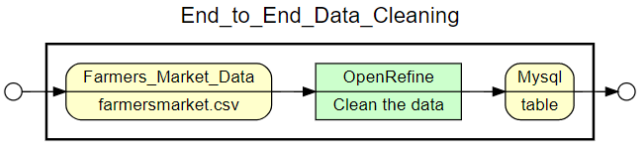
US FARMER MARKET

END-TO-END DATA

CLEANING WORKFLOW



Xiaolong Yang
August 4th, 2017
xyang@illinois.edu (U of I Account)
527536646@qq.com (Coursera Account)

1. Overview and Initial Assessment

Farmers' market is where two or more farm venders selling agricultural products directly to customers at a common, recurrent physical location [1]. The dataset for Farmers Market in US is designed to provide customers with convenient access to information about contact information, market location, opening time, product offerings and accepted forms of payment, and more. The dataset can be found at https://www.ams.usda.gov/local-food-directories/farmersmarkets . It was maintained by Agricultural Marketing Service.

The dataset stored all information in one table. As mentioned above, the dataset contained five parts of information. Contact, Geographic information and operation time should be stored in the same table of the identity, because different market had different locations and time schedule. Product offerings and Payment choices should be stored in separated tables, however, since the dictionary was merely showed user whether they had specific type of product to buy, or the payment methods to choose, there is no need to open a new table.

The contact information included 6 columns: Market Name, Website, Facebook, Twitter, Youtube, and other Media. The Market Name is a text format and others are link format. From an initial assessment using Text Facet in OpenRefine, we can find some of the website and social media was not in link format, I would use regular expression to change the format.

The location part contained 8 columns: street, city, county, state, zip, x (longitude), y (latitude) and locations. The street, city, county, state and location are text format, zip is 5 digits integer, x and y are float point. From an initial assessment, we can find there are different uppercase and lowercase for city, county and state with same meaning. We can also find some of the zip is 4 digits or more than 5 digits. Moreover, the longitude and latitude is in text format. I would use OpenRefine to clean those data.

The operation time contained two part: season date and season time. Season date told users the time periods when farmers market would open and seasons time told users when farmers market would open in a day and day of week. Current date information is stored in text format, I would split the columns into start date and end date in the future. Current time information is stored in json format, I would create 7 columns from Monday to Sunday to store the opening time in each day. There are at most four seasons date and time. There are also an updated time columns to show when is the last time the information modified.

The product offerings and payment choices included 36 columns: Credit, WIC, WICcash, SFMNP, SNAP, Organic, 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. All of them are Yes, No or null. However, for “Organic” column, the null value was replaced with “-”, I would use openrefine to clean the “Organic” column.

The user case should be:

1. import the data into cleaning software,
2. clean exception value,
3. split columns with too much information,
4. transfer data into appropriate format,
5. merge those synonyms cells,
6. export data into files,
7. find the integrity constraint and applied to the data,
8. import the cleaned data into database.

2. Data cleaning with OpenRefine

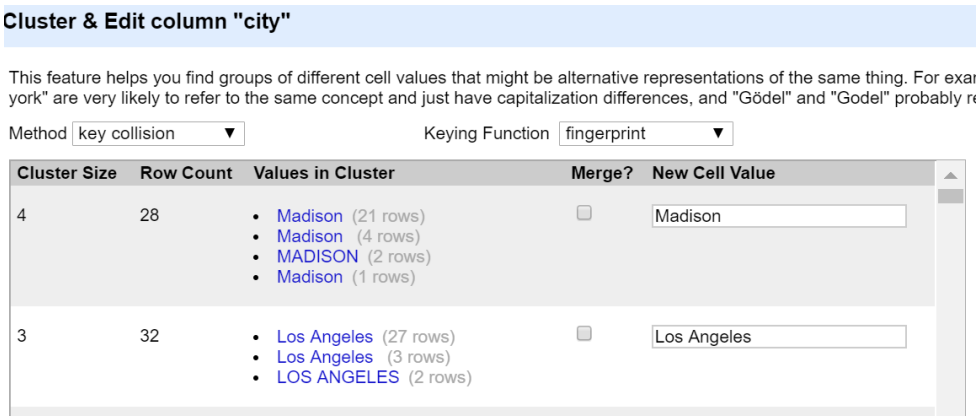
As mentioned above, I divided the dataset into 5 parts: location, contact, operation time, product_offering and payment choice. In the following, I would analyze each part.

