CS 513: Final Project Report

By: Tao Li (taol4), Wanlin Yang(wanliny2)

1. Introduction and Overview

Farmers markets are an important way for citizens to eat healthy fresh food while supporting small family farms. The US Farmers Market dataset lists markets selling agricultural products. This data contains information about all the registered farmers markets in the United States.Users can access brief information, such as geographical coordinate, state, county, website, etc, and product categories of each farmers market in this dataset.

The original dataset has a total of 8817 rows and 59 columns. Each row of data is an instance of a market, while one market may have multiple rows. Each column is an attribute of a market, including market name, webset, categories of product, season time, last update time, etc. Some attribute data could be missing for a market instance.

This dataset is a thorough statistics of farmers markets all over the country, and valuable information for data analysis. However, the raw data in the csv file is not well formatted and very difficult to be processed by script languages. So our goal of this project is to do data cleaning using OpenRefine. We will also construct a cleaning workflow and check its integrity constraint with SQLite.

2. Initial Dataset Assessment & Use Case Discussion

2.1 Structure and Content of dataset

The latest UC Farmers Market dataset file can be exported from https://www.ams.usda.gov/local-food-directories/farmersmarkets. We can open the csv file directly or import it into OpenRefine.

- The dataset contains **8817** market information with **59** different columns or attributes.
- Each row of the dataset contains information of a market instance.
- Columns can be classified into several categories:

Column/Attribute	Data Type	Category	Description
FMID	integer	Identification	Identifier of each farmers market
MarketName	string	Identification	Name of each market
Website	string	Information	Website of each market
Facebook	string	Information	Facebook of each market
Twitter	string	Information	Twitter of each market
Youtube	string	Information	Youtube of each market
OtherMedia	string	Information	Any other media except those mentioned above
street	string	Location	Street
city	string	Location	City
County	string	Location	County
State	string	Location	State
zip	string	Location	Zip Code
Season1Date	date	Seasonality	Date of Season 1
Season1Time	time	Seasonality	Time of Season 1
Season2Date	date	Seasonality	Date of Season 2

Season2Time	time	Seasonality	Time of Season 2
Season3Date	date	Seasonality	Date of Season 3
Season3Time	time	Seasonality	Time of Season 3
Season4Date	date	Seasonality	Date of Season 4
Season4Time	time	Seasonality	Time of Season 4
x	numeric	Location	The longitude of the market
y	numeric	Location	The latitude of market
Location	string	Location	The description of the market location that the market is actually located in
Credit,WIC,WICcash,SFMNP,SNAP	binary	Payment	Payment Accepted
Organic	binary	Product	Is that organic of the products
Bakedgoods,Cheese,Crafts,Flowers,Eggs,Seafood,Herbs,Vegetables,Honey,Jams,Maple,Meat,Nursery,Nuts,Plants,Poultry,Prepared,Soap,Trees,Wine,Coffee,Beans,Fruits,Grains,Juices,Mushrooms,PetFood,Tofu,WildHarvested	binary	Product	Product available
updateTime	time	System	Update time

2.2 Quality issues

Problem 1: Too many missing data
It's obvious that lots of information is missing in the dataset. For example, few markets have Youtube links, or content of Season 2 to 4.
For future data analysis, those columns are not necessary and we can remove the columns with too many missing data in our cleaning.

Problem 2: Bad formatting for some attributes
Season dates are not in the same format. They mostly follow DD/MM/YYYY format, but some are in Month DD, YYYY format.We can also notice that the data is not well formatted. In the "updateTime" column, some instances have a specific date and time of the last update, but others only have the year.
We should make the time expression into the same format to make the content recognizable by script languages.

Problem 3: Inconsistent name or values
In the "MarketName" column, the same market may have multiple formats of name. For example, the three expressions represent one market: Rochester Downtown Farmers Market, Downtown Rochester Farmers Market, Downtown Rochester Farmers' Market. Clustering the market names will solve the quality issue.

2.3 Use cases

The data file is pretty informative and well organized overall, although there are some unformed cells and missing values. With the raw data, several use cases can be satisfied:

- List available products for each market.
- Given FMID, find the corresponding market and its information.
- How many payment methods are available for each market.

For further data analysis using script languages, the raw data is not enough. Lots of tasks can only be performed after data cleaning:

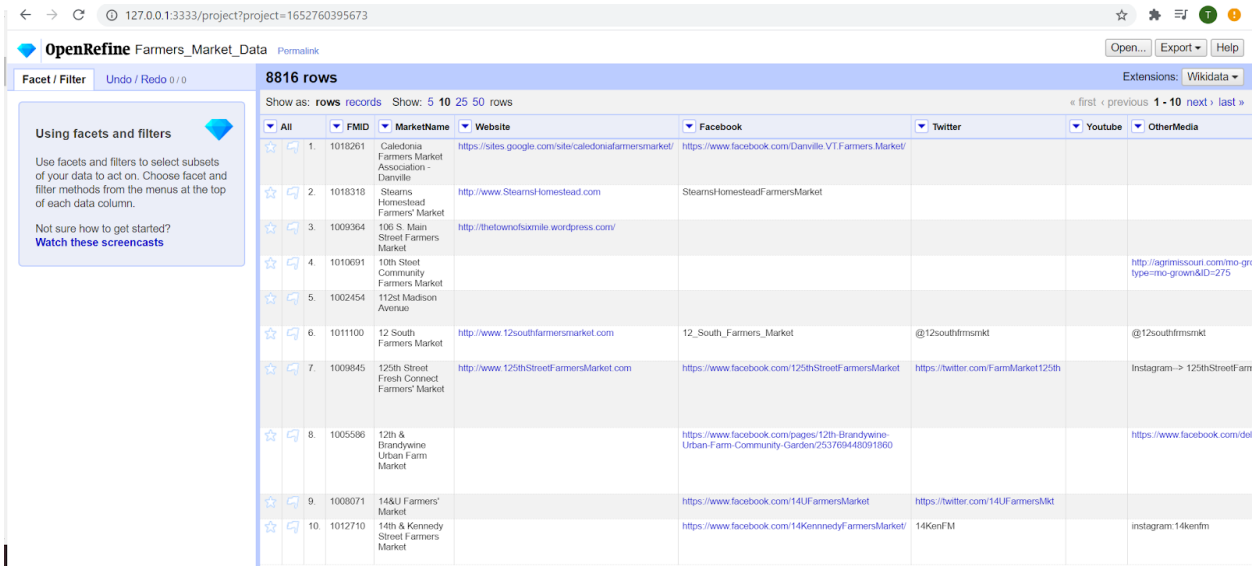
- Locate each Farmers Market on one map using longitude and latitude, which should be well formatted before importing to map.
- Sort Market instances from the oldest to latest based on updateTime. Every date data should be in a standard format.
- List the duration of season1 of markets by start date and end date. The start and end date need to be extracted from one column in raw data.

3. Data Cleaning Process and Details

OpenRefine, (formerly called Google Refine), is one of standalone open source applications which is used for data cleaning, transformation and data wrangling. It is similar to spreadsheet applications (and can work with spreadsheet file formats); however, it behaves more like a database.

In this project, we are going to use OpenRefine to understand, clean and transform the Farmers market datasets to a clean version by cleaning and clustering each of the attributes or columns.

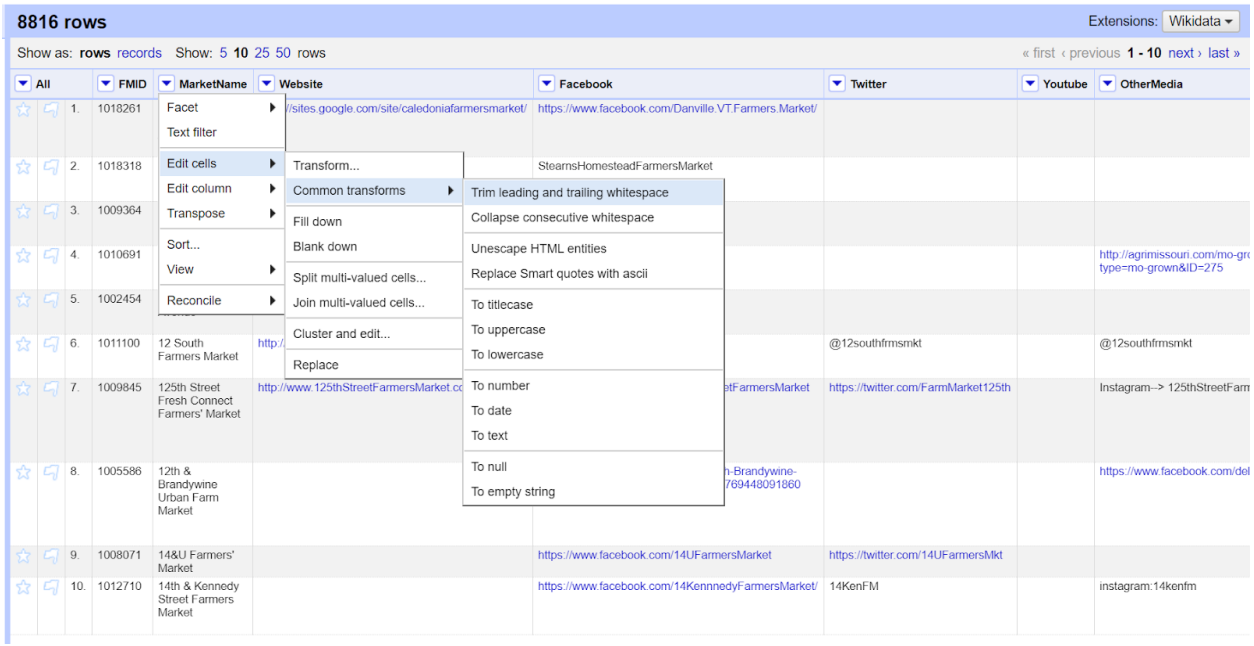
First, let's load the data from local computer and create a project "Farmers_Market_Data", the data will be showing as the following tabular format:



