Team 93
ericca2@, komminen@, rajansp2@

CS513 Project Phase-II Submission

Table of Contents

List of Supplementary Materials

☐ *Submission of supplementary materials (10 P)*

In addition to this PDF document, the following supplementary materials are included in the submission in a zip file name Team93-Phase2SupplementaryMaterials.zip:

- Outer Workflow model – **OuterWorkflow.pdf** (from a chart drawing tool, draw.io) **OuterWorkflow.drawio**
- OpenRefine Inner Workflow model – Three files: **InnerWorkflow-OR.pdf**, **InnerWorkflow-OR.yw** and **InnerWorkflow-OR.gv**
- Python Inner Workflow model – Three files: **InnerWorkflow-Python.pdf**, **InnerWorkflow-Python.yw** and **InnerWorkflow-Python.gv**
- OpenRefine Operation History – **OpenRefineHistory.json** and **sub_ops.json** files
- Python Data Cleaning History – Python script used to clean missing geo-lookup coordinates further, add FacilityID, to split the data into different tables, and to parse the violations column. Violations are parsed by serializing each non-null record into an array of JSON objects with keys for "id", "desc" and "comments". Additionally, a column with an array of violation IDs is provided for a quick count or lookup operations. A standalone lookup table with all unique violation IDs and their descriptions is generated. Finally, a new flat file is produced with records for each individual violation, with the primary key as a combination of (I*spection ID, Violation ID*). Two SQLite databases are generated, one single table database with the flat file of violation records, another relational database with the various entity tables from the schema - **cfi_data_cleaning.py**
- Queries – SQL queries in a file named **queries.txt**. Note that these queries were run against the cleaned dataset loaded into a SQL Server Express database.
- Original and Cleaned Datasets: Uploaded in a U of I Box folder with the link shared in **DataLinks.txt**
- SQLite Databases **–** Created from cleaned .csv file using Python's sqlite3 module. **Chicaco_Food_Inspection_Relational.db** and **Chicaco_Food_Inspection_Single.db**

Introduction

Dataset

We chose the Chicago Food Inspections dataset. This dataset gave us good ideas on use cases as described in our Phase 1 report. It gave us lots of good exposure to different data cleaning methods and techniques ranging from basic to advanced functionality of OpenRefine, additional data cleaning and transformation with Python/Pandas and a prototype realizing the use case through a Tableau dashboard.

Motivation/Use Case

Main Use Case – U$_1$

Build an interactive data dashboard or web application that allows users to explore the food inspection dataset. The dataset will be clean and enable a number of user scenarios:

1. **Explore the data** in aggregate across the city of Chicago. For example, view the data set on a map using the geospatial data, understand types of violations, which violations are the most common, failures by zip code/region, etc.
2. **Zoom and filter** using the dashboard/dataset to find or choose a specific restaurant or a grocery store - for example, in a specific area of the city with no inspection failures.

3. **Dive into details** on a specific establishment. For example, understand the history of inspections (especially failures) for a specific restaurant or grocery store before visiting that establishment
4. **Understand complaints** - Understand which establishments have complaints as the reason for inspection. This can be used to understand complaint history for an establishment you've already visited or to avoid an establishment.

Minor Use Case – U_0

Show count of inspections by date. Extract insights into trends on inspection counts YoY from 2010-2017 and month-over-month within a year.

Note: The column "Inspection Date" is fully specified with no blank data, erroneous dates (at least those which can be discerned from the dataset itself). Similarly, the "Inspection ID" column has no data quality issues that can be inferred and cleaned from the given data set. With these two columns, we can fulfil the use case above with **"zero data cleaning"** as required for U_0.

Minor Use Case – U_2

Enable the analysis of inspection failures by type of cuisine.

Note: A plausible hypothesis is that this might be doable from restaurant name. On further inspection however, it really needs additional metadata not captured in the dataset, i.e. data cleaning is **not sufficient** for this use case

Data Cleaning Workflow (Create a workflow model)

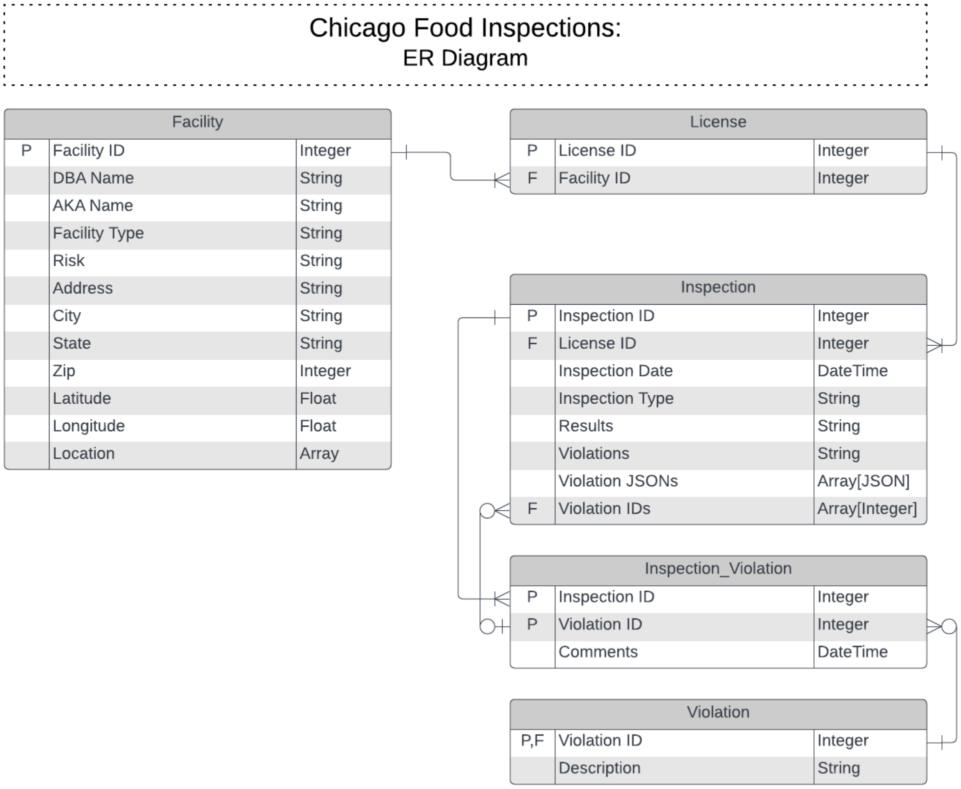
Design of Workflow & Tools Used

Our outer workflow is described in the next section. We used different tools at different stages of our workflow. The tools used and the rationale for the usage of these tools is listed below:

- **Excel** - Provided the easiest way to load and scan through a .csv file and perform initial profiling and inspection of the data. However, due to limitations of Excel for data cleaning, we made use of OpenRefine to clean most of the data and then used Python to perform advanced data profiling with only a few lines of code.
- **OpenRefine** - We learned about OpenRefine through the assignment and really liked the superior data cleaning abilities it provides. It also helps that it is an opensource tool and free vs the paid data cleaning tools that are also available.
- **Python** - Some of the data cleaning & data manipulation can be more easily done in Python than OpenRefine. For example, with the help of Python, we were able to run a script to fill in missing latitude/longitude entries based on the addresses provided. We also used python to perform tasks in parallel with pandas dataframes. Finally, we used python to transform the flat, cleaned CSV into multiple csv files (for easy loading into SQL Server) and to output a standalone SQLite db file.

