1. Points received for initial Phase-1 submission : __91__ (out of 100)
2. Our team chooses the following Phase-1 option:
    - [ ] (A) No change to Phase-1 report
    - [X] (B) Improved (or extended) Phase-1 report

CS513: Theory *and Practice of Data Cleaning* – Final Project (Summer 2021)

Submission: Team37 NetIDs: sunilk2, vs24, amk8

Table of Content

Phase 1 Change Report ***2***

Introduction & Background ***3***

Data Set ***3***

Data Source ***3***

Problem Statement ***3***

Description of Dataset ***4***

Data Exploration ***5***

Data Quality Issues ***8***

Use Case ***10***

Other Potential Use Cases (Dataset "Clean Enough") **10**

Unrealistic Use Cases (Dataset will never be good enough.) **11**

Methodology ***11***

Data Cleaning using OpenRefine 12

Integrity Constraints Violations Check using SQLite: 16

Workflows: YesWorkflow 19

5. Conclusions and Future Work ***21***

5.1 Summary **21**

5.2 Challenges **24**

5.3 Next Steps **24**

Phase 1 Change Report

Phase 1 Remarks (TA feedback):	Student Comments / Summary:
Use cases: [-2] Missing a reason why U0 requires “zero data cleaning”. ;	Reasons added under section: Use case, in boldface font.
Data quality list: [-2] lack supportive evidence to explain the data quality issues (e,g., examples, screenshots)	Screen shots added to section: Data Quality Issues.
[-5] no discussion on the connection between data quality issues and use cases (how these data quality issues affect your use case)	Reasons added under section: Data Quality Issues and in Use case, in boldface font.

Introduction & Background

As part of our dedication to open government, transparency and providing high-value data to citizens, USDA had released the Farmers Market Directory listing over a decade ago. The Farmers Market Directory lists markets that feature two or more farm vendors selling agricultural products directly to customers at a common, recurrent physical location. Maintained by the Agricultural Marketing Service, the Directory is designed to provide customers with convenient access to information about farmers market listings to include: market locations, directions, operating times, product offerings, accepted forms of payment, and more. Local farmers markets have proliferated as a means to distribute fresh produce directly to consumers, skipping the costly distribution and packaging step. In this project, we planned to carry out several data cleaning activities which we learnt throughout the course. Few of such activities include exploring the data, cleaning and standardizing the data, checking integrity violation constraints and producing a final cleaned dataset.

Data Set

In our data cleaning project, we explore the US Farmers Market dataset from the USDA Website: https://www.ams.usda.gov/local-food-directories/farmersmarkets. As defined by Wikipedia, a farmers' market is "a physical retail marketplace intended to sell foods directly by farmers to consumers." The dataset is a directory listing of the various farmers markets in the United States, and includes information such as social media accounts, market location, accepted payments, and agricultural products sold.

Data Source

Input dataset:

- farmersmarkets.csv - https://www.ams.usda.gov/local-food-directories/farmersmarkets

Output dataset(s):

- farmersmarkets_output.csv
- farmersMarket_location.csv
- farmeresmarkets_payments.csv
- farmersmarkets_products.csv

Problem Statement

USDA farmers market dataset is a medium sized dataset with some degree of data quality issues. We found a few broad categories of data quality issues. 1. Missing Data 2. Format Issues such as date format 3. Data Type issues such as numeric columns represented as String 4. Data

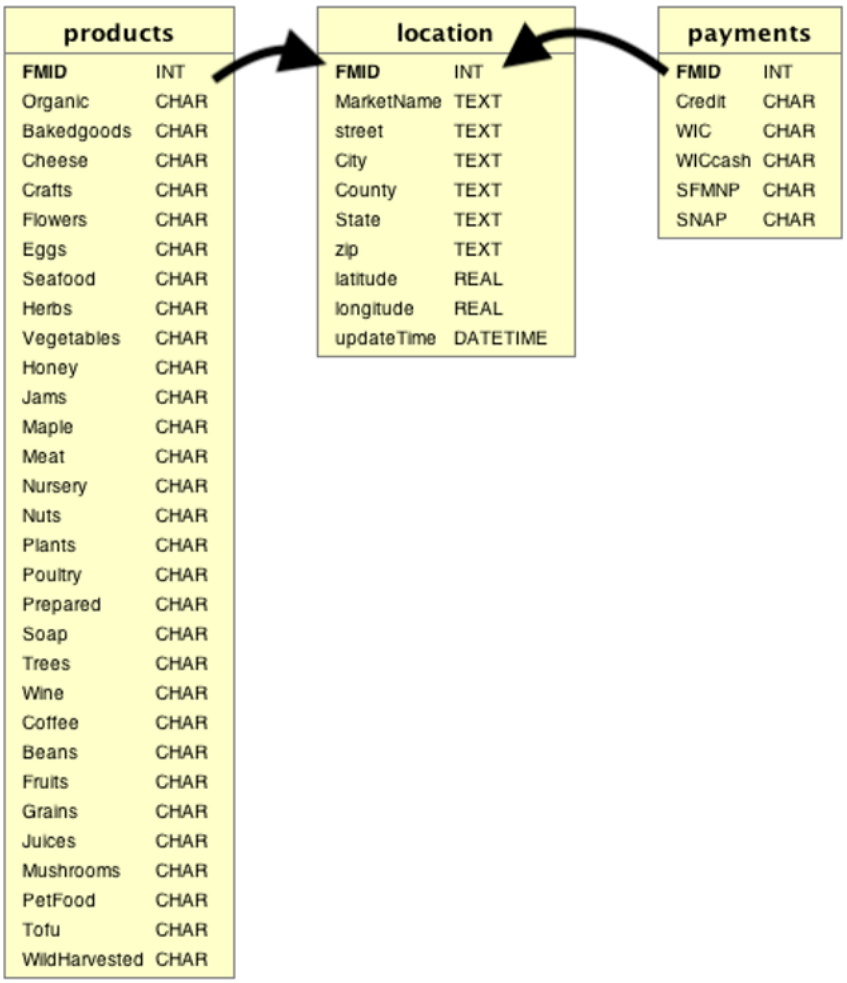
represented in different cases such as upper case, lower case etc., The above-mentioned data quality issues pose problems to uniquely identify the entities, locate the addresses and report the various statistics accurately

Description of Dataset

There are 8675 total observations and 59 columns in this dataset which are described below. The provided html report was generated via a python package called pandas profiling, and allows us to observe some basic, preliminary statistics such number of rows and columns, cardinality, missing values, correlations, etc... as well as the overall schema of the dataset.

FMID - 7 digit integer that uniquely identifies each farmers' market
MarketName - a string containing the name of the farmers' market
Website, Facebook, Twitter, Youtube, Other Media - a string containing URL or other information that identifies the social media site
street, city, County, State, zip - strings that contain data corresponding to the column name that identifies the location of the farmers' market
Season1Date, Season1Time, Season2Date, Season2Time, Season3Date, Season3Time, Season4Date, Season4Time - date fields representing the start date and end date for the given farmers' market or the times in which the farmers' markets are opened
x, y - latitude and longitude coordinates
location - a string describing the location of the farmers' market
Credit, WIC, WICcash, SFMNP, SNAP - Y/N (boolean) character to indicate whether or not a given payment method is accepted
Organic, Bakedgoods, Cheese...PetFood, Tofu, WildHarvested (30 columns) - Y/N (boolean) column to indicate whether or not a given product is offered

The following Entity Relationship shows the schema we developed for our dataset. We broke our cleaned dataset into 4 separate tables: `markets`, `location`, `payments`, and `products`, with the FMID as the primary key for all of them.



Data Exploration

Initial Data Analysis and Exploration:

We performed initial data analysis using python in Jupyter notebook. We looked into the following aspects with the dataset.

