CS 513 Theory and Practice of Data Cleaning

Final Project Report

Team Name: Dream of Data

Mondal, Valentina vmondal2@illinois.edu

Panchal, Harsh Mahendrakumar hmpanch2@illinois.edu

Balasubramanian, Sembian sembian2@illinois.edu

Table of Contents

1. Overview and initial assessment of the dataset [25%]

a. A clear description of the structure and content of the dataset [3%]

Organization:

U.S. Agricultural Marketing Service creates domestic and international marketing opportunities for U.S. producers of food, fiber, and specialty crops. AMS also provides the agriculture industry with valuable services to ensure the quality and availability of wholesome food for consumers across the country.

The Farmers Market Directory lists (https://www.ams.usda.gov/local-food-directories/farmersmarkets) markets that feature two or more farm vendors selling agricultural products directly to customers at a common, recurrent physical location. The U.S. Agricultural Marketing Service maintains the directory listing designed to provide customers with convenient access to farmers' market listings to find operations offering locally grown markers:

The listings include some key details like the market locations, directions, operating times, product offerings, accepted forms of payment, and more. The Farmers Market Directory listing website (Farmers Market, 2020) provides an option for export to CSV format. Additionally, it also has API access, and the listing does not provide a dictionary of all fields or details about how the data is entered and validated.

Architecture:

The Farmers market listing contains 59 columns. The first column FMID is an integer column and can be used as a unique identifier for each farmers' market. The second column MarketName is a text column which allows special characters as well. The next set of columns – Website, Facebook, Twitter, Youtube and OtherMedia allows URL to their website and/or social media pages/handles and contains NULL values. The data in these columns looks legible but contains blank or NULL values as well. The next five columns are related to location of farmers' market – street, city, country, state and zip. Next set of columns are related to seasons – date and time period, this provides market hours and days when it is open. The next three columns – x, y and location, represents latitude, longitude and additional details about market's location. The rest columns except last one (updateTime) are Boolean values (Y or N) representing availability of different products in a market. The last column updateTime is a timestamp column, represents when a record was inserted/updated.

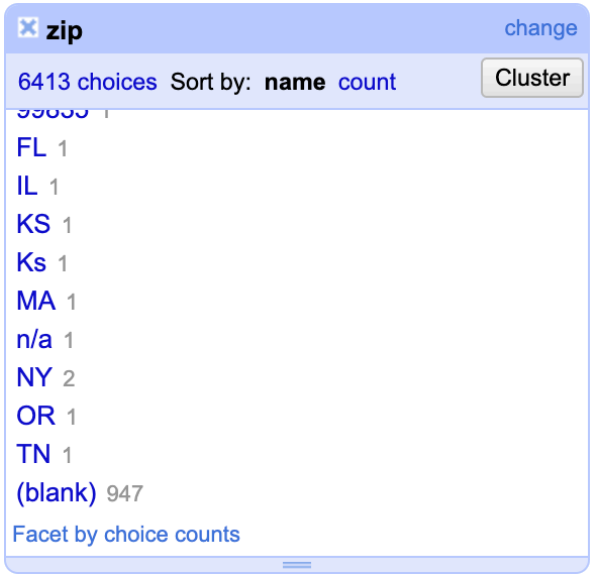
b. A comprehensive list of data quality issues [7%]

This dataset contains lot of quality issues like inconsistent formatting and data values, repetitive information with different column names and redundant values etc. This might be because of lack of validation rules for each column and not following any ISO formatting. We could classify data quality issues using dimensions of data quality:

1. Uniqueness: FMID seems to be unique id associated with each row in the dataset, but it needs to be validated
2. Validity: The columns related to seasons have inconsistent granularity of time period – certain values are date1 to date2 and other month1 to month2.
3. Accuracy:
    i. MarketName is a free text column and allows special characters and hence, this column contains consecutive spaces, leading and trailing spaces and names which are technically same but have few different characters making them look different.

ii. Latitude and Longitude are stored with column names x and y, column names aren't meaningful.

4. Completeness
   The columns related to URL don't have validation; making them allow blank, NULL and text instead of URL.
5. Consistency
   The set of columns related to location have leading and/or trailing white spaces, values in column zip contains more than 940 blank values and two letter state names or "-".



Location column supposed to provide specific geographic location of a market, but it contains irrational values like other, educational institution, healthcare institution and blanks.

The columns related to availability of a product contains not only Y or N values, but it contains blank or "-".

6. Timeliness
   updateTime columns should represent timestamp but it has inconsistent formatting and values like year, MON DD YYYY HH: MM: SS AM/PM, MM/DD/YY HH: MM: SS AM/PM. Moreover, we don't have enough information on how this dataset can be consumed (i.e. API) and can be used in data pipeline as real-time feed.

c. Identifying a feasible use case and the essential data cleaning goals [10%]

This dataset can be used to answer questions like availability of Farmers' Market and markets with availability of group of grocery items like fruits, vegetables, meat etc.

Data Cleaning Goals-

ID	Column Name	Steps	Desired outcome
1	FMID	1.1 Convert to Numbers	FMID needs to be 7-digit unique number to use it as Primary Key
2	Market Name	2.1 Remove trailing and leading spaces including consecutive spaces	Market Names are accurate

		2.2 Remove duplicates using clustering 2.3 Set data type to Text. 2.4 Convert special character '&' to 'and'	
3	Website	3.1 Remove trailing and leading spaces including consecutive spaces3.2 Remove duplicates using clustering 3.3 Convert to all URL to lowercase 3.4 Validate URL	Consistent web URL structure "http(s)://(www.)" Spaces in URL are ignored Leave blank URL's
4	Facebook	4.1 Remove trailing and leading spaces including consecutive spaces 4.3 Remove all duplicates using clustering	Leave blank URL's Spaces in URL are ignored
5	Twitter	5.1 Remove trailing and leading space including consecutive spaces 5.3 Remove all duplicates using clustering	Leave blank URL's Spaces in URL are ignored
6	Youtube	6.1 Remove trailing and leading spaces including consecutive spaces 6.2 Remove duplicates using clustering	Leave blank URL's Spaces in URL are ignored
7	OtherMedia	7.1 Remove trailing and leading spaces including consecutive spaces	Leave blank URL's Spaces in URL are ignored
8	street	8.1 Remove trailing and leading spaces including consecutive spaces 8.2 Identifying redundant 8.3 Change street names case sensitive to title case 8.3 Use clusters to standardize street names with common names (e.g. St > Street) 8.4 Rename column "street" to "Street" to be consistent with other column names	Street names are consistent Ignore or replace blank street names with NULL
9	city	9.1 Remove trailing and leading spaces including consecutive spaces 9.2 Remove duplicates and case sensitive using clustering	City names are consistent
10	County	10.1 Remove trailing and leading spaces including consecutive spaces 10.2 Remove duplicates and case sensitive using clustering	Country names are consistent
11	State	11.1 Remove trailing and leading spaces including consecutive spaces 11.2 Remove duplicates and case sensitive using clustering	State names are consistent
12	zip	12.1 Convert to Numbers 12.2 Remove values other than numbers (10 nonnumeric value like FL, IL, OR and 947 blank value) 12.3. Standardize Zip code to 5 digit and split (99999-99999" > (99999)	The values in Zip codes must be numbers (Assumption - The dataset is of the United States)
13	Season1Date	13.1 Remove trailing and leading spaces including consecutive spaces 13.2 Create new columns From and To based on	Consistent From and To Dates

		separator value “to” 13.3 Convert to ISO Standard Date (or only Month) and remove default timestamp	New columns “Season1FromDate” & “Season1ToDate”
14	Season1Time	14.1 Remove trailing and leading spaces 14.2 Convert to lowercase 14.3 Split into several columns based on the separator “;”	Consistent hours of store
15	Season2Date	15.1 Remove trailing and leading spaces including consecutive spaces 15.2 Create new columns From and To 15.3 Convert to ISO Standard Date (or only Month) and remove default timestamp	Consistent From and To Dates New columns “Season2FromDate” & “Season2ToDate”
16	Season2Time	16.1 Remove trailing and leading spaces 16.2 Convert to lowercase 16.3 Split into several columns based on the separator “;”	Consistent hours of store
17	Season3Date	17.1 Remove trailing and leading spaces including consecutive spaces 17.2 Create new columns From and To 17.3 Convert to ISO Standard Date (or only Month) and remove default timestamp	Consistent From and To Dates New columns “Season3FromDate” & “Season3ToDate”
18	Season3Time	18.1 Remove trailing and leading spaces 18.2 Convert to lowercase 18.3 Split into several columns based on the separator “;”	Consistent hours of store
19	Season4Date	19.1 Remove trailing and leading spaces including consecutive spaces 19.2 Create new columns From and To 19.3 Convert to ISO Standard Date and remove default timestamp	Consistent From and To Dates New columns “Season4FromDate” & “Season4ToDate”
20	Season4Time	20.1 Remove trailing and leading spaces 20.2 Convert to lowercase 20.3 Split into several columns based on the separator “;”	Consistent hours of store
21	x	21.1 Convert to Numbers 21.2 Rename column to Latitude	Accurate Latitude
22	y	22.1 Convert to Numbers 22.2 Rename column to Longitude	Accurate Longitude
23	Location	23.1 Remove trailing and leading spaces including consecutive spaces	Consistent location description values
24	Credit	24.1 Remove trailing and leading spaces 24.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
25	WIC	25.1 Remove trailing and leading spaces 25.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
26	WICCash	26.1 Remove trailing and leading spaces 26.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
27	SFMNP	27.1 Remove trailing and leading spaces 27.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
28	SNAP	28.1 Remove trailing and leading spaces	Consistent Boolean values

		28.2 Replace values to NULL if other than “Y” or “N”	
29	Organic	29.1 Remove trailing and leading spaces 29.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
30	Bakedgoods	30.1 Remove trailing and leading spaces 30.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
31	Cheese	31.1 Remove trailing and leading spaces 31.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
32	Crafts	32.1 Remove trailing and leading spaces 24.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
33	Flowers	33.1 Remove trailing and leading spaces 33.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
34	Eggs	34.1 Remove trailing and leading spaces 34.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
35	Seafood	35.1 Remove trailing and leading spaces 35.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
36	Herbs	36.1 Remove trailing and leading spaces 36.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
37	Vegetables	37.1 Remove trailing and leading spaces 37.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
38	Honey	38.1 Remove trailing and leading spaces 38.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
39	Jams	39.1 Remove trailing and leading spaces 39.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
40	Maple	40.1 Remove trailing and leading spaces 40.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
41	Meat	41.1 Remove trailing and leading spaces 41.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
42	Nursey	42.1 Remove trailing and leading spaces 42.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
43	Nuts	43.1 Remove trailing and leading spaces 43.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
44	Plants	44.1 Remove trailing and leading spaces 44.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
45	Poultry	45.1 Remove trailing and leading spaces 45.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
46	Prepared	46.1 Remove trailing and leading spaces 46.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
47	Soap	47.1 Remove trailing and leading spaces 47.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
48	Trees	48.1 Remove trailing and leading spaces 48.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
49	Wine	49.1 Remove trailing and leading spaces 49.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
50	Coffee	50.1 Remove trailing and leading spaces 50.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
51	Beans	51.1 Remove trailing and leading spaces 51.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
52	Fruits	52.1 Remove trailing and leading spaces 52.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values