- **Draw.io** – We used this to generate our outer workflow diagrams as suggested by a TA on CampusWire. Given that these were manually created, draw.io was easier to use for this purpose.
- **OR2YWTool** – This was the easiest way to generate a YesWorkflow model from OpenRefine.
- **SQL Server Express and SQLite** – These data base packages were used to load both the original and cleaned data sets into SQL databases to verify that the overall data quality has improved through ICV queries and such
- **Tableau** – This was used to realize the use case U1 as a prototype "app". This allowed us to reason about the rationale for the different data cleaning steps (for example, a significant amount of data clean was done on DBA_Name and AKA_Name since users interact with the entity name more than any other field in the data set), provide a way of representing the data, provided comparisons of the UX between unclean & clean data, etc.

A difference vs "standard" data cleaning that we decided to implement as part of our project was the creation of a database schema from the data vs just cleaning the csv/flat file and producing another flat file (though we did this as well). The ER diagram for our final schema is shown below and the supplementary materials has a .png file of the same



Chicago Food Inspections:
ER Diagram

Facility		
P	Facility ID	Integer
	DBA Name	String
	AKA Name	String
	Facility Type	String
	Risk	String
	Address	String
	City	String
	State	String
	Zip	Integer
	Latitude	Float
	Longitude	Float
	Location	Array

License		
P	License ID	Integer
F	Facility ID	Integer

Inspection		
P	Inspection ID	Integer
F	License ID	Integer
	Inspection Date	DateTime
	Inspection Type	String
	Results	String
	Violations	String
	Violation JSONs	Array[JSON]
F	Violation IDs	Array[Integer]

Inspection_Violation		
P	Inspection ID	Integer
P	Violation ID	Integer
	Comments	DateTime

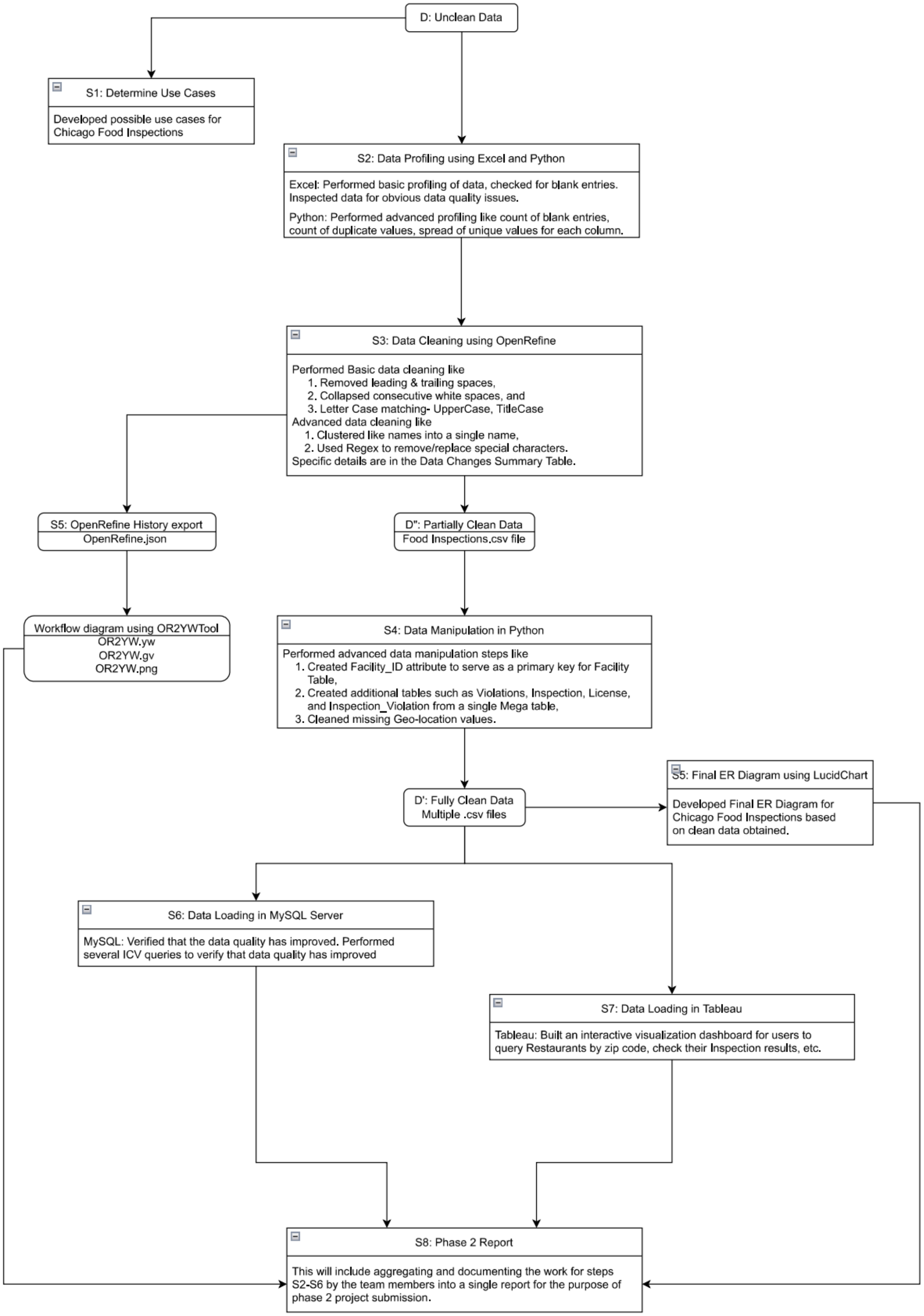
Violation		
P,F	Violation ID	Integer
	Description	String

Data Cleaning Workflow – Outer Workflow

• ☐*A visual representation of your overall (or "outer") workflow W_o, e.g., using a suitable tool such as YesWorkflow. At a minimum, you should identify key inputs, outputs, and steps of the workflow, along with dependencies between these. Key phases and steps of your data cleaning project may include, e.g.,* *data profiling, data loading, data cleaning, IC violation checks, etc. Explain the design of W_o and why you've chosen the tools that you have in your overall workflow. (10 P)*

Our final data workflow is shown below as a visual representation and later in tabular form with additional notes. The visual representation is also in the supplementary materials (**OuterWorkflow.pdf**).

