CS513: Theory & Practice of Data Cleaning

Final Project Phase 1 Report

Team 59: Data Mavericks

Avinash Baldeo, Ashley George, Sotheara Chea

abaldeo2@illinois.edu, ageorge8@illinois.edu, chea3@illinois.edu

Table of Contents

1. Dataset Overview...3
2. Dataset Description...3
2.1 Full Data Narrative...3
2.2 Database Diagram & Schema...5
3. Use cases...10
3.1 U0: Zero Cleaning Use Case...10
3.2 U1: Main Use Case...11
3.3 U2: Never Enough Use Case...11
4. Data Quality Problems...11
4.1 List obvious data quality problems...11
4.2 Why data cleaning is necessary for main use case U1...18
5. Phase-II Initial Plan...19
5.1 Data Cleaning Plan...19
5.2 Who in the team will be responsible for which steps and Timeline...19

1. Dataset Overview

For this project our group is using the Chicago food inspection dataset which is originally released on Kaggle by the City of Chicago: https://www.kaggle.com/datasets/chicago/chicago-food-inspections

The Chicago Department of Public Health's dataset contains information from restaurant inspections since January 1, 2010. As per the description given, the inspections are standardized and conducted by the Food Protection Program staff. The results get input into a database, reviewed, and approved by a Licensed Environmental Health Practitioner. The dataset provided includes a subset of the data elements extracted from the database. A disclaimer is given that the dataset on food inspections may contain duplicates, making it a suitable choice for data cleaning.

2. Dataset Description

2.1 Full Data Narrative

Food establishments undergo annual and complaint-based inspections for compliance with City ordinances. The food inspections ensure food safety in licensed establishments such as restaurants, grocery stores, and bakeries.

The Chicago Department of Public Health (CDPH) conducts these science-based inspections of food establishments, promoting food safety, sanitation, and preventing food-borne illnesses. The inspections cover food handling, temperatures, hygiene, facility maintenance, and pest control.

While Inspections for sanitation are done by the Health Department, inspections are also conducted by the Buildings Department for structural safety and Fire Department for fire exits. The City's Dumpster Task Force also checks for trash disposal compliance with sanitation regulations.

The dataset provided is maintained using Socrata's API and Kaggle's API (Application Programming Interfaces), and the data source is the City of Chicago Data Portal https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5

Uncompressed, the dataset size is 176 MB. In total there are 17 columns and 153,810 records with inspection dates ranging from 01/04/2010 to 08/28/2017.

The table below gives a brief description of each column available in the food_inspection.csv file.

Column Name	**Column Type**	**Description**

Inspection ID	integer	A unique number identifying the inspection occurrence
DBA Name	string	Stands for "Doing Business", it is the legal name of the registered food establishment.
AKA Name	string	Stands for "Also Known As", it is the publicly known name of the food establishment.
License #	integer	A unique license number assigned to the establishment for legal purposes by Department of Business Affairs and Consumer Protection
Facility Type	string	Describes the type/category of the establishment such as a bakery, restaurant, grocery store, etc.
Risk	string	The establishments' risk level of adversely affecting public health (1 being the highest and 3 the lowest risk). Higher risk is inspected more frequently.
Address	string	The full street address of the establishment.
City	string	The city where the establishment is located.
Zip	integer	The zip code associated with the address.
Inspection Date	string	The date when the food inspection occurred.
Inspection Type	string	The type of inspection performed such canvass consultation, complaint, etc.
Results	string	Indicates whether the inspection passed, passed with conditions, or failed.

Violations	string	List of distinct health violations (46 distinct types) with descriptions, found during the inspection
Latitude	float	The GPS latitude coordinate of the establishment location
Longitude	float	The GPS longitude coordinate of the establishment location
Location	string	The GPS point coordinate (latitude, longitude) of the establishment location

Table 2.1 – Food Inspection Dataset Description

2.2 Database Diagram & Schema

The following database diagram(s) represent a better designed & normalized view of the dataset with foreign key constraints enforcing the referential integrity and maintaining the relationships of the original data.

The database consists of the following 5 tables:

FoodEstablishment stores the key establishment information such as license, business name and type of facility.

EstablishmentLocation stores the address & location information of the food establishments.

FoodInspection holds food inspection events including type, date, and result.

InspectionViolation maps the inspection events to the list of violations (if multiple) received along with the health inspectors' comment about each violation.

ViolationCode is the master table that contains the unique list of health inspection violation codes and the descriptions.