53	Grains	53.1 Remove trailing and leading spaces 53.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
54	Juices	54.1 Remove trailing and leading spaces 54.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
55	Mushrooms	55.1 Remove trailing and leading spaces 56.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
56	PetFood	56.1 Remove trailing and leading spaces 56.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
57	Tofu	57.1 Remove trailing and leading spaces 57.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
58	WildHarvested	58.1 Remove trailing and leading spaces 58.2 Replace values to NULL if other than “Y” or “N”	Consistent Boolean values
59	updateTime	59.1 Remove trailing and leading spaces 59.2 Convert to ISO Standard Date	Consistent timestamp value

d. Fitness of Use

Columns	Use/Application
FMID Market Name Season1Time Season2Time Season3Time Season4Time	Market Hour of Operation Per Season?
FMID MarketName Website Facebook Twitter Youtube OtherMedia	Market affiliation in online?
FMID Market Name Season1Date Season2Date Season3Date Season4Date	Markets by Seasonality (Summer/Winter/Fall)?
FMID MarketName Latitude Longitude Zip	Availability of Market in Proximity?

FMID MarketName Credit WIC WICash SFMNP SNAP	Market having specific payment type?
FMID MarketName 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	Products and their type of Products in a specific Market? (Products can be classified into the following categories - a) Organic b) Beverages c) Dairy d) Meat e) Produce f) Dry Goods g) Cleaners h) Canned/Jarred i) Garden j) Other
FMID MarketName updateTime	Market Updated Latest or Oldest?

e. Identifying use cases for which the dataset is already clean and use cases for which it will never be clean enough or usable [5%]

Overall, column zip, x and y are quite clean, and it can be used to map out density/ heat map of availability of Farmers' Market across the United States.

The columns related to Market's website and social media pages are not reliable and can't be used for any kind of analysis. If we want to derive some additional information like how many people have liked a page on Facebook and Twitter, promotional offers on social media, pickup availability, etc. would be difficult.

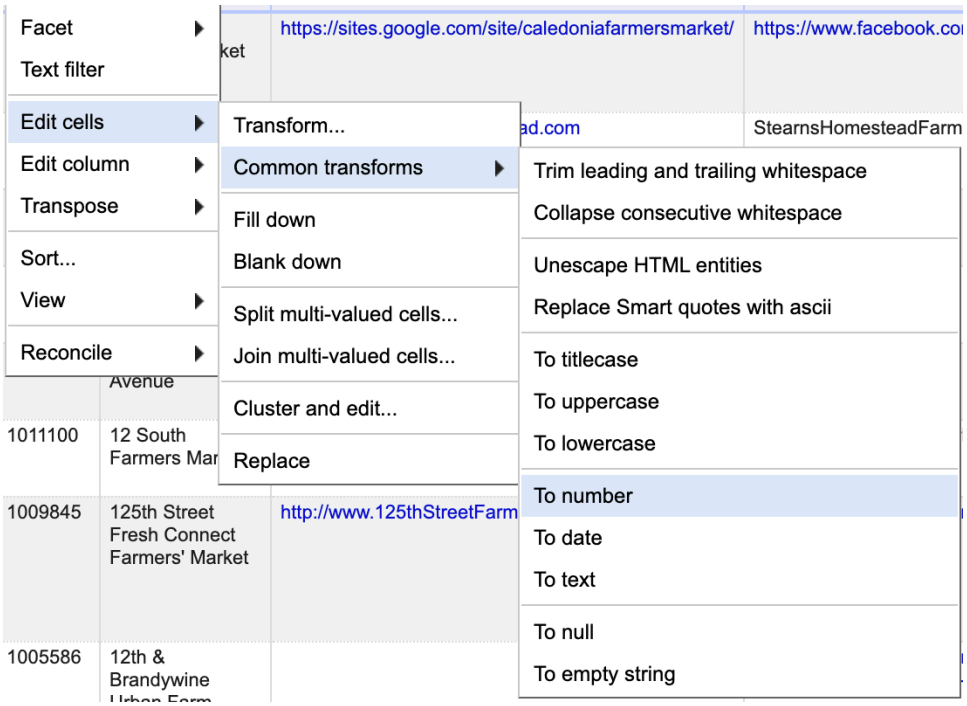
2. Data cleaning with OpenRefine (and other tools) [40%]

a. Identifying the appropriate data cleaning steps for the use case

b. A clear description of the data cleaning steps with supplemental information

1. FMID

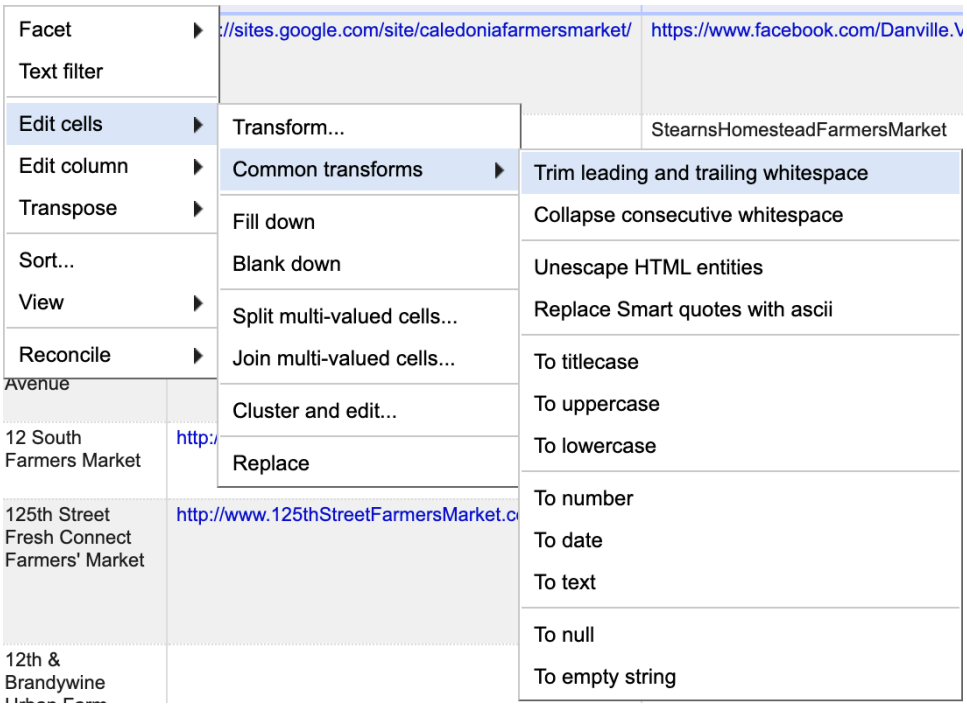
FMID seems to be unique for each row. To validate it, we converted this column into Number and analyzed the results using text facet. The count of each number is "1". The total number of FMID were 8806 and all of them were converted to number, yielding in 8806 unique FMID.



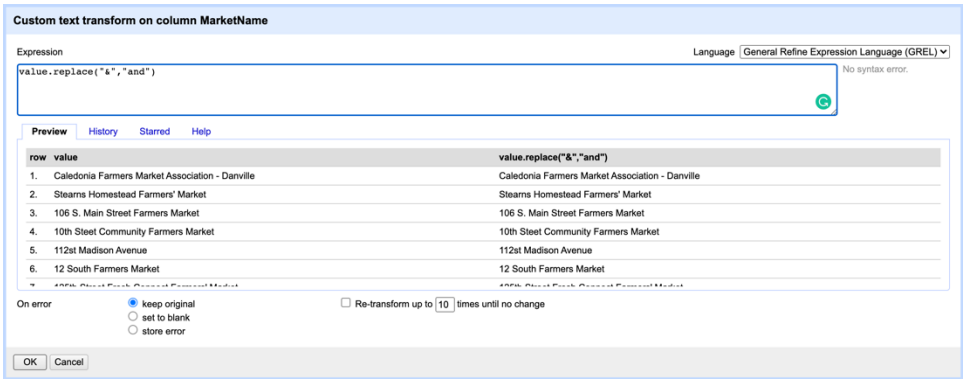
2. Market Name

i) Trim leading and trailing whitespace – The number of cells affected by this transformation were 410

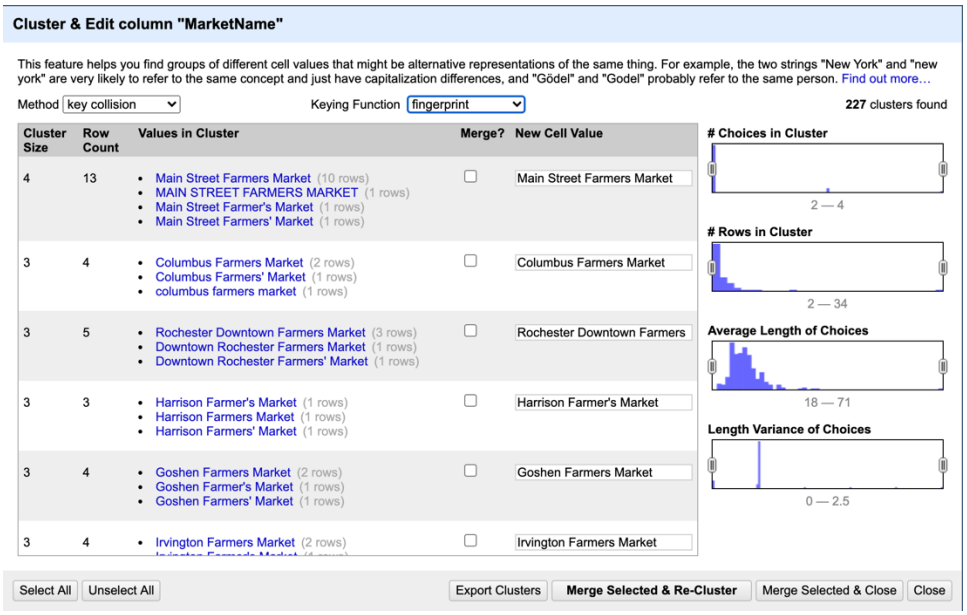
ii) Collapse consecutive whitespace – The number of cells affected by this transformation were 51