Actual Workflow in Phase-II



D: Unclean Data

S1: Determine Use Cases

Developed possible use cases for Chicago Food Inspections

S2: Data Profiling using Excel and Python

Excel: Performed basic profiling of data, checked for blank entries. Inspected data for obvious data quality issues.

Python: Performed advanced profiling like count of blank entries, count of duplicate values, spread of unique values for each column.

S3: Data Cleaning using OpenRefine

Performed Basic data cleaning like
1. Removed leading & trailing spaces,
2. Collapsed consecutive white spaces, and
3. Letter Case matching- UpperCase, TitleCase

Advanced data cleaning like
1. Clustered like names into a single name,
2. Used Regex to remove/replace special characters.

Specific details are in the Data Changes Summary Table.

S5: OpenRefine History export

OpenRefine.json

D'': Partially Clean Data

Food Inspections.csv file

Workflow diagram using OR2YWTool

OR2YW.yw
OR2YW.gv
OR2YW.png

S4: Data Manipulation in Python

Performed advanced data manipulation steps like
1. Created Facility_ID attribute to serve as a primary key for Facility Table,
2. Created additional tables such as Violations, Inspection, License, and Inspection_Violation from a single Mega table,
3. Cleaned missing Geo-location values.

D': Fully Clean Data

Multiple .csv files

S5: Final ER Diagram using LucidChart

Developed Final ER Diagram for Chicago Food Inspections based on clean data obtained.

S6: Data Loading in MySQL Server

MySQL: Verified that the data quality has improved. Performed several ICV queries to verify that data quality has improved

S7: Data Loading in Tableau

Tableau: Built an interactive visualization dashboard for users to query Restaurants by zip code, check their Inspection results, etc.

S8: Phase 2 Report

This will include aggregating and documenting the work for steps S2-S6 by the team members into a single report for the purpose of phase 2 project submission.

Step #	Step Description	Tools Used	Team members	Notes
S1	Project report Phase 1 - Includes description of dataset D and matching use case U1.	UML diagram generator	All	We decided to use the Chicago Food Inspections Database which includes the records of inspection of various types of food facilities across the city of Chicago.
S2	Profiling of D to identify the quality problems P that need to be addressed to support U1	Excel, Python	All	Excel: Performed basic profiling of data, checked for blank entries. Inspected data for obvious data quality issues. Python: Performed advanced profiling like count of blank entries, count of duplicate values, spread of unique values for each column.
S3	Performing the data cleaning process using OpenRefine tool to address the problems P.	OpenRefine	Rajan Patel	Performed Basic data cleaning like 1. Trimmed leading & trailing spaces, 2. Collapsed consecutive white spaces, and 3. Letter Case matching- UpperCase or TitleCase Advanced data cleaning like 1. Clustered like names into a single name, 2. Used Regex to remove/replace special characters. We also used manual Google searches to verify if similar names are referring to the same entity or are different entities. Specific details of data cleaning are in the Data Changes Summary Table.
S4	Performing the data manipulation process using Python to address the problems P.	Python	Eric Alhgren	Performed advanced data manipulation steps like 1. Created Facility_ID attribute to serve as a primary key for Facility Table, 2. Created additional tables such as Violations, Inspection, License, and Inspection_Violation from a single Mega table, 3. Cleaned missing Geo-location values.

S5	Documenting the types and amount of changes that have been executed on D to obtain D’	OpenRefine Export, YesWorkflow, ER Diagramming tool	Rajan Patel, Eric Alhgren	Workflow diagram using OR2YWTool. Created Summary Table of changes. Developed Final ER Diagram for Chicago Food Inspections based on clean data obtained.
S6	Data Loading into a SQL database.	SQL Server Express	Vamshidhar Kommineni	SQL Server Express: Verified that the data quality has improved. Performed several ICV queries to verify that data quality has improved
S7	Build a Tableau dashboard as a prototype to realize the use case U1.	Tableau	Vamshidhar Kommineni	Tableau: Built an interactive visualization dashboard for users to query Restaurants by zip code, check their Inspection results, etc.
S8	Phase 2 Report	Word	All	This includes aggregating and documenting the work for steps S2-S7 by the team members into a single report for the purpose of phase-2 of project submission.