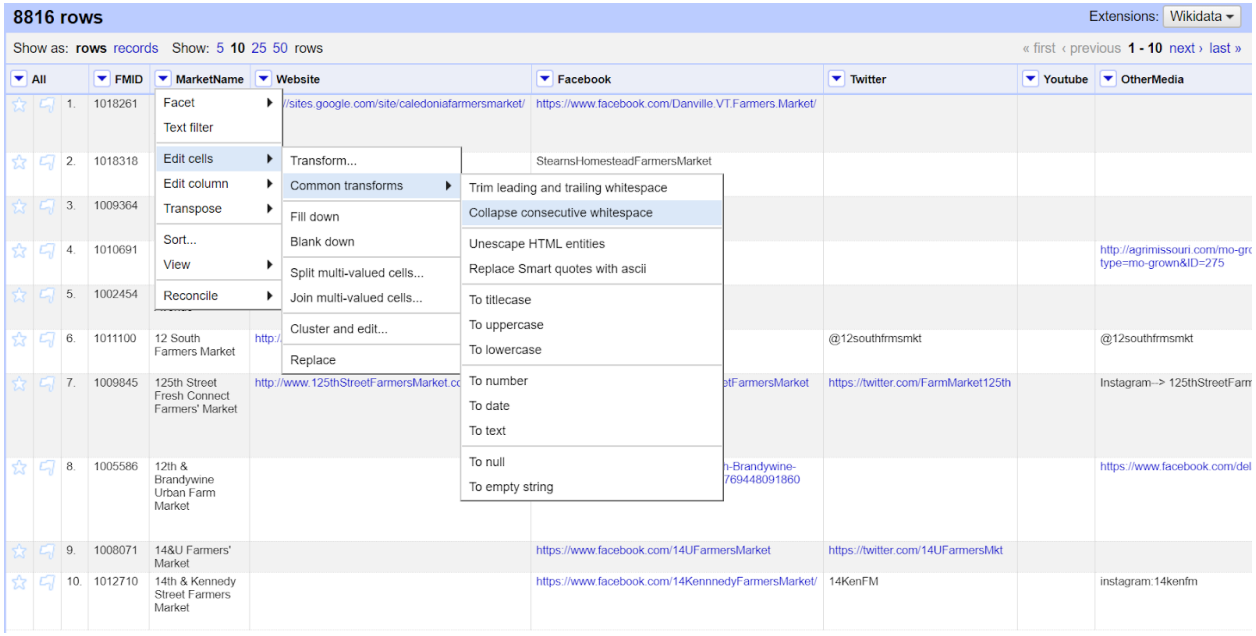
3.1 Identification attributes

MarketName

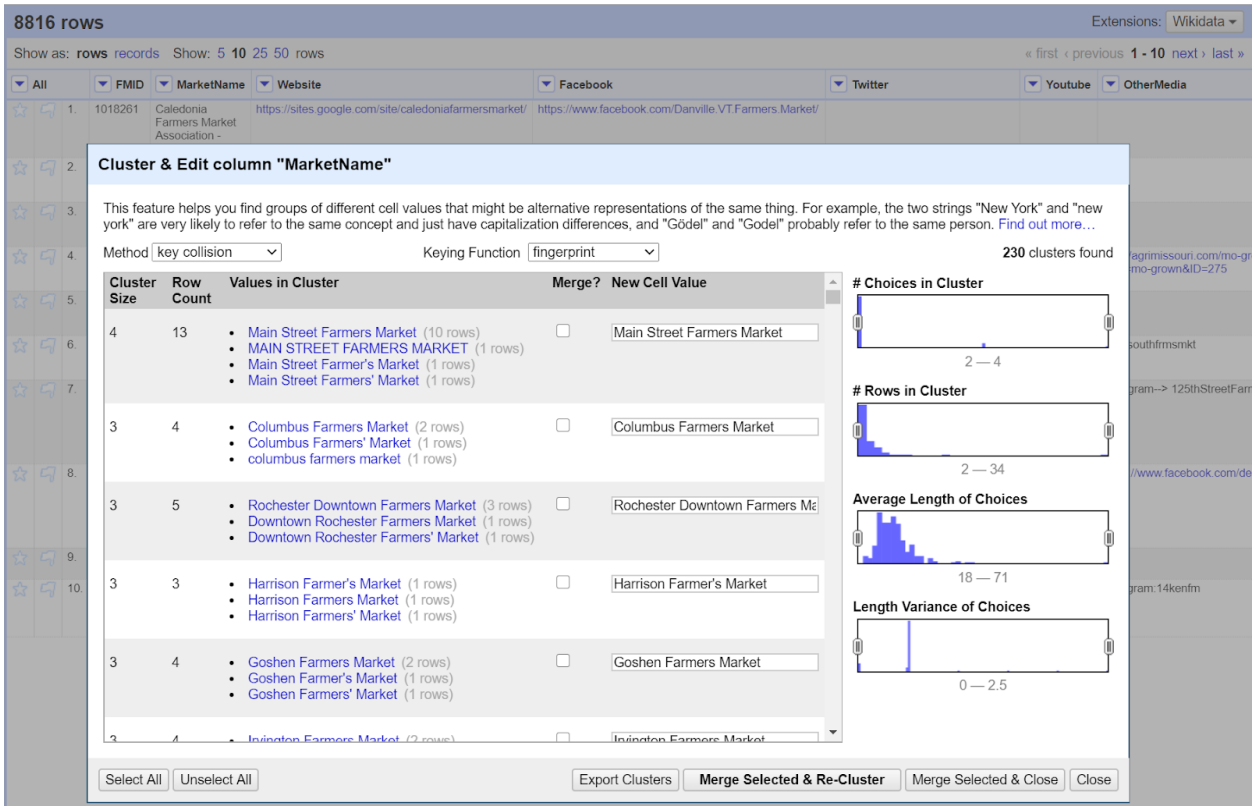
- Trim leading and trailing whitespace



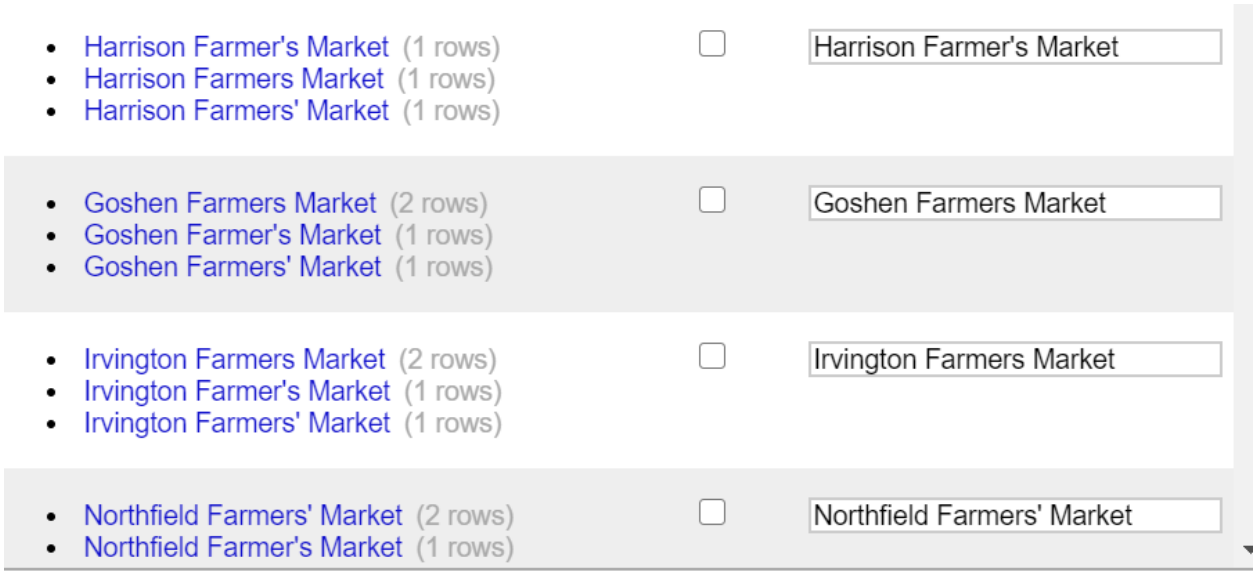
- Collapse consecutive whitespace



- Use text facet and cluster by using **key collision** method and **fingerprint** keying function.

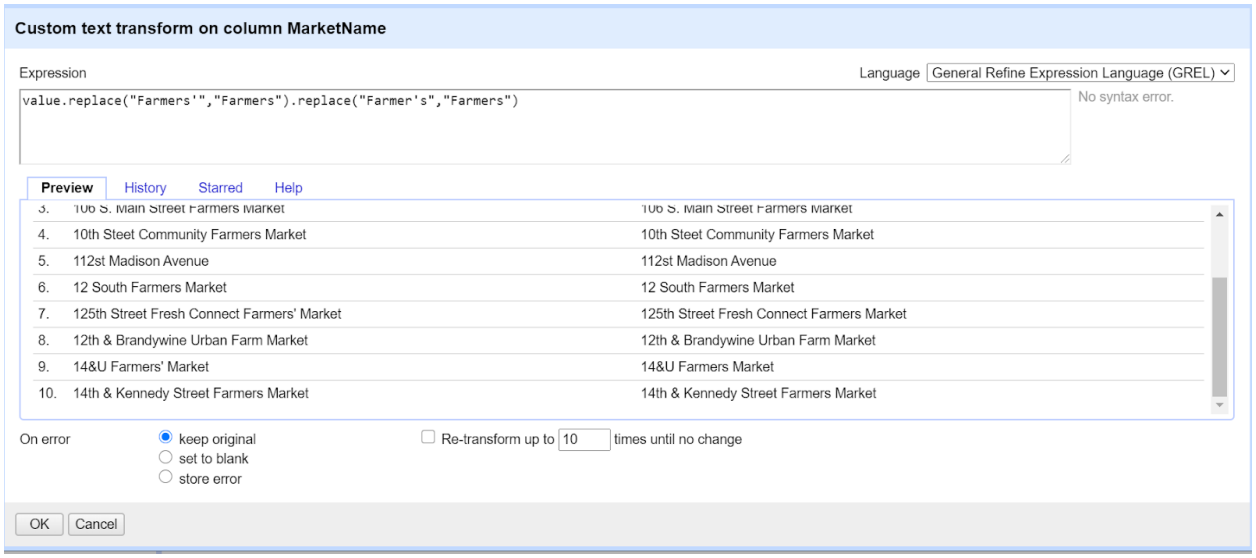


We found one problem that the farmer market name has different ways: Farmer's, Farmers, Farmers' . And it's hard to let the system make auto clustering to choose the consistent names.