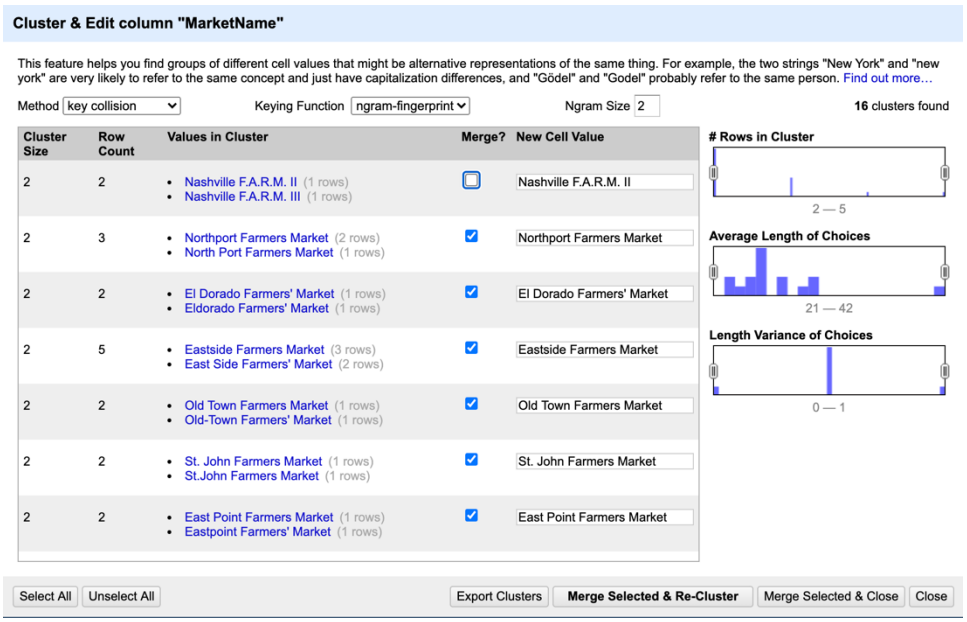
iii) Special character “&” in MarketName values were replaced with “and”. This transformation affected 192 cells.



iv) Since this is a text column, there’s a possibility of having same names with minor difference. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 661 cells.



v) We also transformed this column with Key Collision method but with a variant of fingerprint keying function – ngram-fingerprint with Ngram Size 2. This affected 30 cells.



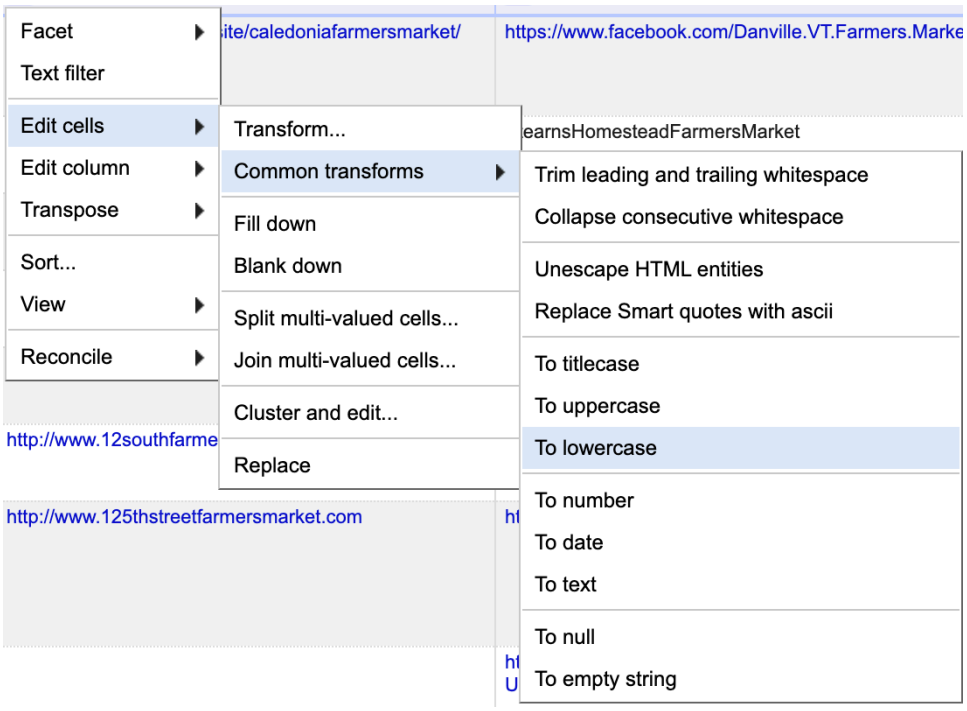
After applying all these steps, now we have clean and consistent MarketName without any duplicates for further use.

3. Website

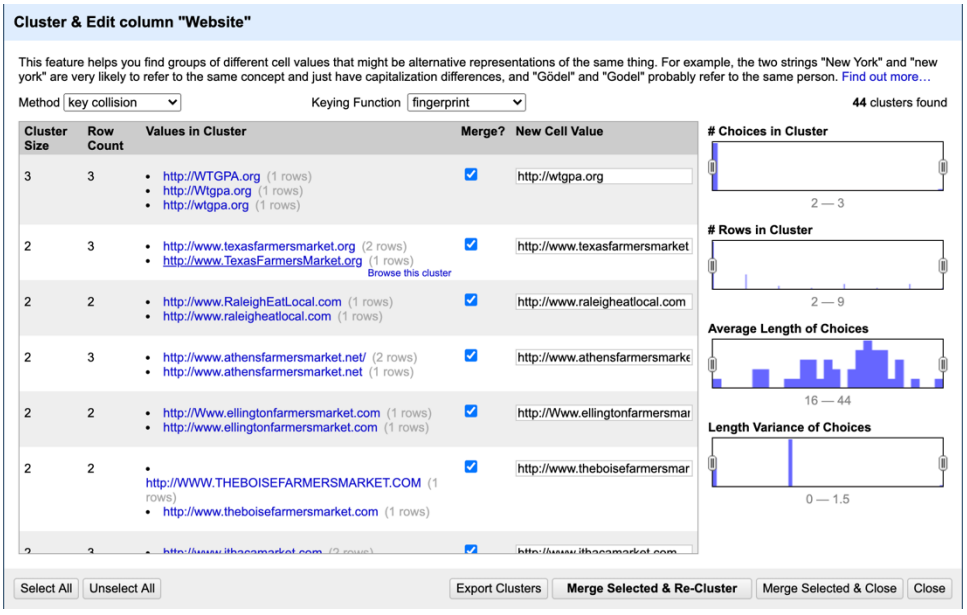
i) Trim leading and trailing whitespace – The number of cells affected by this transformation were 27.

ii) Collapse consecutive whitespace – There were no cells affected by this transformation.

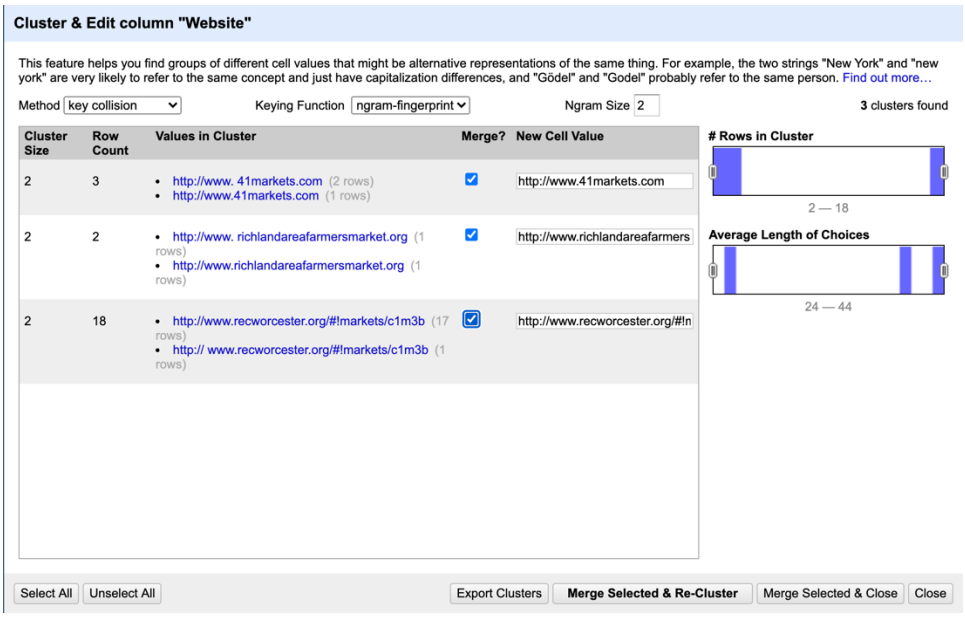
iii) In order to do further cleaning, we converted all URLs to lowercase. This affected 606 cells.



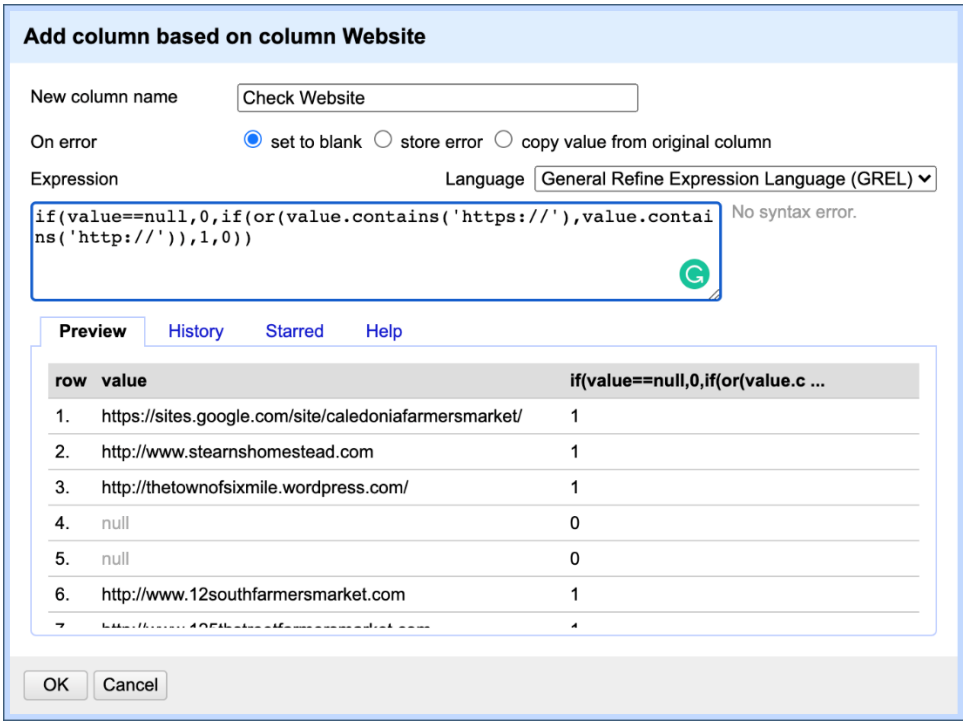
iv) We tried to validate if any URLs are duplicate. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 145 cells.



v) We also transformed this column with Key Collision method but with different keying function called - ngram with Ngram Size 2. This affected 23 cells.



vi) At this point, we have unique URLs in Website column. To validate every URL contains https:// or http://, we used GREL function. We created a new column called "Check Website" and populated 0 and 1 values. 1 means valid URL and 0 means invalid URL or NULL value in Website. This step confirmed that all URLs are valid.