Comparison to Phase-I Plan

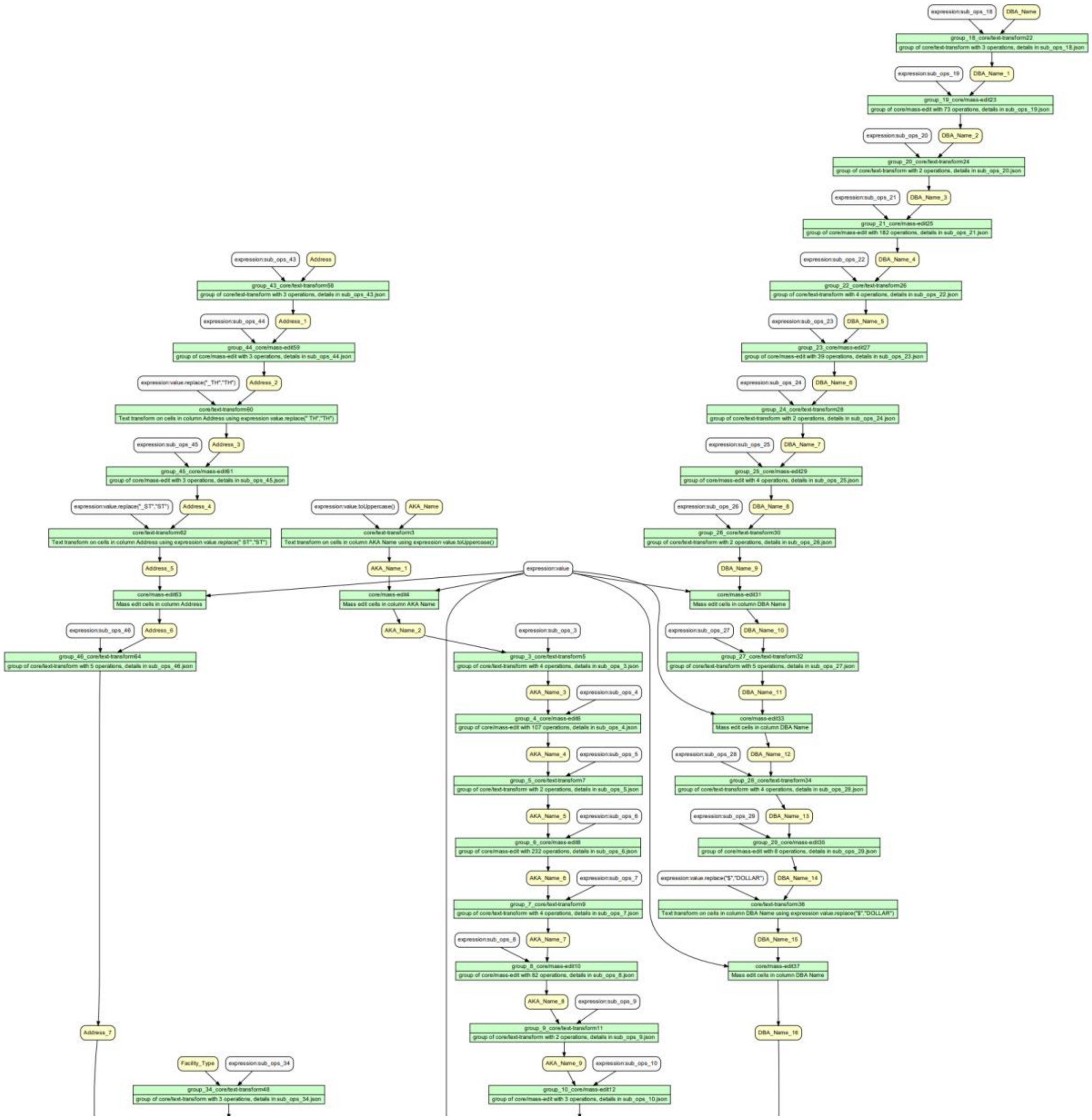
The initial Phase 1 plan is in the Appendix, both in table form and in visual form. Our final workflow (shown above) was very similar to our plan laid out in Phase 1. The main difference was that we realized the need for and were able to perform far more data cleaning than we initially thought was required for the use case U1. We were also able to build a first cut at a Tableau dashboard to realize the use case. This led to additional data cleaning and transformation steps – for example, the creation of Violation and Inspection_Violation tables to make it easier to look up and display the specific violations and comments for an establishment or to search for establishments that had specific kinds of violations.

Data Cleaning Workflow – Inner Workflow

- *☐ A detailed (possibly visual) representation of your “inner” data cleaning workflow W_i (e.g., if you’ve used OpenRefine, you can use the OR2YW tool). (10 P)*

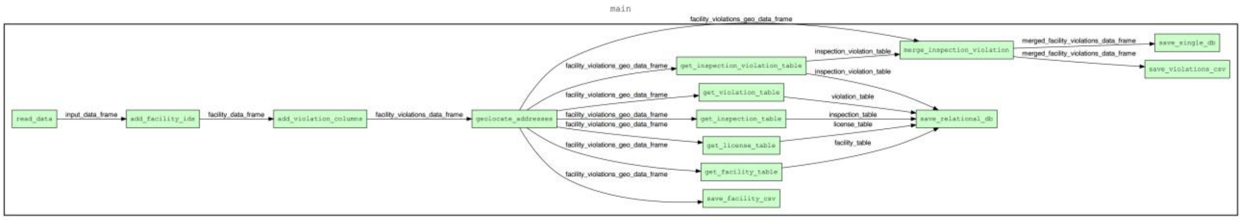
OpenRefine Inner Workflow

Our inner workflow for OpenRefine is included in the supplementary materials (InnerWorkflow-OR.pdf, as well as InnerWorkflow-OR.yw, InnerWorkflow-OR.gv and OpenRefineHistory.json files). The image is too large to show with fidelity in this document, so we included only a partial snippet per the professor’s guidance on CampusWire - please use the pdf or gv to zoom in and look at further details. We also described the inner workflow steps in detail in tabular form in the next section.


expression:sub_ops_18
DBA_Name
group_18_core/text-transform22
group of core/text-transform with 3 operations, details in sub_ops_18.json
expression:sub_ops_19
DBA_Name_1
group_19_core/mass-edit23
group of core/mass-edit with 73 operations, details in sub_ops_19.json
expression:sub_ops_20
DBA_Name_2
group_20_core/text-transform24
group of core/text-transform with 2 operations, details in sub_ops_20.json
expression:sub_ops_21
DBA_Name_3
group_21_core/mass-edit25
group of core/mass-edit with 182 operations, details in sub_ops_21.json
expression:sub_ops_22
DBA_Name_4
group_22_core/text-transform26
group of core/text-transform with 4 operations, details in sub_ops_22.json
expression:sub_ops_23
DBA_Name_5
group_23_core/mass-edit27
group of core/mass-edit with 39 operations, details in sub_ops_23.json
expression:sub_ops_24
DBA_Name_6
group_24_core/text-transform28
group of core/text-transform with 2 operations, details in sub_ops_24.json
expression:sub_ops_25
DBA_Name_7
group_25_core/mass-edit29
group of core/mass-edit with 4 operations, details in sub_ops_25.json
expression:sub_ops_26
DBA_Name_8
group_26_core/text-transform30
group of core/text-transform with 2 operations, details in sub_ops_26.json
DBA_Name_9
core/mass-edit31
Mass edit cells in column DBA Name
expression:sub_ops_27
DBA_Name_10
group_27_core/text-transform32
group of core/text-transform with 5 operations, details in sub_ops_27.json
DBA_Name_11
core/mass-edit33
Mass edit cells in column DBA Name
expression:sub_ops_28
DBA_Name_12
group_28_core/text-transform34
group of core/text-transform with 4 operations, details in sub_ops_28.json
expression:sub_ops_29
DBA_Name_13
group_29_core/mass-edit35
group of core/mass-edit with 8 operations, details in sub_ops_29.json
expression:value.replace("$","DOLLAR")
DBA_Name_14
core/text-transform36
Text transform on cells in column DBA Name using expression value.replace("$","DOLLAR")
DBA_Name_15
core/mass-edit37
Mass edit cells in column DBA Name
DBA_Name_16
expression:sub_ops_43
Address
group_43_core/text-transform58
group of core/text-transform with 3 operations, details in sub_ops_43.json
expression:sub_ops_44
Address_1
group_44_core/mass-edit59
group of core/mass-edit with 3 operations, details in sub_ops_44.json
expression:value.replace(" TH","TH")
Address_2
core/text-transform60
Text transform on cells in column Address using expression value.replace(" TH","TH")
expression:sub_ops_45
Address_3
group_45_core/mass-edit61
group of core/mass-edit with 3 operations, details in sub_ops_45.json
expression:value.replace(" ST","ST")
Address_4
core/text-transform62
Text transform on cells in column Address using expression value.replace(" ST","ST")
Address_5
core/mass-edit63
Mass edit cells in column Address
expression:sub_ops_46
Address_6
group_46_core/text-transform64
group of core/text-transform with 5 operations, details in sub_ops_46.json
Address_7
expression:value.toUppercase()
AKA_Name
core/text-transform3
Text transform on cells in column AKA Name using expression value.toUppercase()
AKA_Name_1
expression:value
core/mass-edit4
Mass edit cells in column AKA Name
AKA_Name_2
expression:sub_ops_3
group_3_core/text-transform5
group of core/text-transform with 4 operations, details in sub_ops_3.json
AKA_Name_3
expression:sub_ops_4
group_4_core/mass-edit6
group of core/mass-edit with 107 operations, details in sub_ops_4.json
AKA_Name_4
expression:sub_ops_5
group_5_core/text-transform7
group of core/text-transform with 2 operations, details in sub_ops_5.json
AKA_Name_5
expression:sub_ops_6
group_6_core/mass-edit8
group of core/mass-edit with 232 operations, details in sub_ops_6.json
AKA_Name_6
expression:sub_ops_7
group_7_core/text-transform9
group of core/text-transform with 4 operations, details in sub_ops_7.json
expression:sub_ops_8
AKA_Name_7
group_8_core/mass-edit10
group of core/mass-edit with 82 operations, details in sub_ops_8.json
AKA_Name_8
expression:sub_ops_9
group_9_core/text-transform11
group of core/text-transform with 2 operations, details in sub_ops_9.json
AKA_Name_9
expression:sub_ops_10
group_10_core/mass-edit12
group of core/mass-edit with 3 operations, details in sub_ops_10.json
Facility_Type
expression:sub_ops_34
group_34_core/text-transform48
group of core/text-transform with 3 operations, details in sub_ops_34.json