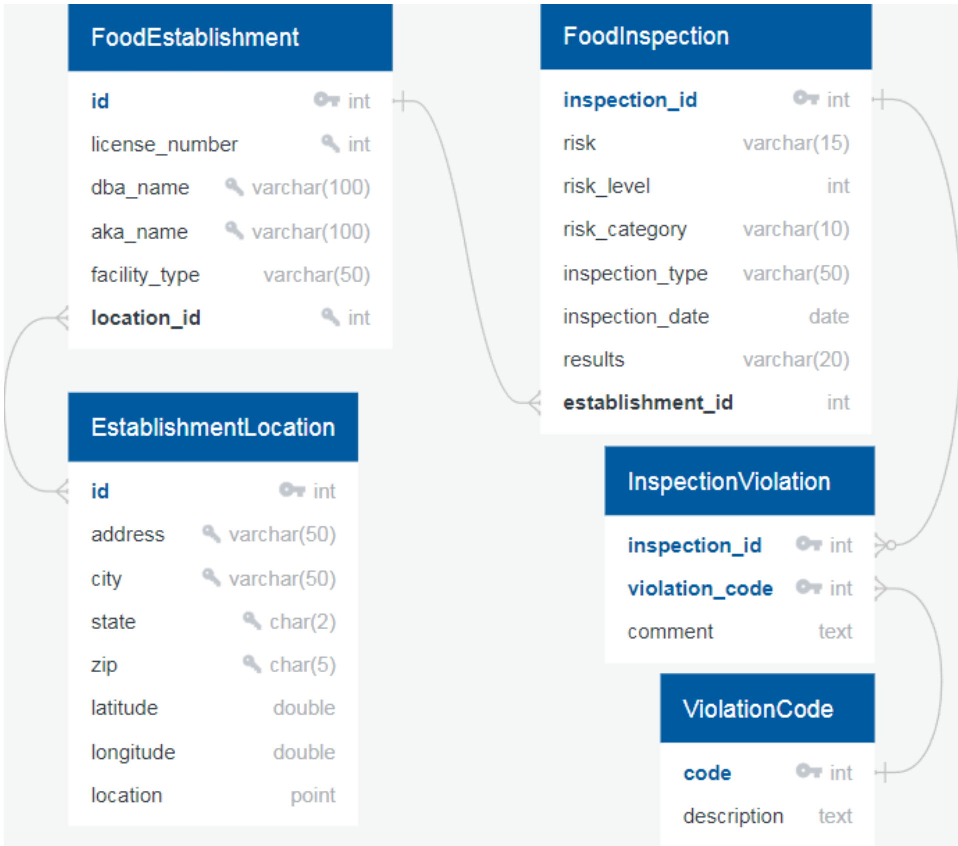

Figure 2A – Entity-Relationship (ER) Diagram of the normalized data

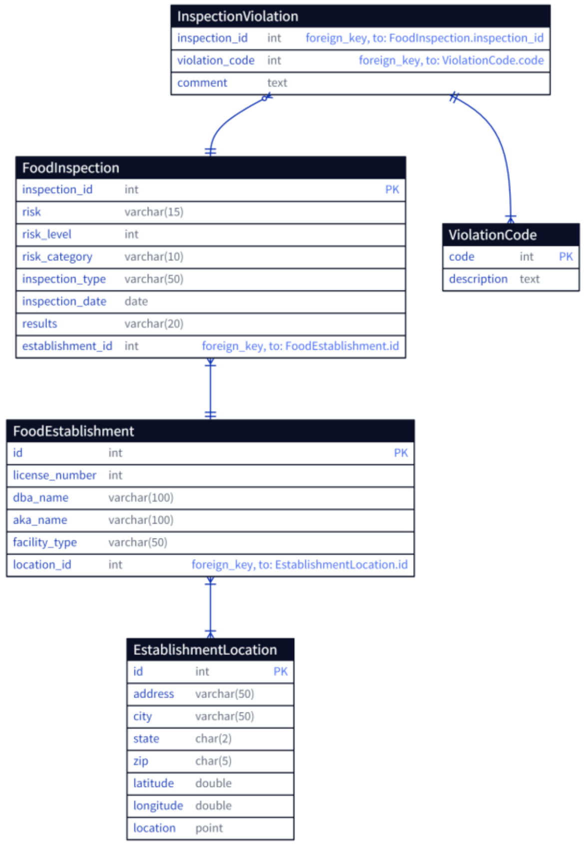

Figure 2B – Entity-Relationship (ER) Diagram with foreign keys

```
CREATE TABLE EstablishmentLocation (
       id INTEGER PRIMARY KEY AUTOINCREMENT,
       address VARCHAR(50) NOT NULL,
       city VARCHAR(50),
       state CHAR(2),
       zip CHAR(5),
       latitude DOUBLE PRECISION,
       longitude DOUBLE PRECISION,
       location POINT
       );
```