2.1 Clean Location information with OpenRefine

The location part contained 8 columns: street, city, county, state, zip, x (longitude), y (latitude) and locations. After checking the text facet, I would focus on city, county, zip, x and y to clean.

2.1.1 Clean City information in Location

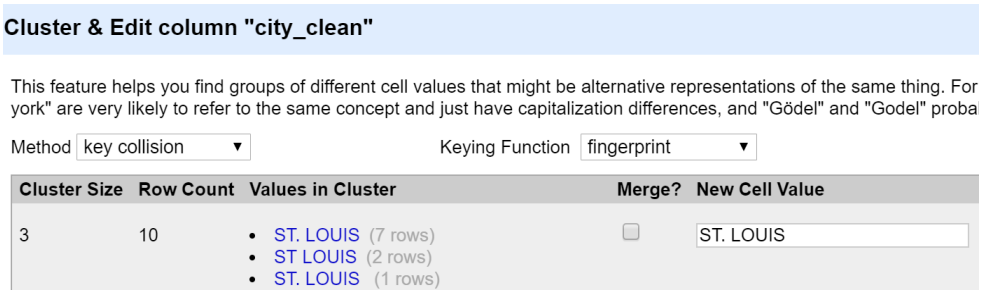
Some of the city name had same meaning but difference in lower and upper characters. We can find an example in the text facet:



Graph1 City Text Facet Cluster

To clean the data, I would add a new column called “city_clean” based on the “city” column.

I used the uppercase function to make the city name in the same format, 8547 out of 8687 cells were affected. The blank values in the cells might also affect the result. I used trim function to cleaned 917 cells and used regular expression to replace blank in 2 cells. After then, the dataset still had columns had key collision:

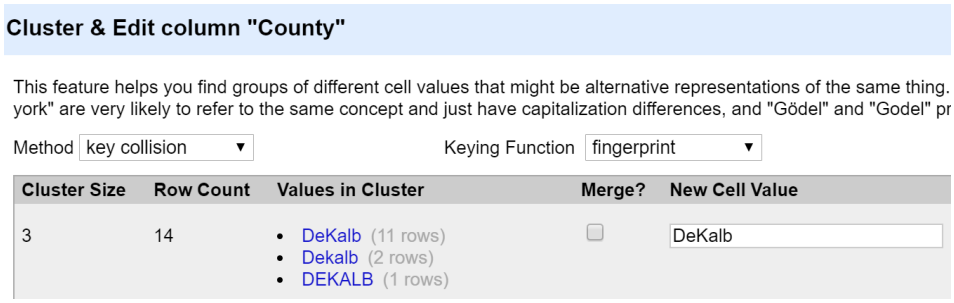


Graph2 City_clean Text Facet Cluster

In this case, I used the merge function in OpenRefine, 58 cells were affected in the first round and 62 cells were affected in the second round.

2.1.2 Clean county information in location

Some of the county name had same meaning but difference in formats. We can find an example in the text facet:



Graph3 City Text Facet Cluster

We can easily find there are collisions. In this case, I first create a new column "county_clean" and then used merge function to clean 734 cells.

