CS 513: Theory and Practice of Data Cleaning - Final Project
US Farmers Markets Data Set

By: James Gaysek (jgaysek2), Adam Reid (adamr3), Matthew Schley (mschley2)

Introduction

We have developed an end-to-end data cleaning workflow, leveraging industry standard tools, including OpenRefine, Python, YesWorkflow, SQLite, and Datalog. Here, we will provide an overview of the dataset we selected for cleaning, as well as an in depth review of each step in our cleaning workflow. We have also attached an *Appendix* containing additional resources for the reader's consideration.

Dataset Overview

The United States Department of Agriculture (U.S.D.A.)[1] maintains the National Farmer's Market Directory, for which we have chosen to develop our end-to-end data cleaning workflow. The dataset is composed of markets "that feature two or more farm vendors selling agricultural products directly to customers at a common, recurrent physical location". The data is in tabular form and is provided in comma-separated value (CSV) file format. The corpus contains 8812 records, each with 59 unique columns. The fields contain information on location data, seasonal hours, payment methods, item availability, and social media hyperlinks. Data types vary depending on the field, meaning many columns lack a uniform type.

The dataset is vast, and many quality issues exist in its uncleaned state. An immediately visible issue centers around duplicate records, whereby multiple records in the dataset reference the same farmer's market. Location data is problematic as well, with many columns (e.g. Street, City, County, State, Zip, etc.) being null. The Street column also acts as a "catch all" field, whose contents vary from valid street addresses to descriptions of specific parking lots or town centers. The columns containing social media data are rife with formatting issues as well as consistency issues, with entries containing a chaotic mess of descriptive strings, hyperlinks, and specific user IDs. Across the entire dataset, individual columns have various formatting and data type discrepancies between their data. Input into the dataset is more or less unrestricted via

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

freeform input boxes with no regular expression checks or validations. Any input requirements that may have at one time existed may very well have been changed at some time during the lifetime of the dataset.

The farmer's market data has several viable use cases that have driven the core focus of our data cleaning objectives. A user may query the uncleaned dataset to locate a nearby farmer's market, or a new startup credit card processing company may be interested in determining which farmers markets they should attend in order to sell more of their credit card processing portals.

For an individual user, the dataset may already be clean enough. A user can search for a specific farmer's market by either its name or location, and will potentially gain insight into what type of products and payment options are available, information about the seasons and times, and the location of the market itself. Although the data quality may not be pristine, a human could gather enough information from the records to find a particular farmer's market of interest, and will likely need to supplement any missing data using Google Maps or one of the associated social media or website links. As most of the records appear to have been entered as unvalidated free-text, any human user will have varied experiences depending on which record or market they happen to focus on.

Another use case we identified for this dataset centers on a hypothetical, new credit card processing company called Chip™. Being the best and the brightest, a data analyst at Chip would seek out which farmer's markets currently use credit processing applications from this dataset. A savvy marketing manager, armed with the insight this dataset has the potential to offer, may customize two different marketing approaches for those who may be looking for the latest and greatest credit card processing companies versus those who would benefit from their service. The analysts at Chip wouldn't have the time or man-power to interact with the data on as personal a level as the lone user looking for locally produced apple butter. They would be performing mass, programmatic operations on the data. Unfortunately for our friends at Chip, the uncleaned dataset is in a very poor state and is nowhere close to being clean enough for their use case. Any automated query may return duplicate records, confusing season date ranges may be returned, and effectively identifying social media will be nontrivial. If left unprocessed and in an uncleaned state, downstream developers will need to handle an unmanageable amount of edge cases.

Taking into account the numerous preceding issues, and considering the Chip use case, we define our data cleaning goals as follows. At a high level, we will normalize data

fields while minimizing record loss, as dropping records will directly result in a loss of potential revenue for Chip (and not to mention those lost farmer's markets themselves). Beyond this, each column should have uniform formatting -- dates as ISO 8601 date times, booleans as 0/1, etc.. Since many columns in a given record are null, malformed records should at minimum retain these null fields rather than being thrown out; we will preserve as much information as possible. Social media columns will be normalized to proper hyperlinks where possible. Ensuring records contain valid locations is critical, with social media hyperlinks and websites providing a potential secondary source of supplementary location data. These data cleaning objectives will result in a dataset that is both uniformly formatted for easier human searchability as well as for machine readability.