Notes on EstablishmentLocation:

- id is the autoincremented primary key.
- address is always present in the dataset given.
- A unique key constraint will be created on address, city, state, and zip.
- Location (latitude, long) is redundant but kept for geospatial queries.
- There can be multiple food establishments at the same location.

```sql
CREATE TABLE FoodEstablishment (
	id INTEGER PRIMARY KEY AUTOINCREMENT,
	license_number INTEGER,
	dba_name VARCHAR(100) NOT NULL,
	aka_name VARCHAR(100),
	facility_type VARCHAR(50),
	location_id INTEGER
	);
```

Notes on FoodEstablishment:

- location_id is foreign key to EstablishmentLocation table.
- id is the autoincremented primary key.
- dba_name is always present in the dataset given.
- A unique key constraint will be created on license_number, dba_name, aka_name, and location_Id after data_cleaning.
- Food establishments such as franchises can have multiple different locations.

```sql
CREATE TABLE FoodInspection (
	inspection_id INTEGER PRIMARY KEY,
	risk VARCHAR(15),
	risk_level INTEGER,
	risk_category VARCHAR(10),
	inspection_type VARCHAR(50),
	inspection_date DATE NOT NULL,
	results VARCHAR(20) NOT NULL,
	establishment_id INTEGER
	);
```

Notes on FoodInspection:

- inspection_id is NOT autogenerated, it is the primary key that identifies the food inspection event and comes directly from Inspection ID column.
- risk_level is the numeric value assigned to the establishment's health risk (1, 2, 3). It is parsed from the Risk column.
- risk_category is the nominal value of risk assigned to establishments health risk (low, medium, high). It is parsed from the original dataset Risk column.
- inpsection_date and results are always present in the dataset given.

```sql
CREATE TABLE ViolationCode (
	code INTEGER PRIMARY KEY,
	description TEXT NOT NULL
	);
```

Notes on ViolationCode:

- code is NOT autogenerated, it is the primary key that identifies the food violation. It is parsed from the Violation column in the dataset.
- description is parsed from the violation code. Each code description is present in the given dataset

```
CREATE TABLE InspectionViolation (
	inspection_id INTEGER,
	violation_code INTEGER,
	comment TEXT,
	PRIMARY KEY (inspection_id,violation_code),
	FOREIGN KEY (inspection_id) REFERENCES FoodInspection(inspection_id),
	FOREIGN KEY (violation_code) REFERENCES ViolationCode(code)
	);
```

The following database indices are created to help with analytical queries and joins between the related tables.

```
CREATE INDEX idx_location ON EstablishmentLocation (location);

CREATE INDEX idx_foodestablishment__location_id ON FoodEstablishment
(location_id);

CREATE INDEX idx_facility_type ON FoodEstablishment (facility_type);

CREATE INDEX idx_foodinspection__establishment_id ON FoodInspection
(establishment_id);

CREATE INDEX idx_risk_category ON FoodInspection (risk_category);

CREATE INDEX idx_inspection_date ON FoodInspection (inspection_date);

CREATE INDEX idx_inspection_type ON FoodInspection (inspection_type);

CREATE INDEX idx_results ON FoodInspection (results);
```

Note, these unique index constraints will be added to the schemas in Phase 2 after cleaning, otherwise it will prevent loading the current dataset into the database.

```
CREATE UNIQUE INDEX idx_uniq_location ON EstablishmentLocation (
	address,city,state,zip
	);


CREATE UNIQUE INDEX idx_uniq_establishment ON FoodEstablishment (
	license_number,dba_name,aka_name,location_id
	);
```

3. Use cases

3.1 U0: Zero Cleaning Use Case

Upon initial the data inspection, it is apparent that the dataset as-is can provide answers to questions about trends in inspection results over time. Specifically, we can check for the following:

- Visualize the count of the results of inspections over a given timeframe (by year/month) as Line/Bar Chart.
- Create a Bar chart showing the count of each inspection result type.
- Create a Bar chart showing the count of the number of inspections by date.