2.1.3 Clean zip code information in location

In common sense, we knew the zip code should be 5 digits integer. However, there are different format in the zip columns:

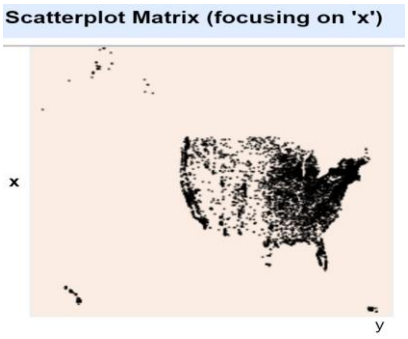
Missing digit	Adding More digit	Full zip code format	Not digit
00000 1 0033 1	513338 1 550424 1	11231 2 11231-1235 1	96815-3738 1 FL 1

Table1 Zip code different format

To clean the zip code, I added a new column called “zip_clean”. For missing digit and adding more digit, I decided to find the zip code based on their other location columns. I manually cleaned 20 cells in zip_clean. For none digit cells, I set all of them to blank in zip_clean. For full zip code format, I used jython code to retrieve the first 5 digits in OpenRefine. It affected 9 cells in zip_clean.

2.1.4 Clean x and y information in location

Since x and y are float point, I used the to number function change that to float format. Using scatter plot facet, we can find it formed the US shape in the map.



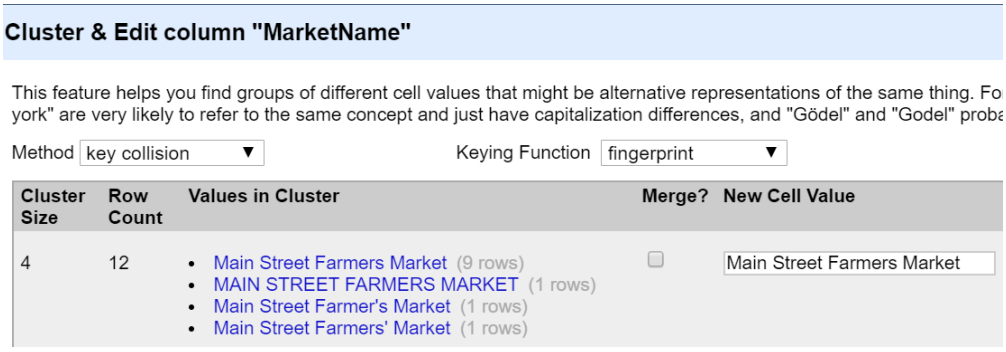
Graph4 X_Y Scatterplot

2.2 Clean Contact information with OpenRefine

The contact information included 6 columns: Market Name, Website, Facebook, Twitter, Youtube, and other Media. I would focus on Market_Name, Website, Facebook and Twitter for cleaning.

2.2.1 Clean Market Name in Contact

Some of the market name is in different format with same meaning:



Graph5 Market Name Clean

I used merge function cleaned 677 cells.

2.2.2 Clean Website in Contact

Some of the website is in different format with same meaning:

Graph6 Website Clean

I used merge function cleaned 160 cells.

2.2.3 Clean Facebook in Contact

We hope the Facebook columns in the link format that viewers can easily use. However, there are different format in the column:

Full link	Not start with https://	Name in Facebook only
www.facebook.com/pages/Haddon-Heights-Farmers-Market/219172298144851 1	www.facebook.com/BrattleboroWinterFar	12_South_Farmers_Market 1 18th Street Farmers Market 1

Table2 Faceboook different format

In this case, I created a new column “facebook_clean”, write jython code in OpenRefine to clean 922 cells and transfer them into valid link. However, some of the website had same meaning might still in different formats. We can find an example in the text facet:

Graph7 Facebook Clean

The difference could be upper and lowercase, with “” and “/” or not, or other difference like “#!”. In this case, I used the merge function to clean. The first round affected 18 cells, the second round affect 106 cells.

2.2.4 Clean Twitter in Contact

The Twitter column had the same issues with Facebook. To clean the columns, I created a new column “twitter_clean”, user lowercase function to clean 603 cells, write jython code in

OpenRefine to clean 334 cells not in link format. After then, I used merge function to clean 15 cells.

2.3 Clean Operation Time information with OpenRefine

The operation time contained two part: season date and season time. Each part had 4 columns, from season 1 to season 4. However, the season is not equal to 3 months, but a duration market set.