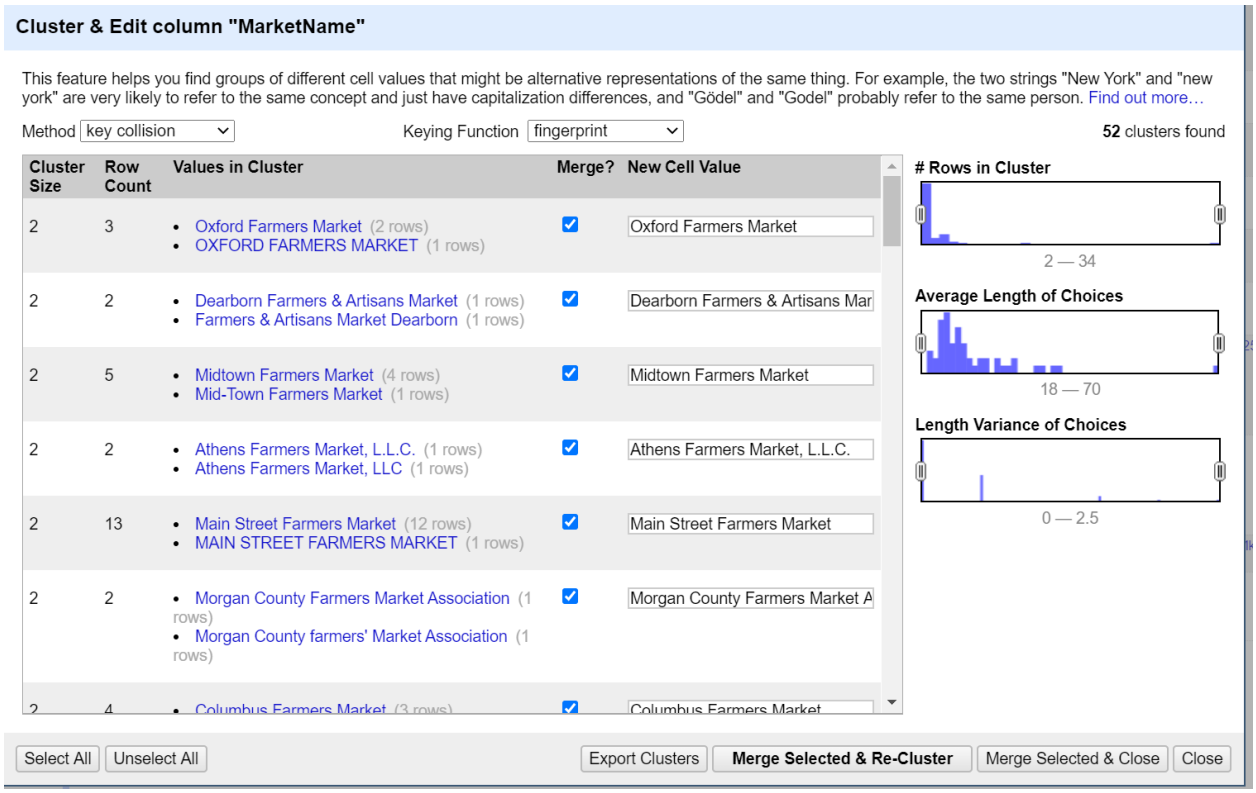


- Custom normalization by using replace characters

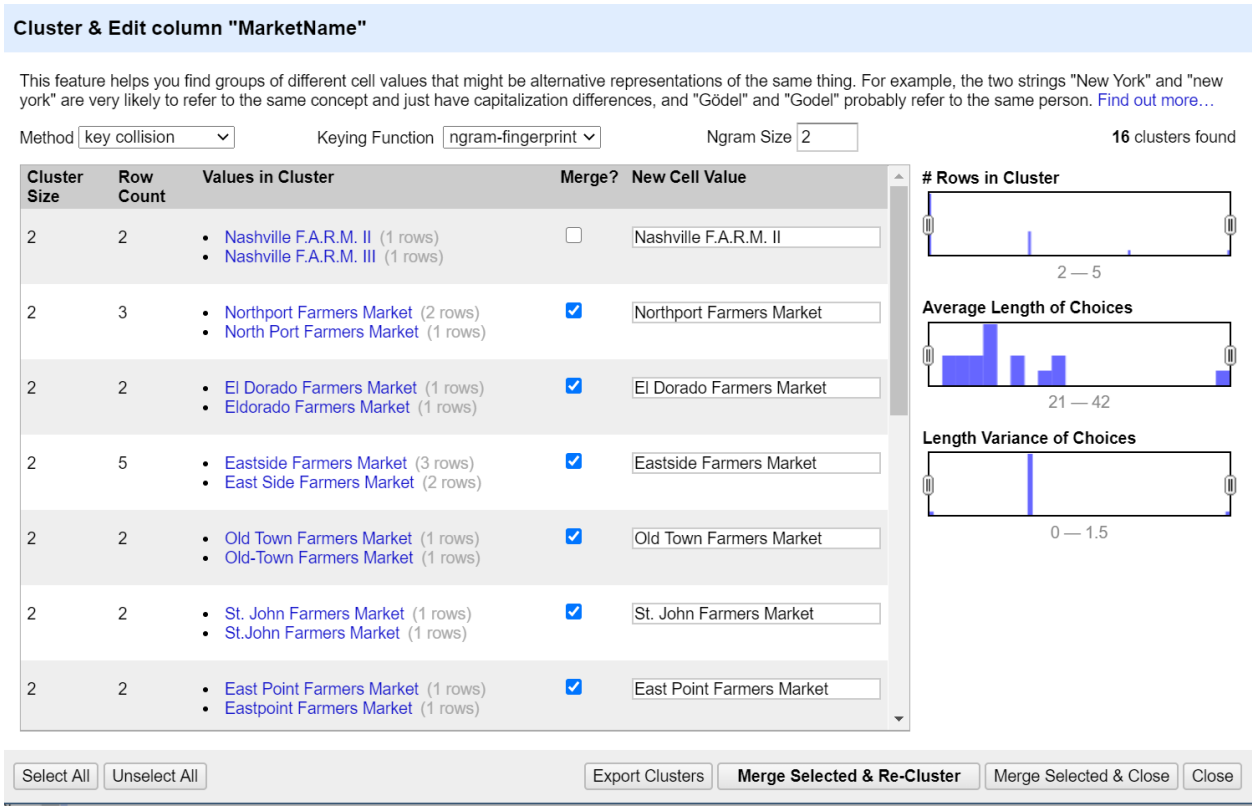
In order to have consistent names, we plan to apply our custom normalizations first before applying the clustering.



- Now, we do the Cluster again by using **key collision** method and **fingerprint** keying function.
  This time, we see less problems, and it's easier to manually review all the clusters and choose the consistent names.



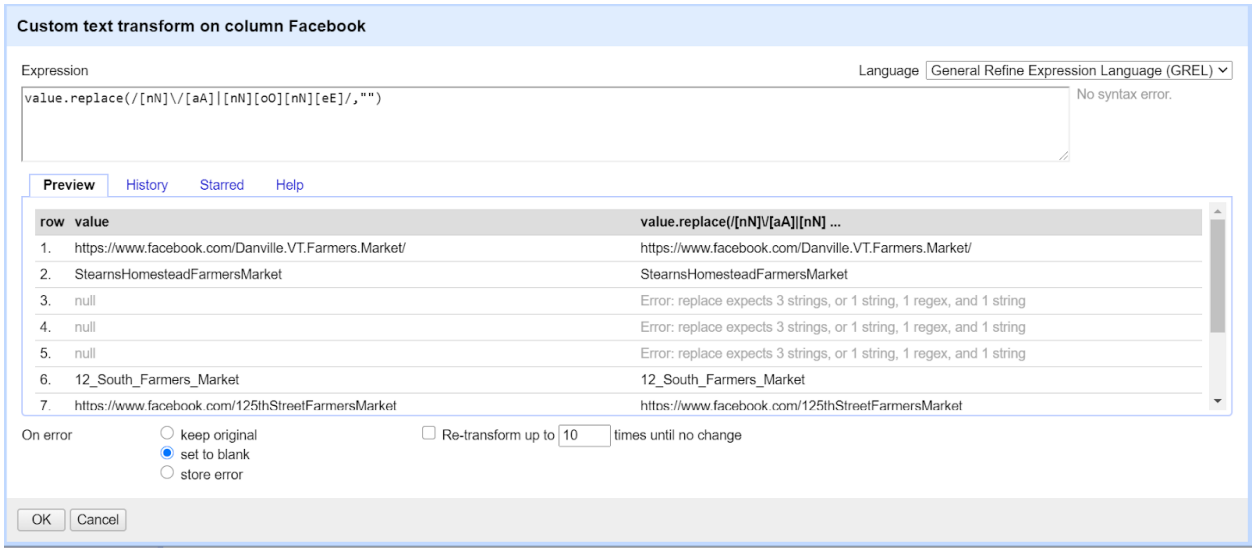
- Use text facet and cluster again by using key collision method and ngram-fingerprint, except those have distinct different names.



3.2 Information attributes

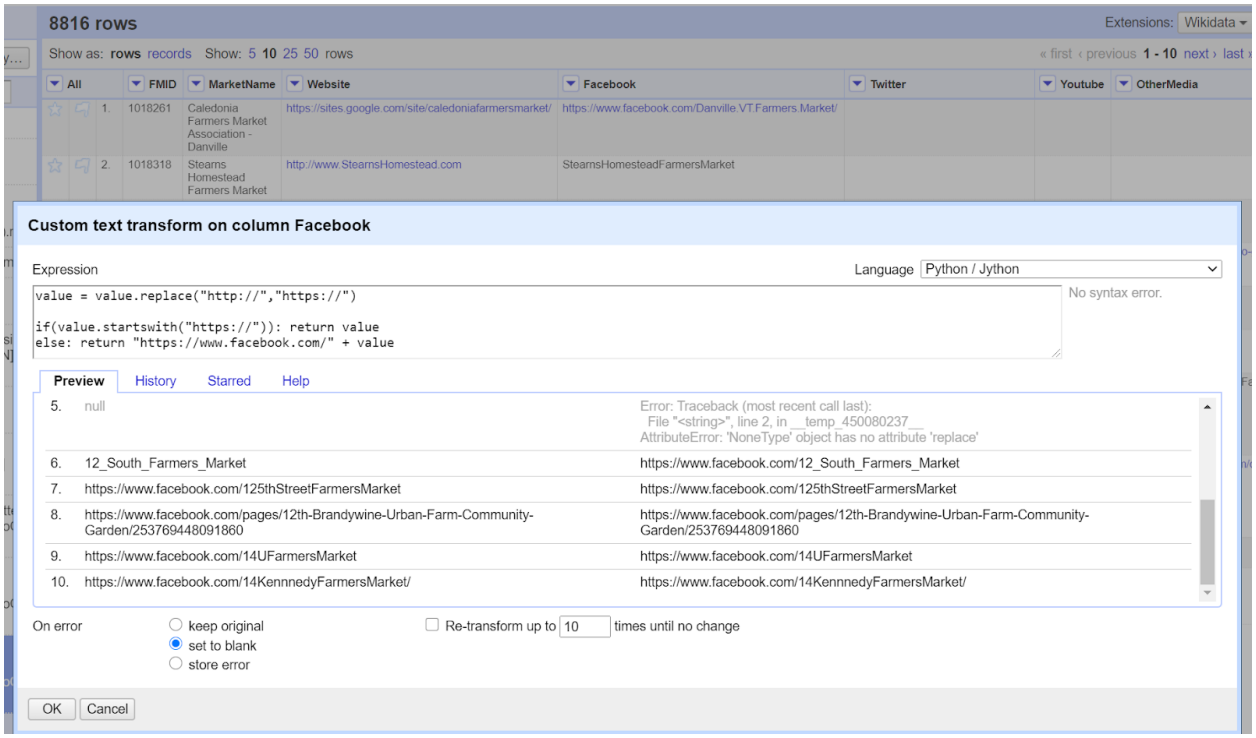
Website|Facebook|Twitter|Youtube|OtherMedia

- Trim leading and trailing whitespace
- By using GREL, change ‘n/a’ or ‘none’ values to blank.