1. Fill rate for each attribute
2. Data types of each attribute
3. Missing values of certain interested attributes

The following table shows the percentage of missing values of each column in the dataset, used to identify unusable columns and information.

```
Table of Missing percentages for each attribute
FMID - 0%
MarketName - 0%
```

Website - 40%
Facebook - 56%
Twitter - 89%
Youtube - 98%
OtherMedia - 93%
street - 3%
city - 0%
County - 6%
State - 0%
zip - 11%
Season1Date - 38%
Season1Time - 36%
Season2Date - 95%
Season2Time - 95%
Season3Date - 99%
Season3Time - 99%
Season4Date - 100%
Season4Time - 100%
x - 0%
y - 0%
Location - 66%
Credit - 0%
WIC - 0%
WICcash - 0%
SFMNP - 0%
SNAP - 0%
Organic - 0%
Bakedgoods - 0%
Cheese - 0%
Crafts - 0%
Flowers - 0%
Eggs - 0%
Seafood - 0%
Herbs - 0%
Vegetables - 0%
Honey - 0%
Jams - 0%
Maple - 0%
Meat - 0%
Nursery - 0%
Nuts - 0%
Plants - 0%
Poultry - 0%
Prepared - 0%
Soap - 0%
Trees - 0%
Wine - 0%
Coffee - 0%
Beans - 0%
Fruits - 0%
Grains - 0%
Juices - 0%
Mushrooms - 0%
PetFood - 0%

Tofu - 0%
WildHarvested - 0%
updateTime - 0%

We see in this table that most of the food and items for sale are completely filled, as well as the update time, payment methods, coordinates, FMID, market name, city and state. We see that the social media sites are mostly missing and are likely unusable for general data analysis, as well as the season 2-4 dates and times. We see that an attribute of interest, zip has 11% of values missing which will need to be addressed in order for future analysis. We wanted the location such as Zip code and City to be available 100%. As you see from the above table zip was missing for ~ 11% and city was missing for about 0.47% records. So, we decided to impute the values for zip and city based on other available attributes.

We also examine the number of unique values for our attributes in the table below to see if they fit with our understanding of the data.

	Number of Unique Values
FMID	8675
MarketName	8102
Website	4273
Facebook	3347
Twitter	748
Youtube	122
OtherMedia	495
street	8196
city	5012
County	1490
State	53
zip	6277
Season1Date	2359
Season1Time	1700
Season2Date	378
Season2Time	205
Season3Date	77
Season3Time	45
Season4Date	6
Season4Time	6
x	8533
y	8533
Location	10
Credit	2
WIC	2
WICcash	2
SFMNP	2
SNAP	2
Organic	3
Bakedgoods	2
Cheese	2
Crafts	2
Flowers	2
Eggs	2

```
Seafood              2
Herbs                2
Vegetables           2
Honey                2
Jams                 2
Maple                2
Meat                 2
Nursery              2
Nuts                 2
Plants               2
Poultry              2
Prepared             2
Soap                 2
Trees                2
Wine                 2
Coffee               2
Beans                2
Fruits               2
Grains               2
Juices               2
Mushrooms            2
PetFood              2
Tofu                 2
WildHarvested        2
updateTime        6154
```

We notice here, that FMID has 8675 unique values, and there are 8675 rows in this data set which leads us to believe that FMID will serve as a primary key in its current state, this will be confirmed later with SQLite integrity constraint checking. We also notice that MarketName has 8102 unique values indicating that some markets share a name, and this attribute cannot be used as a key. We also notice here that some attributes which should be binary, such as Organic have 3 values, instead of 2, which may indicate non-uniform representation of null values and should be explored further.

Data Quality Issues

Using the groups above that describe the dataset contents, we describe some of the quality issues that exist in the dataset, such as non-uniform date formats, Data Type issues such as numeric columns represented as String along with data represented in different cases. We also observe non-uniform null value representation in the same columns.

For the social media columns (Website, Facebook, Twitter, Youtube, OtherMedia), most of the rows appear to be missing, and sometimes, in lieu of an URL, a string is provided.

	Facebook	Twitter	Website	Youtube	OtherMedia
Percent Missing Values	56.27%	88.52%	39.95%	98.17%	92.69%

The string could be a Facebook username or Twitter handle, but the representation is not uniform. This would directly affect any use case which would involve analyzing social media accounts of farmers market due to missing data. The image below shows some non-null values from the Twitter column and demonstrates the different types of entries.

```
array(['@12southfrmsmkt', 'https://twitter.com/FarmMarket125th',
       'https://twitter.com/14UFarmersMkt', '14KenFM', '@21acres',
       'https://twitter.com/31MainFarmMarkt', '39North Downtown',
       'http://twitter.com/61market', 'https://twitter.com/peoplesmarket',
       'https://twitter.com/OrganicGrownDr',
       'https://twitter.com/eventsonthefarm',
       'https://twitter.com/AdaFarmersMkt',
       'https://twitter.com/MaureenDatta',
       'https://twitter.com/Agricenter_Intl',
       'https://twitter.com/AgritopiaFarm',
       'https://twitter.com/RebeccaCAFM',
       'https://twitter.com/teenaspridecsa',
       'https://twitter.com/AVFarmersMarket', '@AlohaFarmMarket',
       'https://twitter.com/AltadenaFM',
       'www.twitter.com/PacCoastFarmers',
       'https://twitter.com/amquistation', '@BenWallach1',
       'https://twitter.com/a2market', 'Anthemfm1',
       'https://twitter.com/MBFarmersMarket', 'Argus Farm Stop',
       'Twitter: @ArlFarmersMkt', 'https://twitter.com/arnoldfarmmkt',
```

The location columns that together comprise an address may have some missing values and basically don't contain all 5 components of the address. There may also be leading/trailing white spaces that need to be trimmed, or case conversions that need to be performed, in order to standardize and clean the address data. **This directly affects our main use case in the ability to analyze credit card usage by location, which cannot be done if the location data is not usable.**

Next, for the dates and times, we see that only Season1 tends to be populated. The values are fairly inconsistent as well - some dates are represented using mm/dd/yyyy and some are represented using month name. I've also noticed some date ranges that don't contain the end date. The Season1Time column is also inconsistent. Also, the x and y columns could be better labeled as latitude and longitude, and even the Location column is somewhat poorly because it appears to be a description about the location.

```
Season1Date Ratio Missing: 0.3801729106628242
Season1Time Ratio Missing: 0.363458213256484l4
Season2Date Ratio Missing: 0.9504322766570605
Season2Time Ratio Missing: 0.9521613832853026
Season3Date Ratio Missing: 0.9908933717579 25
Season3Time Ratio Missing: 0.9913544668587896
Season4Date Ratio Missing: 0.9993083573487032
Season4Time Ratio Missing: 0.9993083573487032
```

Meanwhile, for the boolean columns that contain Y/N values, we also see '-' values which could probably be better represented by a null value. In another words, we want the column to be truly boolean with only 'Y' or 'N'.

```
1 farmersmkt.Organic.unique()
```

```
array(['Y', '-', 'N'], dtype=object)
```

Additionally, there are 948 missing values from in the zip code column, about 10.27% of all values. The zip code data is critical to our main use case, so we will scrape data from *tom tom*'s api [api.tomtom.com], where we can use the latitude and longitude data to obtain the zip code.

Finally, for the updateTime column, we only receive year for some of the records, while others contain the full date time. Also, some of the records contain the month name as opposed to the number.

Use Case

Given this dataset and our interest in the modernization of payment methods, we think an interesting use case to explore would be identifying the adoption of credit card usage. We could do this by either some SQL queries and in the end, by creating a map that portrays the acceptance of credit cards by state and percentage of markets that accept credit cards. **The data in its original quality is not completely fit for this use case as there are missing values for the zip attribute which is critical for this analysis, also non-uniform representation of null values in categorical columns would affect this analysis as well, making data cleaning a necessary process for this task.**

Other Potential Use Cases (Dataset "Clean Enough")

Without (or with very little) additional cleaning, these are just a sample of some of the possible use cases possible with our dataset.

- We could determine the most and least popular products that tend to be sold by farmers' markets by summing the existence of 'Y' for each product's column. We could also do this across certain states or zip codes. **The dataset in its original state would provide enough data to extract this information, we see that the various columns indicating whether or not a certain type of food is sold at a particular market (*columns: Bakedgoods, Cheese, Crafts, Flowers, Eggs, etc.*) are almost entirely populated with very few missing values. Although there is a mix between strings 'N' and '-' for negative values, all positive values are marked 'Y'; allowing us to sum the total count of 'Y' values in each sold item column and rank them to find the most popular options.**
- Another use case would be to determine the most popular type of payment options accepted by farmers markets in general (*cash, credit, food stamps, vouchers, etc.*). **The dataset in it's original quality would support this use case because there are very few missing values in the columns indicating payment type accepted; and similar to the sold item columns, all positive cases are marked with a character 'Y' which can be used to summarize the metrics of different payment types accepted at farmer's markets in general.**
- We could explore competition within certain zip codes by looking at the density or count of farmers' markets in certain zip codes, **due to the original dataset containing very few missing values for the column *zip*.**

Unrealistic Use Cases (Dataset will never be good enough.)

