CS513: Theory & Practice of Data Cleaning

**Final Project Phase 1 Report**

**Team 59: Data Mavericks**

Avinash Baldeo, Ashley George, Sotheara Chea

[abaldeo2@illinois.edu,](mailto:abaldeo2@illinois.edu) [ageorge8@illinois.edu](mailto:ageorge8@illinois.edu), [chea3@illinois.edu](mailto:chea3@illinois.edu)

**Table of Contents**

[**1. Dataset Overview** 3](#_Toc138967167)

[**2. Dataset Description** 3](#_Toc138967168)

[2.1 Full Data Narrative 3](#_Toc138967169)

[2.2 Database Diagram & Schema 5](#_Toc138967170)

[**3. Use cases** 8](#_Toc138967171)

[3.1 U0: Zero Cleaning Use Case 8](#_Toc138967172)

[3.2 U1: Main Use Case 8](#_Toc138967173)

[3.3 U2: Never Enough Use Case 9](#_Toc138967174)

[**4. Data Quality Problems** 9](#_Toc138967175)

[4.1 List obvious data quality problems 9](#_Toc138967176)

[4.2 Why data cleaning is necessary for main use case U1 11](#_Toc138967177)

[**5. Phase-II Initial Plan** 11](#_Toc138967178)

[5.1 Data Cleaning Plan 11](#_Toc138967179)

[5.2 Who in the team will be responsible for which steps and Timeline 12](#_Toc138967180)

# **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.

# **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.

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

The dataset is maintained using Socrata's API (Application Programming Interfaces) and Kaggle's API, 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 1 – Food Inspection Dataset Description

## 2.2 Database Diagram & Schema

The following database diagram(s) represents 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.