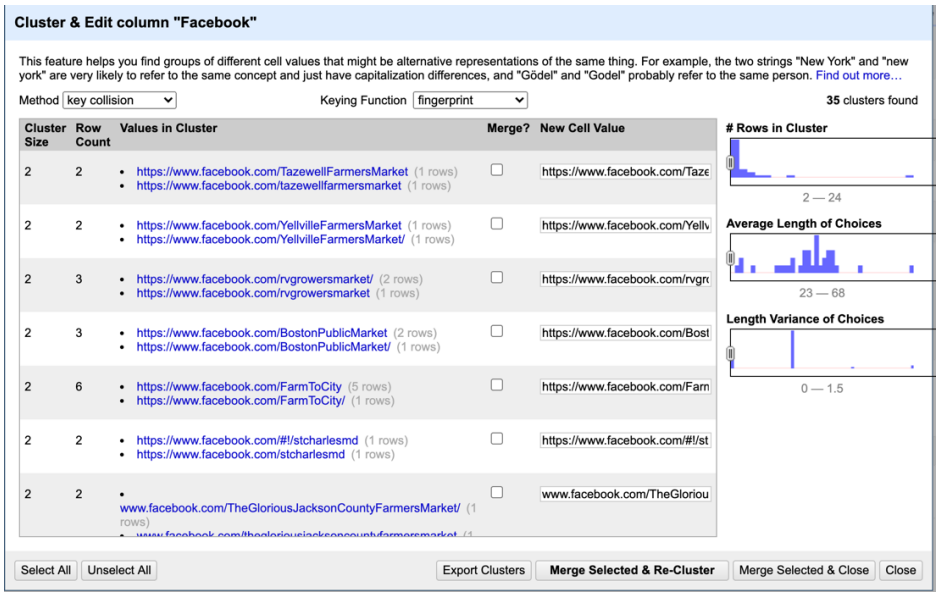
After applying all these steps, now we have valid URLs in Website column without any duplicates.

4. Facebook

i) Trim leading and trailing whitespace – The number of cells affected by this transformation were 40.

ii) Collapse consecutive whitespace – There were 3 cells affected by this transformation.

iii) We tried to validate if any Facebook links are duplicate. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 117 cells.

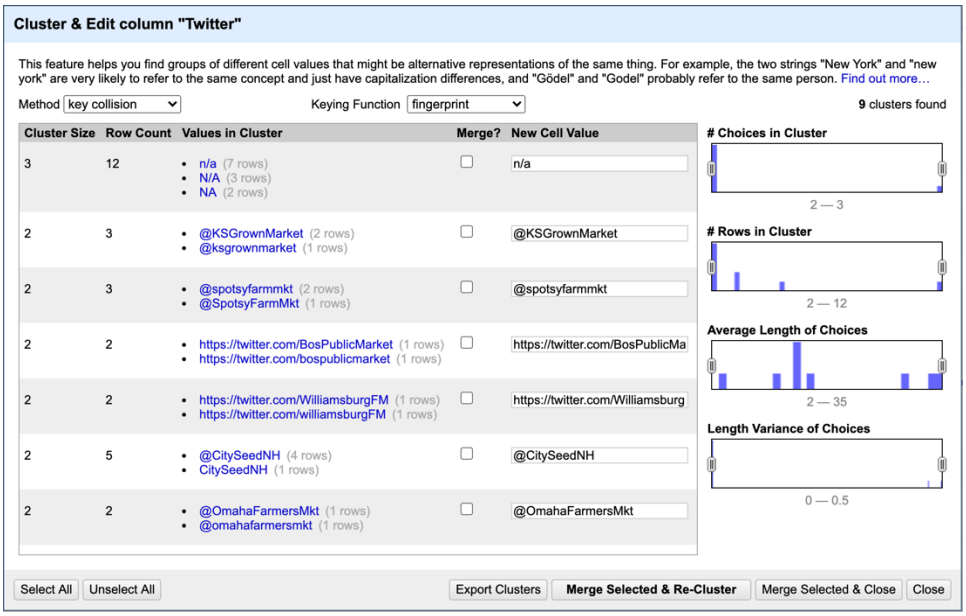


iv) We tried transforming this column with Key Collision method but with different keying function called - ngram with Ngram Size 2 but no clusters were found.

After applying all these steps, now we have clean values without any duplicates in Facebook column.

5. Twitter

i) Trim leading and trailing whitespace – The number of cells affected by this transformation were 9.
ii) Collapse consecutive whitespace – There was 1 cell affected by this transformation.
iii) We tried to validate if any Twitter links are duplicate. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 33 cells.

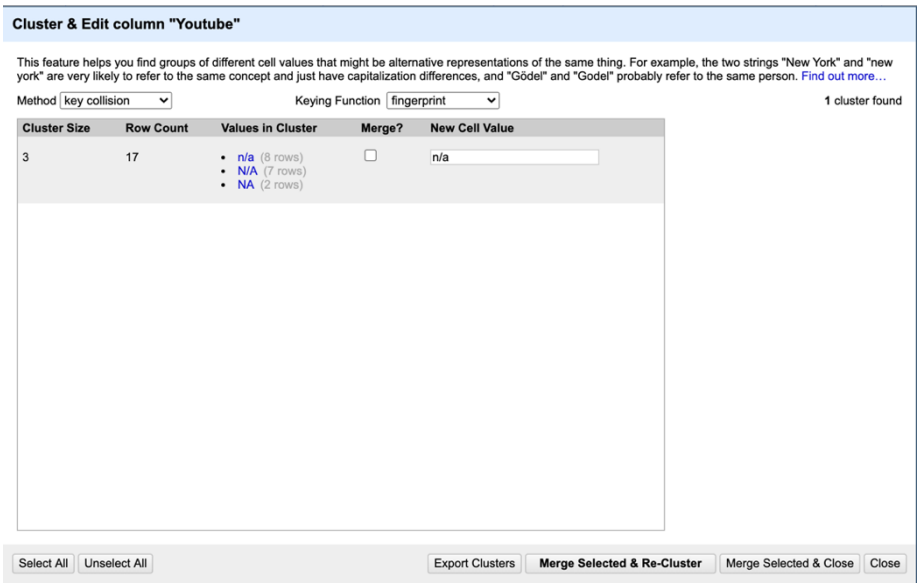


iv) We tried transforming this column with Key Collision method but with different keying function called - ngram with Ngram Size 2 but no clusters were found.

After applying all these steps, now we have clean values without any duplicates in Twitter column.

6. YouTube

i) Trim leading and trailing whitespace – The number of cells affected by this transformation were 4.
ii) Collapse consecutive whitespace – There were no cells affected by this transformation.
iii) We tried to validate if any YouTube links are duplicate. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 17 cells.

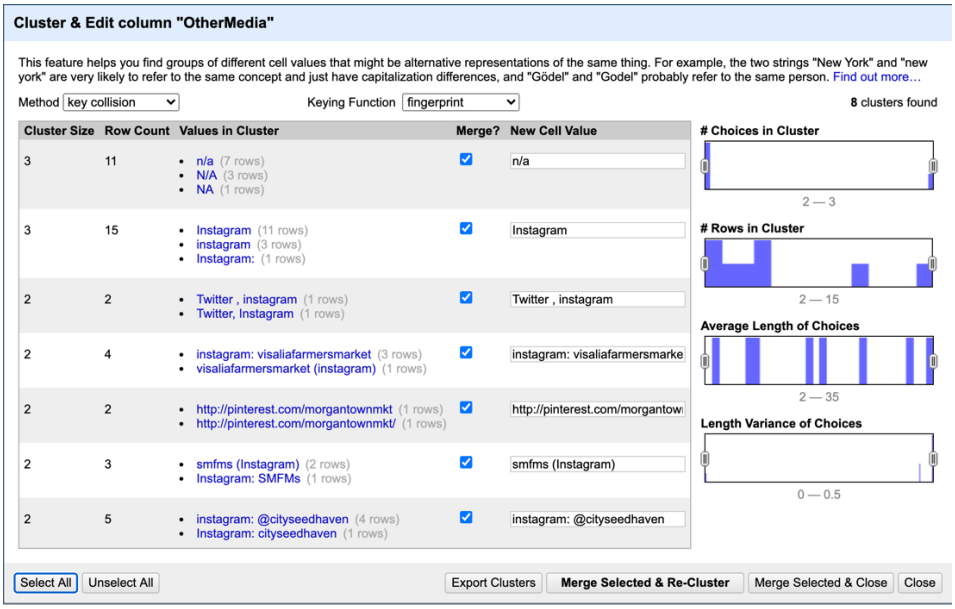


iv) We tried transforming this column with Key Collision method but with different keying function called - ngram with Ngram Size 2 but no clusters were found.

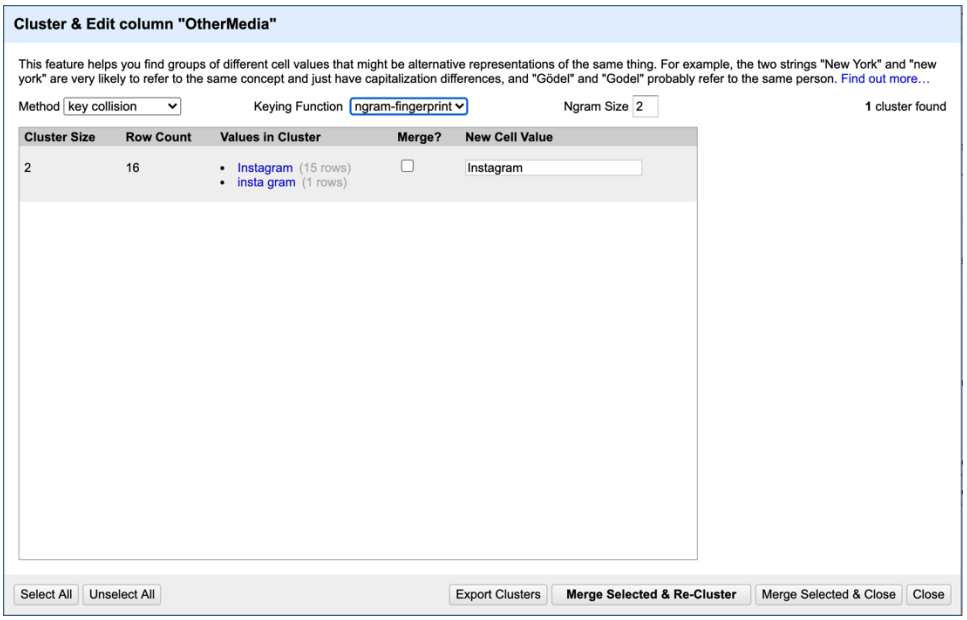
After applying all these steps, now we have clean values without any duplicates in Youtube column.