Each record in the dataset represents an inspection event, as denoted by the Inspection ID field, which is unique and has no missing values. The dataset also contains both the Inspection Date for each inspection and the Results of the inspections. Both fields contain no missing values, and the inspection date follows a consistent date format dd/mm/yyyy which can easily be converted from string to date without any data cleaning.

Column Name	Null Values
Inspection ID	0
DBA Name	0
AKA Name	2543
License #	15
Facility Type	4560
Risk	66
Address	0
City	159
State	8
Zip	98
Inspection Date	0
Inspection Type	1
Results	0

Figure 3.1a - U0 Columns

By using these fields to create these type visualizations, we should be able to see if there are there any seasonal patterns to inspection result outcomes (i.e., more failures during colder months vs. warmer months)

3.2 U1: Main Use Case

Consumers choose food establishments based on several factors. Food safety is one of, if not the most important factor for them. With this dataset on Chicago food inspections, for our main use we can build a visualization dashboard (using either Python or Tableau) to provide insights on the safety of Chicago food establishments based on the history of inspection violations, the frequency of each violation type, and the risk level for consumers. Specifically, we would hope to answer the following questions by performing this data cleaning and visualization:

- Is there a correlation between the type of facility and the risk level assessed from inspections?
- How do inspection results (pass, pass with conditions, fail, etc.) vary among different types of food establishment?
- What are the most common violation codes received based on type of inspection & type of facility?
- What is the distribution of inspection results by geographic locations (i.e., which areas have the highest number of failed inspections?)
- What is the distribution of risk severity by geographic locations?

The list of all the violations cited is currently included in the same column for each inspection. The description of each violation is duplicated as there are only 46 distinct violations. Also, comments providing the details of each violation are bundled together in the same column.

3.3 U2: Never Enough Use Case

Even with the data cleaning performed on this dataset, we still are unable to provide general food establishment recommendations to consumers as the data available is only a view of food safety, which is just one aspect in determining where to eat. The dataset currently does not incorporate customer reviews on food taste, indicate the type of cuisines available, or include indicators on the food prices. We would need to build a separate workflow to pull such information via external APIs such as Yelp, Foursquare, and Google and then we would have to combine it with this food inspection dataset to create a proper recommendation system.

4. Data Quality Problems

4.1 List obvious data quality problems

1. Missing Values – there are values missing in some of the columns in the dataset such as Violations, Facility Type, AKA Name, and Longitude and Latitude. The most is the violations column with 30798 nulls.

a. For the main use case, these inspections with null in the violation column may need to be filtered out if they truly represent that the inspection resulted in no violations.
b. 4560 missing values in Facility Type column may or may not be an issue depending on the type of visualization being shown on the dashboard, but likely they will be filtered out.
c. AKA Name has 2543 missing values. Since we are planning on using AKA Name as part of a composite key to identify establishments, we shall coalesce these with DBA Name to deal with the nulls in this column.
d. Latitude and Longitude are missing for 548 address records. Depending on if these are mapped to valid businesses, we can try to backfill them (using address and zip) to increase the coverage for our geographic visualizations, otherwise these records may need to be dropped.
e. For 3 records the address is blank (empty string) which can be dropped.