Data Cleaning with OpenRefine

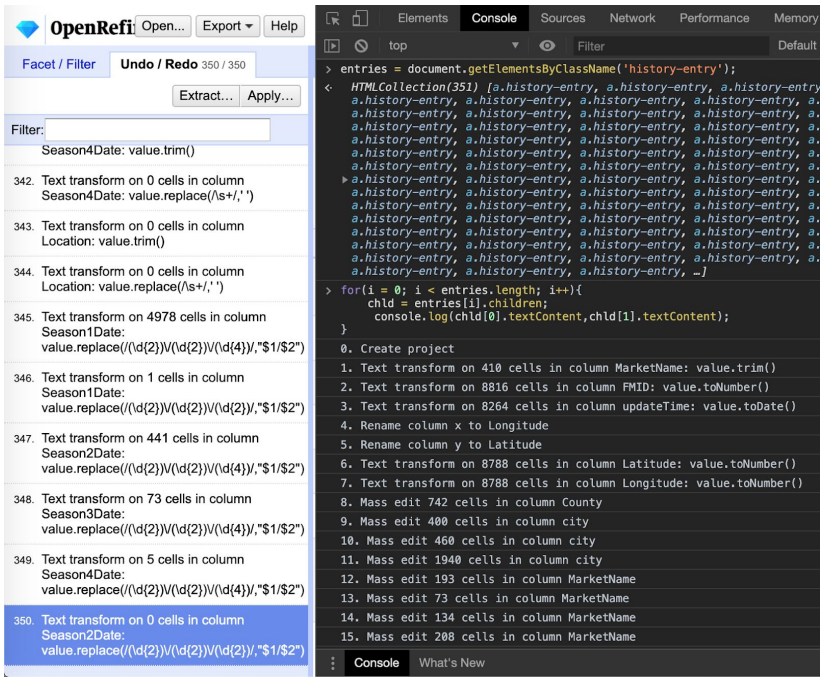
The first step in our data cleaning process after data acquisition was to load our dataset into OpenRefine. Our first set of changes were related to capitalizing all of the column headers and renaming columns such as 'x' and 'y' to Latitude and Longitude. The intent was to add clarity as to what each column's data values were actually representing. For instance, ‘x’ and ‘y’ imply a cartesian coordinate system or possibly some transformation thereof, which does not align with the geodetic representation presented in the columns. To demonstrate the value in this step, we have included **Appendix E**, which shows a plot created with the cleaned coordinate data from the dataset. Once coordinates were established, we performed clustering with the fingerprint algorithm on the County, City, and MarketName columns to ensure uniformity. Then, we conducted a manual inspection of all suggested clusters, and in cases of ambiguity, a deeper inspection into the datum identified for the cluster. In cases where a cluster was not appropriate, perhaps due to the overeagerness of the algorithm, the clustering was not performed. Next, we transformed column data types to more accurately present the data’s intrinsic meaning to the record. For example, we converted the UpdateTime column from string to date value type, and we transformed the FMID (unique farmer's market identifier) from string to a numeric type. After ensuring all columns were of the appropriate type, we trimmed whitespaces from the front and back of each value in every column. We removed duplicate whitespaces, something we performed a second time as our final step once all changes had been made.

After we had completed the more generic value type and whitespace cleanup changes, we then focused on the more esoteric changes that we would have to make. For example, we encountered a problem where one record’s Season1Date, Season2Date,

Season3Date, and Season4Date fields contained a date range between two dates. This range would come in a number of different date formats, so we were forced to manually change the string representations of the range to MM/DD (i.e. July 1st, 2014 to September 28th, 2014 became 07/01 to 09/28). Some dates had years included, whereas others did not. We removed these years using Regex, as they were a bit confusing and made the data appear stale. We also converted boolean Y/N values representing certain product offerings or payment methods into proper bit values -- 1 and 0 -- for more efficient storage in our SQL database. One of these Y/N columns also had a third option of '-', which we converted to be blank to match the other binary columns.

Once completed, we had 350 steps in our OpenRefine provenance history (see Appendix A). Aside from the manual intervention for the Season1Date through Season4Date columns, we implemented the vast majority of the changes to provide uniform formatting across the corpus, while ensuring as little data loss as possible.

