The NYPL Menu data is not "fit for use" for most practical use cases outside of an overall count of menus, or dishes over a given time period. It is also specifically not useful for building regression models , or analytics based on the menus.

Our initial plan is to clean out a number of the fields like venue, location , event, sponsor . And then additionally clean and create a main_ingredient column which would allow us to build better analytics or regression models.

As part of the initial cleaning we will also clean out the price columns . We will then run relational integrity checks on the overall data.

Doing all of this will provide a better dataset for analytics and regression models that we plan.

Target Use Case

Use Case 0: We would like to analyse the data for price fluctuations of dishes based on location, and year.

This use case does not need data cleaning. The Menu Item data has both price , and high price of the Item and also has when that was updated (date in specific format) . Menu Item also has the x , y position for that Menu Item.

We can then query that against which Dish that is referring to.

Given our analysis this initial use case does not require data cleaning.

Use Case 1: We would like to analyse the data of price fluctuations of dishes in reference to the main ingredient (Chicken, Beef, Seafood) , based on location, year, sponsor , venue.

This use-case is not just 1 query but a few different queries/dashboard , or something that can also be eventually built into a regression model.

The model as a whole would be something like this

price ~ main_ingredient, location_cleaned, sponsor_cleaned, venue_cleaned

This use case does require data cleaning . We had to clean and extract the main ingredient , also had to clean the location, year, sponsor , venue fields. Also we have to make sure that values of currency are correct.

Fields used for data cleaning:

Dish: lowest_price

Dish: highest_pric

Menu: sponsor

Menu: venue

Menu: occasion
Menu: currency
Menu: sponsor
Dish: lowest_price
Dish: highest_price
Dish: first_appeared
Dish: last_appeared

Fields cleaned:

Dish: main_ingredient
Dish: lowest_price
Dish: highest_price
Menu: sponsor_cleaned
Menu: venue_cleaned
Menu: occasion_cleaned
Menu: currency
Dish: lowest_price
Dish: highest_price
Dish: first_appeared
Dish: last_appeared

Use Case 2:

Use case 2 is to look at price fluctuations based on ingredients of the dish. For this use-case no amount of data cleaning would be sufficient . Because we have a limited set of ingredients based on the name of the dish . Given that it is hard to project which ingredients have the most impact on price. Also given ingredients are varied in quality it would be difficult to account for that.

Impact of Data Cleaning on U1

Since data in the original Menu data set sponsor, venue, occasion, currency, and sponsor had some odd characters and spaces, we needed to make sure it fits the referential key constraints it would impact the analytics model created in U1.

Once these fields were cleaned, and we further extracted the main_ingredient we should be able to build much better analytics and models for this data set.

Technology

We used OpenRefine to examine and construct recipes to clean our datasets. Then we used Python to load the datasets into a SQLite database. We used the SQLite database to store our dataset and perform integrity checks. Finally, we used YesWorkflow and or2yw to construct workflow diagrams to document what steps we took to clean our dataset.

Data Cleaning Process

Cleaning Steps

The following is a description of operations performed when cleaning the data sets using OpenRefine. Each file of the data set was loaded into OpenRefine, cleaned step by step as outlined in each section; upon completion of the cleaning steps the data was exported and saved to a corresponding CSV as <entity>-Cleaned.csv, along with the supporting provenance in a JSON format.

Menu

Menu is the parent for all the entities within the NYPL dataset; each Menu has a unique identifier and associated data, including data on the venue and or the event the menu was created for. The Menu data also contains the location that the menu was used, the currency indicated for the prices and other descriptive and quantitative fields.

The complete list of attributes and metadata are as follows:

- **Id** - a unique identifier for menu items.
- **Name** - the name of the menu (mostly blank or the name of the restaurant).
- **Sponsor** - the sponsor of this menu (often the name of the restaurant).
- **Event** - the event the menu was created for (the special event that the menu was created for).
- **Venue** - the venue where dishes on this menu were served.
- **Place** - geographical description of the menu location (includes street address, city, state, country or name of venue).
- **Physical_Description** - description of the menu physical characteristics, including paper stock, dimensions, colors, design.
- **Occasion** - as in holiday, daily menu, etc., often left blank.
- **Notes** - notes pertaining to this menu.
- **Call_Number** - the menu call number within the NYPL collection.
- **Keywords** - keyword for the menu (all rows within the collection blank as of most recent publication).
- **Language** - language of the menu (all rows within the collection blank as of most recent publication).
- **Date** - date the menu was added to the collection.
- **Location** - physical location where the dishes were served.
- **Location_type** - type of location (all rows within the collection blank as of most recent publication).
- **Currency** - currency used for pricing (mostly blank).
- **Currency_symbol** - currency symbol for pricing (mostly blank).
- **Status** - the digitization or transcription status of the menu ("complete" or "under review").
- **Page_count** - number of pages in this menu.
- **Dish_count** - number of dishes on this menu.