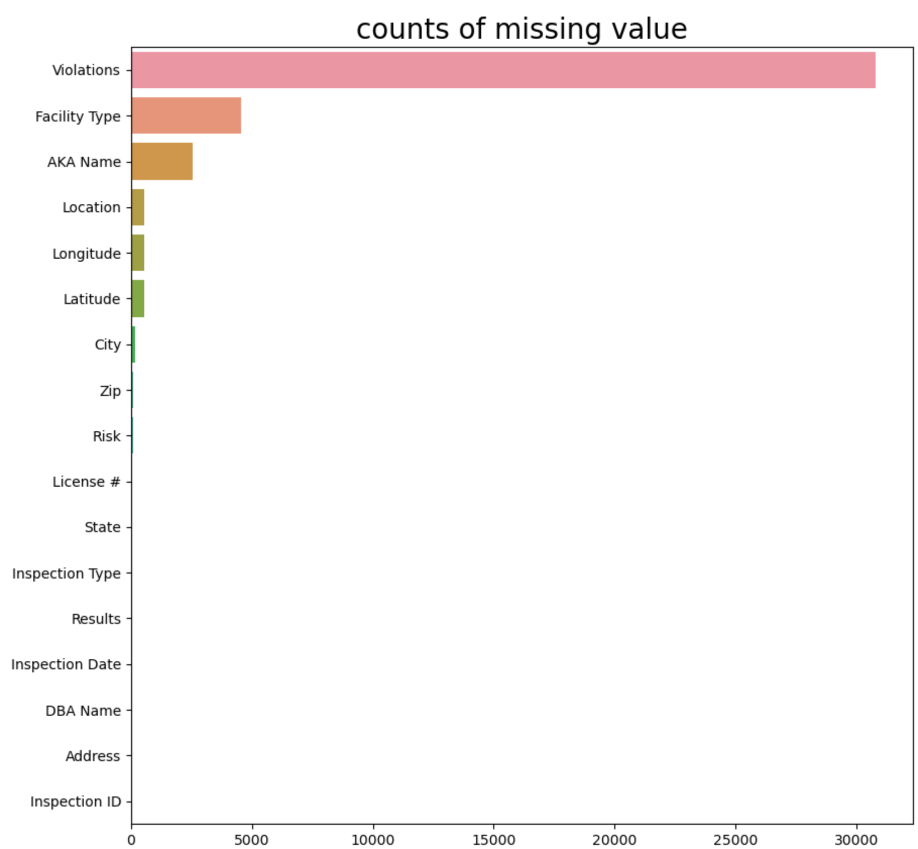

Figure 4.1a - Bar Chart for Count of Missing Values in Dataset

Number of null values in each column:

	Column Name	Null Values	Percentage of Null Values	Non-Null Values
0	Inspection ID	0	0.00	153810
1	DBA Name	0	0.00	153810
2	AKA Name	2543	1.65	151267
3	License #	15	0.01	153795
4	Facility Type	4560	2.96	149250
5	Risk	66	0.04	153744
6	Address	0	0.00	153810
7	City	159	0.10	153651
8	State	8	0.01	153802
9	Zip	98	0.06	153712
10	Inspection Date	0	0.00	153810
11	Inspection Type	1	0.00	153809
12	Results	0	0.00	153810
13	Violations	30798	20.02	123012
14	Latitude	544	0.35	153266
15	Longitude	544	0.35	153266
16	Location	544	0.35	153266

Figure 4.1b - Counts of Null Values in Dataset

```
SELECT Address, * FROM data where TRIM(Address)=''
```

	Address object	Inspection ID int...	DBA Name object	AKA Name object	License # float64
0		1763245	EAT N RUN CHICKEN AND...	EAT N RUN CHICKEN AND...	2442868
1		114452	Starfruit Cafe	Starfruit Cafe	2031651
2		60405	CASA CENTRAL LA POSADA	CASA CENTRAL LA POSADA	0

Figure 4.1c - Records with blank address

2. Duplicated Values – Since the dataset is given as a flat file of inspection events, there are many duplicates among food establishments and location entities. For example, as you can see in figure 4.1d below, License #, DBA Name, Address, and Location only have a fraction (10-20%) of unique values as compared to overall dataset size. Using the SQL queries shown below, we can see there should be around 17k unique locations and 34k food establishments. It is important to verify the duplicates are eliminated before loading the data into our database otherwise the numbers shown in our visualizations will be off due to double counting.

```
Number of unique values in each column:
Inspection ID       153810
Violations          121916
License #            32850
DBA Name             24685
AKA Name             23591
Address              17017
Longitude            15908
Latitude             15908
Location             15908
Inspection Date       1946
Facility Type          447
Inspection Type        108
Zip                    100
City                    57
Results                  7
Risk                     4
State                    1
```

Figure 4.1d - Number of unique values in Dataset

```sql
SELECT COUNT(1) Num_Distinct_locations
FROM (SELECT DISTINCT address,city,state,zip
FROM data) T
```

	Num_Distinct_locations int64
0	17077

Figure 4.1e - Number of distinct locations in Dataset

```sql
SELECT COUNT(1)  Num_Distinct_establishments FROM (
SELECT DISTINCT "License #","DBA Name",COALESCE("AKA Name","DBA Name"),"Address","City","Zip"
from data ) AS T
```

	Num_Distinct_e...
0	34015

Figure 4.1f - Number of distinct establishments in Dataset

3. Inconsistent Naming and Typos- As shown in figure 4.1e below, the same Facility Types are listed in different ways. This will need to be clustered and combined using OpenRefine.