Figure 1

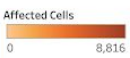


Quantifying the Results

We performed 350 operations in cleaning our data. Many cells were affected by either a mass edits, text transformations, or both. Although we have already described in detail our OpenRefine cleaning steps, it can be difficult to digest the magnitude and the vector of these changes. The provenance history that OpenRefine exports as JSON does not include information on the magnitude of the affected cells for each operation. To get this information, we scripted the browser console dump the magnitude information from the provenance chart in OpenRefine. This simple process is demonstrated in **Figure 1**. We then converted the output from the console log into CSV and ingested the data into Tableau. Having done this, we are able to present **Figure 2**: a complete view of all column operations, rename operations, mass edits, text transformations, expressions, and each affected cell. In total, we cleaned *273,516* cells of data. This represents changes to over 52% of the entire corpus.

Figure 2

Column	Operation	Affected Cells
*Special	Create project	
	Rename column city to City	
	Rename column street to Street	
	Rename column updateTime to UpdateTime	
	Rename column x to Longitude	
	Rename column y to Latitude	
	Rename column zip to Zip	
Bakedgoods	Mass Edit	5,996
Beans	Mass Edit	5,996
Cheese	Mass Edit	5,996
city	Mass Edit	2,800
City	value.replace(/\s+/," ")	1
	value.trim()	436
Coffee	Mass Edit	5,996
County	Mass Edit	742
	value.replace(/\s+/," ")	0
	value.trim()	0
Crafts	Mass Edit	5,996
Credit	Mass Edit	8,816
Eggs	Mass Edit	5,996
Facebook	value.replace(/\s+/," ")	3
	value.trim()	40
Flowers	Mass Edit	5,996
FMID	value.toNumber()	8,816
Fruits	Mass Edit	5,996
Grains	Mass Edit	5,996
Herbs	Mass Edit	5,996
Honey	Mass Edit	5,996
Jams	Mass Edit	5,996
Juices	Mass Edit	5,996
Latitude	value.toNumber()	8,788
Location	value.replace(/\s+/," ")	0
	value.trim()	0
Longitude	value.toNumber()	8,788
Maple	Mass Edit	5,996
MarketName	Mass Edit	672
	value.replace(/\s+/," ")	50
	value.trim()	413
Meat	Mass Edit	5,996
Mushrooms	Mass Edit	5,996
Nursery	Mass Edit	5,996
Nuts	Mass Edit	5,996
Organic	Mass Edit	8,816
OtherMedia	value.replace(/\s+/," ")	22
	value.trim()	18
PetFood	Mass Edit	5,996
Plants	Mass Edit	5,996
Poultry	Mass Edit	5,996
Prepared	Mass Edit	5,996
Seafood	Mass Edit	5,996
Season1Date	Mass Edit	883
	value.replace(/(\d{2})\/(\d{2})\/(\d{2})/,"$1/$2")	1
	value.replace(/(\d{2})\/(\d{2})\/(\d{4})/,"$1/$2")	4,978
	value.replace(/\s+/," ")	0
	value.trim()	5
Season1Time	value.replace(/\s+/," ")	0
	value.trim()	0
Season2Date	Mass Edit	38
	value.replace(/(\d{2})\/(\d{2})\/(\d{2})/,"$1/$2")	0
	value.replace(/(\d{2})\/(\d{2})\/(\d{4})/,"$1/$2")	441
	value.replace(/\s+/," ")	0
	value.trim()	2
Season2Time	value.replace(/\s+/," ")	0
	value.trim()	0
Season3Date	Mass Edit	8
	value.replace(/(\d{2})\/(\d{2})\/(\d{4})/,"$1/$2")	73
	value.replace(/\s+/," ")	0
	value.trim()	0
Season4Date	value.replace(/(\d{2})\/(\d{2})\/(\d{4})/,"$1/$2")	5
	value.replace(/\s+/," ")	0
	value.trim()	0
SFMNP	Mass Edit	8,816
SNAP	Mass Edit	8,816
Soap	Mass Edit	5,996
State	value.replace(/\s+/," ")	0
	value.trim()	0
Street	value.replace(/\s+/," ")	90
	value.trim()	318
Tofu	Mass Edit	5,996
Trees	Mass Edit	5,996
Twitter	value.replace(/\s+/," ")	1
	value.trim()	9
updateTime	value.toDate()	8,264
Vegetables	Mass Edit	5,996
Website	value.replace(/\s+/," ")	0
	value.trim()	27
WIC	Mass Edit	8,816
WICcash	Mass Edit	8,816
WildHarvested	Mass Edit	5,996
Wine	Mass Edit	5,996
Youtube	value.replace(/\s+/," ")	0
	value.trim()	4
Zip	value.replace(/\s+/," ")	0
	value.trim()	0
Grand Total		273,516



Affected Cells broken down by Measure Names vs. Column and Operation. Color shows Affected Cells. The marks are labeled by Affected Cells. The view is filtered on Column, which excludes Null.