Python Inner Workflow

This is included in the supplementary materials (InnerWorkflow-Python.pdf, as well as InnerWorkflow-Python.yw and InnerWorkflow-Python.gv) and shown below. We also described the inner workflow steps in detail in tabular form in the next section.


main
read_data
input_data_frame
add_facility_ids
facility_data_frame
add_violation_columns
facility_violations_data_frame
geolocate_addresses
facility_violations_geo_data_frame
get_inspection_violation_table
facility_violations_geo_data_frame
get_violation_table
facility_violations_geo_data_frame
get_inspection_table
facility_violations_geo_data_frame
get_license_table
facility_violations_geo_data_frame
get_facility_table
facility_violations_geo_data_frame
save_facility_csv
facility_violations_geo_data_frame
merge_inspection_violation
inspection_violation_table
inspection_violation_table
violation_table
inspection_table
license_table
facility_table
save_relational_db
merged_facility_violations_data_frame
save_single_db
merged_facility_violations_data_frame
save_violations_csv

Data cleaning performed, Document data quality changes

- *☐ Identify and describe all (high-level) data cleaning steps you have performed. (20 P)*
- *☐ For each high-level data cleaning step you have performed, explain its rationale. Was it required to support the use case U_1? If not, explain why those steps were still useful. (20 P)*
- *☐ Quantify the results of your efforts, e.g., by providing a summary table of changes: Which columns changed? How many cells (per column) have changed, .. ? (10 P)*

In the first column below, we indicate all data cleaning steps that were required for our use case U1 with a *. Other steps were taken to improve the data cleanliness, build a good table structure for our database, etc. but weren't strictly required for the use case U1

OpenRefine Data Cleaning Performed

Data Cleaning Step #	Data Cleaning Step Description	Data Cleaning Rationale (Requirement for U1)	Data Changes Summary - Columns affected / # of Cells affected
1	Text transform value.toDate	Required so that the value can be treated as Date for a variety of chronological queries. One of our aims with U_1 was to provide an inspection timeline.	Inspection Date / 153810
2	Cluster "like" names into a single name	Required to reduce the spread of different names pointing to the same entity. Eg. Walgreens & walgreens pharmacy #12345 point to the same "Walgreens" name, CHICAGO, chicago, and Cchicago point to Chicago. This provides a cleaner view to the user.	City / 153518 AKA_Name / 69473 DBA_Name / 63087 Facility Type / 9309 Address / 649 Inspection Type / 88720
3	Text transform value.toUppercase	Required as values transformed into one common letter case helped in multiple ways like better organization, increased readability, and affects clustering to some extent.	City / 1 AKA_Name / 9585 DBA_Name / 10443 Facility Type / 146729 Address / 9488 Inspection Type / 153692 Results / 153810
4	Text transform value.toNumber	Required for implementing domain constraints, and helped in performing numeric comparisons. Also, we have multiple sub-use cases like show facilities by zip code, plot facilities on Geo Map using lat/long, etc.	License # / 153795 Inspection ID / 153810 Latitude / 153266 Longitude / 153266 Zip / 153712
5	Trim leading & trailing White Spaces	Required operation for improved clustering & querying. E.g. Select * from TABLE where DBA_Name = "Starbucks Coffee" will give results not having "Starbucks Coffee " or " Starbucks Coffee ".	AKA_Name / 8787 DBA_Name / 18039 Address / 153366 Violation / 18580