Sponsor_Clean - Detailed

1. Created Sponsor_clean copying data from Sponsor.
2. Trim leading and trailing white spaces to reduce extraneous characters which can lead to inconsistent values.
3. Collapse consecutive white spaces to reduce extraneous characters which can lead to inconsistent values.
4. Convert column values to upper case to make all values consistent.
5. Remove the characters % # ! / () [] ? \ " ' using GREL to reduce extraneous characters which can lead to inconsistent values.
6. Replace Semicolon (;) to reduce extraneous characters which can lead to inconsistent values.
7. Make a facet and perform the cluster operation using the key-collision method and fingerprint function. Merge the relevant clusters.
8. Repeat last step with n-gram fingerprint, metaphone3, cologne-phonetic methods.
9. Make a facet and perform the cluster operation using the nearest neighbor method and levstein distance function . Merge the relevant clusters.
10. Make a facet and perform the cluster operation using the nearest neighbor method and PPM distance function. Merge the relevant clusters.

Cluster & Edit column "sponsor_clean"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. [Find out more...](#)

Method **nearest neighbor** levenshtein Radius **1.0** Block Chars **6** 10 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value	# Rows in Cluster
2	2	<ul style="list-style-type: none"> Jansen's Restaurant (1 rows) Janssen's Restaurant (1 rows) 	<input type="checkbox"/>	Jansen's Restaurant	2 — 11
2	4	<ul style="list-style-type: none"> RIGGS HOUSE (3 rows) BRIGGS HOUSE (1 rows) 	<input type="checkbox"/>	RIGGS HOUSE	Average Length of Choices
2	4	<ul style="list-style-type: none"> Hotel Lacher (2 rows) Hotel Sacher (2 rows) 	<input type="checkbox"/>	Hotel Lacher	7 — 50
2	4	<ul style="list-style-type: none"> Breakfast Menu, S. S. Vaderland, March 18th, 1910. (2 rows) Breakfast Menu, S. S. Vaderland, March 19th, 1910. (2 rows) 	<input type="checkbox"/>	Breakfast Menu, S. S. Vaderland, M	Length Variance of Choices
2	11	<ul style="list-style-type: none"> The Cortland (7 rows) THE PORTLAND (4 rows) 	<input type="checkbox"/>	The Cortland	0 — 0.5
2	2	<ul style="list-style-type: none"> La Camelia (1 rows) Le Camelia (1 rows) 	<input type="checkbox"/>	La Camelia	
2	2	<ul style="list-style-type: none"> Bellini (1 rows) Collini (1 rows) 	<input type="checkbox"/>	Bellini	

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

11. Correct obvious spelling errors, collapsing into correct neighbors to reduce multiple values for the same sponsor.

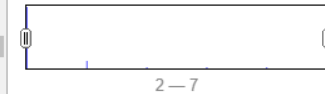
Cluster & Edit column "sponsor"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. [Find out more...](#)

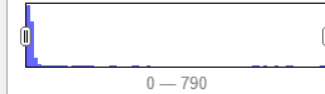
Method **nearest neighbor** Levenshtein Radius **1.0** Block Chars **6** 470 clusters found

3	783	<ul style="list-style-type: none"> Waldorf Astoria (691 rows) Waldorf-Astoria (85 rows) WALDORF-ASTORIA (7 rows) 	<input type="checkbox"/>	Waldorf Astoria
3	695	<ul style="list-style-type: none"> Waldorf Astoria (691 rows) WALDORF ASTORIA (2 rows) WALDORF-ASTORIA (2 rows) 	<input type="checkbox"/>	Waldorf Astoria
3	8	<ul style="list-style-type: none"> FLAT IRON RESTAURANT AND CAFE (3 rows) FLATIRON RESTAURANT AND CAFE (1 rows) FLAT IRON RESTAURANT AND CAFE (2 rows) 	<input type="checkbox"/>	FLAT IRON RESTAURANT AND CA
3	7	<ul style="list-style-type: none"> NIPPON YUSEN KAISHA - S.S.KOBE MARU (4 rows) NIPPON YUSEN KAISHA - S.S.KOBI MARU (2 rows) NIPPON YUSEN KAISHA -S.S.KOBE MARU (1 rows) 	<input type="checkbox"/>	NIPPON YUSEN KAISHA - S.S.KO
3	24	<ul style="list-style-type: none"> HOTEL SAVOY (17 rows) Hotel Savoy (6 rows) Hotel Saboy (1 rows) 	<input type="checkbox"/>	HOTEL SAVOY
3	8	<ul style="list-style-type: none"> RED STAR LINE - ANTWERPEN,NY (5 rows) 	<input type="checkbox"/>	RED STAR LINE - ANTWERPEN,N

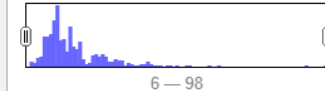
Choices in Cluster



Rows in Cluster



Average Length of Choices



Length Variance of Choices



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

Repeated same steps for **event_clean**, **venue_clean**, **place_clean**, **occasion_clean** and **Location_clean**.