There is also a related column "UpdateTime" tells user when is the information last updated. The updated time could occur both before and after the season's date, but they should appear in same year.

2.3.1 Clean Season Date in Operatioin Time

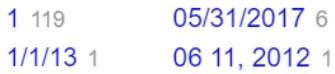
There are mainly 3 types of date format in the Season Date:

US Date Format	Month Full Name_day, year	Only Month Full Name
11/22/2015 to 04/01/2016 1 11/22/2015 to 12/20/2015 1	June 17, 2012 to June 17, 2012 1	July to August 2 July to November 6

Table3 Season Date different format

To clean the Season date, I create a new column "SeasonDateCopy", replace the month full name to the corresponding number using jython in OpenRefine. It affected 808 cells in Season 1, 24 cells in Season 2, 6 cells in season3 and no cells in season4.

The next step is to split the column into two columns, start date and end date. I used jython split function splited by "to". The new columns called "SeasonDateStart" and "SeasonDateEnd". For both Season Date data, we still need to transform the "MM DD, YYYY" into "MM/DD/YYYY" format, and fix the column only contained the month.



Graph8 Season Date different format

I tried to use jython for the former, but after I transfer the columns into date format, I found OpenRefine can also recognized “MM DD, YYYY” format.

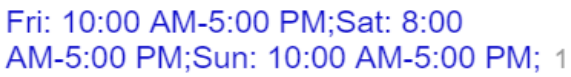
For those only contained the month, we can estimate the day is the beginning of the month, but we cannot estimate the year. As mentioned above, the Season Date information should be in the same year of Update Time information, I used that part to fix the missing part.

I created a new column “updateyear” based on “UpdateTime”, combined with cells only had month information in Season Date, and then transformed into “MM/01/YYYY” using jython. It affected 29 cells in season 1 start date, 27 cells in season 1 end date; 24 cells in season 2 start date, 22 cells in season 2 end date; 5 cells in season 3 start date, 5 cells in season 3 end date.

After then, I used the toDate function transform the cells into date format. It affected 5478 cells in season 1 start date, 5362 cells in season 1 end date, 449 cells in season 2 start date, 434 cells in season 2 end date, 81 cells in season 3 start date, 6 cells in season 3 end date, 5 cells in season 4 start date, 5 cells in season 4 end date.

2.3.2 Clean Season time in Operation Time

Season time date is in JSON format, only few days in a week would market open, and each day had different operation time schedule.



Graph9 Season Time format

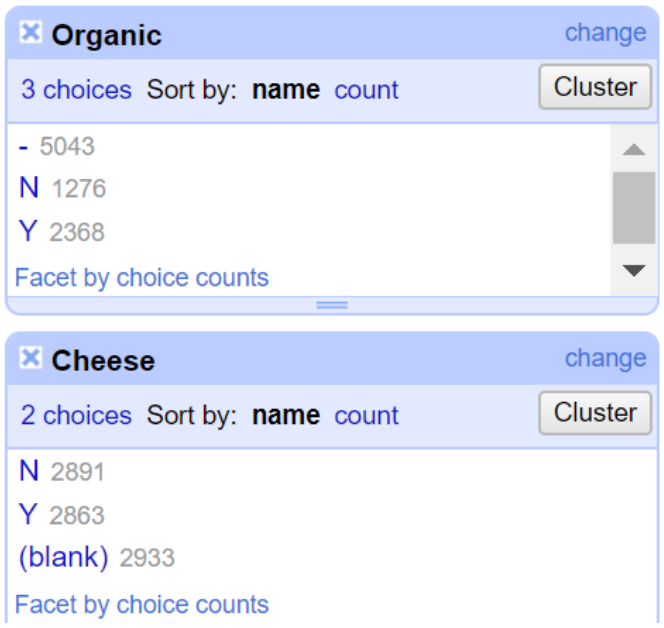
In this case, I used jython to split the season time into 7 columns from Monday to Sunday.