Data Cleaning with Python

We supplemented the data cleaning we performed in OpenRefine with Python to harmonize the malformatted social media hyperlinks. Cleaning Facebook and Twitter hyperlink data via mass edit proved quite cumbersome, so we used Python to programmatically address semantic issues in the data - specifically, to convert Facebook page names and Twitter handles into valid urls so that less emphasis would be placed on a non-technical user's ability to piece the data together or navigate to the link. Additionally, the update time field did not use a common time format, so ISO 8601 was selected and applied to the column for consistency. We chose ISO 8601 as it is the defacto standard for date and time formatting.

Developing a Relational Schema

We used OpenRefine to generate an initial SQL schema and SQL data, which we were able to leverage directly. We performed manual inspection of the data to derive the schema, which is included in **Appendix B**. After manually setting the schema, we loaded the entire SQL into SQLite to test our integrity constraints.

The first IC we devised considered the uniqueness of the primary key. If the FMID is to be a primary key, it must be unique. We ran the following query to validate the uniqueness of the FMID, where a result of *0* indicates that no duplicates were found:

```
SELECT FMID, count(*) as NUMBER
FROM FarmersMarket
GROUP BY FMID
HAVING COUNT(*) > 1
ORDER BY NUMBER desc;
```

Next, we wanted to ensure our non-nullable fields were compliant with our schema, and so we defined our IC as follows:

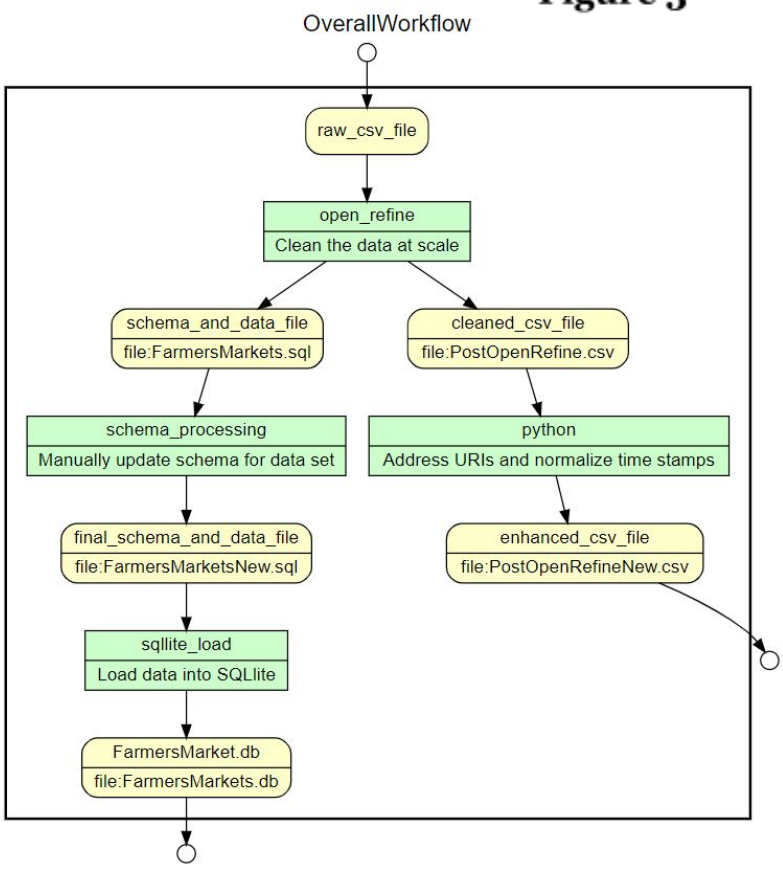
```
SELECT count(*) AS icv
FROM FarmersMarket
WHERE credit = NULL
OR wic = NULL
OR wiccash = NULL
```

```
OR sfmnp = NULL
OR snap = NULL
OR fmid = NULL
OR marketname = NULL
OR updatetime = NULL;
```