6	Text transform value.replace(".","")	Required to remove special characters providing better readability on Tableau dashboard.	AKA_Name / 7076 DBA_Name / 11072 Facility Type / 2
7	Text transform value.replace(",","")	Required to remove special characters providing better readability on Tableau dashboard.	AKA_Name / 4098 DBA_Name / 8149
8	Text transform value.replace("/"," / ")	Required to remove special characters providing better readability on Tableau dashboard.	AKA_Name / 2914 DBA_Name / 1830 Facility Type / 537
9	Text transform value.replace("ANNEX","")	Required to remove business designations as users do not want to see them in names, thus increasing readability and searchability on Tableau dashboard.	AKA_Name / 92 DBA_Name / 77
10	Text transform value.replace("INC","")	Required to remove business designations as users do not want to see them in names, thus increasing readability and searchability on Tableau dashboard.	AKA_Name / 8722 DBA_Name / 17111
11	Text transform value.replace("LLC","")	Required to remove business designations as users do not want to see them in names, thus increasing readability and searchability on Tableau dashboard.	AKA_Name / 1003 DBA_Name / 2771
12	Text transform value.replace("/\s+/,"") also known as Collapse consecutive white spaces	Required operation for improved clustering & querying. Eg Select * from TABLE where DBA_Name = "Starbucks Coffee" will give results not having "Starbucks Coffee " or "Starbucks Coffee".	AKA_Name / 1512 DBA_Name / 559 Facility Type / 99 Address / 821 Violations / 84606
13	Text transform value.replace("&,"AND")	Required to replace special characters increasing readability and searchability on Tableau dashboard.	AKA_Name / 12108 DBA_Name / 13067 Facility Type / 3
14	Text transform value.replace("@,"AT")	Required to replace special characters increasing readability and searchability on Tableau dashboard.	AKA_Name / 378 DBA_Name / 1105
15	Text transform value.replace("?,"")	Required to remove special characters providing better readability on Tableau dashboard.	AKA_Name / 21 DBA_Name / 21
16	Text transform value.replace("!,"")	Required to remove special characters providing better readability on Tableau dashboard.	AKA_Name / 123 DBA_Name / 131
17	Text transform value.replace("- ,"-")	Required to remove space between special characters to provide uniformity in names and increase readability on Tableau dashboard.	AKA_Name / 385 DBA_Name / 279
18	Text transform value.replace("+,"+ ")	Required to add a space between special characters to provide uniformity in names and increase readability on Tableau dashboard.	AKA_Name / 116 DBA_Name / 85
19	Text transform value.replace("$,"DOLLAR")	Required to replace special characters increasing readability on	DBA_Name / 9

		Tableau dashboard and also helped in merging similar names like “Dollar Store” and “$ store” into single entity.	
20	Text transform value.replace(" TH,"TH")	Required to remove space between numbers and their ordinal indicators (suffixes) to increase readability and searchability on Tableau dashboard.	Address / 29
21	Text transform value.replace(" ST,"ST")	Required to remove space between numbers and their ordinal indicators (suffixes) to increase readability and searchability on Tableau dashboard.	Address / 16
22	Text transform value.replace(/(\(.+\))/,"")	Required to remove special characters increasing readability and searchability on Tableau dashboard.	Address / 3223
23	Text transform value.toTitlecase	Required transformation of all strings to TitleCase increasing readability on Tableau dashboard.	City / 153651 AKA_Name / 151201 DBA_Name / 153749 Facility Type / 146729 Address / 9488 Inspection Type / 153808 Results / 153810 Violations / 123012

Python Data Cleaning Performed

Data Cleaning Step #	Data Cleaning Step Description	Data Cleaning Rationale (Requirements for U1)	Data Changes Summary - Columns affected / # of Cells affected
1	Read data cleaned in OR into pandas DataFrame	Perform more advanced data cleaning and manipulation using the pandas Python library.	All / 153810
2	Generate column for *Facility ID* from unique combinations of *(DBA Name, Address)*	Assigns unique IDs for each physical location of businesses that have many locations with the same name.	Facility ID / 153810
3	Parse free text from *Violations* column into serialized JSON objects *Violation JSONs*, and arrays of integers *Violation IDs*	Provides more robust access to violation information for each record.	Violation JSONs / 123012 Violation IDs / 123012
4	Identify records with missing lat/long geocoordinates and attempt to lookup via Web API	Provides more robust dataset for map-based visualizations.	Latitude / 129 Longitude / 129 Location / 129
5	Output the cleaned/updated flat file in csv format for use cases that require only steps 1-4	Provides a light-weight solution for certain use cases that don’t require a database.	All / 153810
6	Create a separate Facility table with unique *Facility ID* as primary key	Required to generate a relational database from the input flat file, to support use cases which require it.	Facility ID / 28153 DBA Name / 28153 AKA Name / 28153

			Facility Type/ 28153 Risk / 28153 Address / 28153 City / 28153 State / 28153 Zip / 28153 Latitude / 28153 Longitude / 28153 Location / 28153
7	Create a separate License table with unique *License ID* as primary key and *Facility ID* as foreign key	Required to generate a relational database from the input flat file, to support use cases which require it.	License # / 32850 Facility ID / 32850
8	Create a separate Inspection table with unique *Inspection ID* as primary key and *License #* as foreign key	Required to generate a relational database from the input flat file, to support use cases which require it.	Inspection ID / 153810 License # / 153810 Inspection Date / 153810 Inspection Type / 153810 Results / 153810 Violations / 153810 Violation JSONs / 153810 Violation IDs / 153810
9	Create a separate non-relational Violation lookup table with unique *Violation ID* as primary key	Required to provide a mapping from *Violation ID* to violation description.	ID / 46 Description / 46
10	Create a separate Inspection + Violation table with unique *(Inspection ID, Violation ID)* as primary key	Required to provide an easy lookup of comments for each inspection, for use cases that do not want the overhead of parsing the violation JSON arrays as part of data analysis (at the expense of larger file size).	Inspection ID / 599452 Violation ID / 599452 Comments / 599452
11	Output relational database in SQLite format	Required for use cases that need a relational database solution.	[TableName] / [# Rows] Facility / 28153 License / 32850 Inspection / 153810 Violation / 46 Inspection Violation / 599452
12	Output the full-sized flat file in csv format with each violation for each inspection as a unique record	Required for use cases that do not need a database solution but do need violations broken out into separate records with associated comments.	All / 599452
13	Output single database in SQLite format	Required for use cases that need a database solution with unique records for every	[TableName] / [# Rows]

	inspection/violation pair, but do not need a relational database.	Food-Inspections_Violations / 599452

Documentation of Data Quality Improvements

- ☐ *Demonstrate that data quality has been improved, e.g., by devising ICV queries and showing the difference between “before” and “after” (cleaning). (10 P)*

Query Number	$Q_U(D)$ and $Q_U(D')$	$Q_U(D)$ Number of results	$Q_U(D')$ Number of results	Notes

1	`SELECT COUNT(DISTINCT DBA_Name)`
	`FROM Original_Food_Inspections`
	`WHERE DBA_Name LIKE '%Starbucks%'`
	
	`SELECT COUNT(DISTINCT DBA_Name)`
	`FROM C_Food_Inspections_Facility`
	`WHERE DBA_Name LIKE '%Starbucks%'`
		150	1	Clustered similar DBA_Name and AKA_Names

2	`SELECT COUNT(DISTINCT City) FROM`
	`Original_Food_Inspections`
	`WHERE City LIKE 'CHI%'`
	`OR City LIKE 'CCHI%'`
	
	`SELECT COUNT(DISTINCT City) FROM`
	`C_Food_Inspections_Facility`
	`WHERE City LIKE 'CHI%'`
	`OR City LIKE 'CCHI%'`
		5	2	Remove spelling errors of Chicago