7. OtherMedia

i) Trim leading and trailing whitespace – The number of cells affected by this transformation were 18.
ii) Collapse consecutive whitespace – There were 22 cells affected by this transformation.
iii) We tried to validate if any OtherMedia links are duplicate. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 47 cells.



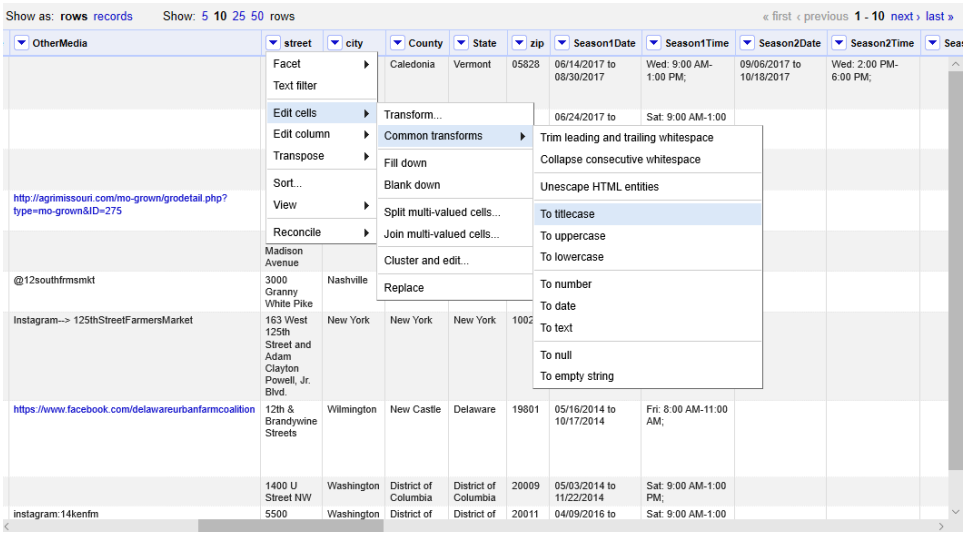
iv) We also transformed this column with Key Collision method but with different keying function called - ngram with Ngram Size 2. This affected 16 cells.



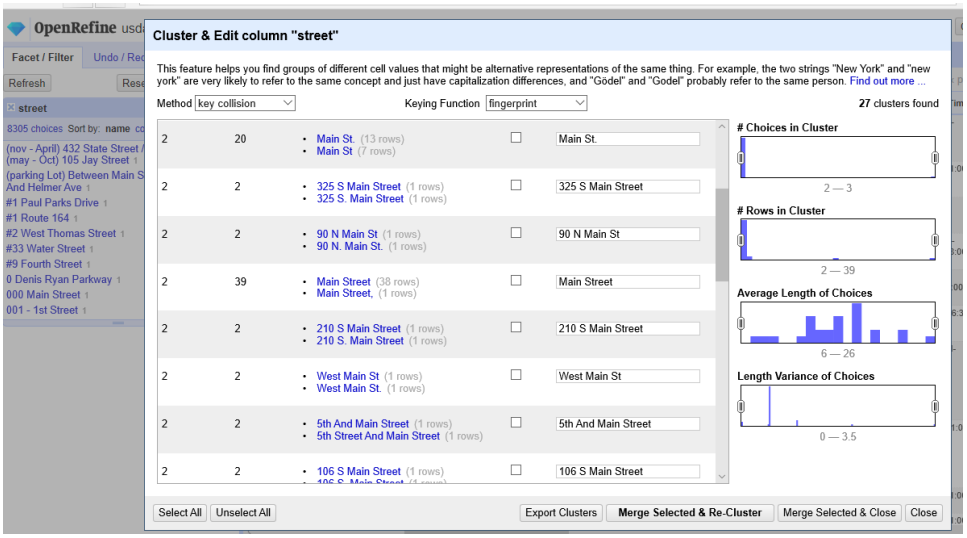
After applying all these steps, now we have clean values without any duplicates in OtherMedia column.

8. street

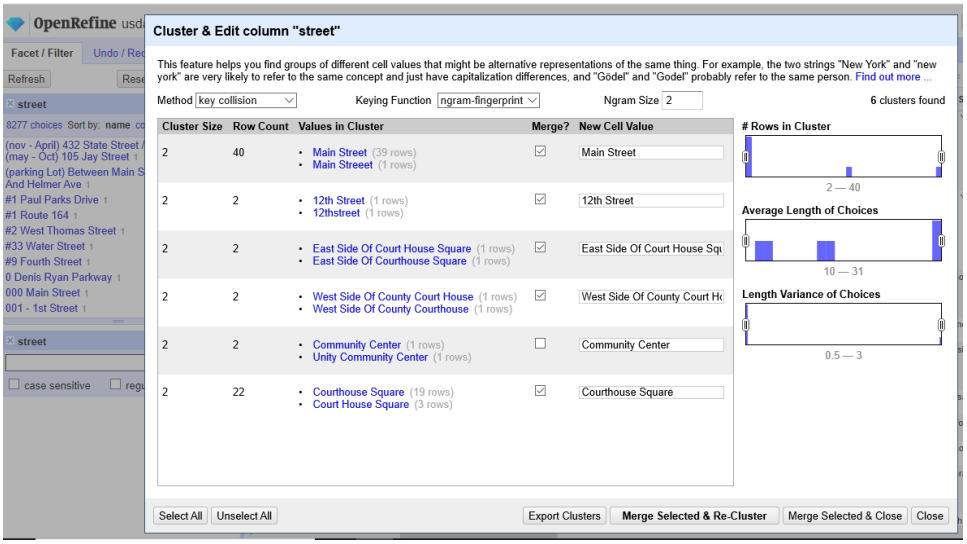
i) Trim leading and trailing whitespace – The number of cells affected by this were 317
ii) Collapse consecutive whitespace – The number of cells affected by this were 90
iii) To further make the Street name consistent, we convert Street names to title case. This affected 2507 cells



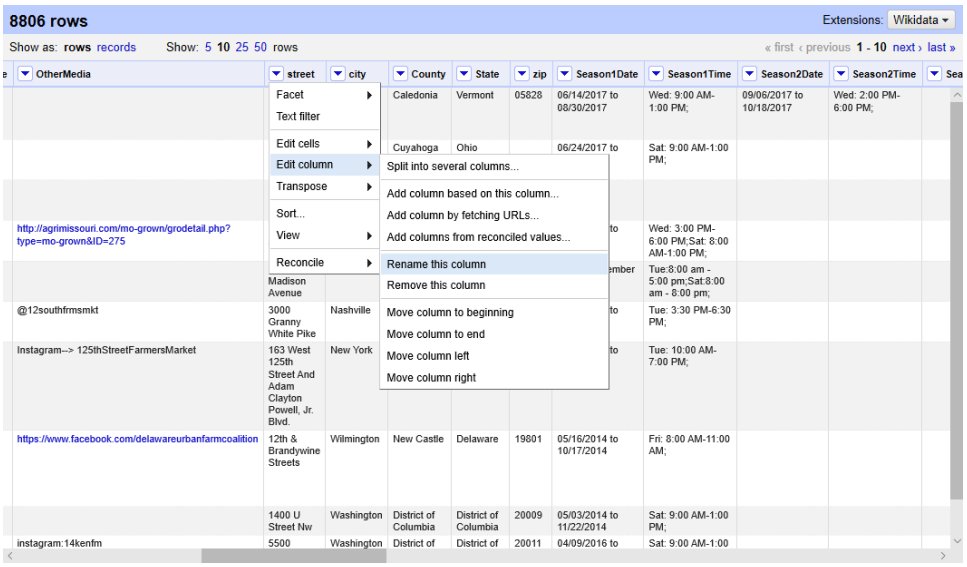
iv) Since this is a text column, there's a possibility of having same names with minor difference. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 111 cells.



v) We also transformed this column with Key Collision method but with different keying function called - ngram with Ngram Size 2. This affected 68 cells. Note that have intentionally not merged the values Community Center and Unity Community Center as they are different values.



vi) To keep the column street to be consistent with other column names, we have renamed the column"street" to titlecase "Street"