a. Other string columns that can be clustered include Inspection Type, DBA Name, AKA Name, Address, and City.
b. For string columns like DBA Name, Facility Type, Address and City the values should be made same case (uppercase/titlecase).

```
(convenience store)
(gas station)
1005 NURSING HOME
1023
1023 CHILDERN'S SERVICE FACILITY
1023 CHILDERN'S SERVICE S FACILITY
1023 CHILDERN'S SERVICES FACILITY
1023 CHILDREN'S SERVICES FACILITY
1023-CHILDREN'S SERVICES FACILITY
1584-DAY CARE ABOVE 2 YEARS
A-Not-For-Profit Chef Training Program
AFTER SCHOOL CARE
AFTER SCHOOL PROGRAM
ALTERNATIVE SCHOOL
ART GALLERY
ART GALLERY W/WINE AND BEER
ASSISSTED LIVING
ASSISTED LIVING
Adult Family Care Center
```

Figure 4.1e - Facility Type Clustering Examples

```
select distinct "Inspection Type" from data
```

	Inspection Type object (License 0.9%; 107 others 98.2%; Missing 0.9%)
40	TASK FORCE LIQUOR 1470
41	Sample Collection
42	license task 1474
43	FIRE/COMPLAIN
44	Task Force for liquor 1474
45	ADDENDUM
46	1315 license reinspection
47	Task force liquor inspection 1474
48	Task Force Liquor Catering

```
select distinct city from data
```

	City object
0	CHICAGO
1	CCHICAGO
2	CHICAGOCHICAGO
3	312CHICAGO
4	SCHILLER PARK
5	CHCHICAGO
6	CHICAGOI
7	CHICAGO HEIGHTS

Figure 4.1g&h – Inspection Type & City Clustering Examples

4. Correct Datatypes - Zip needs to convert from float to string and trim to 5 characters. License should be converted to integer and convert NaN to 0, so can filter them out. Inspection Date needs to be parsed into date object.

```
Data columns (total 17 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Inspection ID    153810 non-null  int64
 1   DBA Name         153810 non-null  object
 2   AKA Name         151267 non-null  object
 3   License #        153795 non-null  float64
 4   Facility Type    149250 non-null  object
 5   Risk             153744 non-null  object
 6   Address          153810 non-null  object
 7   City             153651 non-null  object
 8   State            153802 non-null  object
 9   Zip              153712 non-null  float64
 10  Inspection Date  153810 non-null  object
 11  Inspection Type  153809 non-null  object
 12  Results          153810 non-null  object
 13  Violations       123012 non-null  object
 14  Latitude         153266 non-null  float64
 15  Longitude        153266 non-null  float64
 16  Location         153266 non-null  object
dtypes: float64(4), int64(1), object(12)
```

Figure 4.1i – Datatype Conversion Examples

5. The Violations column contains the code, description and comments all listed in a single column. These will need to be parsed using Regex in order for us to store it into the DB schema. In total there are 46 distinct violation codes, each having a unique description. This description should be stored with the code in a master table and should be made to have the same sentence case. The comments should be stored with each inspection violation code. Note that there can multiple lines of comments for each violation.

N
Violations
33. FOOD AND NON-FOOD CONTACT EQUIPMENT UTENSILS CLEAN, FREE OF ABRASIVE DETERGENTS - Comments: INSTRUCTED TO DETAIL CLEAN AND MAINTAIN INTERIOR SURFACES OF ICE MACHINE. | 40. REFRIGERATION AND METAL STEM T
2. FACILITIES TO MAINTAIN PROPER TEMPERATURE - Comments: VIOLATION CORRECTED PREMISES HAS 1 MILK COOLER, 2 2 DOOR COOLERS, 1 FREEZER, AND 1 HOME REFRIGERATOR. ALL AT PROPER TEMPER COOLERS/FREEZER. | 11. ADEQUAT
35. WALLS, CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAIR, SURFACES CLEAN AND DUST-LESS CLEANING METHODS - Comments: FOUND THREE WATER DAMAGED CEILING TILES AT THE ENTRANCE LEADING INTO THE
2. FACILITIES TO MAINTAIN PROPER TEMPERATURE - Comments: OBSERVED INADEQUATE REFRIGERATION PREMISES HAS ONE HOUSE REFRIGERATOR. ONE FREEZER. NEED TO PROVIDE REFRIGERATION FOR ALL FOOD PRODUCT/MILK FOR 1200/
33. FOOD AND NON-FOOD CONTACT EQUIPMENT UTENSILS CLEAN, FREE OF ABRASIVE DETERGENTS - Comments: MUST CLEAN INSIDE ALL CABINETS ON SHELF TO REMOVE DUST AND DIRT,INSTRUCTED TO CLEAN AND MAINTAIN AREA | 40. RE
21. * CERTIFIED FOOD MANAGER ON SITE WHEN POTENTIALLY HAZARDOUS FOODS ARE PREPARED AND SERVED - Comments: Violation corrected. Summer festivals licenses obtained. | 35. WALLS, CEILINGS, ATTACHED EQUIPMENT CONSTRU