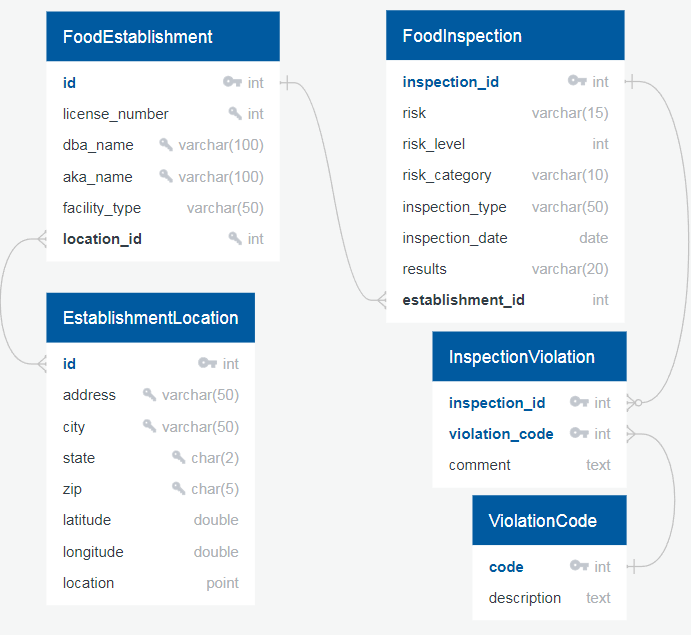


Figure 1A – Entity-Relationship (ER) Diagram of the normalized data

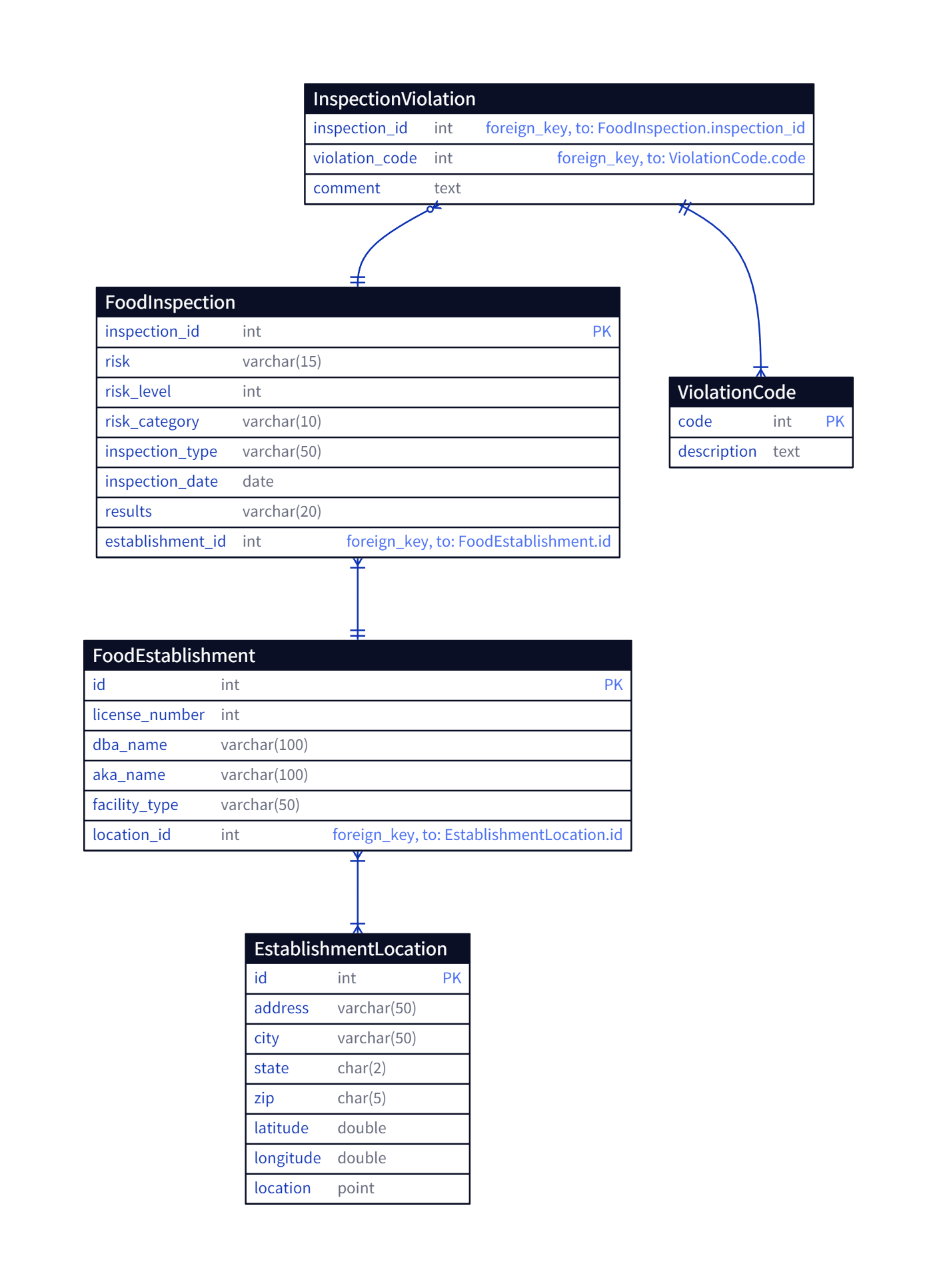


Figure 1B – Entity-Relationship (ER) Diagram with foreign keys

**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.

**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.

**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.
  + 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 risk column.
  + Inpsection\_date and results are 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.

**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 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);**

These unique index constraints will be added to the schemas in Phase 2 after cleaning, otherwise it will prevent loading the data 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

To provide neighborhood-based insights on relative food safety. The data set already contains the name, location, results of the inspections and associated risk are available.

* Which geographical areas have the highest concentration of failed inspections?
* What is the geographic distribution of the inspection risk categories?
* Summary of inspections per timeframe (year/month) and counts by results.