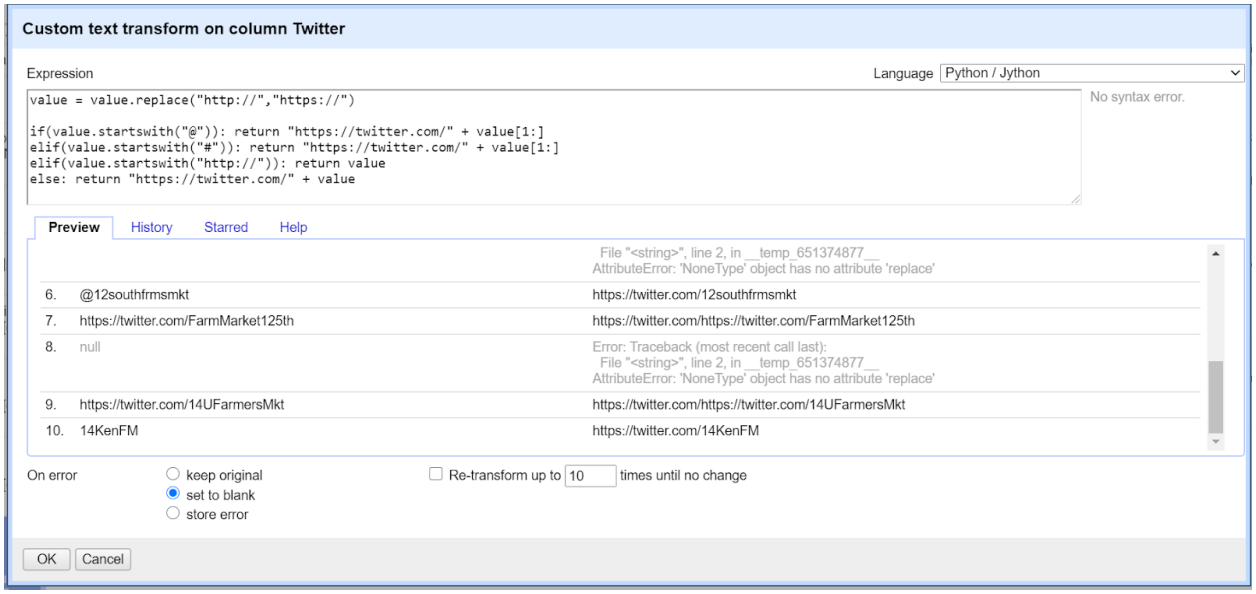
Facebook

Specially for Facebook, we also want to keep the url consistent as we found a lot of the values have the alias without facebook.com prefix. We plan to update that to keep data consistent.



Twitter

Similarly for Twitter, we also need to normalize and unify the format. We found 4 patterns in the Twitter data: @XXX, #XXX, XXX, https://twitter.com/XXX. We applied the python code to unify the formats to https://twitter.com/XXX.



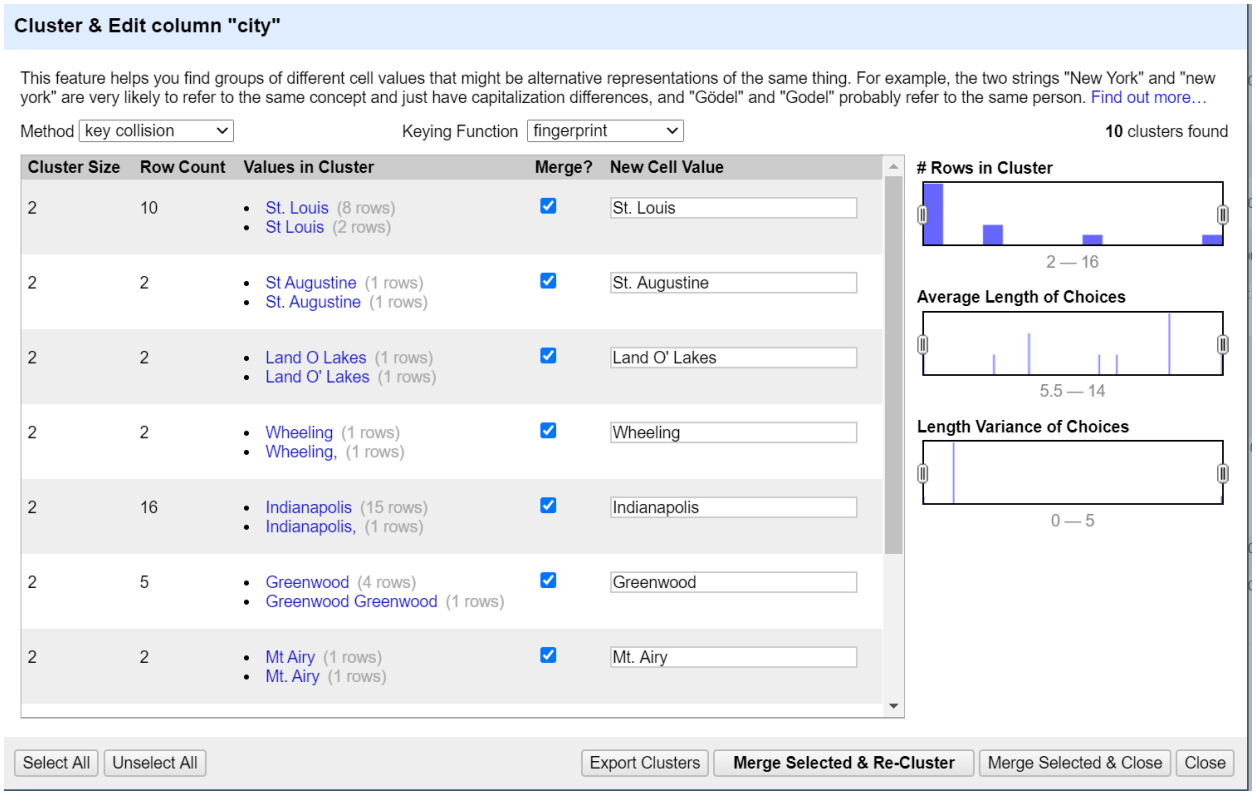
3.3 Location attributes

Street|City|State

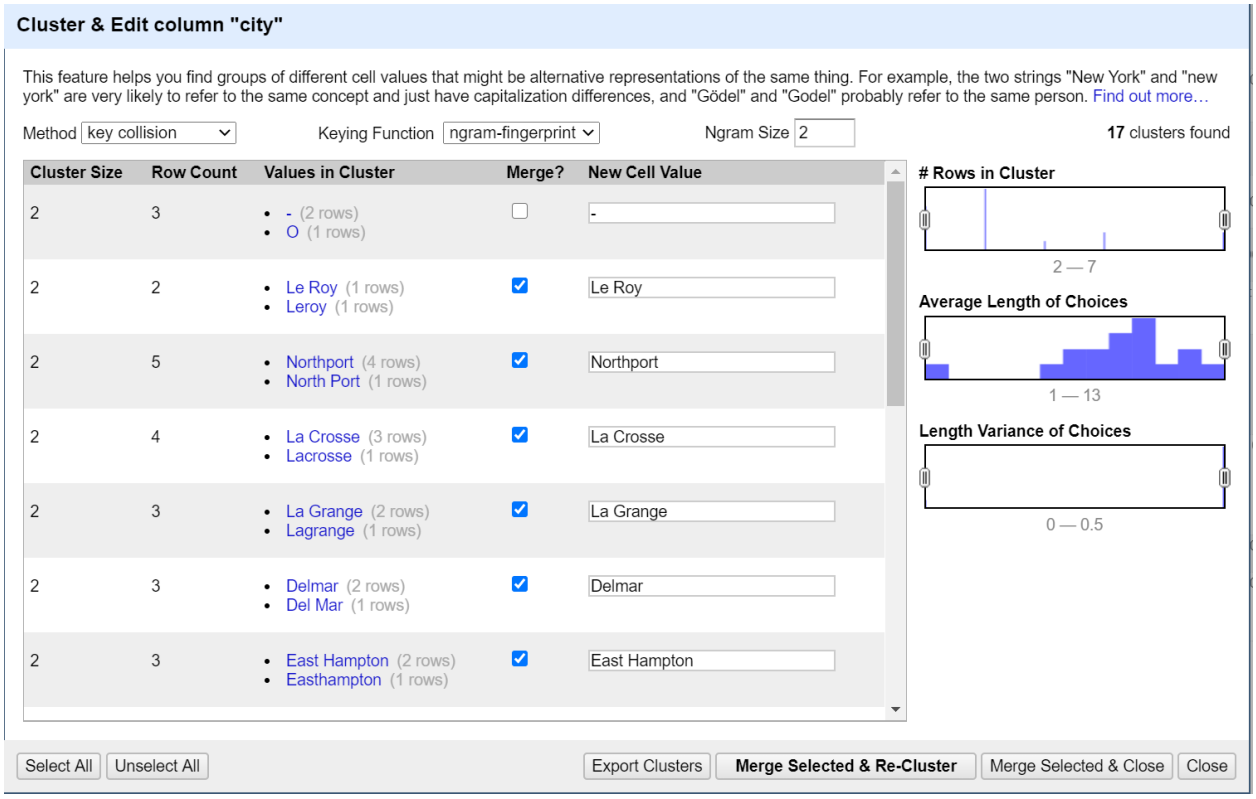
- Trim leading and trailing white space.
- Collapse consecutive white spaces.
- To Title Case.

City:

- Use text facet and cluster by using key collision method and fingerprint keying function.

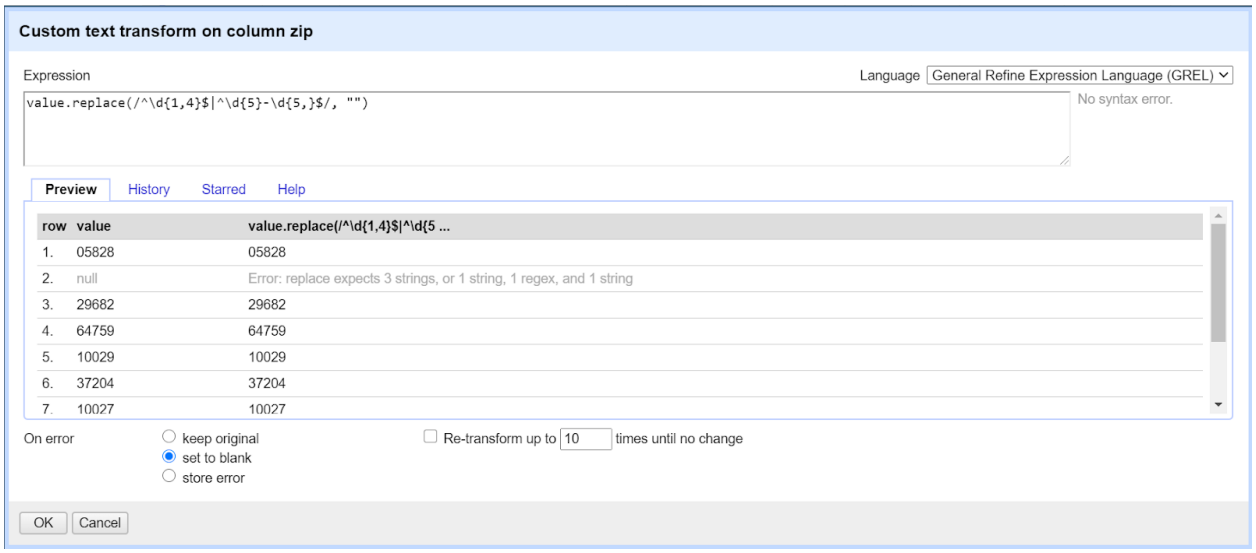


- Use text facet and cluster again by using key collision method and ngram-fingerprint, except those have distinct different names.