- Detailed analysis of social media options for the farmers markets is also highly unlikely due to missing data. For instance, Youtube, Twitter, and Other Media columns have around 90% missing values. If some of these columns were better populated with links, then a web-scraping pipeline could potentially be developed to augment the current dataset.

Methodology

Handling Missing value for Zip Code:

We filtered the records where zip was missing and geo attributes (Longitude & Latitude) available. Then used TomTom's Reverse Geocode API to derive zip code. (Refer farmersmarket_impute_zip.ipynb). After this step, we had 100% coverage of zip code and only 3 records which were not having any geographics attributes such as street, city, state or Longitude, Latitude.

Handling Missing value for City:

We filtered the records where city was missing but zip code available. Then we used USPS '`zip_code_database.csv`' database to impute missing city. (Refer

farmersmarket_impute_zip.ipynb). After this step, we had 100% coverage of City and only 3 records which were not having any geographics attributes such as zip.

The generated new file with imputed values: `farmersmarkets_imputed.csv`

The following code confirms that these values were filled and the columns contain 0 missing value now.

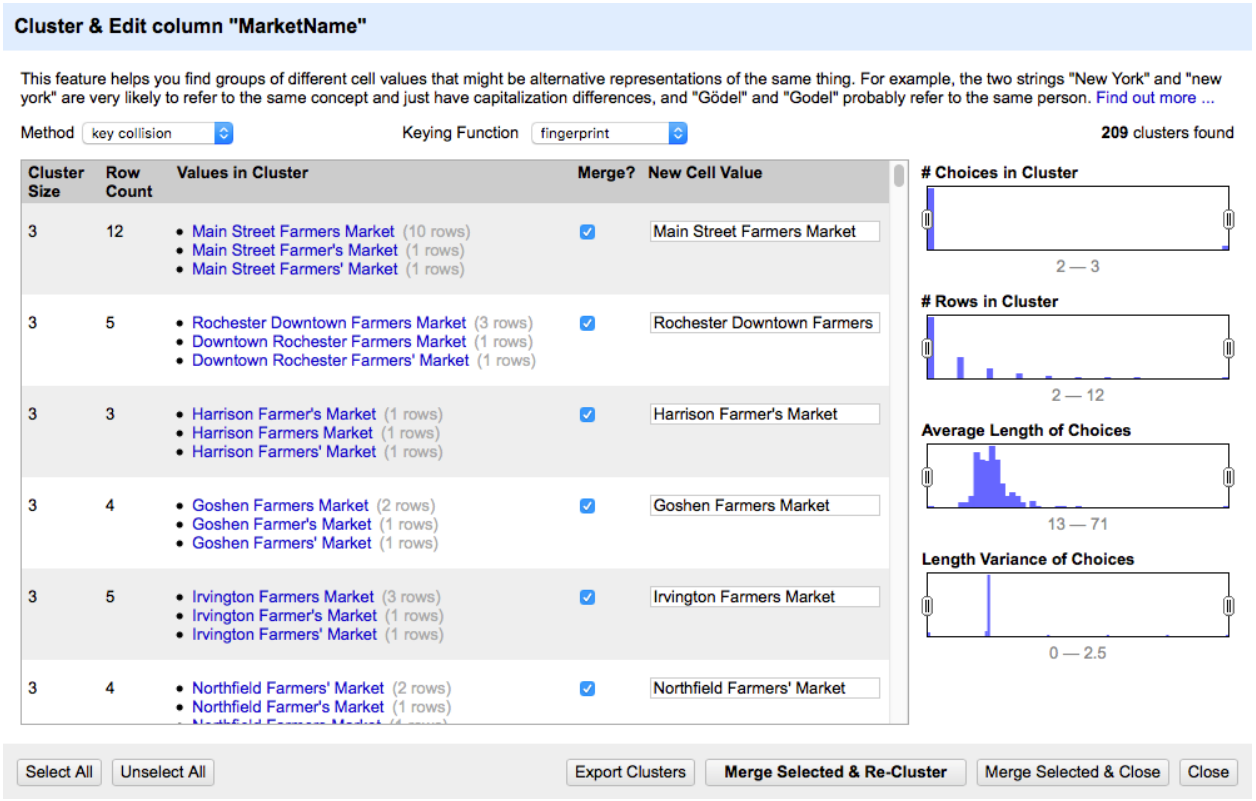
```
for col in df1.columns:
    percentage_missing = np.mean(df1[col].isnull())
    print('{}  - {}%'.format(col.ljust(20), round(percentage_missing*100,2)))
```

```
city - 0%
zip - 0%
```

Data Cleaning using OpenRefine

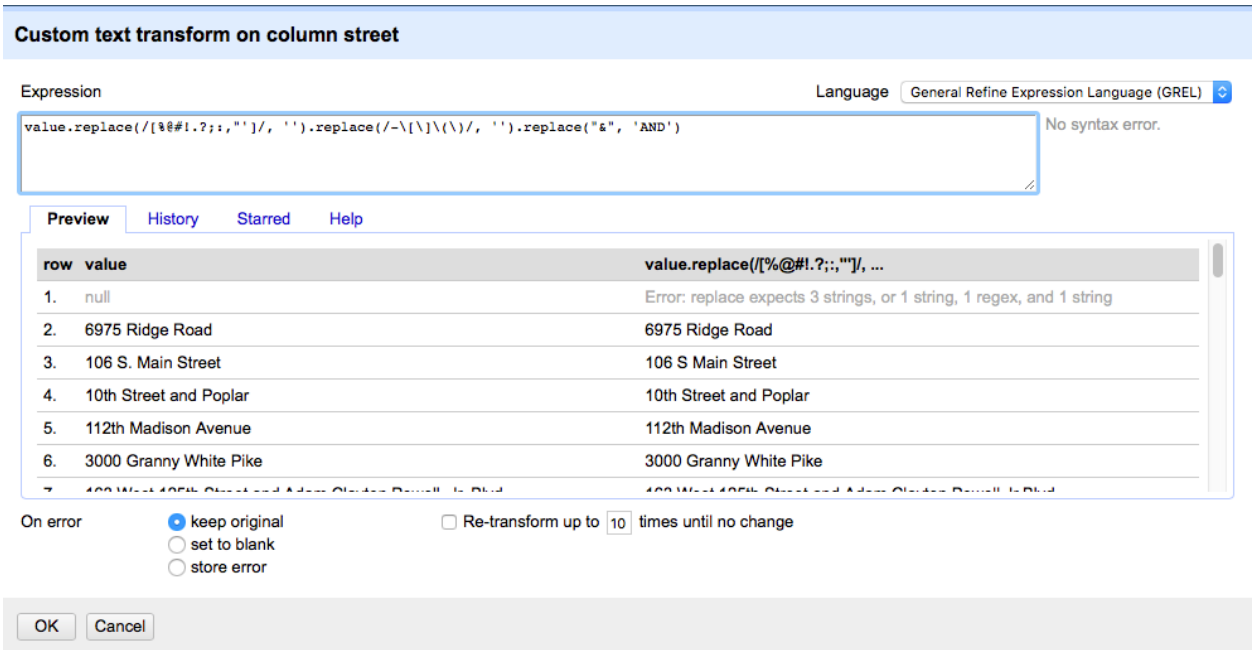
Next, we perform some manual data cleaning on the updated data set [farmersmarkets_imputed.csv] in OpenRefine. This is done to ensure that queries that depend on these values are performed correctly, with the values being uniform. If the format of these values are not uniform the queries performed for our use case may not perform as expected, for instance if we would like to select records within a certain date range, all dates attributes must be in a standard form. Another example would be if we would like to find records in that match a certain location by street, state, or city, we need to make sure that all of the values in these column follow a standard form (not leading or trailing whitespace, consecutive white space or difference in case.)

Step 1. We begin with the MarketName column by first trimming the leading and trailing whitespace and then collapsing any consecutive whitespaces. We identify key collisions within the MarketName column and cluster and merge these names into common strings.