The latitude and longitude of our data is expected to be in the northwestern hemisphere. From visual inspection, we know that the dataset contains markets as far south as the Virgin Islands, as far north as Alaska, and as far west as Hawaii. We determined that reasonable bounds for our data would be latitude: 0 to 90, and longitude: 0 to -180.

```
SELECT count(*) as icv
FROM FarmersMarket
WHERE latitude < 0
OR latitude > 90
OR longitude < -180
OR longitude > 0;
```

Figure 3



Creating Workflow Models

The overall workflow of our data cleaning pipeline for this project is shown in **Figure 3**. As one can see, there were a number of steps along the data's cleaning journey. We initially input the raw CSV file, taken from the U.S.D.A.'s website, into OpenRefine, which outputs what we call "Cleaned CSV File": the result of the data cleaning process discussed above. A more detailed provenance model can be found in **Appendix A**. This cleaned CSV file is then input into Python for the enhanced processing that we were unable to achieve within OpenRefine for social media hyperlinks and UpdateTime modifications. Although listed as a single step in the diagram, there were two separate Python processes used. Finer detailed models for these Python processes can be found in **Appendix C**, under ModifySocialMedia and

ModifyUpdateTime. The Python cleaning outputs our final cleaned dataset which we then used to generate relational schema in SQLLite. Our logical integrity constraints are included in the construction of the table in the form of Primary Key constraints and NULL constraints. We then generated insert scripts to import the data into our SQLite database, and subsequently ran the aforementioned SQL queries to verify that our integrity constraints were not violated by the imported data.

Developing provenance

Figure 4

```
e("core/text-transform0","MarketName_1").
e("MarketName","core/text-transform0").
e("core/mass-edit11","MarketName_2").
e("MarketName_1","core/mass-edit11").
e("core/mass-edit12","MarketName_3").
e("MarketName_2","core/mass-edit12").
e("MarketName_3","core/mass-edit13").
e("CombineDataCleaningChanges","CleanData").
e("MarketName_3","CombineDataCleaningChanges").


tc(X,Y) :- e(X,Y).
tc(X,Y) :- e(X,Z), tc(Z,Y).

upstream(X) :- tc(X,"CleanData").
```

An ancillary step in our workflow that we've included was made to show more detail about the inputs and outputs that our OpenRefine workflow depends on. We developed a series of provenance queries in DLV (**Figure 4**) that are designed to reveal the dependencies needed to arrive at our final cleaned data set. To demonstrate this we have included an example provenance query on a subset of edges within the OpenRefine provenance graph showing the history of the MarketName column in **Appendix D** (the graph is too large to demonstrate all edges). As you can see in **Appendix A**, the Clean Data CSV file that is generated at the end of our OpenRefine steps depends on many inputs, namely all the previous transformations applied to MarketName (MarketName_1, MarketName_2, MarketName_3) and CombineDataCleaningChanges which is an aggregate step that brings all of the other columns together (they've been left out for brevity). The full list of dependencies is much larger, consisting of every change applied to every column.

Figure 5

```
upstream("CombineDataCleaningChanges") upstream("MarketName_3") upstream("core/mass-edit12") upstream("MarketName_2")
upstream("core/mass-edit11") upstream("MarketName_1") upstream("core/text-transform0") upstream("MarketName")
```

Appendix A. OpenRefine Provenance History.