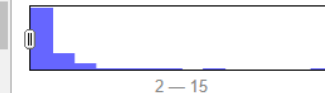
Cluster & Edit column "event_clean"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. [Find out more...](#)

Method **key collision** Keying Function **ngram-fingerprint** Ngram Size **2** 67 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
15	2150	<ul style="list-style-type: none"> DINNER (1827 rows) dinner (142 rows) [DINNER] (74 rows) Dinner (70 rows) (DINNER) (10 rows) DINNER; (10 rows) DINNER (?) (5 rows) DINNER(?) (3 rows) DINNNER (2 rows) [DINNNER?] (2 rows) (?DINNNER?) (1 rows) DINNER, (1 rows) DINNER? (1 rows) [?DINNER?] (1 rows) [DINNER ?] (1 rows) 	<input type="checkbox"/>	DINNER
10	637	<ul style="list-style-type: none"> LUNCH (520 rows) lunch (55 rows) Lunch (33 rows) [LUNCH] (13 rows) (?LUNCH?) (7 rows) LUNCH; (3 rows) [LUNCH?] (3 rows) (LUNCH) (1 rows) LUNCH (?) (1 rows) LUNCH. (1 rows) 	<input type="checkbox"/>	LUNCH

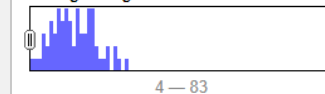
Choices in Cluster



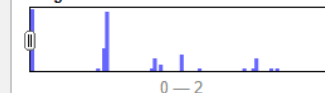
Rows in Cluster



Average Length of Choices



Length Variance of Choices



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

Physical_description

1. Split the column values using semicolon (;) as separator, resulting in seven new columns.
2. Rename the first column as "physical_description_type".

Date

1. Converted the "date" column values to date format "YYYY-MM-dd".

Name_clean, call_number, keywords, language, status

1. Trimmed leading and trailing white spaces.
2. Collapsed consecutive white spaces.

Id, page_count, dish_count

1. Transformed each "To Number".

Dish

Dish is a parent entity that defines a unique dish name, however dishes could appear on multiple menus, ex. Brandy or Mashed Potatoes, etc. So we could not remove rows due to child table dependencies. Dish has a primary key, i.e. id, that can be used to match this table with the other tables, i.e. MenuItem.

The complete list of attributes and metadata are:

- **id** - Primary key for dishes per dish recorded.
- **name** - Name of the dishes as extracted from the historic menus.
- **Description (removed)** - Description was empty so removed from the original set.
- **Main_ingredient (added)** - Added multiple columns to extract the main ingredient from the name of a dish.
- **Menus_appeared** - Total count of menus a particular dish title has appeared in
- **Times_appeared** - Total count of appearances of the dish with this id across all menus.
- **First_appeared** - Year the dish first appears in a menu inclusive from the 17 century to the current year.
- **Last_appeared** - Year the dish last appears in a menu inclusive from the 17 century to the present.
- **Lowest_price** - The lowest price that a dish was offered historically. Some menu items were in other currencies, as well as cents were not differentiated from dollars.
- **Highest_price** - The highest price that a dish is offered historically.

This dataset required processing and cleaning due to Object Character Recognition (OCR) quality issues. Leading and trailing whitespaces were cleared, consecutive whitespaces were removed. Next quotations were removed from the name column. The name column was also changed to title case. There were issues OCR with the years for the first_appeared and last_appeared columns.

Sometimes '1' would be misinterpreted as '2' so years would appear as '2924' instead of '1924'. Using regular expressions leading 2s were replaced in both columns. Next years would not be picked up at all, often provided as '1'. A regular expression representing valid dates was negated to remove erroneous data. All steps are captured in the provenance file related to Dish. Clustering was not pursued as the software was limited due to

the memory limitations for the large data set. An alternative method of extracting the main ingredient was pursued.

MenuPage

MenuPage is a child entity with reference to Menu as a parent (Menu:id).

MenuPage has a unique identifier and primary key (id). Associated MenuPage attributes include the page number of the MenuPage, an identifier for the scanned image of the page, and the dimensions of the page. Each MenuPage is associated with some number of MenuItem values.

The complete list of attributes and metadata are:

- **Id** - The unique identifier of this page in the MenuPage table
- **Menu_id** - The identifier of the menu that this page is associated with. Foreign key
- **Page_number** - The page number in the menu this page is associated with
- **Image_id** - The unique identifier of the image of this page
- **Full_height** - The height of the page
- **Full_width** - The width of the page
- **Uuid** - A unique identifier string

This dataset requires very little cleaning. The only problem with this dataset is that some of the height and width values are null. We do not believe that this information is relevant to our use case, so we believe this data quality issue does not need to be addressed to fit our use case. The only steps in our OpenRefine recipe are converting every column except UUID to numeric.

MenuItem

This data is related to the menu, dish and menu page files and provides the information related to each menu item. For each menu item, it provides the dish id from dish.csv and menu page id from menupage.csv. It also provides other details of menu items such as price, creation type, and the location coordinate of the place.

The complete list of attributes and metadata are:

- **Id** - A unique identifier for each menu item
- **Menu_page_id** - A foreign key to the MenuPage table for the menu page associated with this menu item
- **Price** - The price of the item
- **High_price** - The highest price of this item found in the dataset
- **Dish_id** - A foreign key to the Dish table for the dish associated with this menu item
- **Created_at** - The time this menu item was created
- **Updated_at** - The time this menu item was updated
- **Xpos** - The X coordinate of this menu item on the associated menu page
- **Ypos** - The Y coordinate of this menu item on the associated menu page

This dataset also required very little cleaning. We validated in this the price, high_price was a numeric, and numbers such as Xpos, Ypos were numeric.