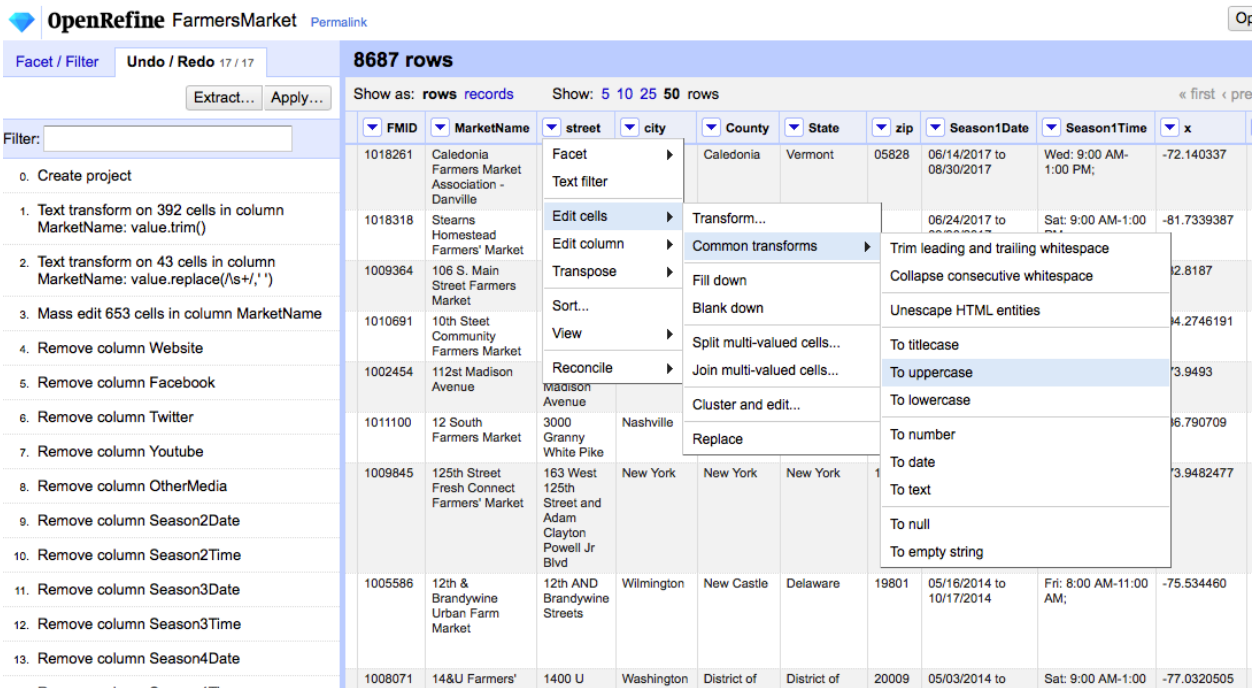


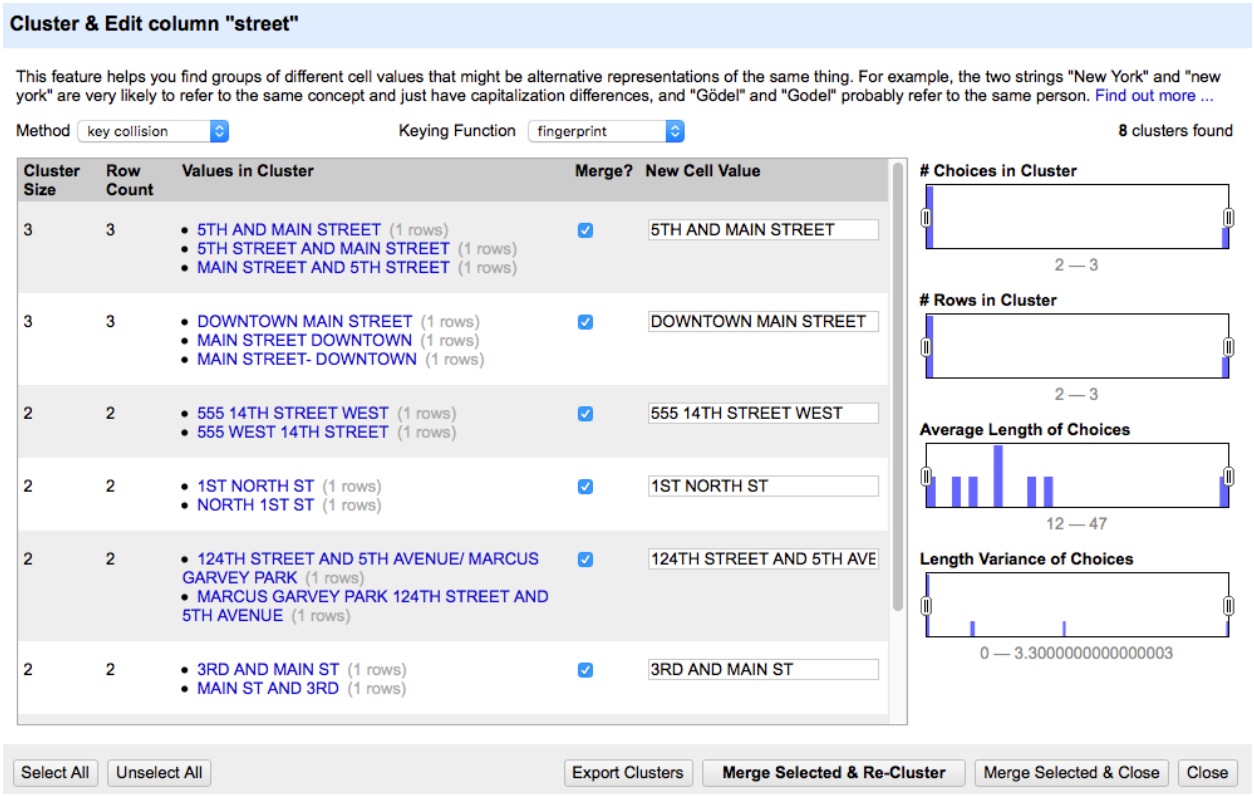
Step 2. The data quality and fill rates for these columns were poor: Website, Facebook, Twitter, Youtube, OtherMedia. For academic purpose, we have applied the regex URL validation Website, Facebook, Youtube and OtherMedia and created new columns appended 1 with original column names.

Step 3. Then, we focus on the location columns - street, city, County, State, and zip.
For street, we substitute the ampersand and ‘And’ with 'AND' for uniform representation. Then, we trim the leading and trailing whitespace and collapsed any consecutive whitespaces and then convert to title case.



Step 4. We repeat the process of trimming the leading and trailing whitespace and collapsing consecutive whitespaces, along with converting to title case for city, County, and State columns.





Step 5. We then check for key collisions on street, city, County and State columns. Both County and State show no key collisions, so no further operations are performed on these columns. There are a few key collisions that our found using the fingerprint keying function which appear to be legitimate collisions, so these instances are clustered and corrected. There are also key collisions identified within the street column which are merged.

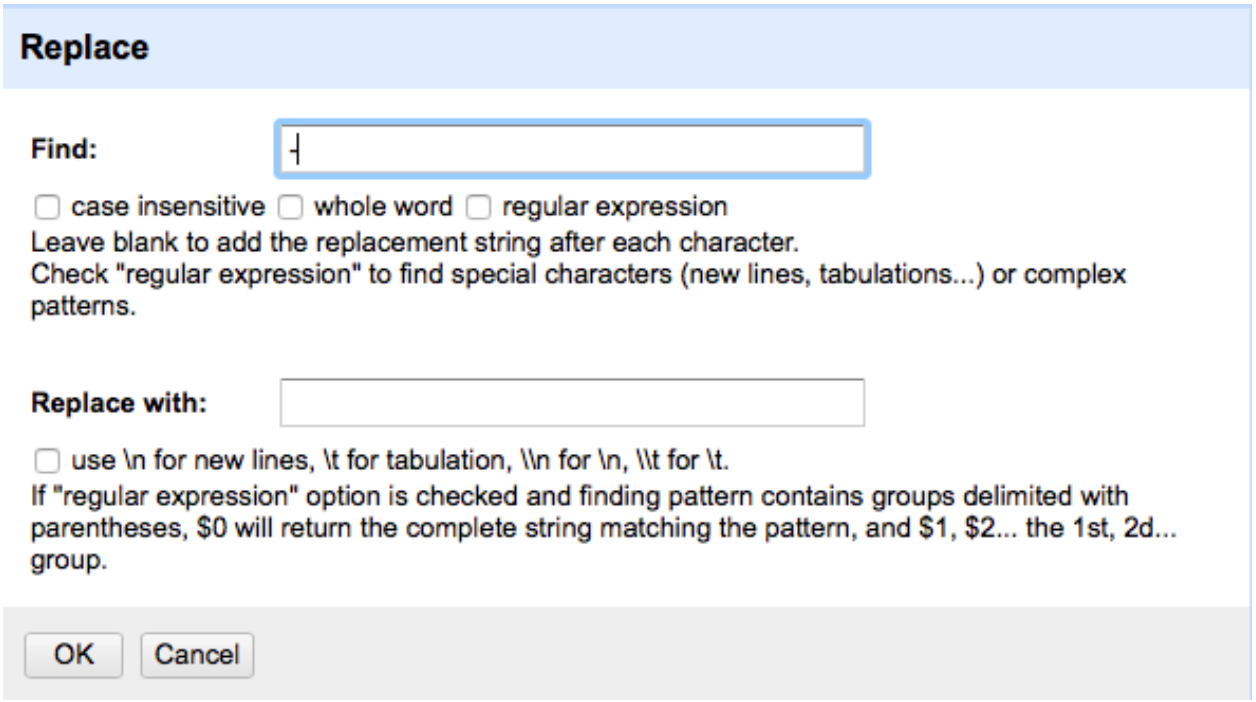
Step 6. Season1Date was freeform text which had value of season start and end date in a verbal format separated by the word “to”. We split this field into two namely: Season1Date_Start and Season1Date_End. We can use this new fields in sqlite to answer queries such as “List the markets which are open in summer or any particular month.
Also, some of the columns populated with Month name where imputed that as proper date with default year. Ex: Season1Date = “July to November” resulted in Season1Date_Start = “07-01-2010” and Season1Date_End = “11/30/2010”.