https://github.com/adam-reid/cs-513-final-project/blob/master/ywf.png

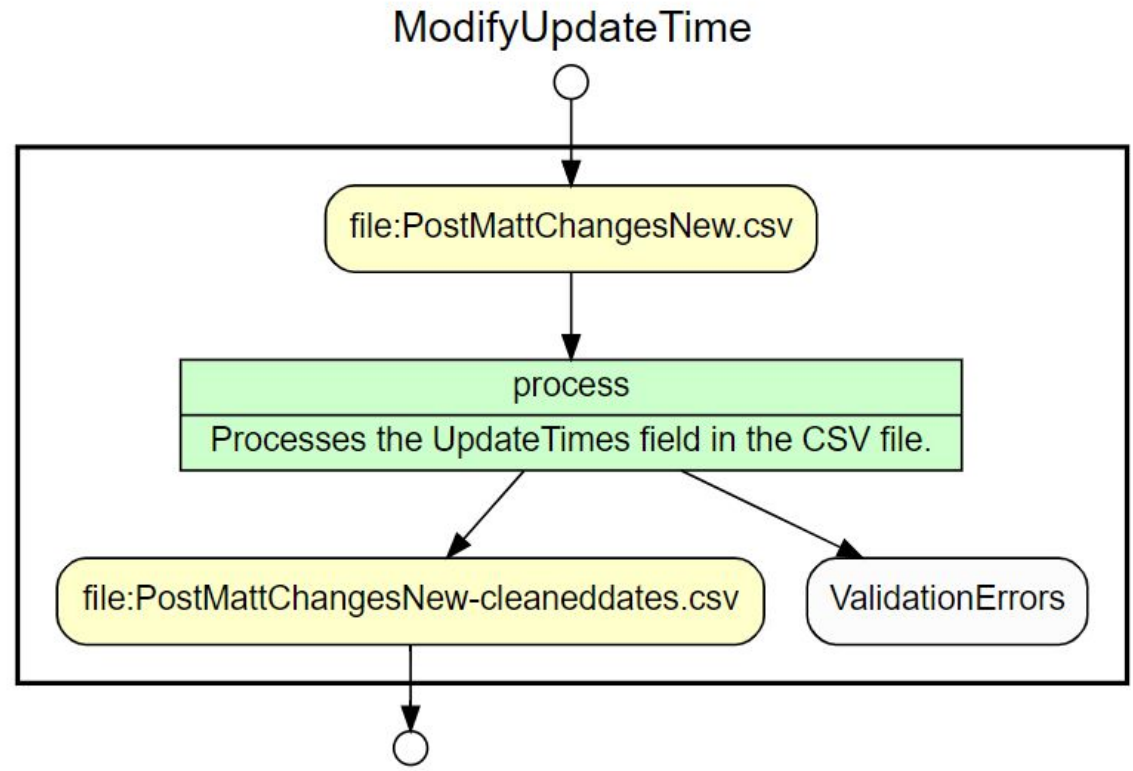
Appendix B. Relational Schema.

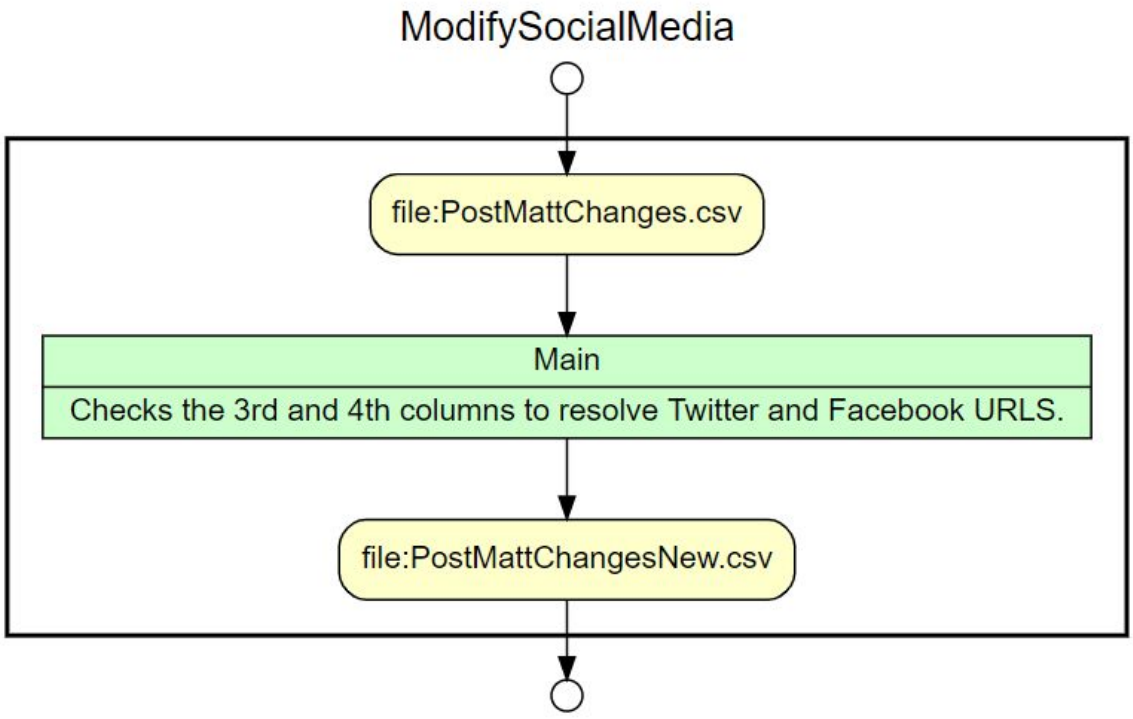
Field	Data Type	Nullable	Key Type
FMID	INT	NO	PRIMARY
MarketName	VARCHAR(255)	NO	
Website	VARCHAR(255)		
Facebook	VARCHAR(255)		
Twitter	VARCHAR(255)		
Youtube	VARCHAR(255)		
OtherMedia	VARCHAR(255)		
Street	VARCHAR(255)		
City	VARCHAR(255)		
County	VARCHAR(255)		
State	VARCHAR(255)		
Zip	VARCHAR(5)		
Season1Date	VARCHAR(255)		
Season1Time	VARCHAR(255)		
Season2Date	VARCHAR(255)		
Season2Time	VARCHAR(255)		
Season3Date	VARCHAR(255)		
Season3Time	VARCHAR(255)		
Season4Date	VARCHAR(255)		
Season4Time	VARCHAR(255)		
Longitude	NUMERIC		