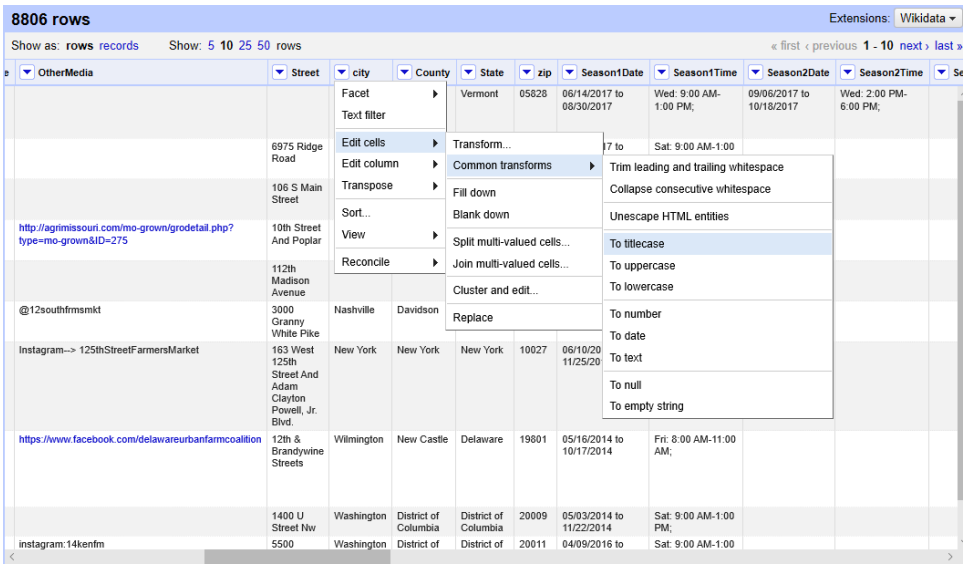


9. city

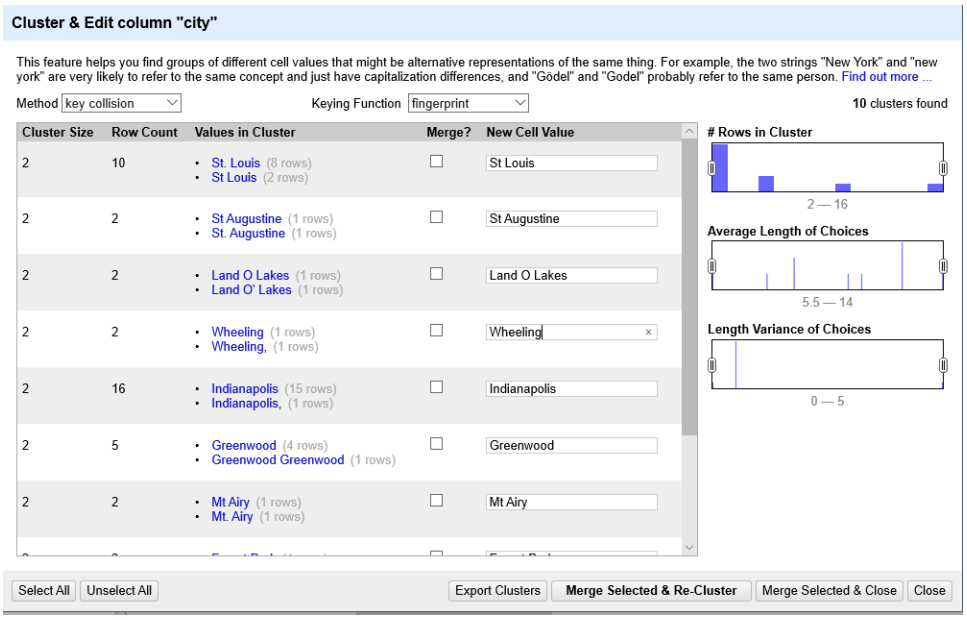
i) Trim leading and trailing whitespace – The number of cells affected by this were 967

ii) Collapse consecutive whitespace – The number of cells affected by this were 2

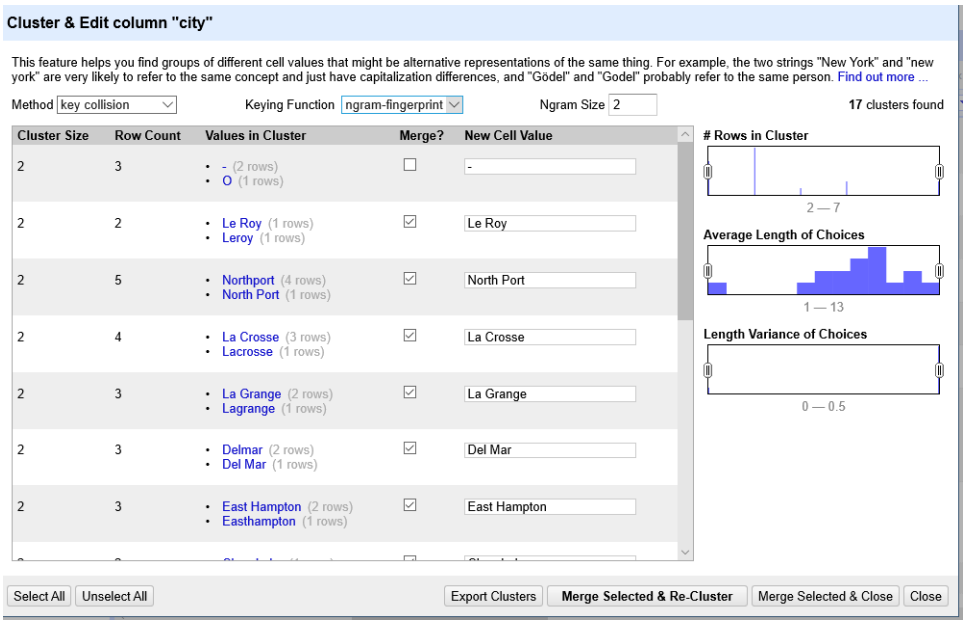
iii) To further make the City name consistent, we convert city names to title case. This affected 342 cells



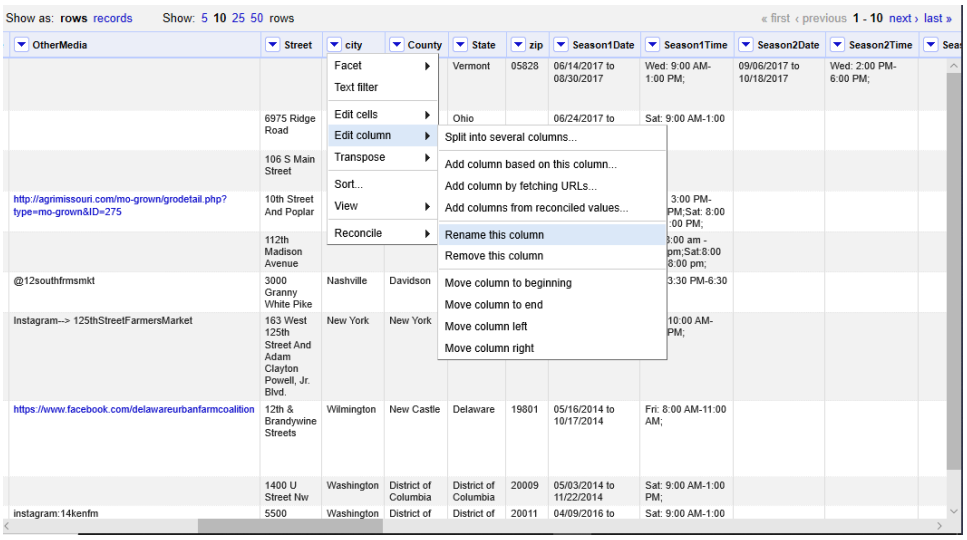
iv) Since this is a text column, there's a possibility of having same names with minor difference. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and Fingerprint keying function. This transformation affected 48 cells.



v) We also transformed this column with Key Collision method but with different keying function called - ngram with Ngram Size 2. This affected 56 cells. Note that have intentionally not merged the values blank with O as they are different values.



vi) To keep the column city to be consistent with other column names, we have renamed the column “city” to titlecase “City”

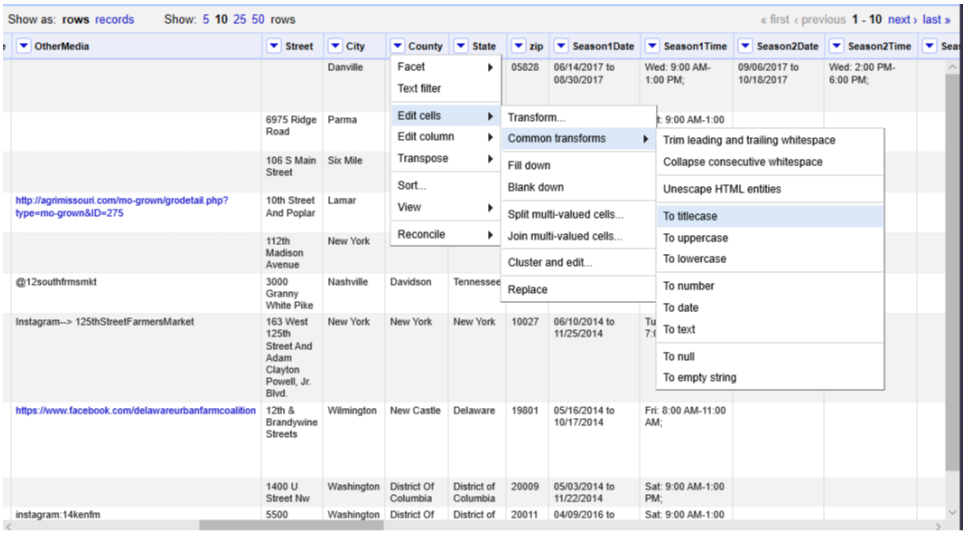


10. County

i) Trim leading and trailing whitespace – The number of cells affected by this were 0

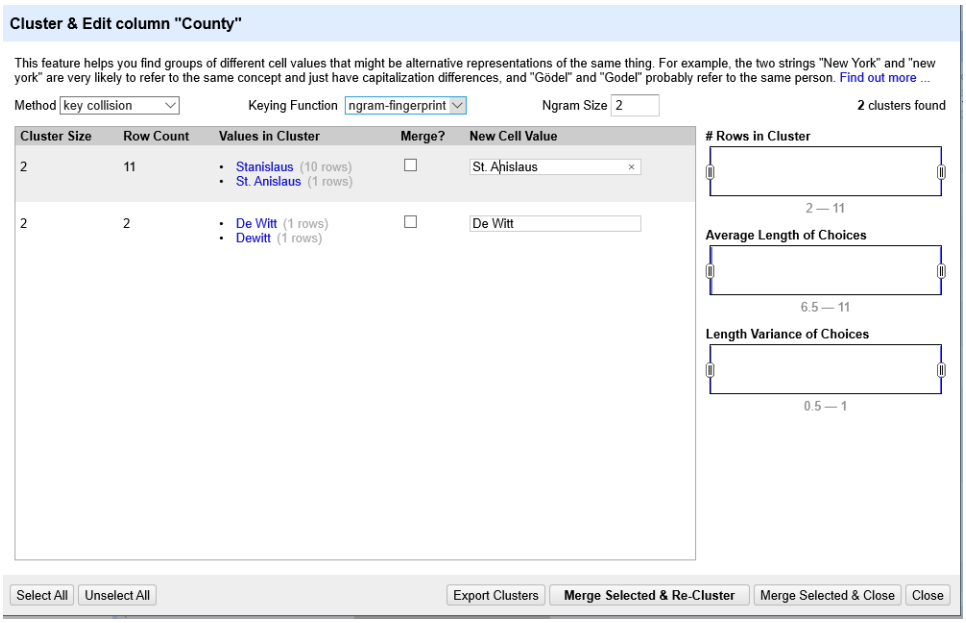
ii) Collapse consecutive whitespace – The number of cells affected by this were 0

iii) To further make the County name consistent, we convert County names to title case. This affected 232 cells



iv) Since this is a text column, there's a possibility of having same names with minor difference. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and n-gram-fingerprint keying function. This transformation affected 13 cells.

v) We also checked for clustering transformation using fingerprinting keying function and there were no duplicates found.