Step 7. Season1Time was freeform text which had value of season start and end time for days it was open separated by “;”. We split this field into 7 namely: Season1Time 1 to Season1Time 7. We can use this new fields in sqlite to answer queries such as “List the markets which are open at particular time.

Step 8. Important to our analysis later, are the x and y columns, which we rename to latitude and longitude respectively, and then convert to numeric

Step 9. Location column contains landmark details and sparsely filled. We just performed trimming white spaces and collapsing multiple white spaces.

Step 10. For some finishing touches, we remove the occurrence of "-" in the Organic column, so that missing values are just left blank.



Step 11: We also converted the values in the updateTime column to ISO format using the GREL expression: *value.toDate('d/M/y H:m:s')* after trimming and collapsing whitespaces.

The output file name from this OpenRefine step: `farmersmarkets_openrefined.csv`

Integrity Constraints Violations Check using SQLite:

Next, we ensure our integrity constraints hold in the data. This is performed to ensure our queries for our main use case perform as expected. For example, if we would like to use the column FMID as a primary key to identify a certain record, we need to ensure the Primary Key integrity constraint holds for this column.

We created four tables from the cleaned dataset in this jupyter notebook- farmersmarket_data_cleaning.ipynb

```
In [279]: conn = sql.connect('farmersmarkets.db')
          farmers_df.to_sql('farmersmarkets', conn)

In [280]: farmers_df.to_csv("farmersmarkets_openrefined.csv",index=False)

In [292]: Date 1","Season4Date 2","Season4Time","Season4Time 1","Season4Time 2"]].to_sql("markets",conn,if_exists="replace")
          Date 1","Season4Date 2","Season4Time","Season4Time 1","Season4Time 2"]].to_csv("farmersmarkets_output.csv",index=False)

In [293]: eet","city","County","State","zip","Longitude","Latitude","Location"]].to_sql("location",conn,if_exists="replace")
          eet","city","County","State","zip","Longitude","Latitude","Location"]].to_csv("farmersMarket_location.csv",index=False)

In [294]: farmers_df[["FMID","Credit","WICcash","WIC","SFMNP","SNAP"]].to_sql("payments",conn,if_exists="replace")
          farmers_df[["FMID","Credit","WICcash","WIC","SFMNP","SNAP"]].to_csv("farmeresmarkets_payments.csv",index=False)

In [295]: farmers_df[["FMID","Organic","Bakedgoods","Cheese","Crafts","Flowers","Eggs","Seafood","Herbs","Vegetables","Honey","Ja
          ]].to_sql("products",conn,if_exists="replace")
          farmers_df[["FMID","Organic","Bakedgoods","Cheese","Crafts","Flowers","Eggs","Seafood","Herbs","Vegetables","Honey","Ja
          ]].to_csv("farmersmarkets_products.csv",index=False)
```

Then, we develop a few integrity constraints which we ran in SQL/SQL.ipynb notebook. (Refer - farmersmarket-sqlite-IntegrityChecking.ipynb)

- Check that FMID is an appropriate primary key: non-null and unique
- Ensure that data for my use case is non-null (specifically latitude, longitude, state, credit)
- latitude must be in [0,90] and longitude should be [-180, 180]
- Every FMID has single address (street, City, County, State, zip) if it exists
- List all Markets which are open in Summer – i.e., between May - August

The following screen shots provide validation that these integrity constraints hold.

```
# 1. Check that FMID is an appropriate primary key: non-null and unique
ID_NULL_DF = pd.read_sql("select * from farmersmarkets where FMID is NULL",con_1)
ID_NULL_DF
```

index	FMID	MarketName	Website	Website_1	Facebook	Facebook_1	Twitter	Twitter_1	Youtube	...	Season4Date_ismissing	Season4Date 1_ismissing	Season4Date 2_ismissing	S

0 rows × 150 columns

```
#2. Check Uniqueness
print(pd.read_sql("select count(distinct FMID) from farmersmarkets",con_1))
```

```
   count(distinct FMID)
0                  8675
```

```
#3 Check MarketName all populated - NULL Check
MKTNAME_NULL_DF = pd.read_sql("select * from farmersmarkets where MarketName is NULL",con_1)
MKTNAME_NULL_DF
```

index	FMID	MarketName	Website	Website_1	Facebook	Facebook_1	Twitter	Twitter_1	Youtube	...	Season4Date_ismissing	Season4Date 1_ismissing	Season4Date 2_ismissing	S

0 rows × 150 columns

Here we confirm that there is a FMID for each row, and there are 8675 unique values for FMID which is equal to the total amount of rows in this dataset, confirming that each instance has a

unique FMID which can be therefore used as a Primary Key. We also confirm that each row has a MarketName value, helpful for human readability and identification.

Next we confirm that all latitude and longitude values are within the correct range, as confirmed below.

```
#latitude and longitude should be in the interval [0,90], [-180,180]
geo_check_df = pd.read_sql("select m.FMID, m.MarketName, m.Longitude,m.Latitude from farmersmarkets m where (cast(m.Lor
geo_check_df
```

	FMID	MarketName	Longitude	Latitude
0	2000001	Center For Design Practice - Mobile Farmers Ma...	None	None
1	1011689	Charlotte Regional Farmers Market	None	None
2	2000002	Dig It!	None	None
3	1002854	East Goshen Farmers Market	None	None
4	2000004	Farm A La Carte	None	None
5	2000005	Farm Fresh Mobile Market	None	None
6	2000006	Farm To Family	None	None
7	2000007	Farmer's Market Express	None	None
8	2000008	Food Shuttlle	None	None
9	2000009	Freshest Cargo: Mobile Farmers' Market	None	None
10	2000010	Fulton Fresh Mobile Farmer's Market	None	None
11	2000011	Go Fresh Mobile Farmers Market	None	None
12	2000012	Gorge Grown Mobile Farmers Market	None	None
13	2000013	Green Mountain - Mobile Farmers Market	None	None
14	2000014	Greensgrow Farms Mobile Food Delivery System	None	None
15	2000016	Honeybee Mobile Farmers Market	None	None
16	2000017	Hub City Mobile Farmers Market	None	None
17	2000019	Merced County's Mobile Farmers Market	None	None

```
# Active vs Non Active markets
print(pd.read_sql("select count(FMID) from farmersmarkets where Season1Date is not null",con_1))
print(pd.read_sql("select count(FMID) from farmersmarkets where Season1Date is null",con_1))
```

```
   count(FMID)
0         5377
   count(FMID)
0         3298
```

```
# LOCATION (ZIP) CHECK for MARKETS
print(pd.read_sql("select count(FMID) from farmersmarkets where zip is not null",con_1))
print(pd.read_sql("select count(FMID) from farmersmarkets where zip is null",con_1))
```

```
   count(FMID)
0         8672
   count(FMID)
0            3
```

```
# CITY NAME CHECK for MARKETS
print(pd.read_sql("select count(FMID) from farmersmarkets where city is not null",con_1))
print(pd.read_sql("select count(FMID) from farmersmarkets where city is null",con_1))
```

```
   count(FMID)
0         8672
   count(FMID)
0            3
```

```
print(pd.read_sql("select FMID,city,zip,Longitude,Latitude from farmersmarkets where city is null",con_1))
```

```
      FMID  city   zip Longitude Latitude
0  2000007  None  None      None     None
1  2000016  None  None      None     None
2  2000030  None  None      None     None
```

We then confirm that most values which are critical for our use case are not null, (zip and city). We see that there are only 3 missing values for these cases, which would be dropped or ignored.