3	`SELECT COUNT(DISTINCT DBA_Name)`
	`FROM Original_Food_Inspections`
	
	`SELECT COUNT(DISTINCT DBA_Name)`
	`FROM C_Food_Inspections_Facility`
		24418	20762	Cleaning and clustering

4	`SELECT COUNT(DISTINCT AKA_Name)`
	`FROM Original_Food_Inspections`
	
	`SELECT COUNT(DISTINCT AKA_Name)`
	`FROM C_Food_Inspections_Facility`
		23343	20291	Cleaning and clustering

5	`SELECT COUNT(DISTINCT City) FROM`
	`Original_Food_Inspections`
	
	`SELECT COUNT(DISTINCT City) FROM`
	`C_Food_Inspections_Facility`
		52	45	Cleaning and clustering

6	`SELECT COUNT(DISTINCT`
	`Facility_Type) FROM`
	`Original_Food_Inspections`
	
	`SELECT COUNT(DISTINCT`
	`Facility_Type) FROM`
	`C_Food_Inspections_Facility`
		400	305	Cleaning and clustering

7	`SELECT COUNT(DISTINCT Address)`
	`FROM Original_Food_Inspections`
		16867	16813	Cleaning and clustering

	SELECT COUNT(DISTINCT Address)
	FROM C_Food_Inspections_Facility			

8	SELECT COUNT(DISTINCT Inspection_Type) FROM Original_Food_Inspections
	
	SELECT COUNT(DISTINCT Inspection_Type) FROM C_Food_Inspections_Facility
		96	88	Cleaning and clustering

9	SELECT COUNT(*) FROM Original_Food_Inspections
	WHERE Location IS NULL
	
	SELECT COUNT(*) FROM C_Food_Inspections_Facility
	WHERE Location IS NULL
		544	295	Fixed NULL locations through Geolookup in Python scripts

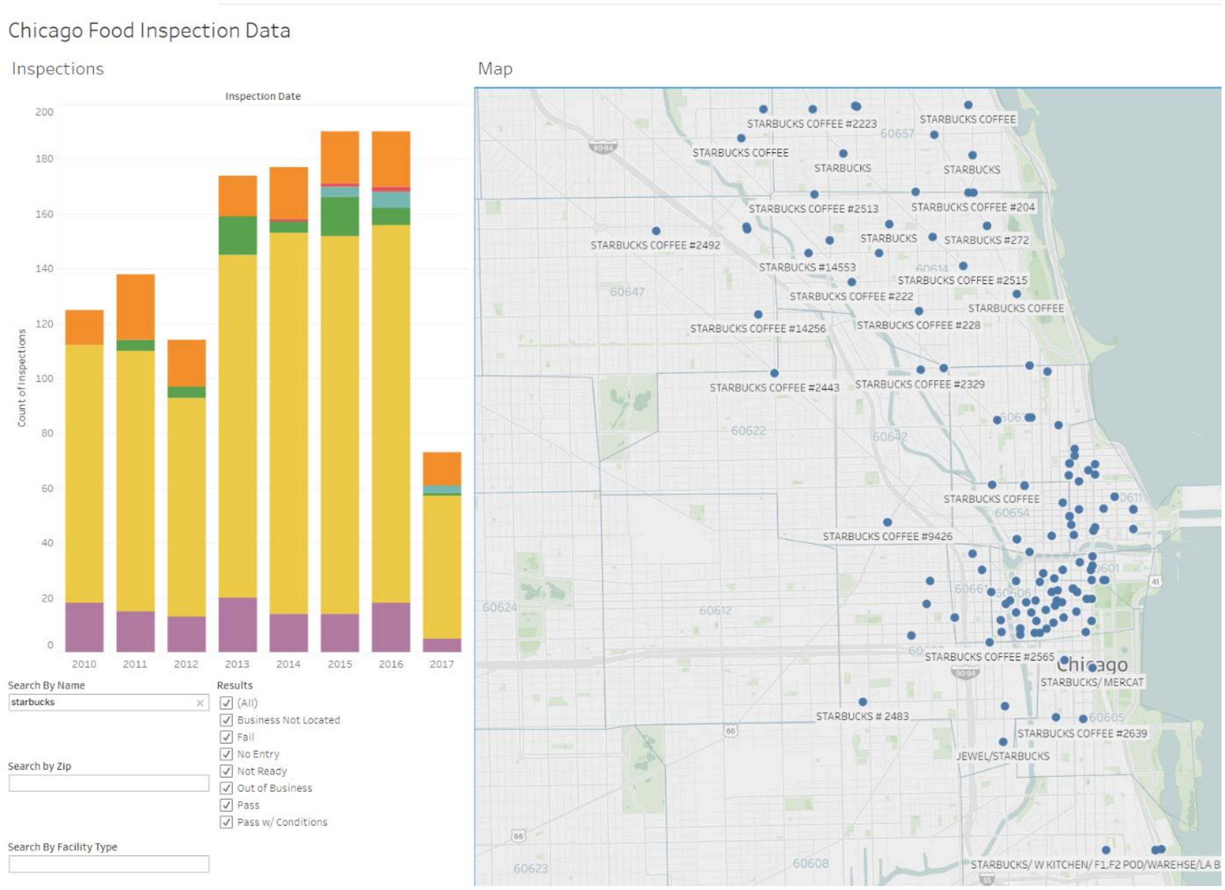
10	SELECT COUNT(*) FROM Original_Food_Inspections WHERE DBA_NAME LIKE '%.%' OR DBA_NAME LIKE '%,%' OR DBA_NAME LIKE '%ANNEX%' OR DBA_NAME LIKE '%INC%' OR DBA_NAME LIKE '%&%' OR DBA_NAME LIKE '%@%' SELECT COUNT(*) FROM C_Food_Inspections_Facility WHERE DBA_NAME LIKE '%.%' OR DBA_NAME LIKE '%,%' OR DBA_NAME LIKE '%ANNEX%' OR DBA_NAME LIKE '%INC%' OR DBA_NAME LIKE '%&%' OR DBA_NAME LIKE '%@%'	35190	0	Clean text for readability

11	SELECT count(DISTINCT DBA_Name)
	FROM Original_Food_Inspections
	
	SELECT count(DISTINCT Facility_ID)
	FROM C_Food_Inspections_Facility
		24418	26873	Disambiguate DBA_Name by introducing Facility_ID which is a composite of DBA_Name and Address

Realization of Use Case U1

We built out a rough prototype of the use case U$_1$ as a Tableau dashboard to guide some of our thinking around the data cleaning requirements for the use case. Some screenshots are shown below and the dashboard can be accessed at: https://public.tableau.com/app/profile/vamshi.kommineni/viz/ChicagoFoodInspections-Originaldataset/ChicagoFoodInspectionData?publish=yes