## 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 of it. We attempt to provide insights based on past food safety inspection violation types, their frequency, and their severity for consumers. All violations cited are included in the same column now. Comments are provided on the details of the violation too in the same column. The main use case is to provide insights and suggestions to the consumer public based on violations and their trends and distribution. Below are some examples

* Are there any correlations between the type of facility and the frequency or severity of violations?
* Are there any trends or patterns in inspection results or violations by certain popular chains and brands.
* Are there patterns in distribution of violation types by neighborhoods?
* Are there any trends or patterns in inspection results or violations over time?
* How effective are re-inspections in improving the compliance of food establishments?
* Is there any seasonal patterns in inspection outcomes or specific violations?

## 3.3 U2: Never Enough Use Case

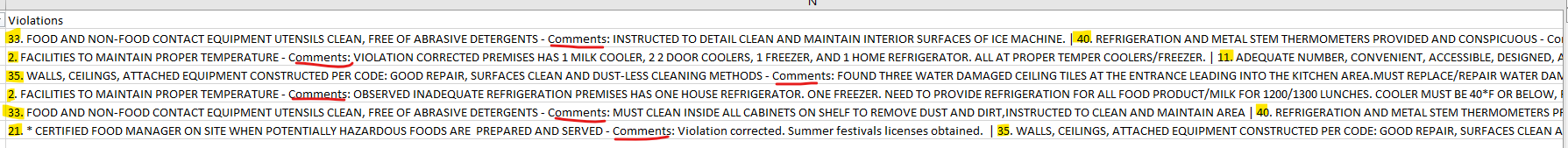
Be able provide overall recommendations to consumers on food establishments as the data available here is only a view of food safety. Does not incorporate customer reviews on taste, customer service, price, etc. This would need customer reviews from the likes of Google, yelp, etc. combined with this food safety data used to create a recommendation system.

# **4. Data Quality Problems**

## 4.1 List obvious data quality problems

1. Empty cells- there are many cells missing in some of the columns in the dataset like the violations, facility type, aka name, and longitude and latitude. The most is the violations column with 30798 null cells. Depending on the use case, these locations with no violation may need to be filtered out. In other use cases, these null cells will show that inspection resulted in no violations.
   1. A picture containing text, screenshot, display, software

      Description automatically generated
2. Inconsistent Naming and Typos- the same facility types are listed in different ways. Many of this will need to be cleaned and combined.
   1. A screenshot of a computer program

      Description automatically generated with medium confidence
3. Violations and comments are all listed in one column
   1. 
4. License # as 0- there are 439 records with a license number of 0.
   1. A screenshot of a computer

      Description automatically generated with medium confidence

## 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’ column 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\_\_\_
   1. Check for missing values
   2. Check for and delete Duplicate records
   3. Fix inconsistent data
   4. Data type conversions for dates
   5. Formatting columns to title case.
   6. Text cleaning to remove extraneous special characters, punctuations, and leading, trailing and consecutive whitespaces
   7. Delete unnecessary columns
   8. 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
   1. Extract each violation to its own table
   2. Extract comments out for each violation
4. Fix facility type column using OpenRefine and python
   1. Standardized spelling and facility type name
   2. Combine and merge errors to use standardized name
5. SQL constraint for unique location & food establishment
6. Load data to SQLite database and perform data quality improvement by SQL.
7. Generate YesWorkflow model using OR2YW Tool
8. Demonstrate Data quality improvements

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** |
| General data cleaning (trailing spaces, missing values, duplicates, outliers) | Theara | Jul, 6 |
| Data cleaning using Clustering | Ashley | Jul, 11 |
| Clean violation column | Avinash | Jul, 6 |
| Fix facility type column S4 | Avinash | Jul, 13 |
| Create workflow model using YesWorkflow tool | Ashley | Jul 14 |
| Create the python script to load the sqllite tables with the cleaned data | Avinash | Jul, 20 |
| Perform the U1 analysis and visualizations | Theara | Jul 21 |
| Write detailed description of data cleaning performed. Describe all data cleaning steps performed (high-level) and explain the Rationale. | All | Jul, 24 |
| Document data quality changes, show how the data improved and quantify it. Which columns and how many cells affected? | All | Jul, 24 |
| Work on Conclusions and Summary | All | Jul, 24 |
| Finish Final Report | All | Jul, 24 |