Figure 4.1k – Violations Format

6. License # as zero – there are 439 records with a license number of 0.

```
Top values for column 'License #':
0.0          439
1354323.0    198
14616.0      172
1574001.0     79
1974745.0     58
1490035.0     45
20481.0       44
1596210.0     42
1142451.0     40
1884255.0     40
Name: License #, dtype: int64
```

	Inspection	DBA Name	AKA Name	License #	Facility Type	Risk	Address
2	2078644	ST. MARY'S CHURCH	ST. MARY'S CHURCH	0	Special Event	Risk 1 (High)	4225 N CENTRAL AVE
3	2069393	THE WAY OF TRUTH BAPTIST CHURCH	THE WAY OF TRUTH BAPTIST CHUR	0	SUMMER FEEDING PREP AREA	Risk 1 (High)	6456 S CALIFORNIA AVE
4	2064703	ANNUNCIATION GREEK ORTHODOX	ANNUNCIATION GREEK ORTHODO	0	CHURCH	Risk 2 (Medium)	1017 N LA SALLE DR
5	2065073	THE WAY OF TRUTH BAPTIST CHURCH	THE WAY OF TRUTH BAPTIST CHUR	0	SUMMER FEEDING PREP AREA	Risk 1 (High)	6456 S CALIFORNIA AVE
6	2064639	IMMACULATE CONCEPTION CHURCH		0	CHURCH/SPECIAL EVENT	Risk 1 (High)	8756 S COMMERICAL AVE
7	2064631	OLD ST. PATRICK'S CHURCH	OLD ST. PATRICK'S CHURCH	0	Church	Risk 2 (Medium)	700 W ADAMS ST
8	2060022	CHICAGO BEST NAAN	CHICAGO BEST NAAN	0	Bakery	Risk 2 (Medium)	6352 N OAKLEY AVE
9	2059984	LOVE TACO	LOVE TACO	0	Restaurant	Risk 1 (High)	109 E 51ST ST
10	2059731	OLD ST. PATRICK'S CHURCH	OLD ST. PATRICK'S CHURCH	0	Church	Risk 2 (Medium)	700 W ADAMS ST
11	2059546	LOVE TACO	LOVE TACO	0	Restaurant	Risk 1 (High)	109 E 51ST ST
12	2059363	BBQ SUPPLY	BBQ SUPPLY	0	Restaurant	Risk 1 (High)	6948 N WESTERN AVE
13	2050791	ST. EUGENE PARISH	ST. EUGENE PARISH Shaunnessy Ce	0	Special Event	Risk 2 (Medium)	5220 N CANFIELD AVE
14	2050319	BBQ SUPPLY	BBQ SUPPLY	0	Restaurant	Risk 1 (High)	6948 N WESTERN AVE
15	2050069	LITTLE BLACK PEARL	LITTLE BLACK PEARL	0	School	Risk 2 (Medium)	1060 E 47TH ST
16	2049900	ST. GEORGE GREEK ORTHODOX CHUR	ST. GEORGE GREEK ORTHODOX CH	0	Special Event	Risk 1 (High)	2701 N SHEFFIELD AVE
17	2049466	ST. ANDREWS GREEK ORTHODOX CH	ST ANDREW'S GREEK ORTHODOX C	0	CHURCH/SPECIAL EVENTS	Risk 1 (High)	5649 N Sheridan RD
18	2049301	CHICAGO BEST NAAN	CHICAGO BEST NAAN	0	Bakery	Risk 2 (Medium)	6352 N OAKLEY AVE
19	2010062	JACKSON PARK SLF, LLC	JACKSON PARK SLF, LLC	0	Long Term Care	Risk 1 (High)	1440-1448 E 75TH ST
20	1995369	LUBAVITCH GIRLS HIGH SCHOOL	CONGREGATION BNEIRUVEN	0	PRIVATE SCHOOL	Risk 1 (High)	6350 N WHIPPLE ST
21	1979104	FIVE STARZ FOODS	FIVE STARZ FOODS	0	Grocery Store	Risk 3 (Low)	7855 S HALSTED ST