We then use the clean data to run an example query which can be performed now with the clean data and not with the original data, to demonstrate the usefulness of the data cleaning process. This query could not be performed with the original data being a csv file.

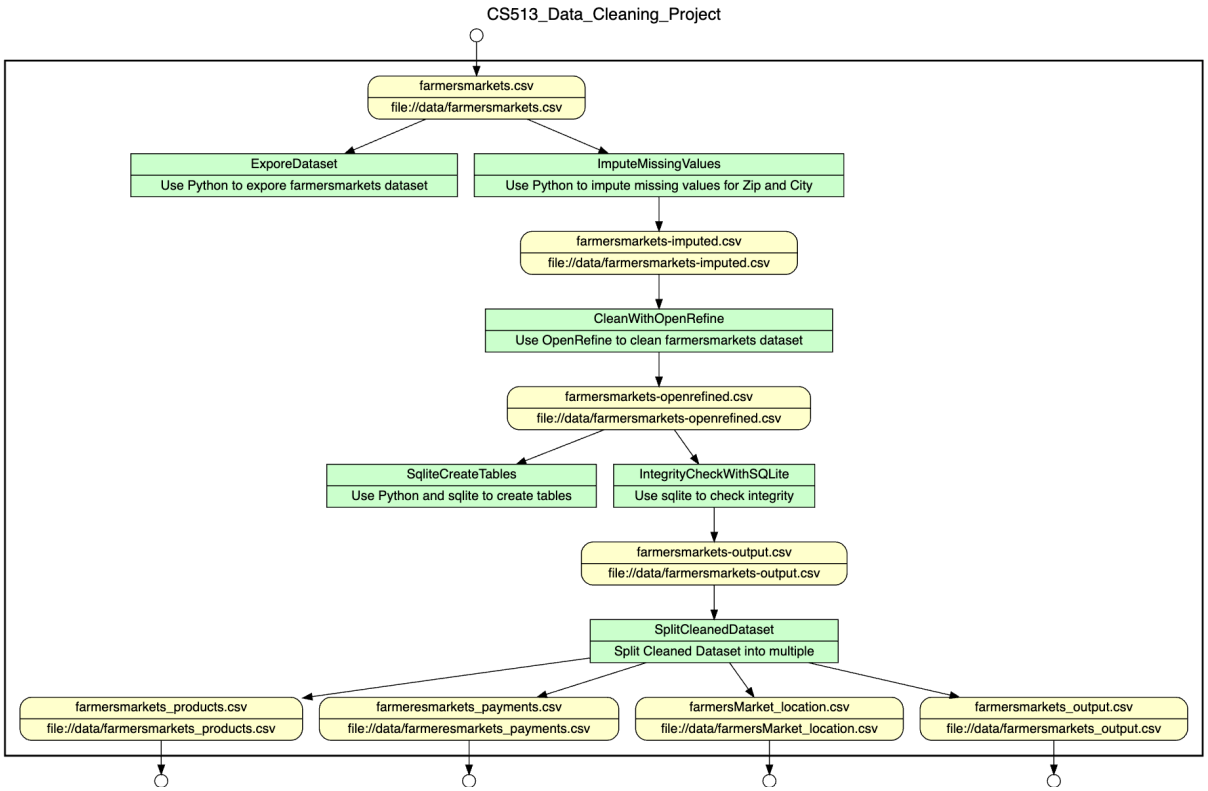
```
# List FarmersMarkets and location details which are open in Summer May - Aug

df = pd.read_sql("select MarketName,street,city,State,zip,Season1Date,strftime('%m',Season1Date_Start) as start, strfti
df.head(20)
```

	MarketName	street	city	State	zip	Season1Date	start	end1
0	39 North Marketplace	Downtown Sparks Victorian Ave	Sparks	Nevada	89431	06/09/2016 to 08/18/2016	06	08
1	84 West Farmers Market (dothan)	None	Dothan	Alabama	36305	06/06/2013 to 08/08/2013	06	08
2	Abbeville Farmers Market	Kirkland Street	Abbeville	Alabama	36310	06/06/2014 to 07/25/2014	06	07
3	Acreage Green Market	6701 140th Ave. North	Loxahatchee	Florida	33470	05/31/2015 to 05/29/2016	05	05
4	Amqui Station Farmers Market	301 B Madison Street	Madison	Tennessee	37115	05/04/2014 to 08/30/2015	05	08
5	Anderson County Farmers Market	402 N. Murray Ave.	Anderson	South Carolina	29621	05/03/2014 to 05/31/2014	05	05
6	Anselma Farmers And Artisans Market	1730 Conestoga Road/rt 401	Chester Springs	Pennsylvania	19425	May to August	05	08
7	Appleton Area Farmers Market	None	Appleton	Minnesota	56208	06/08/2016 to 08/31/2016	06	08
8	Asu Regional Farmers Market	3300 Aggie Road	Jonesboro	Arkansas	72401	05/02/2015 to 05/30/2015	05	05
9	Athens Saturday Market	409 W Green Street	Athens	Alabama	35611	06/04/2016 to 08/27/2016	06	08
10	Auroras Farmers Market East	701 South Eola Road	Aurora	Illinois	60607	07/07/2016 to 08/25/2016	07	08
11	Austintown Farmers Market	6000 Kirk Road (austintown Township Park)	Austintown	Ohio	44515	06/08/2015 to 08/29/2016	06	08
12	Barker Farmers Market	Main St., & Quaker Rd.	Barker	New York	14012	06/12/2014 to 08/28/2014	06	08
13	Beaverton Farmers Market - Wednesday	Hall Blvd.	Beaverton	Oregon	97075	06/17/2015 to 08/26/2015	06	08
14	Bensenville French Market	10 S Center Street	Bensenville	Illinois	60105	06/15/2016 to 08/31/2016	06	08
15	Benton Farmers Market	495 Simpson	Benton	Louisiana	71006	05/31/2015 to 07/26/2015	05	07
16	Berlin Farmers And Artists Market	N State Street And E Huron St	Berlin	Wisconsin	54923	06/07/2016 to 08/30/2016	06	08
17	Bethany Beach Farmers Market	Garfield Parkway And Pennsylvania Avenue	Bethany Beach	Delaware	19930	06/14/2015 to 08/30/2015	06	08

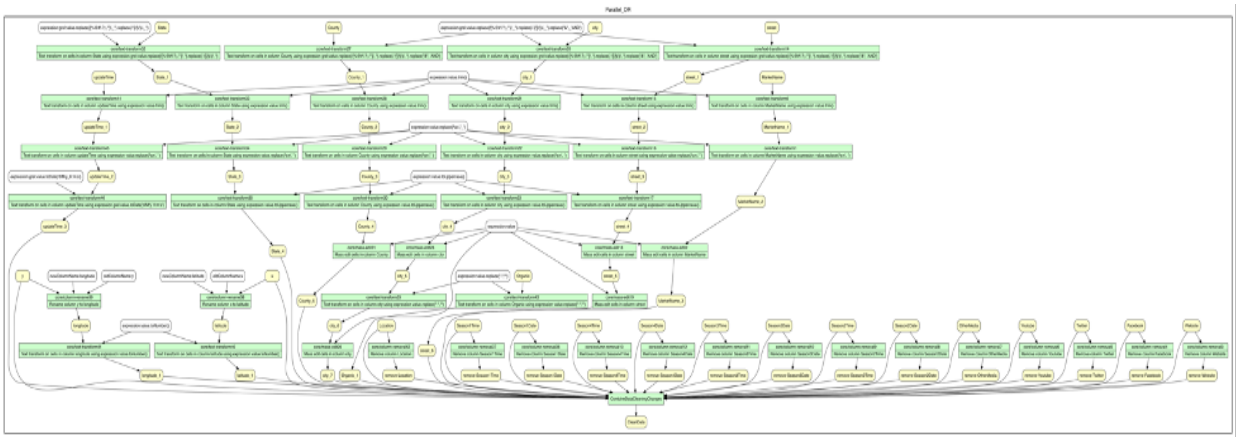
Workflows: YesWorkflow

1. High Level YesWorkflow



We see here an overall workflow of our data cleaning process. We begin with the original dataset and perform data exploration and use an external client tool to fill some missing values. We then use this updated dataset to perform cleaning with OpenRefine, critical to ensure our use case queries perform as expected. For this same reason, we then use the clean dataset to check for integrity constraints using SQLite. We then separate this clean dataset into different csv files in order to use with a database system.