Season1Time	Season1Sunday	Season1Saturda	Season1Friday	Season1Thursd	Season1Wednes	Season1Tuesda	Season1Monday	Season2Date
Wed: 9:00 AM-1:00 PM;					9:00 AM-1:00 PM			09/06/2017 to 10/18/2017
Sat: 9:00 AM-1:00 PM;		9:00 AM-1:00 PM						
Wed: 3:00 PM-6:00 PM;Sat: 8:00 AM-1:00 PM;		8:00 AM-1:00 PM			3:00 PM-6:00 PM			
Tue:8:00 am - 5:00 pm;Sat:8:00 am - 8:00 pm;		8:00 am - 8:00 pm				8:00 am - 5:00 pm		
Tue: 3:30 PM-6:30 PM;						3:30 PM-6:30 PM		
Tue: 10:00 AM-7:00 PM;						10:00 AM-7:00 PM		

Graph10 Season Time split into 7 columns

2.4 Clean Product Offering information with OpenRefine

The product offerings included a lot of columns, all of cells are among “Yes”, “No” or “null”. However, for “Organic” column, the null value was “-”.



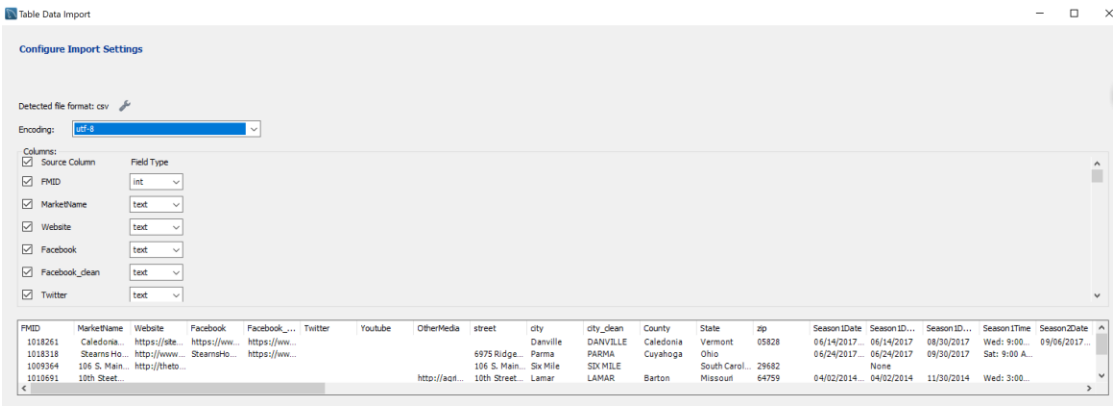
Graph11 Organic Clean

To make all of them had same format, I created a new column “Organic_clean” used the jython function to replace that “-” with null.

3. Develop a relational database schema

3.1 Load data table into relational database

As mentioned before, all the farmer markets data stored in one table. I could split the table based on the different function above, however, all of them had different locations, operation time, contact, and product offering and payment type contained only true or false, split the table into different table would be redundant. In this case, I would upload the table into one table in my schema in Mysql.



Graph12 Import cleaned data into Mysql

3.2 Check the integrity constraints

There are 4 logical integrity constraints in the database: product buyable, market payable, start date smaller then end date and season year equal to update year.