Zip

- Trim leading and trailing white space.
- Convert any string values to blank using GREL.
- [GREF] Unify the format. (United States zip code follows NNNNN or NNNNN-NNNN format)



x|y

- Trim leading and trailing white space.

- Convert the values to number

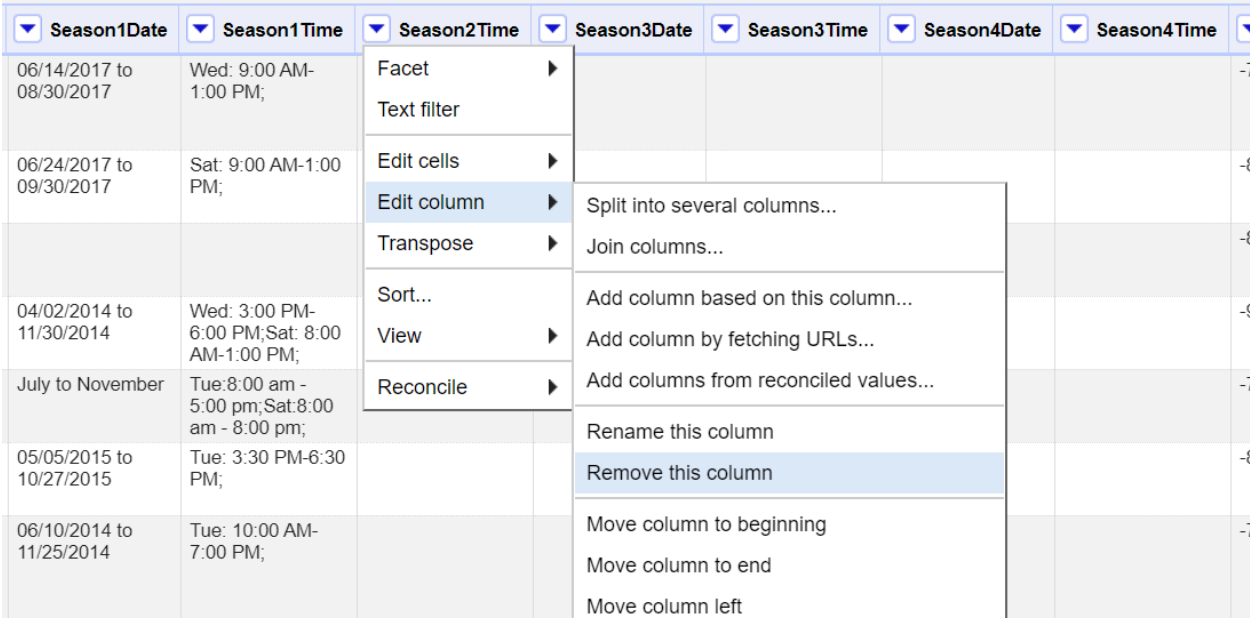
x	y	Location	Credit	WIC	WICcash	SFMNP	SNAP	
-72.140335	44.411037		Y	Y	N	Y	N	Y
-81.73394	41.374802		Y	N	N	Y	N	-
-82.818703	34.804199		Y	N	N	N	N	-
-94.27462	37.495628		Y	N	N	N	N	-

Location:

- Remove the column as not useful for our analysis and too sparse with no useful information.

3.4 Seasonality

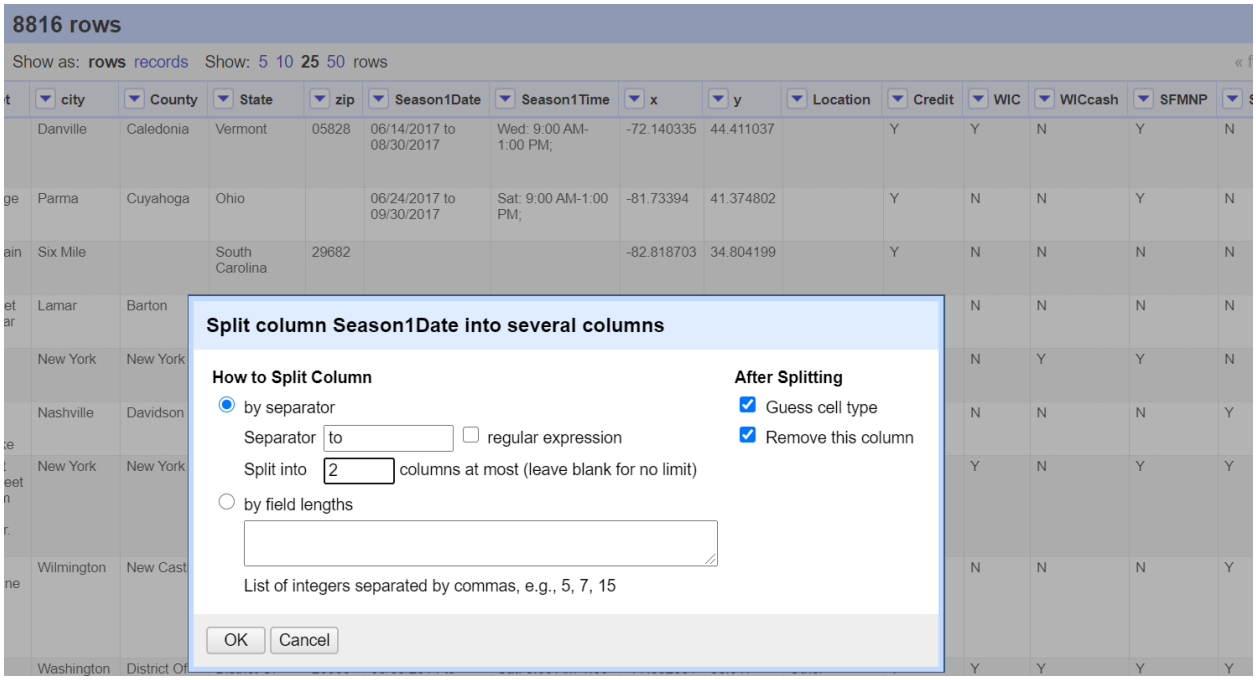
The data provides the attributes Date and Time for Season 1, Season 2, Season 3, Season 4. However, The Season 2|3|4 values are pretty sparse with nearly all empty values there. So, we consider removing these attributes for our analysis.



Season1Date

- Trim leading and trailing white space.
- Collapse consecutive white spaces.

- In order to analyze the date range of start and end for this season. We plan to split the data to 2 columns: Season1StartDate, Season1EndDate

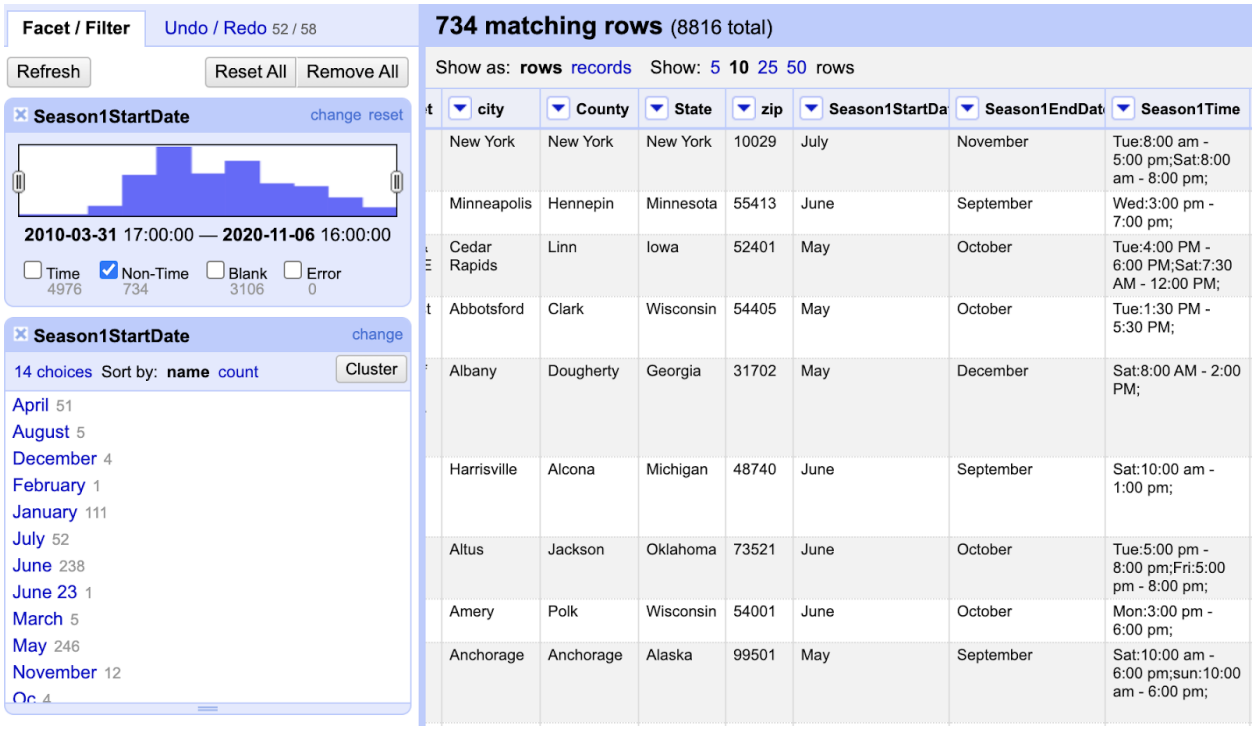


For the new splitted columns:

- Trim leading and trailing white space.
- Apply the transform to date
- Rename the column to Season1StartDate, Season1EndDate

	County	State	zip	Season1StartDa	Season1EndDat	Season1Time	x	y	Location	
	Caledonia	Vermont	05828	2017-06-14T00:00:00Z	2017-08-30T00:00:00Z	Wed: 9:00 AM-1:00 PM;	-72.140335	44.411037		Y
	Cuyahoga	Ohio		2017-06-24T00:00:00Z	2017-09-30T00:00:00Z	Sat: 9:00 AM-1:00 PM;	-81.73394	41.374802		Y
		South Carolina	29682				-82.818703	34.804199		Y
	Barton	Missouri	64759	2014-04-02T00:00:00Z	2014-11-30T00:00:00Z	Wed: 3:00 PM-6:00 PM;Sat: 8:00 AM-1:00 PM;	-94.27462	37.495628		Y
rk	New York	New York	10029	July	November	Tue:8:00 am - 5:00 pm;Sat:8:00 am - 8:00 pm;	-73.949303	40.7939	Private business parking lot	N
le	Davidson	Tennessee	37204	2015-05-05T00:00:00Z	2015-10-27T00:00:00Z	Tue: 3:30 PM-6:30 PM;	-86.79071	36.11837		Y
rk	New York	New York	10027	2014-06-10T00:00:00Z	2014-11-25T00:00:00Z	Tue: 10:00 AM-7:00 PM;	-73.94825	40.808952	Federal/State government building grounds	Y