2. Detailed Yesworkflow



5. Conclusions and Future Work

5.1 Summary

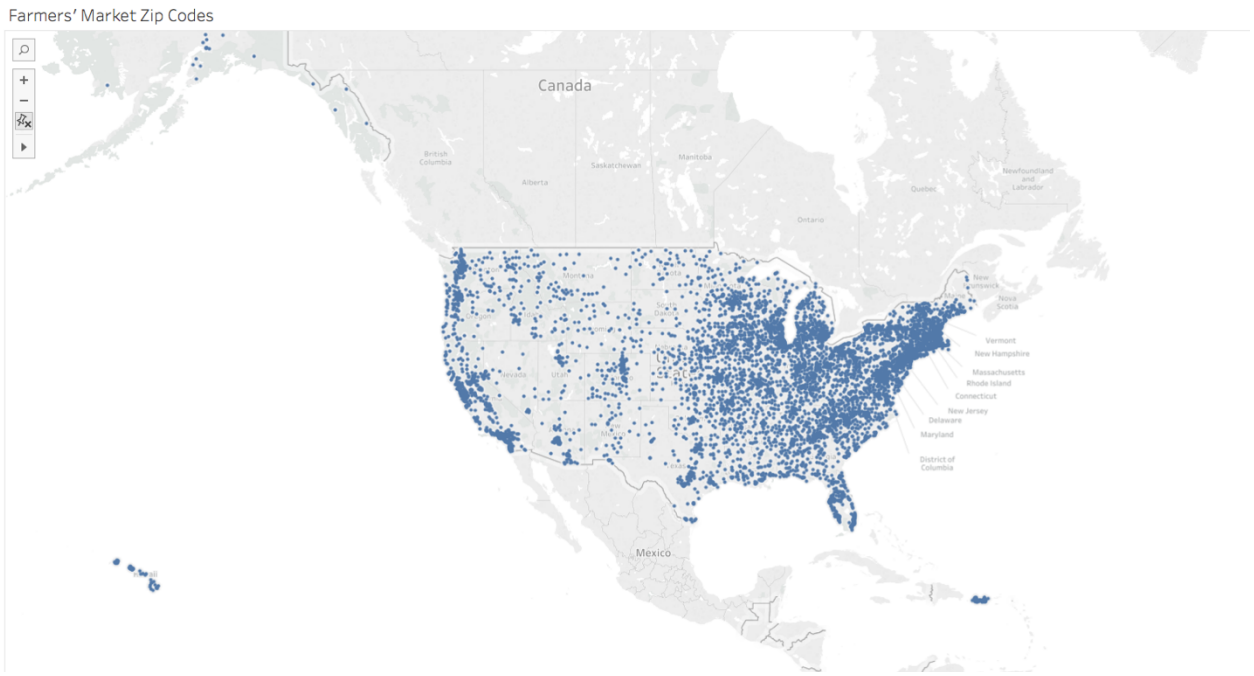
After our data cleaning exercise in OpenRefine, we are finally able to dive into our use cases. Our use case was to explore the adoption of credit card usage of the farmers' markets in our dataset. In order to accomplish this task, we used external client API's in order to fill missing values, used open refine to standardize and merge matching data represented in different ways, and used SQLite to check and assert integrity constraints, our process was captured through YesWorkflow.

We found that in some cases, where the initial dataset would appear to contain useful data, such as social media sites in this case, upon deeper examination, these columns were too bare in order to perform useful analysis on social media sites of farmers markets. Although it would appear to contain useful information in this case on first glance, the dataset was not actually useful in this case.

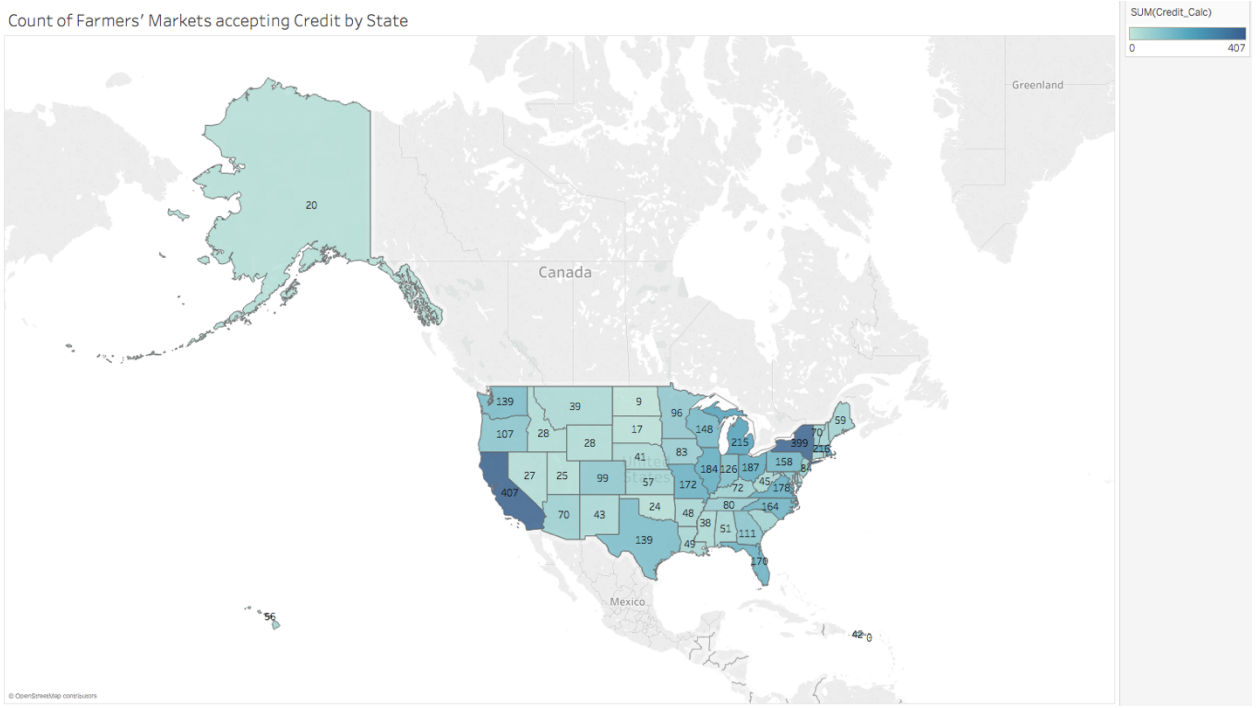
We also found that although some data was missing, such as zip code in this case, we were able to use external tools in order to obtain the missing values based on other attributes (longitude, and latitude).

We also found the YesWorkflow representation to be extremely helpful through this process, where there is a mix of automated and manual workflows. In this case the YesWorkflow helps in our understanding of the different data files that are generated throughout this process and a map of the manual and automated steps that were performed.