Integrity Constraint Checks

Dish

1. Distinct **id** check, evaluating for duplicates.

```
1 --ICV Dish duplicate ID
2
3 select id, count(id) as duplicateCount
4 from dish
5 group by id
6 having count(id) > 1;
```

1:24 [24] INS

Log 1: dish [0] ×

Log 1: dish [0] ×

* id	duplicateCount
------	----------------

2. NULL value evaluation for **Id**.

```
1 --ICV Dish ID NULL
2
3 select *
4 from dish
5 where id is NULL;
```

5:18 [59] INS

Log 1: dish [0] ×

Log 1: dish [0] ×

* id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
------	------	-------------	----------------	----------------	----------------	---------------	--------------	---------------

3. NULL value evaluation for **menus_appeared** and **times_appeared**.

```
1 --ICV Dish menu and times appeared.
2
3 select * from dish
4 where menus_appeared is NULL or times_appeared is NULL ;
```

4:57 [113] INS

Log 1: dish [0] ×

* id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
------	------	-------------	----------------	----------------	----------------	---------------	--------------	---------------

4. Year value validation for **first_appeared** and **last_appeared**, where **first_appeared** should not be greater than **last_appeared**.

```
1 --ICV Dish evaluation of first_appeared occurring before last_appeared.
2 select * from dish
3 where cast(first_appeared as year) > cast(last_appeared as year)
4 and last_appeared <> 0;
```

1:1 [1] INS

Log 1: dish [0] ×

* id	name	description	menus_appeared	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
------	------	-------------	----------------	----------------	----------------	---------------	--------------	---------------

5. **First_appeared** and **last_appeared** should have values between 1851 and 2021. The following query was used to measure for outliers:

```
--ICV Dish evaluation for first_appeared and last_appeared between 1851 and 2021
select count(*) from dish as total_rows;
select count(*) from dish where (first_appeared between '1851' and '2021');
select count(*) from dish where (last_appeared between '1851' and '2021');
```

Total rows were greater than the count for first_appeared and last_appeared evaluation. The rows that violated this constraint were filtered out of the data set.

6. **Lowest_price** should be less than or equal to **highest_price**.

```
1 --ICV Dish evaluation, lowest_price should be less than or equal to highest_price.
2 select count(*) from dish where lowest_price > highest_price;
```

2:62 [145] INS

Log 1: dish [1] X

	* count(*)
1	0

7. Violation of the relationship between Dish and MenuItem

```
62 SELECT count(Dish.id) as MenuItem_Dish_violations
63 FROM Dish
64 LEFT JOIN MenuItem
65 ON MenuItem.dish_id = Dish.id
66 WHERE MenuItem.dish_id is NULL;
67
```

35:30 [2136] INS

Log 1: Dish [1] X

	* MenuItem_Dish_violations
1	12754

Menu

1. Unique **Id**, and non NULL **Id** evaluation with the following query and output.

```
1 --ICV Menu duplicate ID
2 select id, count(id) as duplicateCount
3 from menu
4 group by id
5 having count(id) > 1;
6
7
```

5:23 [108] INS

Log 1: menu [0] X

* id	duplicateCount
------	----------------

```
1 --ICV Menu NULL ID evaluation
2 select count(*)
3 from menu
4 where id is NULL;
5
6
```

4:18 [76] INS

Log 1: menu [1] X

* count(*)
1 0

2. The following query was used to evaluate **sponsor_clean** for NULL and blank values:


select * from menu where sponsor_clean is NULL or ";

1618 rows violated this constraint and were omitted from the final dataset.

3. The following query was used to evaluate **page_count**:

```
1 --ICV page_count NULL or Blank
2 select count(*) from menu where page_count is NULL or '';
3
4
```

1:1 [1] INS

 Log  1: menu [1] ×



	* count(*)
1	0

MenuPage

1. Unique **Id**, and non NULL **Id** evaluation with the following query and output.

```
1 --ICV MenuPage duplicate ID
2 select id, count(id) as duplicateCount
3 from MenuPage
4 group by id
5 having count(id) > 1;
6
```

5:23 [116] INS

Log 1: MenuPage [0] X



* id	duplicateCount
------	----------------

```
1 --ICV MenuPage NULL ID
2 select count(*)
3 from MenuPage
4 where id is NULL;
5
```

5:1 [74] INS

Log 1: MenuPage [1] X



* count(*)
1 0

2. The following query was used to measure for valid **page_number**:

--ICV MenuPage page_number 0 and NULL check

```
select count(*)
from MenuPage
where page_number is NULL
or page_number = 0;
```

There are 1202 MenuPage records with this constraint violation.