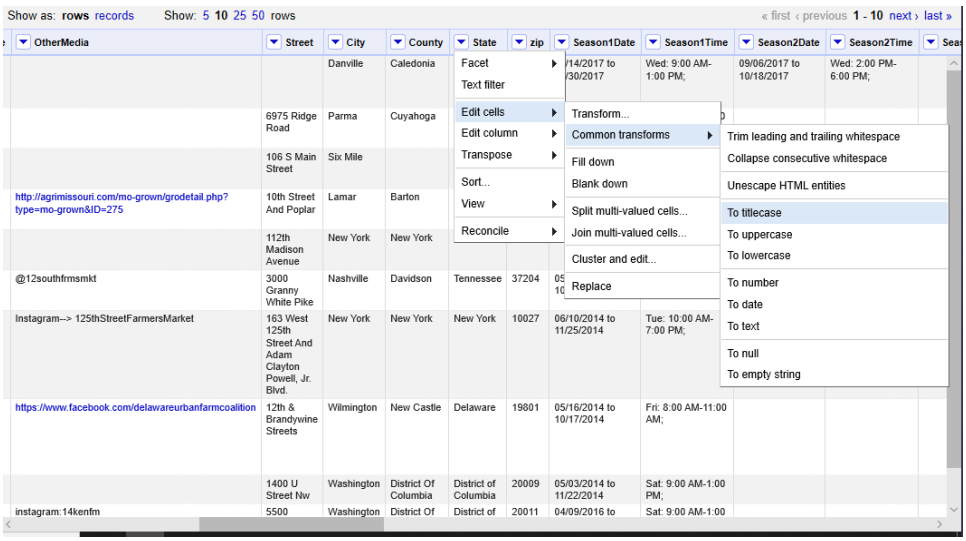


11. State

i) Trim leading and trailing whitespace – The number of cells affected by this were 0

ii) Collapse consecutive whitespace – The number of cells affected by this were 0

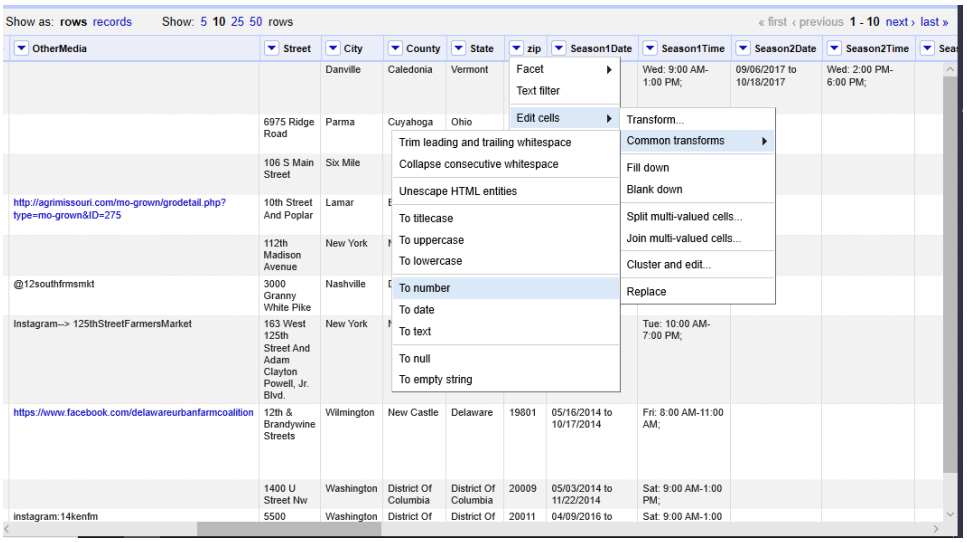
iii) To further make the State name consistent, we convert State names to title case. This affected 60 cells



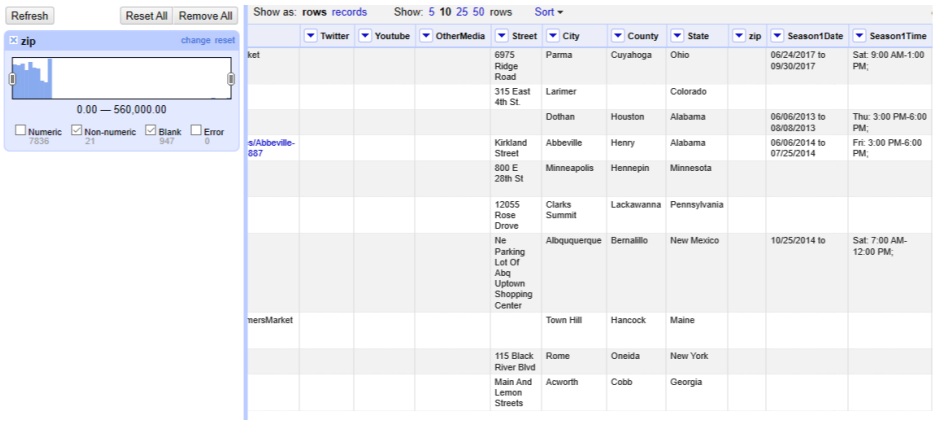
iv) Since this is a text column, there's a possibility of having same names with minor difference. To remove these kinds of duplicates, we used clustering transformation using Key Collison method and functions – fingerprinting and n-gram-fingerprint keying function. This transformation affected 0 cells. This is an indication of consistent State names provided in the data. It can be inferred that the information in State column has overall high accuracy.

12.Zip

i) Convert to Number – Since zip is supposed to be a numeric column, column is converted to number. 7838 cells were affected.

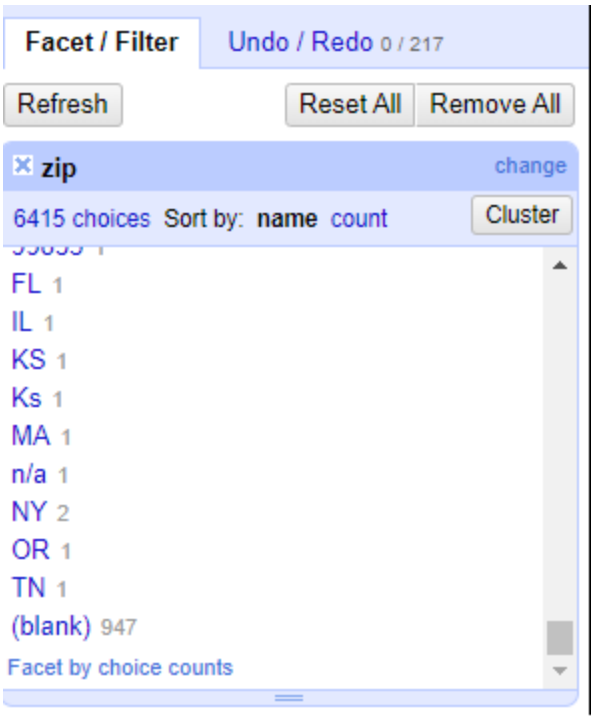


There is total 21 non-numeric values and 947 blank values as seen in below screenshot. This shows zip column is inconsistent and not completely reliable.



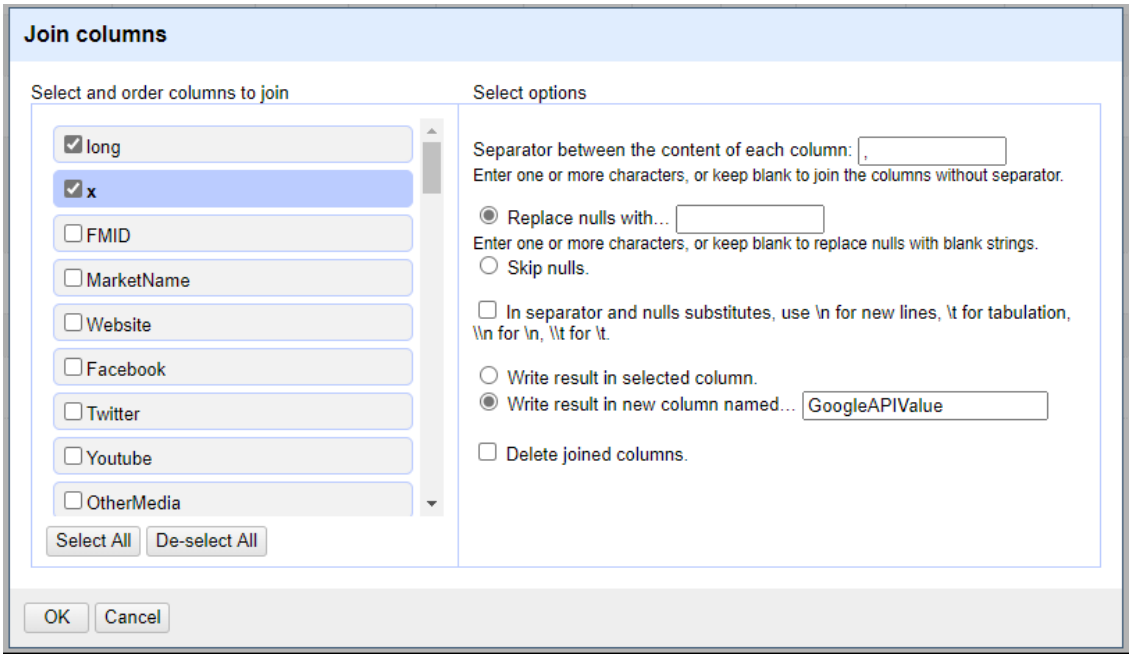
ii) We used Google API to Recover blank Zip codes using location and compared it to existing city values and recovered 584 zip codes matching the exact city and latitude and longitude values as provided by Google API

We initially filtered thru applying a Text Facet and included the blank zip codes



Google API URL using a API Key https://maps.googleapis.com/maps/api/geocode/json?latlng=" + value + "&key=AIzaSyCM0SHbIG6XqWqhDaGPV-RoPiDvvGe0cD8&result_type=postal_code" – the value accepts “lat,long”

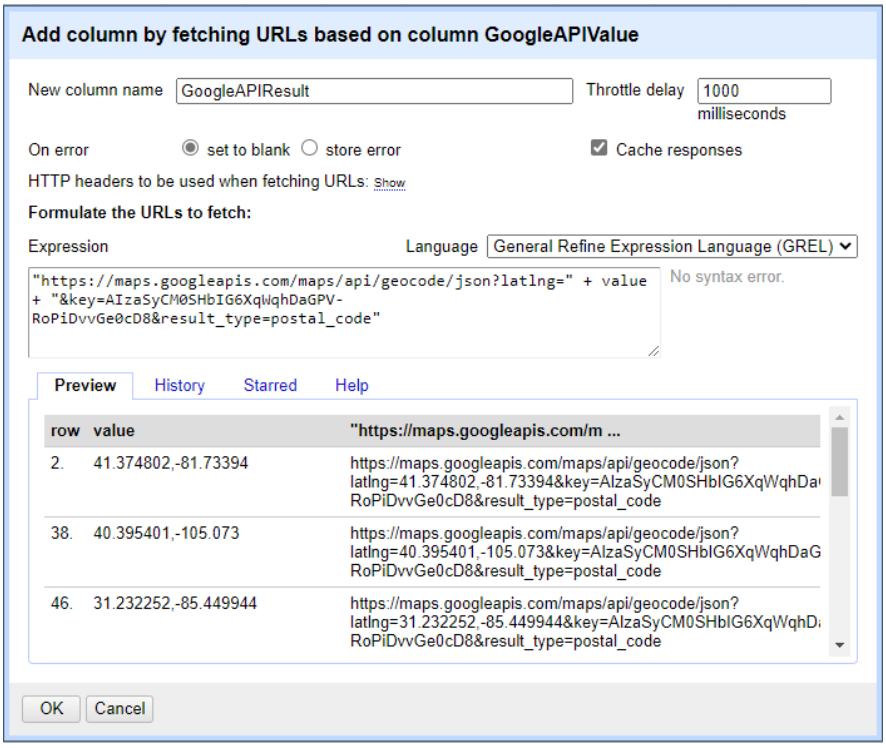
Created a new column based on y value and appended the x value using Transform Join columns using a comma (,) separator so it can be passed to the API