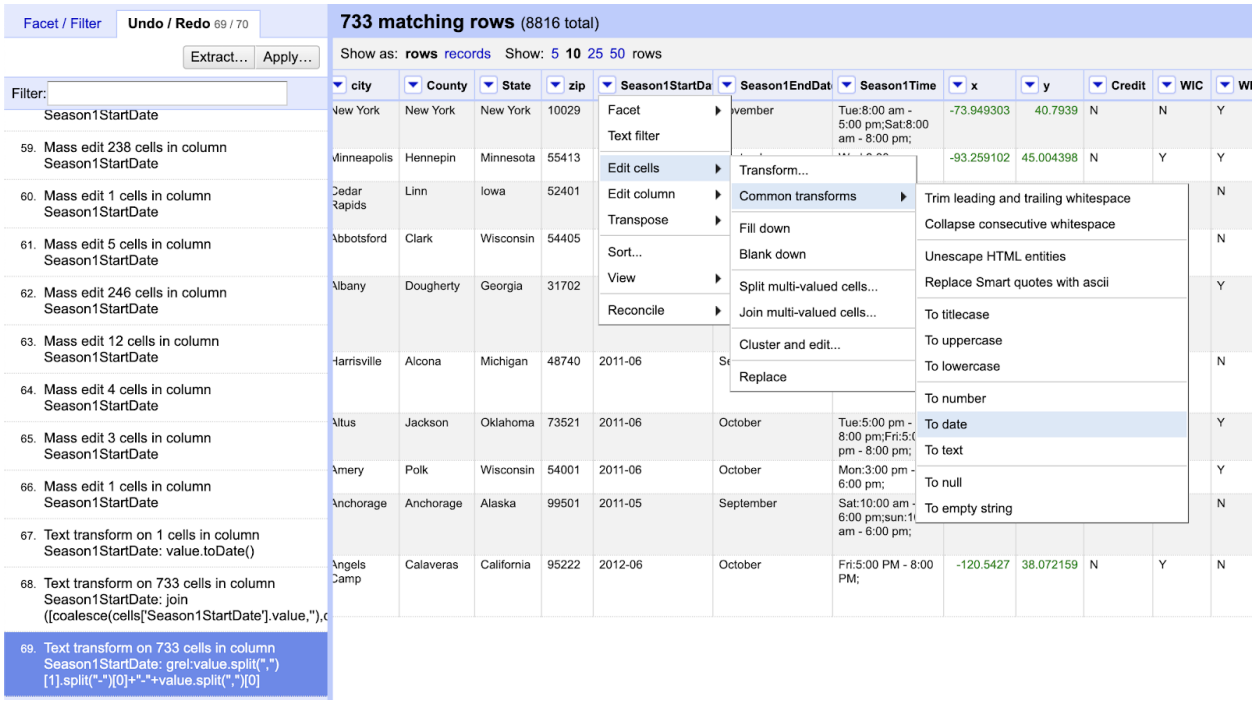
After converting the Season1StartDate and Season1EndDate to date format, we found that formats like “July” and “November” are not convertible, and the year is missing in those cells.



For those cells that are not convertible to date, we assume the year is the same as the year of corresponding updateTime.

For each cell:

- Convert the month name to numeric format.
- Join the updateTime column and only keep the year.
- Convert the column to date.



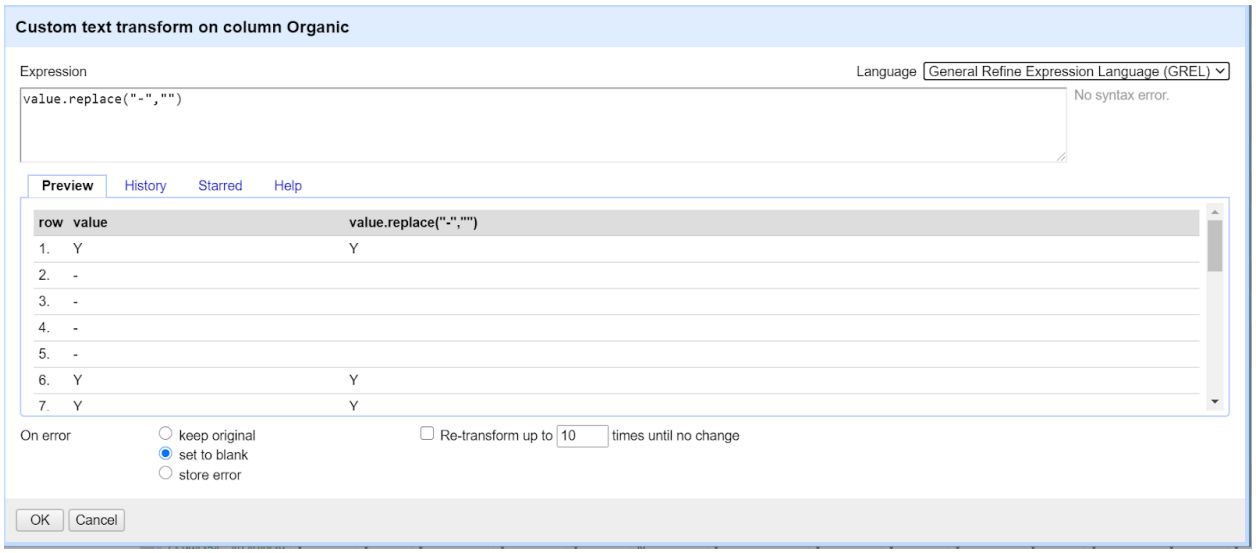
Then we can follow the same process for the Season1EndDate column.

Season1Time

- Trim leading and trailing white space.
- Collapse consecutive white spaces.
- The data contains one or multiple times, each with the weekday, start time and end time. In our analysis, we plan to keep the raw format. If any scenario is needed, we may need to process this attribute for better analysis.

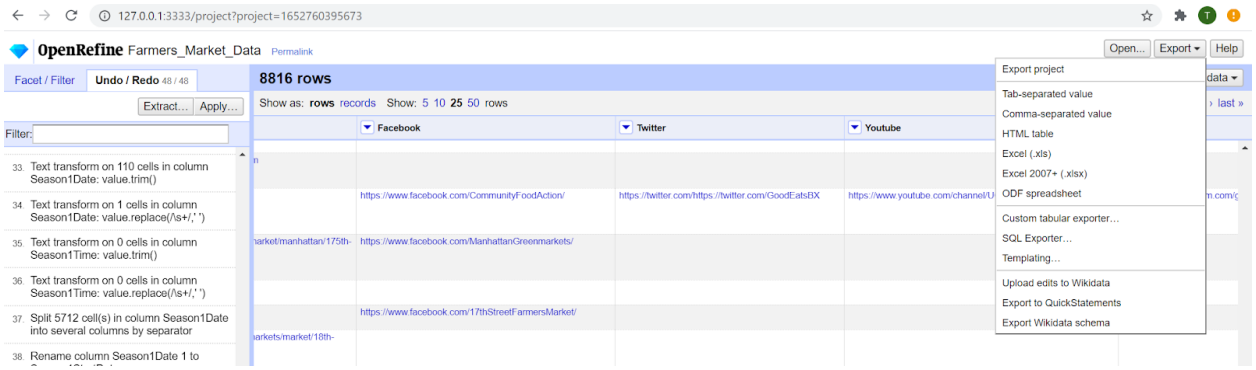
3.5 Payments & Products

Organic:
Better to replace “-” to empty



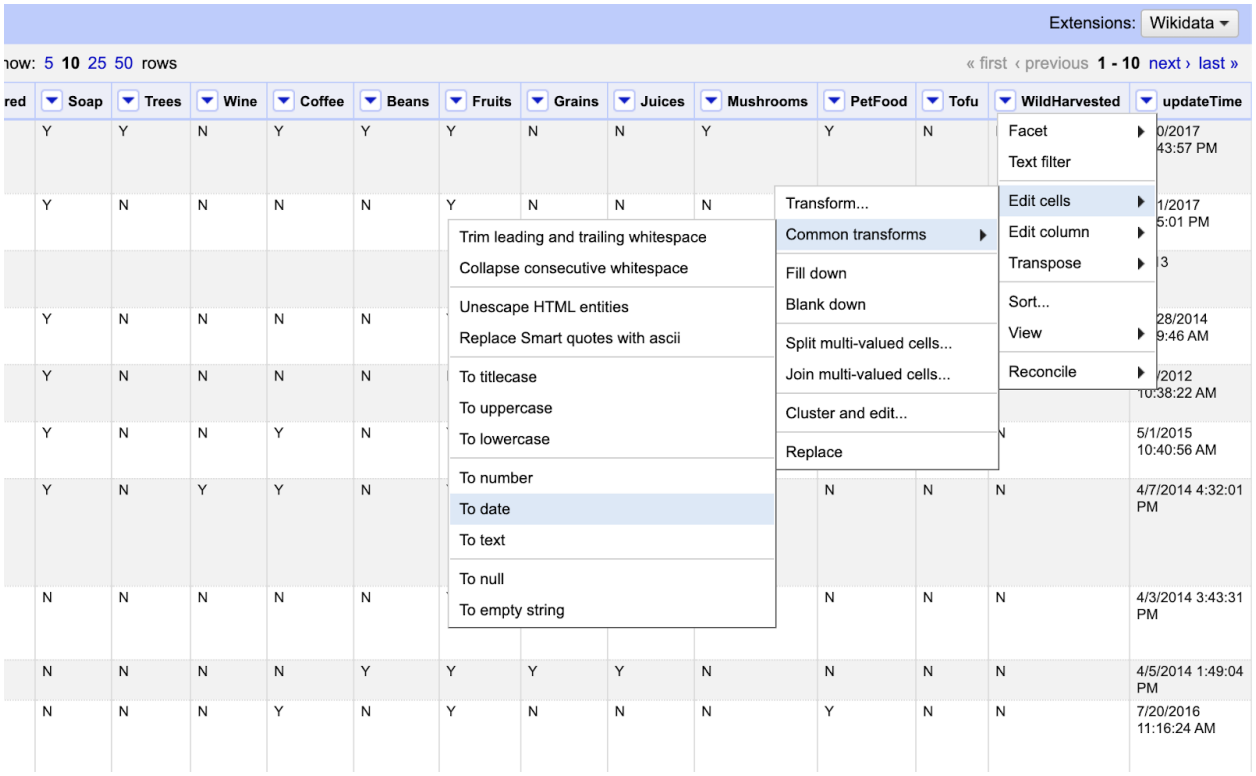
All the rest of values contain binary or boolean values. No need to clean these fields.

After the data cleaning, we can use the export feature to export projects and data results into various formats.