3.2.1 Product buyable

The farmers market must have at least one of the product offerings, otherwise customers had nothing to buy. The integrity rules in SQL is:

```
SELECT
   FMID, IC
FROM
   (SELECT
      Organic_clean+ Bakedgoods_clean+Cheese_clean+ Crafts_clean+
Flowers_clean+ Eggs_clean+ Seafood_clean+ Herbs_clean+  Vegetables_clean+
Honey_clean+ Jams_clean+ Maple_clean+ Meat_clean+ Nursery_clean+
Nuts_clean+ Plants_clean+ Poultry_clean+ Prepared_clean+ Soap_clean+
Trees_clean+ Wine_clean+ Coffee_clean+ Beans_clean+ Fruits_clean+
Grains_clean+ Juices_clean+ Mushrooms_clean+ PetFood_clean+ Tofu_clean+
WildHarvested_clean AS IC,
      FMID
   FROM
      `598`.farmersmarkets
   WHERE
      Organic_clean IN (0 , 1) and  Bakedgoods_clean IN (0 , 1) and Cheese_clean IN
(0 , 1) and  Crafts_clean IN (0 , 1) and  Flowers_clean IN (0 , 1) and  Eggs_clean IN
(0 , 1) and  Seafood_clean IN (0 , 1) and  Herbs_clean IN (0 , 1) and
Vegetables_clean IN (0 , 1) and  Honey_clean IN (0 , 1) and  Jams_clean IN (0 , 1)
and  Maple_clean IN (0 , 1) and  Meat_clean IN (0 , 1) and  Nursery_clean IN (0 , 1)
and  Nuts_clean IN (0 , 1) and  Plants_clean IN (0 , 1) and  Poultry_clean IN (0 , 1)
and  Prepared_clean IN (0 , 1) and  Soap_clean IN (0 , 1) and  Trees_clean IN (0 , 1)
and  Wine_clean IN (0 , 1) and  Coffee_clean IN (0 , 1) and  Beans_clean IN (0 , 1)
and  Fruits_clean IN (0 , 1) and  Grains_clean IN (0 , 1) and  Juices_clean IN (0 , 1)
and  Mushrooms_clean IN (0 , 1) and  PetFood_clean IN (0 , 1) and  Tofu_clean IN
(0 , 1) and  WildHarvested_clean
      ) AS Q
WHERE Q.IC = 0;
```

There are 0 cells violate the rules, the least market offered 6 types of product:

FMID	IC	Organic	Bakedgoods	Cheese	Crafts	Flowers	Eggs	Seafood	Herbs
1011705	6	N	N	N	N	N	N	N	Y
1018166	7	Y	N	N	N	N	Y	N	N
1011319	8	-	Y	N	Y	Y	N	N	N

Graph13 Product_buyable_IC

3.2.2 Market payable

The farmers market must have at least one methods for customers to pay their purchase. The integrity rules in SQL is:

```
SELECT
   FMID, Credit,  WIC,   WICcash, SFMNP,   SNAP
FROM
   (SELECT
      Credit_clean + WIC_clean + WICcash_clean + SFMNP_clean + SNAP_clean
AS IC,
         FMID, Credit, WIC, WICcash, SFMNP, SNAP
   FROM
      `598`.farmersmarkets
   WHERE

      Credit_clean IN (0 , 1)

         AND WIC_clean IN (0 , 1)

         AND WICcash_clean IN (0 , 1)

         AND SFMNP_clean IN (0 , 1)

         AND SNAP_clean IN (0 , 1)) AS Q

WHERE

   Q.IC = 0;
```

There are 2677 cells violate the rules, here is sample result:

FMID	Credit	WIC	WICcash	SFMNP	SNAP
1006234	N	N	N	N	N
1006494	N	N	N	N	N
1007585	N	N	N	N	N
1002947	N	N	N	N	N
1004031	N	N	N	N	N

Graph14 Market_payable_IC

It could happen when the "N" is the default setting for the payment choices and maintainers kept all unknown as default. However, those cells are not integrity.

3.2.3 Start date should smaller than End data

The market operation start date should smaller then end data, otherwise the market would not open. The integrity rules in SQL is:

```
SELECT
   FMID, Season1startDate,
Season1EndDate, Season1Date
FROM
   `598`.farmersmarkets
WHERE
   date(Season1startDate) >
date(Season1EndDate);
```

There are 17 cells violate the rules, here is sample result:

FMID	Season1startDate	Season1EndDate	Season1Date
1004929	2012-11-01T00:00:00Z	2012-03-01T00:00:00Z	November to March
1011959	2016-10-01T00:00:00Z	2016-05-07T00:00:00Z	10/01/2016 to 05/07/2016

Graph15 Date_Start_before_End_IC

We can find some of the cells was wrong because their Season Date is from this year month to next year month, but I used the same update year for them. Others went wrong because they are actually record the wrong time in the Season Date columns.