There are over 40,000 MenuPages that do not show up in the MenuItem table. This is not an integrity constraint violation because it is possible for menu pages to not contain menu items. Below is a screenshot demonstrating the presence of these 40,000+ items.

```
SELECT count(MenuItem.id) as MenuItem_MenuPage_violations
FROM MenuPage
LEFT JOIN MenuItem
ON MenuItem.menu_page_id = MenuPage.id
WHERE MenuItem.menu_page_id is NULL;
```

39 [2157] INS

Log 1: MenuPage [1] ×

MenuItem_MenuPage_violations

40329

MenuItem

1. Unique **Id**, and non NULL **Id** evaluation with the following query and output.

```
1 --ICV MenuItem duplicate ID
2 select id, count(id) as duplicateCount
3 from MenuItem
4 group by id
5 having count(id) > 1;
```

5:23 [116] INS

Log 1: MenuItem [0] ×

* id	duplicateCount
------	----------------

```

1 --ICV MenuItem NULL ID
2 select count(*)
3 from MenuItem
4 where id is NULL;

```

4:18 [72] INS

Log 1: MenuItem [1] ×

	* count(*)
1	0

2. Values for columns **xpos** and **ypos** should be between 0 and 1, shown if in figure below:

```

1 --ICV MenuItem NULL ID
2 select count(*) as xpos_violation
3 from MenuItem
4 where xpos < 0 and xpos > 1;
5
6
7 --ICV MenuItem NULL ID
8 select count(*) as ypos_violation
9 from MenuItem
10 where ypos < 0 and ypos > 1;

```

10:29 [204] INS

Log 1: MenuItem [1] × 2: MenuItem [1] ×

	* xpos_violation
1	0

Both columns passed validation checks.

Results of Data Cleaning for Purposes of U1

Summary Table - Quality Changes

Column Name	Number of Cells Transformed	Query
Menu.Location	4078	<pre>select count(*) as location_cells_transformed from menu where location <> location_clean;</pre>
Menu.Event	1134	<pre>select count(*) as event_cells_transformed from menu where event <> event_clean;</pre>
Menu.Venue	1927	<pre>select count(*) as venue_cells_transformed from menu where venue <> venue_clean;</pre> <p>Was cleaned as the initial plan but was not used for the initial analytics built in the report.</p>
Menu.Occasion	3173	<pre>select count(*) as occasion_cells_transformed from menu where occasion <> occasion_clean;</pre> <p>Was cleaned as the initial plan but was not used for the initial analytics built in the report.</p>

Summary of Quality - ICV Queries

Menu.Location

This query evaluated to 6280:

```
select count(distinct location) as location_pre_clean  
from menu;
```

This query evaluated to 5628:

```
select count(distinct location_clean) as location_post_clean  
from menu;
```

The cleaning resulted in the number of unique location names decreasing by: 652.

Menu.Event

This query evaluated to 1769:

```
select count(distinct event) as event_pre_clean  
from menu;
```

This query evaluated to 1625:

```
select count(distinct event_clean) as event_post_clean  
from menu;
```

The cleaning resulted in the number of unique Event names decreasing by: 144.

Menu.Venue

This query evaluated to 233:

```
select count(distinct Venue) as Venue_pre_clean  
from menu;
```

This query evaluated to 129:

```
select count(distinct Venue_clean) as Venue_post_clean  
from menu;
```

The cleaning resulted in the number of unique Event names decreasing by: 104.

Menu.Occasion

This query evaluated to 423:

```
select count(distinct Occasion) as Occasion_pre_clean  
from menu;
```


This query evaluated to 197:

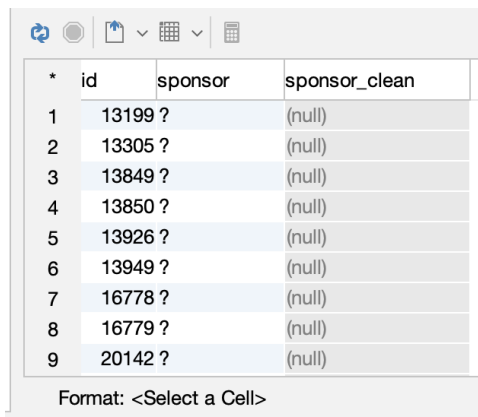
```
select count(distinct Occasion_clean) as Occasion_post_clean
from menu;
```

The cleaning resulted in the number of unique Event names decreasing by: 226.

As a result of our cleaning, the needed columns for U1 in Menu are much more useful than they were beforehand. Below are screenshots of queries illustrating that our cleaning resulted in better quality data.

The result set below is from the following query:

```
select id, sponsor, sponsor_clean
from menu
where sponsor is not NULL and (sponsor_clean is NULL or sponsor_clean = "");
```



*	id	sponsor	sponsor_clean
1	13199 ?		(null)
2	13305 ?		(null)
3	13849 ?		(null)
4	13850 ?		(null)
5	13926 ?		(null)
6	13949 ?		(null)
7	16778 ?		(null)
8	16779 ?		(null)
9	20142 ?		(null)