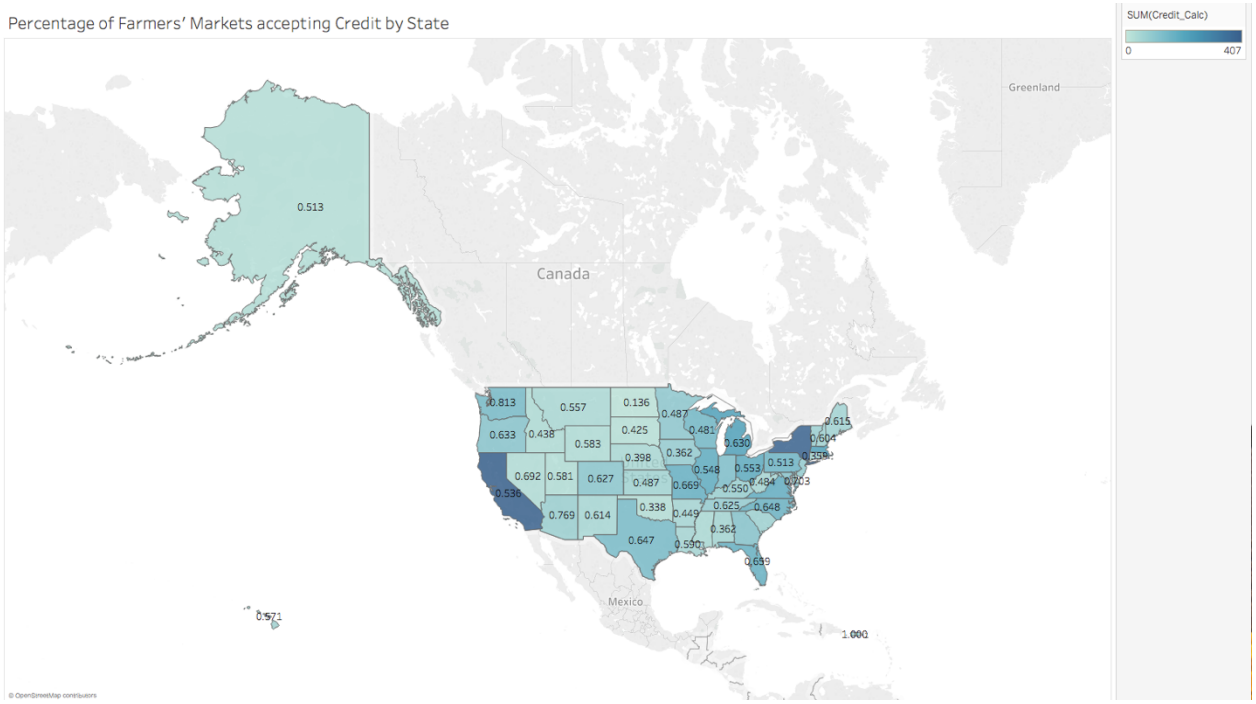
We visualized the cleaned dataset through Tableau to give ourselves a few views. The first one plots the zip codes corresponding to the farmers' markets, and it is essentially like a density plot that allows us to see that some of the more densely populated regions, for example in the northeast, have many farmers' markets, while the midwest and Alaska appears to be sparser.



Next, we see exactly how many farmers' markets each state has, and it is no surprise that more heavily populated states such as California and New York are shaded darker.

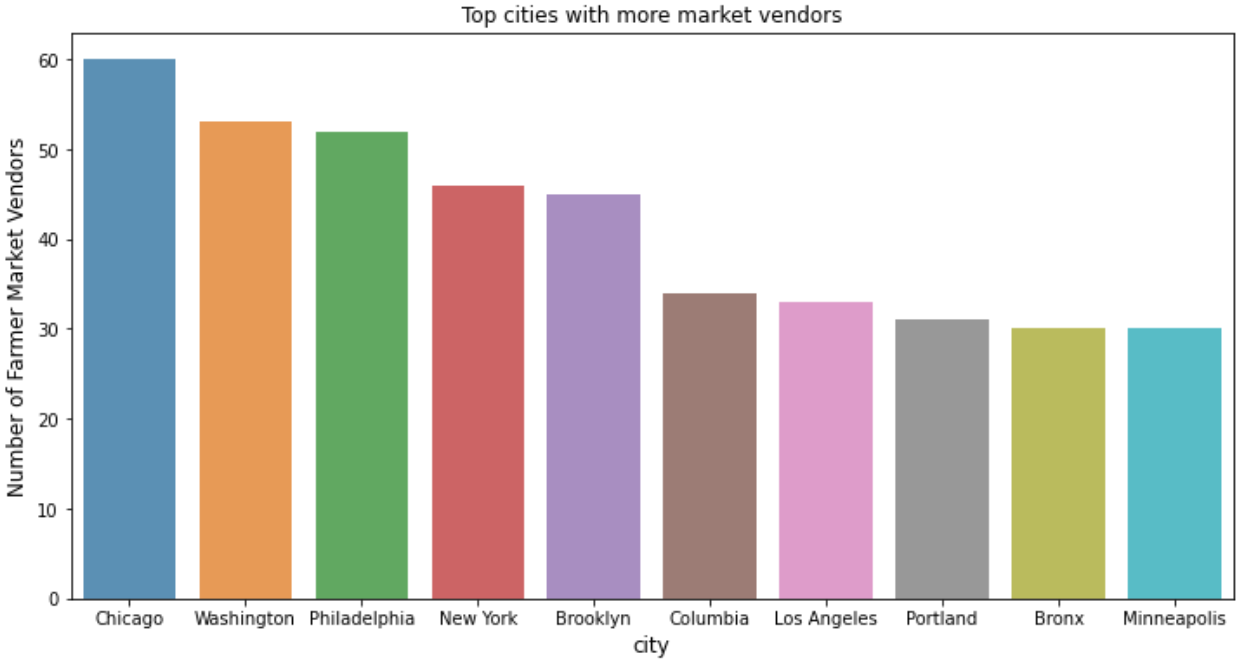


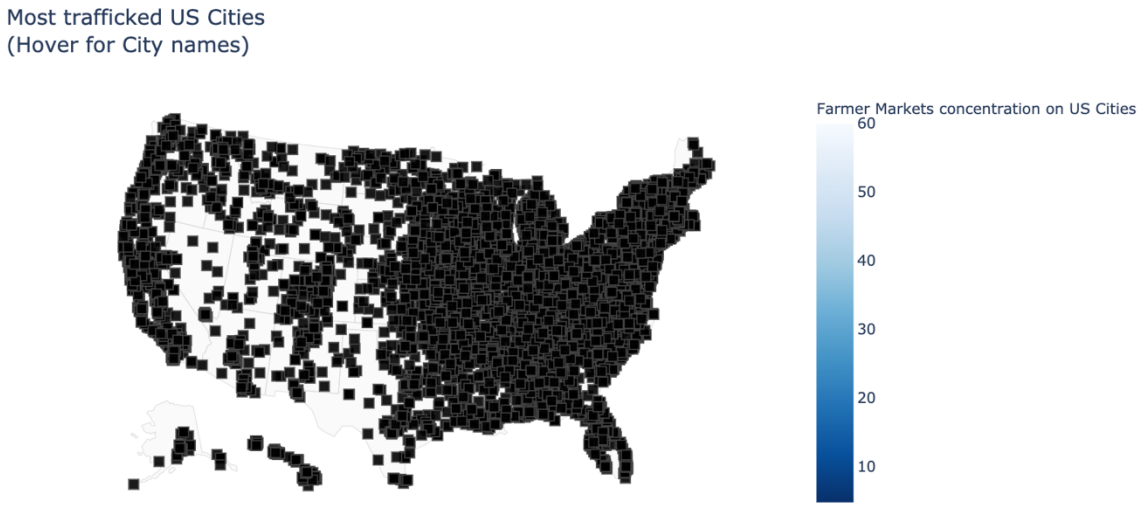
However, a more appropriate view that allows us to understand how much each state has adopted/accepted credit card usage is below. Here, we depict the percentage of farmers' markets that accept credit cards. The color intensity is the same as in the previous view which allows us to compare the overall number of farmers' markets between states.



Based off of the provided descriptions in this report, it should be very clear to the client what we have changed from the original dataset, as well as the challenges we faced in dealing with certain quality issues, and how we chose to resolve them.

Top Cities with more farmer Markets





5.2 Challenges

The most challenging problem within data cleaning and curation remains the correction of inconsistent values and invalid entries. In many cases, the available information on such anomalies is limited and insufficient to determine the necessary transformations or corrections, leaving the deletion of data, though, leads to loss of information; this loss can be particularly costly if there is a large amount of deleted data. Because of this reason we haven't deleted any data from the original dataset. Also, many farmer markets were mobile, so it was not possible to add an accurate geographic location. The social media information were inaccurate and many didn't exist.

5.3 Next Steps

We have listed a few other use cases that our farmers' market data supports, and with additional time, they could certainly be explored. For instance, we could delve into the specific product offerings of the farmers' markets and their distributions by region. We could also look into combinations of product offerings to determine whether people could get all their shopping done at specific farmers' market locations. We could also look into the season1Date to get a sense for how long some of these farmers' markets have been around for. There are many more questions that could be answered, and with some more time, the Tableau visualizations or dashboards could be advanced. It was nice that the dataset included location data that allowed us to utilize the map plot, but other visualization types could be used. Predictive models could also potentially build (e.g. predict whether or not credit card is accepted based on location and product offerings), but some additional work with OpenRefine or Python might be necessary to further prepare the dataset.