3.2.4 Season end date should be in the same or larger year of updateyear

The farmer market operation time should include the current and future time schedule when it updated. The integrity rules in SQL is:

```
SELECT

   FMID, Season1EndDate, updateyear

FROM

   `598`.farmersmarkets

WHERE

   YEAR(Season1EndDate) < updateyear;
```

There are 351cells violate the rules, here is sample result:

FMID	Season1EndDate	updateyear
1009994	2014-09-27T00:00:00Z	2016
1005636	2013-12-31T00:00:00Z	2014
1008391	2013-08-08T00:00:00Z	2014
1000986	2013-10-20T00:00:00Z	2016
1006988	2013-09-21T00:00:00Z	2014

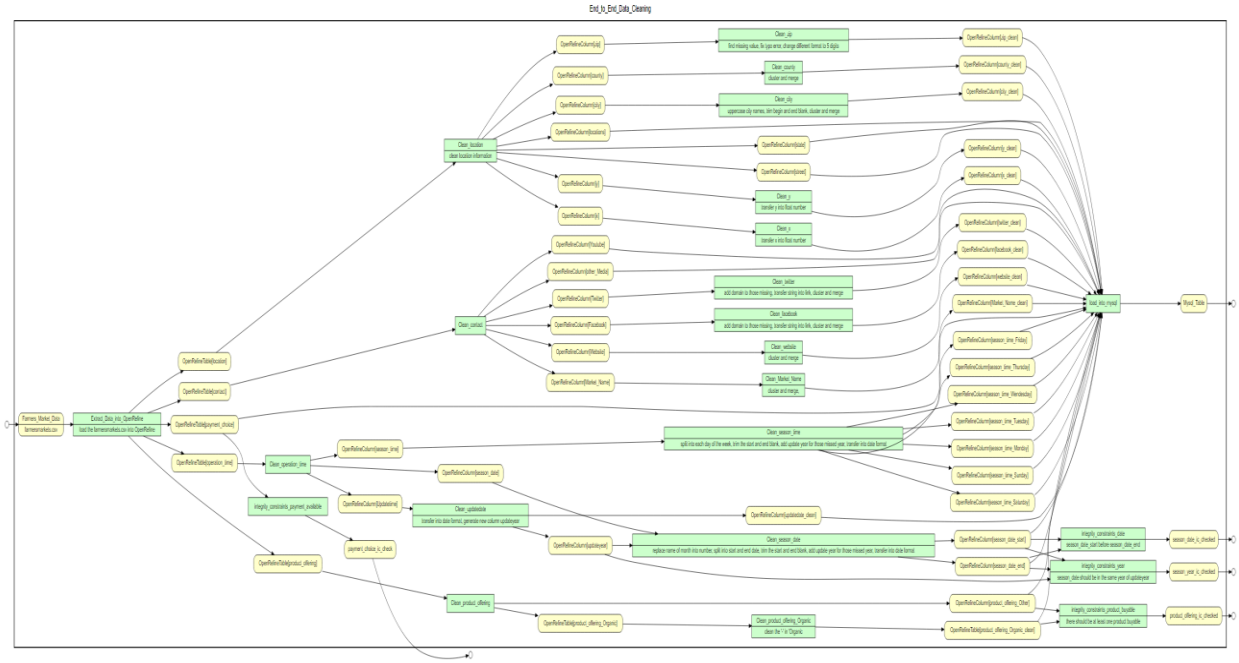
Graph16 Season_End_date_before_last_update_IC

Some of the season date is 3 years before the last update. It could happen when the maintainers keep the same season date or they forget to update. Whatever reason beneath, the 351 cells is not integrity.

4. Workflow Model

The key input is the farmersmarket.csv file and the output is the Mysql table.

Here is the overall view:



You can use YW Model File.txt to replot the graph to view the details.

Reference

[1] https://www.ams.usda.gov/local-food-directories/farmersmarkets