For example, a search by a user for "Starbucks" returns a lot of different names for Starbucks which is something we can remedy through data cleaning:



Chicago Food Inspection Data

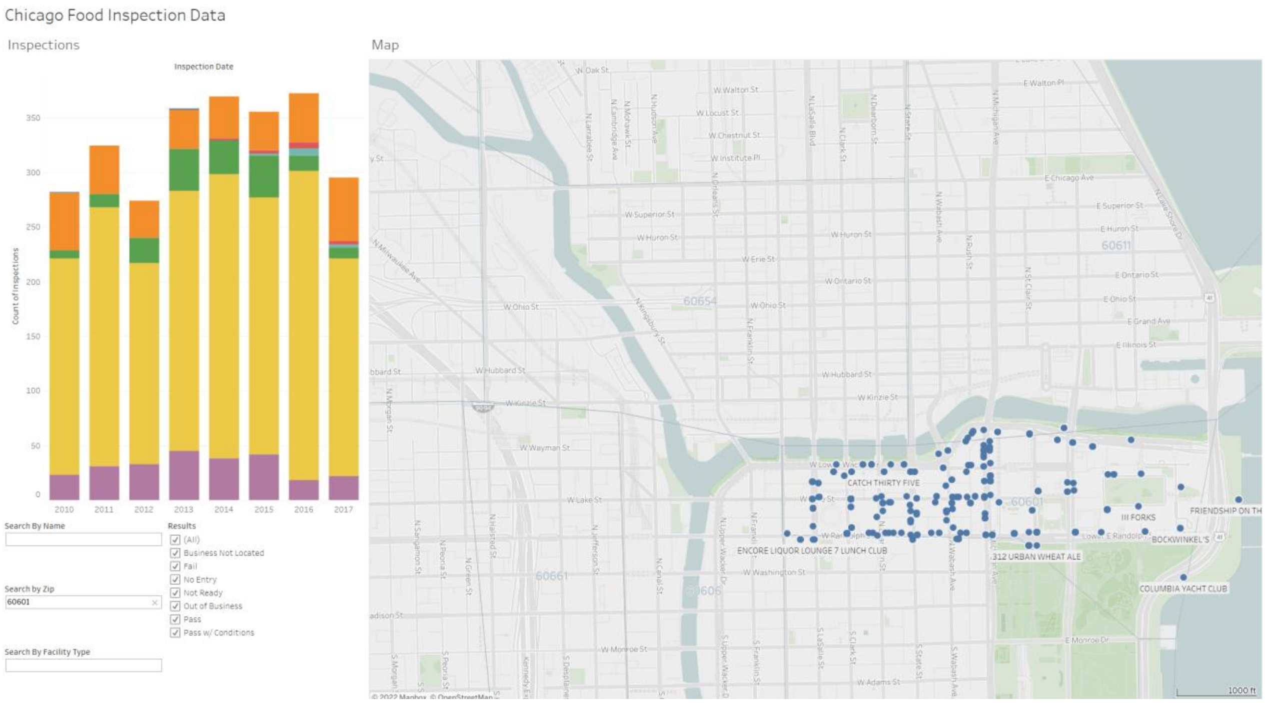
Search By Name: starbucks

Results
☑ (All)
☑ Business Not Located
☑ Fail
☑ No Entry
☑ Not Ready
☑ Out of Business
☑ Pass
☑ Pass w/ Conditions

Search by Zip

Search By Facility Type

A user can perform other searches, for example using a zip code to filter their search:



Search By Name

Search by Zip
60601

Search By Facility Type

Results
☑ (All)
☑ Business Not Located
☑ Fail
☑ No Entry
☑ Not Ready
☑ Out of Business
☑ Pass
☑ Pass w/ Conditions

Summary and Conclusions

- *☐ Conclusions & Summary (10 P).*
  - *Please provide a concise summary and conclusions of your project, including lessons learned.*
  - *If you haven't done so earlier in the report, you should also summarize the contributions of each team member here (for teams with >= 2 members).*

Findings, Problems and Lessons Learned

Overall, this was a very positive and interesting learning experience to supplement the lectures, fireside chats and assignments. Doing this as group work allowed us to share our diverse perspectives and bring different skillsets to the table. Group composition is always critical for a group project, and we were lucky to have a good group of professional fellow students that were able to execute independently as well as meet to discuss, collaborate and agree on next steps.

Some of the findings through our project experience:

- OpenRefine documentation was lacking. In particular, the "-t merge" option was very helpful to produce a usable final inner workflow from many hundreds of data cleaning steps but wasn't documented (we found it through a TA's Campuswire post).
- OpenRefine doesn't cleanly log rows affected in the operation history or have any summaries to this effect. Rajan, our team member who worked on OpenRefine had to manually tally up the cells and columns effected as shown in the table above.

Problems that we encountered include:

- Problem (minor): Stability of OpenRefine. Had occasional crashes with this larger dataset.

- Problem: We expected License numbers to be more useful, i.e. closer to a unique id for a business. Instead, there were a number of data cleanliness issues like license number 0. License numbers were also incorrect when we tried to look up a sample set on the City of Chicago data website
- Problem: Looking up geo-coordinates. We had the idea of filling in missing lat/long coordinates with a geo-lookup from address. But we found that a number of these had data quality problems with the address as well.

Lessons learned from the project:

- Profiling and building a plan for data cleaning was surprisingly important, even for a smaller data set and project like this. Having a solid plan as part of the phase 1 proposal allowed us to make quicker progress towards the completion of Phase 2
- Keeping the end goal/use case in sight was also very important. Building the very simple/rough POC app/dashboard in Tableau allowed us to think more deeply about what kinds of data cleaning steps were useful. For example, we thought of the geo-lookup cleaning while trying to plan out the use case in a more detailed way.

Next Steps

There are other data transformation steps that we could take if we wanted to follow through and build a full application as described in our use case. In terms of building out the use case/application, we could definitely do a lot more work on the Tableau dashboard or potentially build a web application.

Team Member Contributions

- We split up the work amongst ourselves, and met regularly over the ~7 weeks that we worked on it.
- To enable working in isolation, we split up the work as follows among our team:
  - **Rajan Patel (rajansp2@)**: All OpenRefine based data cleaning, investigation and creation of workflow diagrams
  - **Eric Ahlgren (ericca2@)**: All Python based data cleaning. Creation of tables to realize the ER diagram for a more database-centric solution rather than just using a flat file.
  - **Vamshidhar Kommineni (komminen@)**: Loading the data into SQL, data quality queries, building the Tableau POC
  - **All**: Profiling the data, building a plan, discussing the ideas and issues over the course of the project, writing the Phase 1 & Phase 2 reports