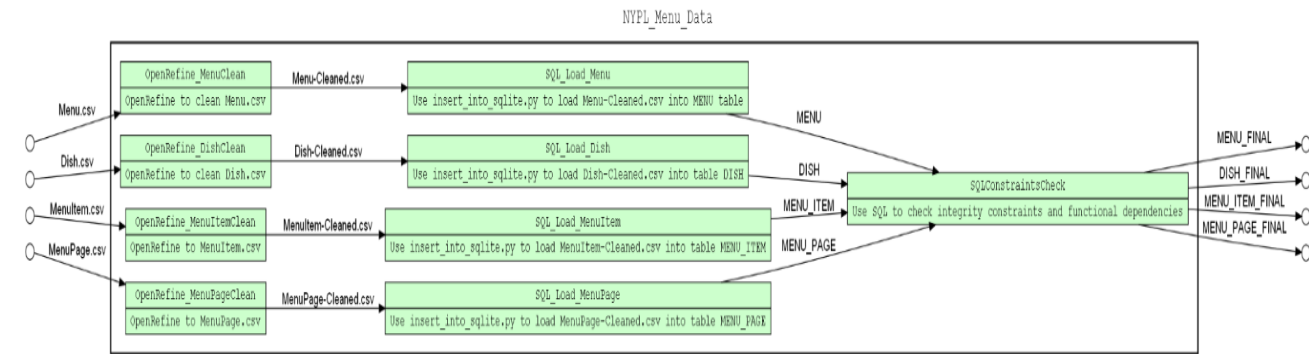
Format: <Select a Cell>

As shown by these results, the sponsor column contained many rows with "?", which is useless information and should be converted to NULL. This increased the number of rows that are null, but improved the quality of the information in the sponsor column by making it clear which rows contained useful data and which rows did not. We performed similar operations on the event, place, physical_description, occasion, and location columns and got similar results. By making these "?" cells null, we enhanced the quality of the information by making sure that any values indicating no information contained the same value.

Workflow Models

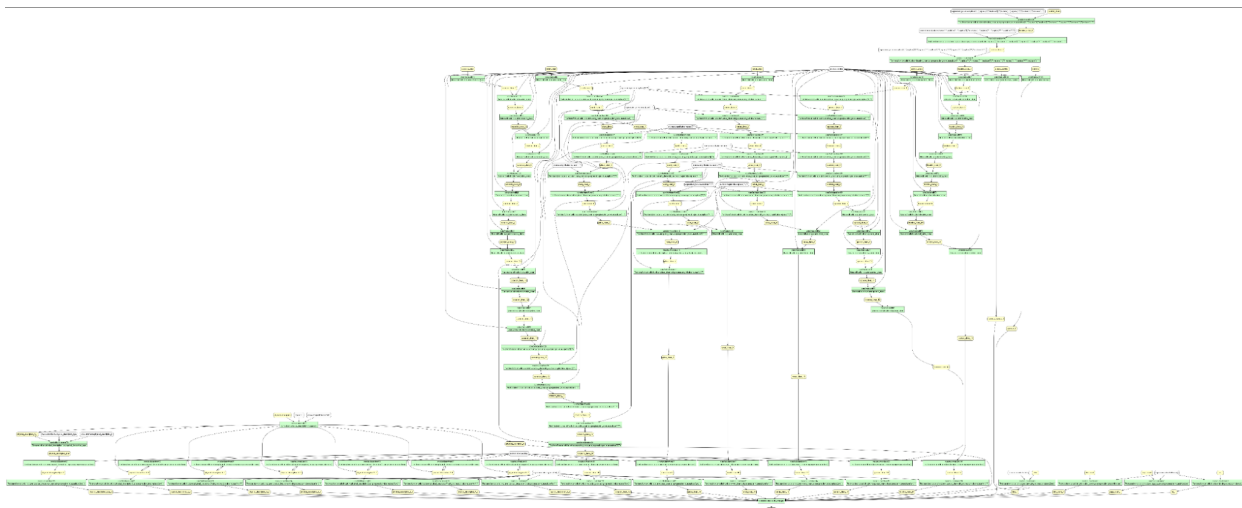
We used YesWorkflow and or2yw to create the workflow graphs to document the provenance for the cleaning process. The scripts and output files have been added to the packaged files for assignment submission.