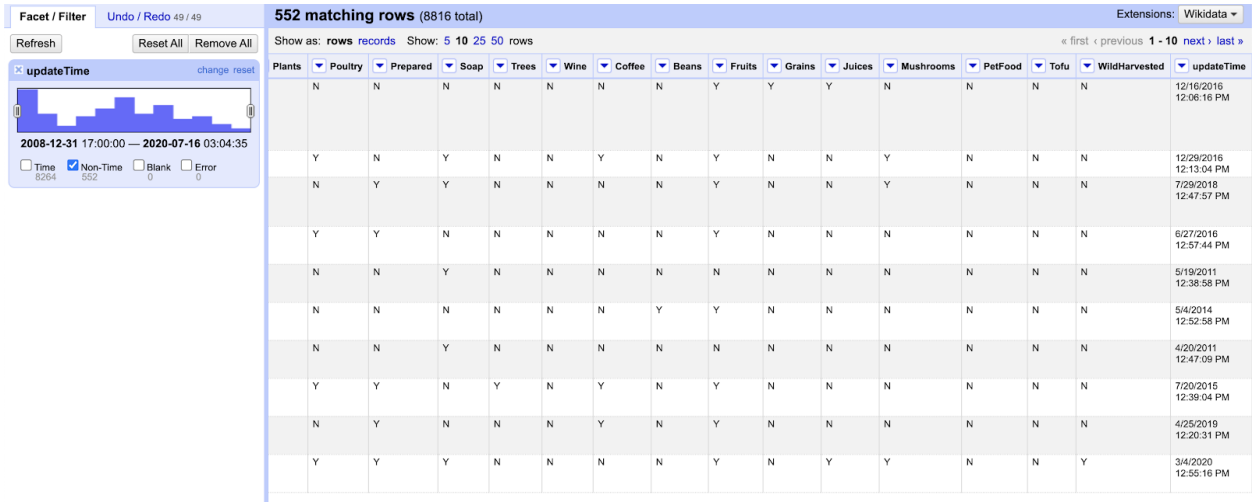


3.6 UpdateTime

The simplest way to format the updateTime column is convert values to date format directly.



But there are several cells that OpenRefine is unable to convert. The problem is that cells with 12PM are not convertible.



For the unconvertible cells:

- Remove PM appendix.

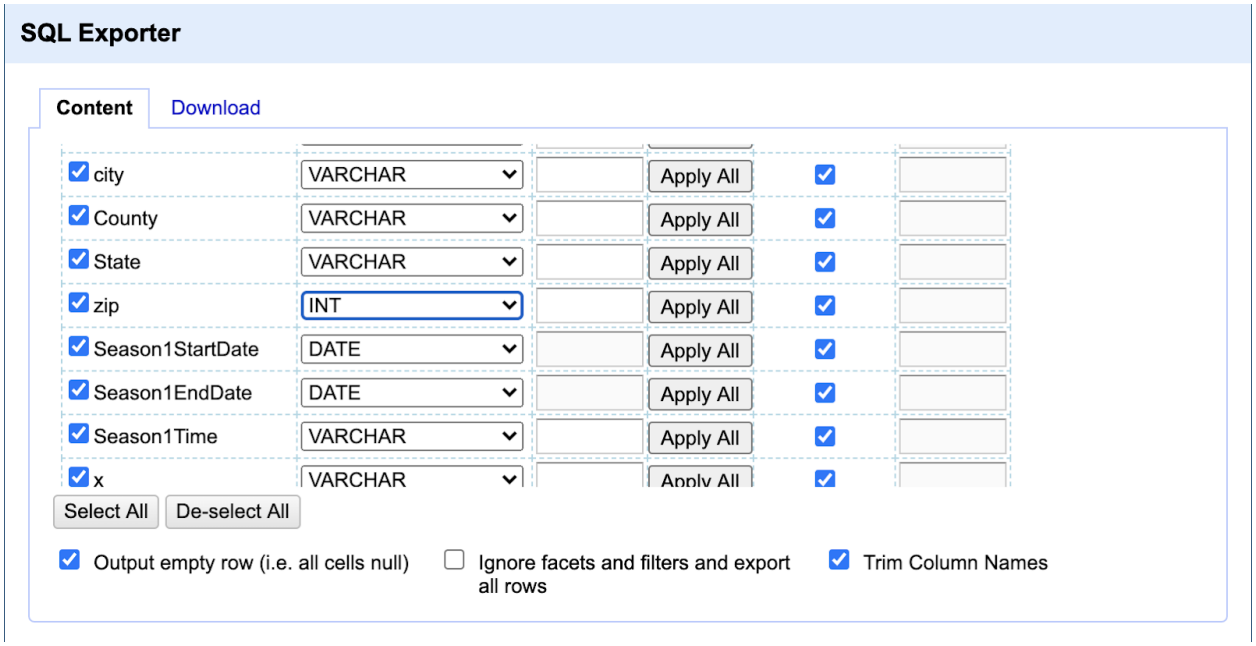
- Trim leading and trailing white space.
- Convert to date format.

4. Results

In this section, we will firstly export SQLite database from OpenRefine, then do integrity constraint analysis with SQLite scripts.

4.1 Relational Database Schema

OpenRefine is able to specify the SQL Type of each column before exporting the sql file.



The FMID and zip column is INT format. Season1StartData, Season1EndDate, are updateTime are DATE format. x and y are NUMERIC format. Others are VARCHAR format.

The following is the schema generated by SQL Exporter of OpenRefine:

```
CREATE TABLE Farmers_Market_Data (
FMID INT NULL,
MarketName VARCHAR(255) NULL,
Website VARCHAR(255) NULL,
Facebook VARCHAR(255) NULL,
Twitter VARCHAR(255) NULL,
Youtube VARCHAR(255) NULL,
OtherMedia VARCHAR(255) NULL,
street VARCHAR(255) NULL,
city VARCHAR(255) NULL,
County VARCHAR(255) NULL,
```

```
State VARCHAR(255) NULL,
zip INT NULL,
Season1StartDate DATE NULL,
Season1EndDate DATE NULL,
Season1Time VARCHAR(255) NULL,
x NUMERIC NULL,
y NUMERIC NULL,
Credit VARCHAR(255) NULL,
WIC VARCHAR(255) NULL,
WICcash VARCHAR(255) NULL,
SFMNP VARCHAR(255) NULL,
SNAP VARCHAR(255) NULL,
Organic VARCHAR(255) NULL,
Bakedgoods VARCHAR(255) NULL,
Cheese VARCHAR(255) NULL,
Crafts VARCHAR(255) NULL,
Flowers VARCHAR(255) NULL,
Eggs VARCHAR(255) NULL,
Seafood VARCHAR(255) NULL,
Herbs VARCHAR(255) NULL,
Vegetables VARCHAR(255) NULL,
Honey VARCHAR(255) NULL,
Jams VARCHAR(255) NULL,
Maple VARCHAR(255) NULL,
Meat VARCHAR(255) NULL,
Nursery VARCHAR(255) NULL,
Nuts VARCHAR(255) NULL,
Plants VARCHAR(255) NULL,
Poultry VARCHAR(255) NULL,
Prepared VARCHAR(255) NULL,
Soap VARCHAR(255) NULL,
Trees VARCHAR(255) NULL,
Wine VARCHAR(255) NULL,
Coffee VARCHAR(255) NULL,
Beans VARCHAR(255) NULL,
Fruits VARCHAR(255) NULL,
Grains VARCHAR(255) NULL,
Juices VARCHAR(255) NULL,
Mushrooms VARCHAR(255) NULL,
PetFood VARCHAR(255) NULL,
Tofu VARCHAR(255) NULL,
WildHarvested VARCHAR(255) NULL,
updateTime DATE NULL
);
```

We can then create the database from this schema through command:

```
sqlite3 Farmers_Market_Data.db < Farmers_Market_Data.sql
```

4.2 Integrity Constraints

There are several integrity problems that should be checked. After this section, we will fix some logical problems in this dataset.

FMID

Since FMID is the identifier of each farmers market, each FMID should be unique in the dataset. We can output duplicates through the command:

```
SELECT FMID
  FROM Farmers_Market_Data
  GROUP BY FMID
  HAVING COUNT(FMID)>1;
```

The output is empty, so there are no duplicates in FMID.

Then we can check whether there are any null FMID:

```
SELECT COUNT(FMID)
  FROM Farmers_Market_Data
  WHERE FMID IS NULL;
```

The output is 0. So there is no null FMID.

OtherMedia

The column of OtherMedia is the supplement of the columns of Website, Facebook, Twitter, and Youtube. If content in OtherMedia is the same as those in one of the four columns, it is considered as redundant.
We can check whether data in OtherMedia is same as any other or not:

```
SELECT "Website  == Othermedia cases ", COUNT(OtherMedia)
  FROM Farmers_Market_Data
  WHERE Website = OtherMedia AND OtherMedia!="";

SELECT "Facebook == Othermedia cases ", COUNT(OtherMedia)
  FROM Farmers_Market_Data
```

```
  WHERE Facebook = OtherMedia AND OtherMedia!="";

SELECT "Twitter  == Othermedia cases ", COUNT(OtherMedia)
  FROM Farmers_Market_Data
  WHERE Twitter = OtherMedia AND OtherMedia!="";

SELECT "Youtube  == Othermedia cases ", COUNT(OtherMedia)
  FROM Farmers_Market_Data
  WHERE Youtube = OtherMedia AND OtherMedia!="";
```

The output is:

```
Website  == Othermedia cases     1
Facebook == Othermedia cases     1
Twitter  == Othermedia cases     0
Youtube  == Othermedia cases     5
```

Then we can remove the redundant data:

```
UPDATE Farmers_Market_Data
SET OtherMedia = ""
  WHERE FMID IN
  (SELECT FMID
    FROM Farmers_Market_Data
    WHERE Website=OtherMedia AND OtherMedia!="");

UPDATE Farmers_Market_Data
SET OtherMedia = ""
  WHERE FMID IN
  (SELECT FMID
    FROM Farmers_Market_Data
    WHERE Facebook=OtherMedia AND OtherMedia!="");

UPDATE Farmers_Market_Data
SET OtherMedia = ""
  WHERE FMID IN
  (SELECT FMID
    FROM Farmers_Market_Data
    WHERE Twitter=OtherMedia AND OtherMedia!="");

UPDATE Farmers_Market_Data
SET OtherMedia = ""
  WHERE FMID IN
  (SELECT FMID
    FROM Farmers_Market_Data
    WHERE Youtube=OtherMedia AND OtherMedia!="");
```

After these steps, duplicated content in OtherMedia is removed, and the updated database is saved in the .db file automatically.