Field	Data Type	Nullable	Key Type
Latitude	NUMERIC		
Location	VARCHAR(255)		
Credit	INT(1)	NO	
WIC	INT(1)	NO	
WICcash	INT(1)	NO	
SFMNP	INT(1)	NO	
SNAP	INT(1)	NO	
Organic	INT(1)		
Bakedgoods	INT(1)		
Cheese	INT(1)		
Crafts	INT(1)		
Flowers	INT(1)		
Eggs	INT(1)		
Seafood	INT(1)		
Herbs	INT(1)		
Vegetables	INT(1)		
Honey	INT(1)		
Jams	INT(1)		
Maple	INT(1)		
Meat	INT(1)		
Nursery	INT(1)		
Nuts	INT(1)		

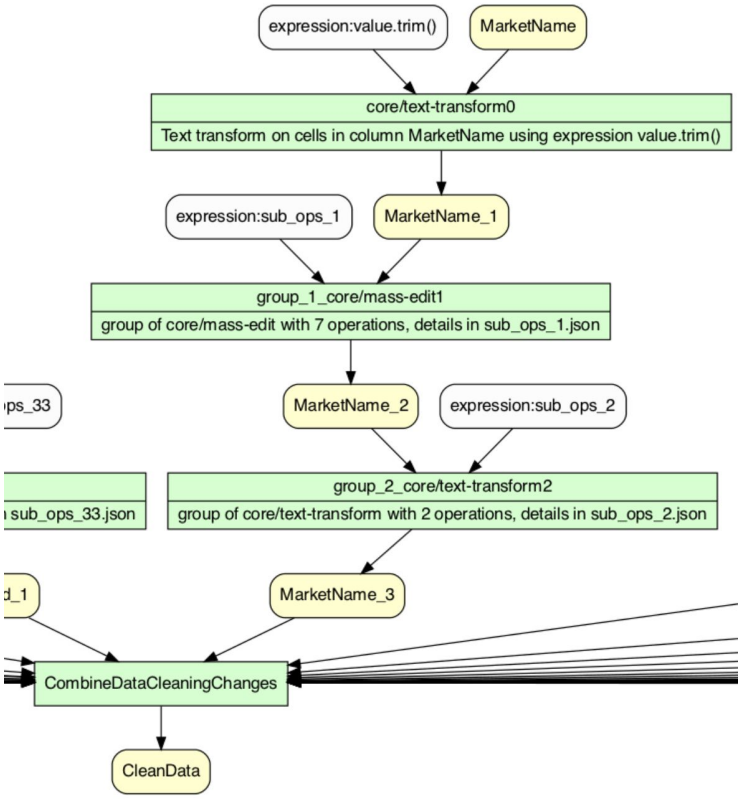
Field	Data Type	Nullable	Key Type
Plants	INT(1)		
Poultry	INT(1)		
Prepared	INT(1)		
Soap	INT(1)		
Trees	INT(1)		
Wine	INT(1)		
Coffee	INT(1)		
Beans	INT(1)		
Fruits	INT(1)		
Grains	INT(1)		
Juices	INT(1)		
Mushrooms	INT(1)		
PetFood	INT(1)		
Tofu	INT(1)		
WildHarvested	INT(1)		
UpdateTime	TIMESTAMP	NO	

Appendix C. Python Workflow Models.





Appendix D. Slice of OpenRefine Workflow Used for Provenance Demo



Appendix E. Cleaned Coordinate Data Plotted Against Map