Outer Model

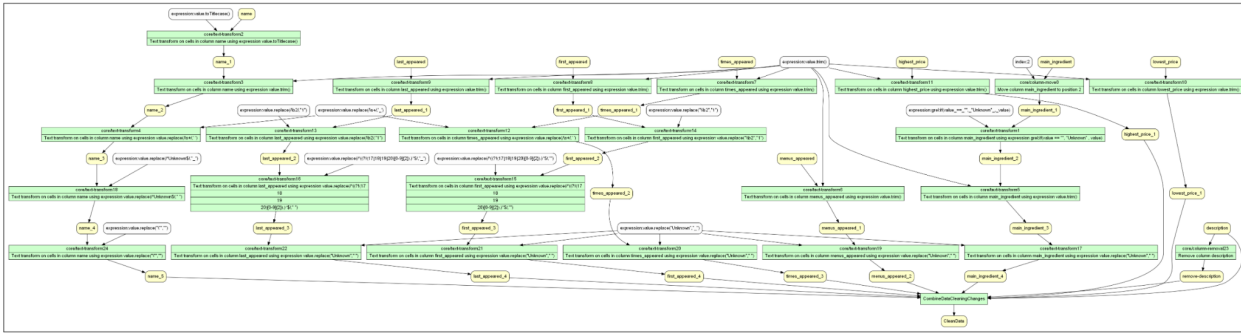


Inner Models

Menu



Dish



Analytics

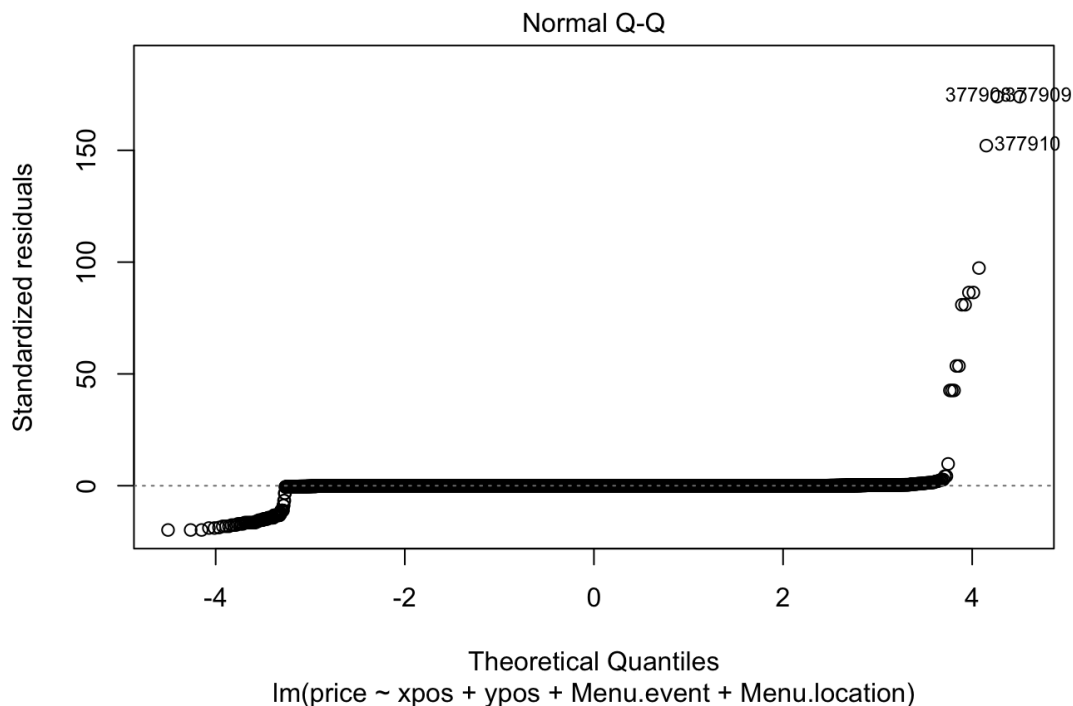
We built an initial model using the uncleaned data in R.

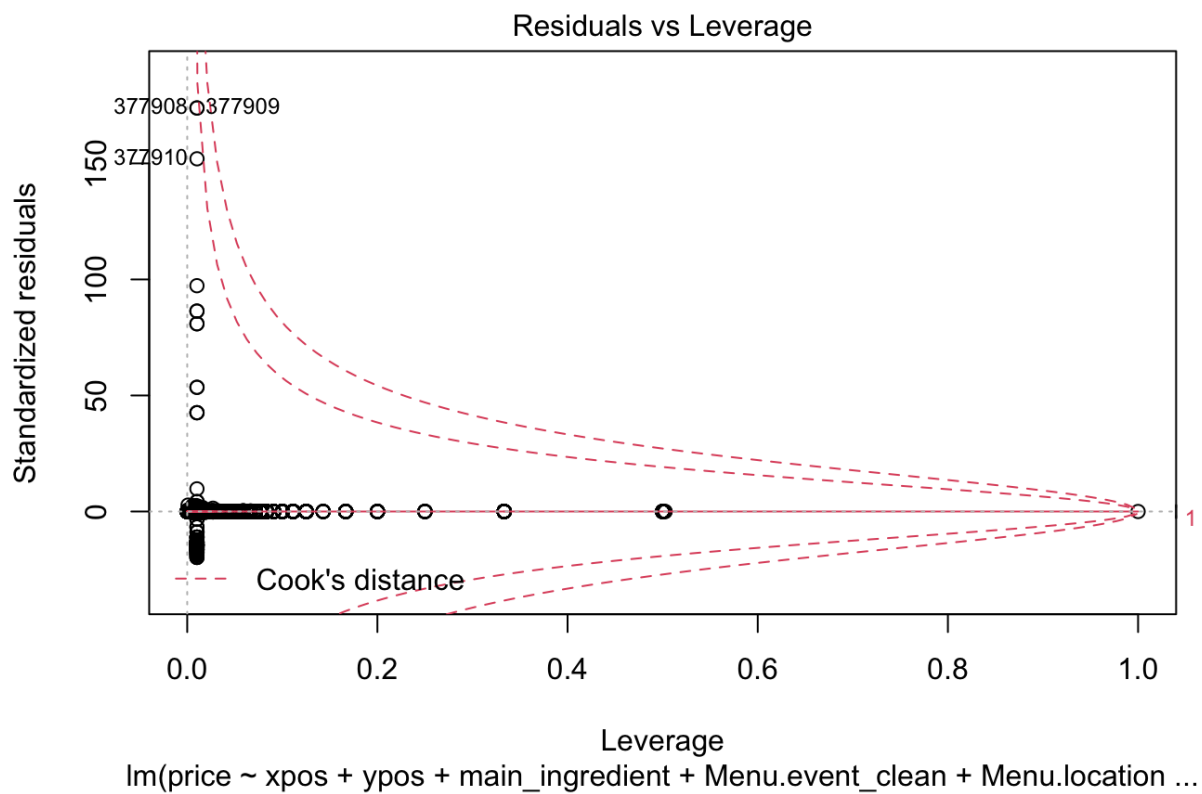
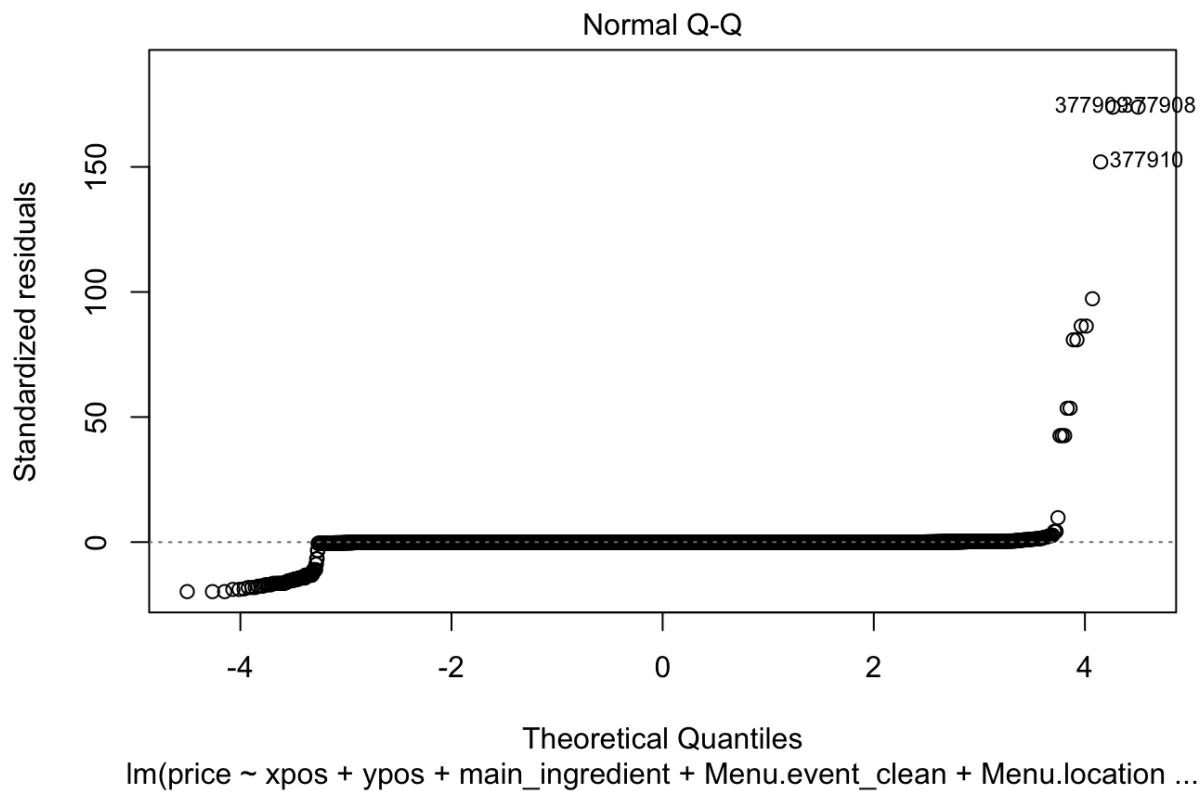
```
```{r}
```