Figure 4.1l&m– License # 0 Examples

7. License # is not enough to uniquely to a food establishment. As can see in screenshot below, there are cases where two different businesses share the same license #. Food establishments can also have different licenses either due to having multiple locations in case of franchises or a business might have had to be temporarily closed and then reassigned new license. In such case, we would need to correlate based on the inspection date to figure out most recent license # as valid one, but as of now that is out of scope for our use case. To uniquely identify a food establishment, we plan to create constraint on composite key index using following columns License #, DBA Name, AKA Name, and Location

```sql
SELECT "License #", COUNT(DISTINCT "DBA Name") "Distinct_Business_Count"
FROM data
GROUP BY "License #"
HAVING COUNT(distinct "DBA Name") > 1
ORDER BY 2 DESC
```

	License # float64 0.0 - 2535148.0	Distinct_Busine... 2 - 211
0	0	211
1	14616	7
2	nan	6
3	1354323	5
4	1514802	4
5	1893935	3
6	1542362	3
7	1042664	3
8	1120537	3
9	1933945	3

Here is link our Python notebook used for this analysis:

https://deepnote.com/@mcs-ds/CS513FinalProjectTeam59-5c00fda9-e408-45c6-ac1e-09c6fb54e7f2

4.2 Why data cleaning is necessary for main use case U1

Data cleaning is a crucial step to support the main use case of providing insights and suggestions based on food safety inspections. Currently the 'Violation' column contains various types of violations and their corresponding comments in a single column. To pave the way for a more detailed analysis and provide reliable insights and suggestions, data cleaning will be needed to extract these violations and comments to their own tables and columns. With cleaning, we plan to standardize the violation types and leverage it for analysis. Next to provide safety insights by facility types, we need a high level of consistency of the data. There are noticeable inconsistencies with the 'Facility Types' & 'Inspection Type' columns like spelling mistakes, inconsistent casing, and lack of standardization.

Without data cleaning of this dataset, we will not be able to ensure integrity and validity of the data, which will not achieve our goal of providing accurate and reliable insights for our consumers.

5. Phase-II Initial Plan

5.1 Data Cleaning Plan

1. General data cleaning using OpenRefine, Python, SQLite, & YesWorkflow tools
    a. Check for missing values
    b. Check for and remove Duplicate records
    c. Fix & standardize inconsistent data
    d. Data type conversions
    e. Formatting columns to same case (uppercase/title case).
    f. Perform text cleaning to remove extraneous special characters, punctuations, and leading, trailing and consecutive whitespaces
    g. Check for and Remove outliers
2. Use clustering to group and clean categorical data column like Risk, Inspection Type, Results are inconsistencies or misspellings
3. Parse violation columns using Python and RegEx
    a. Extract each violation to its own table
    b. Extract comments out for each violation and store with Inspection Violation mapping
4. Fix Facility Type & Inpsection Type column using OpenRefine and python
    a. Standardized spelling and facility type name
    b. Cluster and merge similar names to use standardized name
5. Create SQL constraints to ensure unique locations & food establishments
6. Load data to SQLite database and perform data integrity checks via SQL.
7. Generate YesWorkflow model using OR2YW Tool
8. Demonstrate Data quality improvements
9. Document Data Cleaning procedure into final report

The cleaned data will be loaded to the SQLite database matching the ERD. It will be used for the data analysis steps further to answer the main use case questions and visualizations.

5.2 Who in the team will be responsible for which steps and Timeline

Action	Who	Deadline
S1 – General data cleaning (trailing spaces, missing values, duplicates, outliers)	Theara	Jul, 11
S2 - Data cleaning using Clustering	Ashley	Jul, 11
S3 – Clean violation column	Avinash	Jul, 11
S4 – Fix facility type column	Avinash	Jul, 13
S7– Create workflow model using YesWorkflow tool	Ashley	Jul, 14
S6 – Create the python script to load the SQLite tables with the cleaned data	Avinash	Jul, 18
S8 – Perform the U1 analysis and visualizations	Theara	Jul, 21

S9 Write detailed description of data cleaning performed. Describe all data cleaning steps performed (high-level) and explain the Rationale.	All	Jul, 24
S9 Document data quality changes, show how the data improved and quantify it. Which columns and how many cells affected?	All	Jul, 24
S9 Work on Conclusions and Summary	All	Jul, 24
S9 Finish Final Report	All	Jul, 24