Season1StartDate, Season1EndDate

In previous cleaning section, we splitted Season1Date into Season1StartDate and Season1EndDate, and formatted them into standard date. We can now use SQLite to and output instances that the start is later than the end date in this database:

```
SELECT FMID, Season1StartDate, Season1EndDate
  FROM Farmers_Market_Data
  WHERE Season1StartDate>Season1EndDate
  AND Season1EndDate!="";
```

The output:

```
1006039     2011-11-01T00:00:00Z  2011-04-01T00:00:00Z
1004929     2012-11-01T00:00:00Z  2012-03-01T01:00:00Z
1011959     2016-10-01T00:00:00Z  2016-05-07T00:00:00Z
1000865     2012-11-01T00:00:00Z  2012-03-01T01:00:00Z
1000108     2012-11-01T00:00:00Z  2012-03-01T01:00:00Z
1001963     2012-10-01T00:00:00Z  2012-03-01T01:00:00Z
1002273     2012-12-01T01:00:00Z  2012-05-01T00:00:00Z
1003898     2012-09-01T00:00:00Z  2012-04-01T00:00:00Z
1002777     2012-10-01T00:00:00Z  2012-05-01T00:00:00Z
1005022     2011-09-01T00:00:00Z  2011-05-01T00:00:00Z
1001511     2012-11-01T00:00:00Z  2012-04-01T00:00:00Z
1005379     2012-11-01T00:00:00Z  2012-04-01T00:00:00Z
1002027     2011-11-01T00:00:00Z  2011-05-01T00:00:00Z
1003932     2012-11-01T00:00:00Z  2012-05-01T00:00:00Z
1002724     2012-10-01T00:00:00Z  2012-05-01T00:00:00Z
1008935     2013-01-01T00:00:00Z  2012-12-31T00:00:00Z
1006056     2012-11-01T00:00:00Z  2012-04-01T00:00:00Z
1006508     2012-12-01T01:00:00Z  2012-04-01T00:00:00Z
```

To fix this integrity problem, we can swap the start date and the end date:

```
UPDATE Farmers_Market_Data
SET Season1StartDate=Season1EndDate, Season1EndDate=Season1StartDate
WHERE FMID IN
  (SELECT FMID
    FROM Farmers_Market_Data
    WHERE Season1StartDate>Season1EndDate
    AND Season1EndDate!="");
```

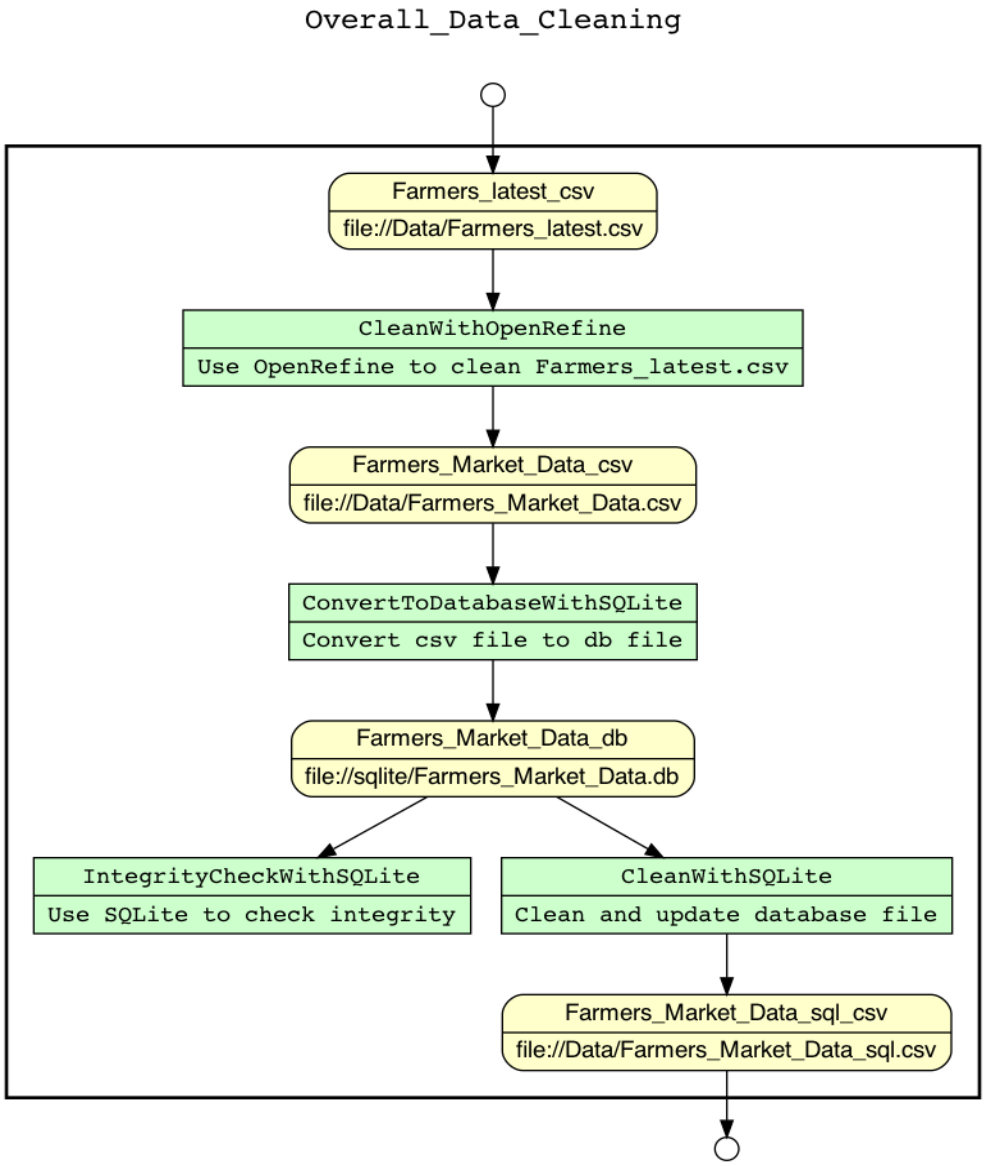
Then the Season1StartDate and Season1EndDate are reasonable in the database.

5. Workflow

YesWorkflow aims to provide a number of the benefits of using a scientific workflow management system without having to rewrite scripts and other scientific software. In this project, we are going to use Workflow to create and manage the scripts for data cleaning, and SQL scripts for data modeling and analysis.

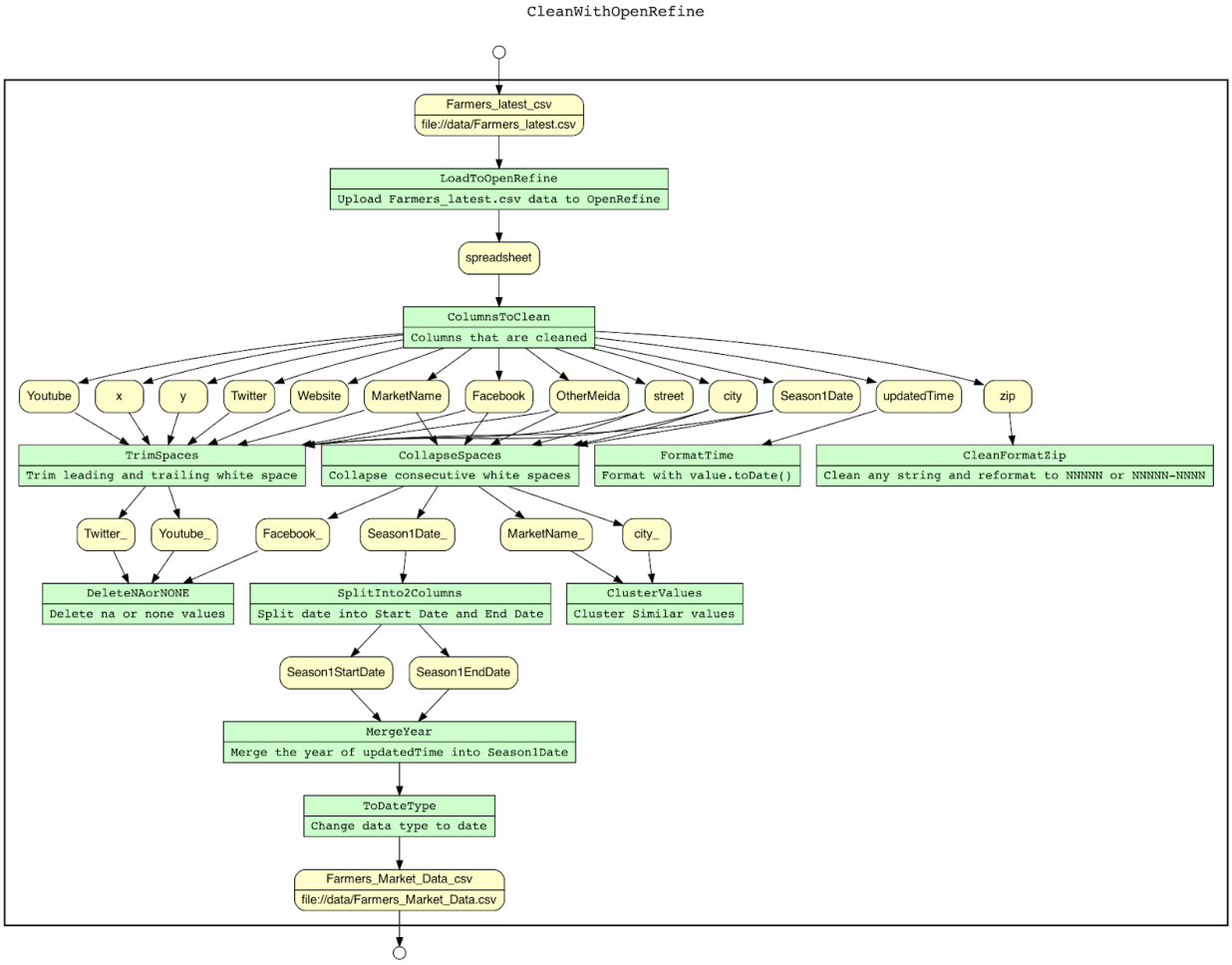
5.1 E2E Workflow:

The following is the end-to-end workflow for our project including data cleaning by OpenRefine,Data modeling and querying in sqlite.



5.2 Data Cleaning by OpenRefine

In the following workflow, it highlights each part of the data cleaning by OpenRefine for each attribute of the dataset.



6. Conclusions and Future Work

In this project, we analyzed the data from Farmers Market datasets which contain very informative attributes but too much noisy, diverse formats and empty information. We first analyzed all these attributes and categorized them into several categories based on the semantic meaning and business scenarios of the attributes to **identification**, **information**, **location**,**Seasonality payment** and **products** categories.

In order to have high quality data for better analysis, we used OpenRefine to do the data cleaning with text cleaning, and clustering to have clean data and consistent formats. Although

there is still some data not being cleaned or reformatted into a perfect way. One problem is that the data contains too much missing values for several attributes. But overall, the data quality after data cleaning is good enough to support our analysis.

In the data analysis part, we leveraged the SQL to convert the data to structured data with data types and formats. On top of this SQL data, we experimented with diverse analysis on the data to analyze the business insights behind the Farmers Market datasets. In this process, we also double checked the data quality when performing the data analysis.

In the last, we also created the workflow model to track our end-to-end pipeline including the data cleaning, data modeling and analysis.

In the next step, we want to continue the data cleaning using OpenRefine and other tools (For example Python Script, SQL Script) And more data modeling and analysis will also continue to have better understanding and insights.