```
nypl_model_unclean = lm(price ~ xpos + ypos + Menu.event + Menu.location , data = nypl_menu)
summary(nypl_model_unclean)
```
```

The output for that was

```
Residual standard error: 918 on 149067 degrees of freedom
(258248 observations deleted due to missingness)
Multiple R-squared: 0.2579, Adjusted R-squared: 0.2539
F-statistic: 64.27 on 806 and 149067 DF, p-value: < 2.2e-16
```





We can see the adjusted R-Squared has increased just slightly given the overall work done in cleaning the data and adding an attribute for main_ingredient.

Summary / Conclusions

We chose the NYPL Menu dataset, which contained extremely messy data, particularly in the Menu table. The dirty data prevented us from performing good analysis on the dataset.

We cleaned the dataset extensively to remove whitespaces, confusing characters, formatting errors, ambiguous values, misspellings. In addition, we were able to extract the main ingredients of various dishes. The steps we took to clean our dataset enhanced the information contained in the dataset and allowed us to make a better analysis of the dataset for our use cases. The overall cleaning resulted in a slightly better regression model for price but we think that from this we can iterate to other models.

Data provenance differs between data analytics and statistical inference, ex. ML/AI. The data provenance used for our project is unstructured data. For ML/AI, our data provenance would require further processing to structure the data. For example, the tables would need to be reorganized chronologically for time related inference, etc.

Contribution Summary

Jason Lundquist: Menu.csv cleaning using OpenRefine, ICV queries for Menu, Dish, MenuItem, and MenuPage using SQLite, Outer and Inner Models using YesWorkflow and or2yw.

Atif Malik: MenuItem.csv cleaning using OpenRefine, ICV queries for Menu, Dish, MenuItem, and MenuPage. Worked on providing main_ingredient in Dish. And then building regression model for sample analytics.

Dan Crosson: MenuPage.csv cleaning, SQL profiling and ICV queries, loading data into SQLite database, data cleaning enumeration in report, summary and conclusions, technology summary, and report editing.

Roy Alda: Dish.csv, cleaning, formatting document, and research into data provenance differences between statistical inference and analytics. Assisted with GV file generation for data provenance. Reviewed and contributed content to various headings in the above report as required.