The join operation combined the latitude and longitude in a format compliant with Google API requirements

x	y	long	GoogleAPIValue
-81.73394	41.374802	41.374802	41.374802,-81.73394
-105.073	40.395401	40.395401	40.395401,-105.073
-85.449944	31.232252	31.232252	31.232252,-85.449944
-85.250366	31.571272	31.571272	31.571272,-85.250366
-93.262901	44.955601	44.955601	44.955601,-93.262901
-75.774597	41.424999	41.424999	41.424999,-75.774597
-106.56584	35.103989	35.103989	35.103989,-106.56584

Using the GoogleAPIValue > Edit Column > Add Columns by fetching URLs passed in the value and concatenating the Google API URL we were able to generate the fetching URL including only the postal code details which returns a Json for the latitude and longitude and postal code along with city the final GREL expression with 10ms Throttle delay:
grel:\"https://maps.googleapis.com/maps/api/geocode/json?latlng=\" + value + \"&key=AIzaSyCM0SHbIG6XqWqhDaGPV-RoPiDvvGe0cD8&result_type=postal_code\""



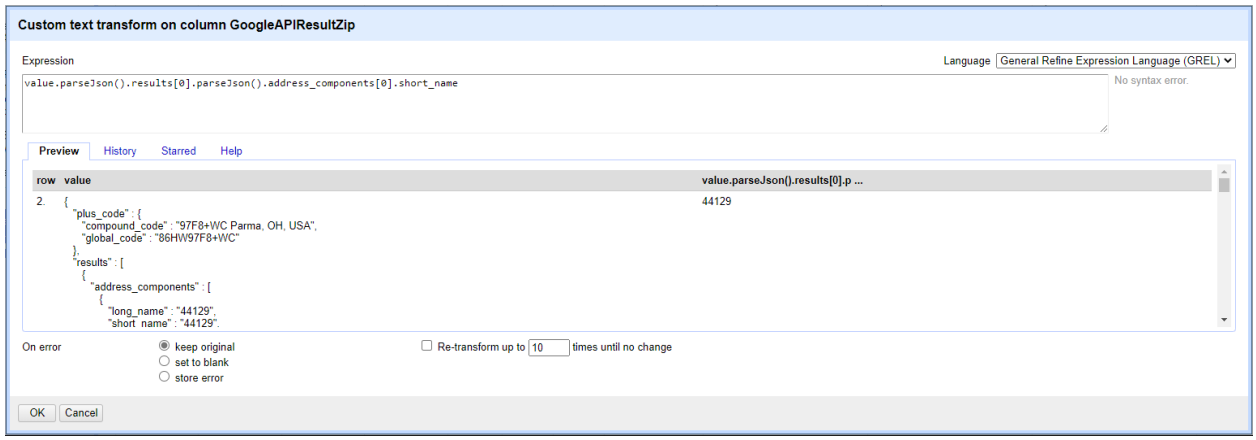
The API executed for around 30 minutes and returned zip files for all latitude longitude values resulting JSON file is now updated in the GoogleAPIResult column

GoogleAPIValue	GoogleAPIResult
41.374802,-81.73394	{ "plus_code" : { "compound_code" : "97F8+WC Parma, OH, USA", "global_code" : "86HW97F8+WC" }, "results" : [{ "address_components" : [{ "long_name" : "44129", "short_name" : "44129", "types" : ["postal_code"] }, { "long_name" : "Parma", "short_name" : "Parma", "types" : ["locality", "political"] }, { "long_name" : "Cuyahoga County", "short_name" : "Cuyahoga County", "types" : ["administrative_area_level_2", "political"] }, { "long_name" : "Ohio", "short_name" : "OH", "types" : ["administrative_area_level_1", "political"]

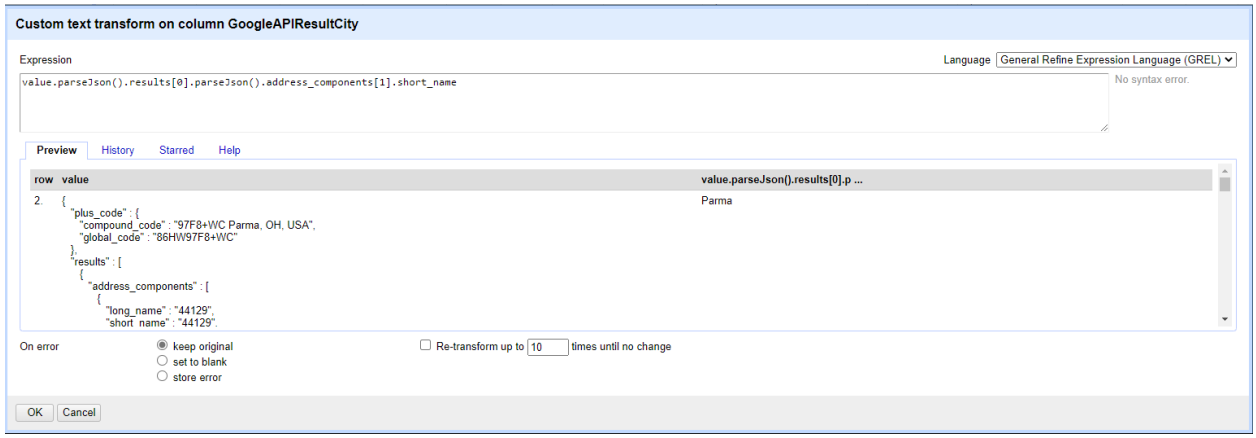
Duplicated the GoogleAPI Result to GoogleResultAPIZip and GoogleAPIResultCity using Add New Column based on the values to retain the JSON results for extracting Zip and City

GoogleAPIResult	GoogleAPIResultZip	GoogleAPIResultCity
{ "plus_code" : { "compound_code" : "97F8+WC Parma, OH, USA", "global_code" : "86HW97F8+WC" }, "results" : [{ "address_components" : [{ "long_name" : "44129", "short_name" : "44129", "types" : ["postal_code"] }, { "long_name" : "Parma", "short_name" : "Parma", "types" : ["locality", "political"] }, { "long_name" : "Cuyahoga County", "short_name" : "Cuyahoga County", "types" : ["administrative_area_level_2", "political"] }, { "long_name" : "Ohio"	{ "plus_code" : { "compound_code" : "97F8+WC Parma, OH, USA", "global_code" : "86HW97F8+WC" }, "results" : [{ "address_components" : [{ "long_name" : "44129", "short_name" : "44129", "types" : ["postal_code"] }, { "long_name" : "Parma", "short_name" : "Parma", "types" : ["locality", "political"] }, { "long_name" : "Cuyahoga County", "short_name" : "Cuyahoga County", "types" : ["administrative_area_level_2", "political"] }, { "long_name" : "Ohio"	{ "plus_code" : { "compound_code" : "97F8+WC Parma, OH, USA", "global_code" : "86HW97F8+WC" }, "results" : [{ "address_components" : [{ "long_name" : "44129", "short_name" : "44129", "types" : ["postal_code"] }, { "long_name" : "Parma", "short_name" : "Parma", "types" : ["locality", "political"] }, { "long_name" : "Cuyahoga County", "short_name" : "Cuyahoga County", "types" : ["administrative_area_level_2", "political"] }, { "long_name" : "Ohio"

Using the GoogleAPIResultZip column performed a transform to parse the JSON and extract the Zip by adding a custom text transform and using parseJson() GREL command value.parseJson().results[0].parseJson().address_components[0].short_name



Using the GoogleAPIResultCity column performed another text transform to parse the JSON and extract the City by adding a custom transform GREL command: value.parseJson().results[0].parseJson().address_components[1].short_name



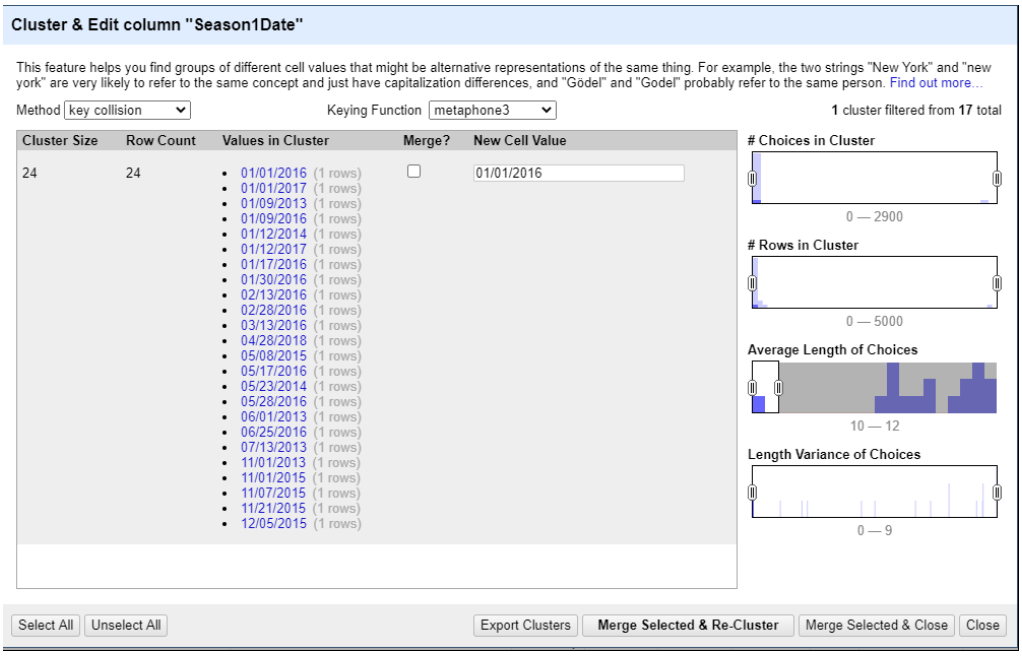
Removed the GoogleAPIResult column and used the Zip and City to extract FMID, city, zip GoogleAPIResultZip, GoogleAPIResultCity into a separate Excel and filtered using condition where the city column is equal to GoogleAPIResultCity.

The Final output we were able to recover 584 correct zip code matching the city names

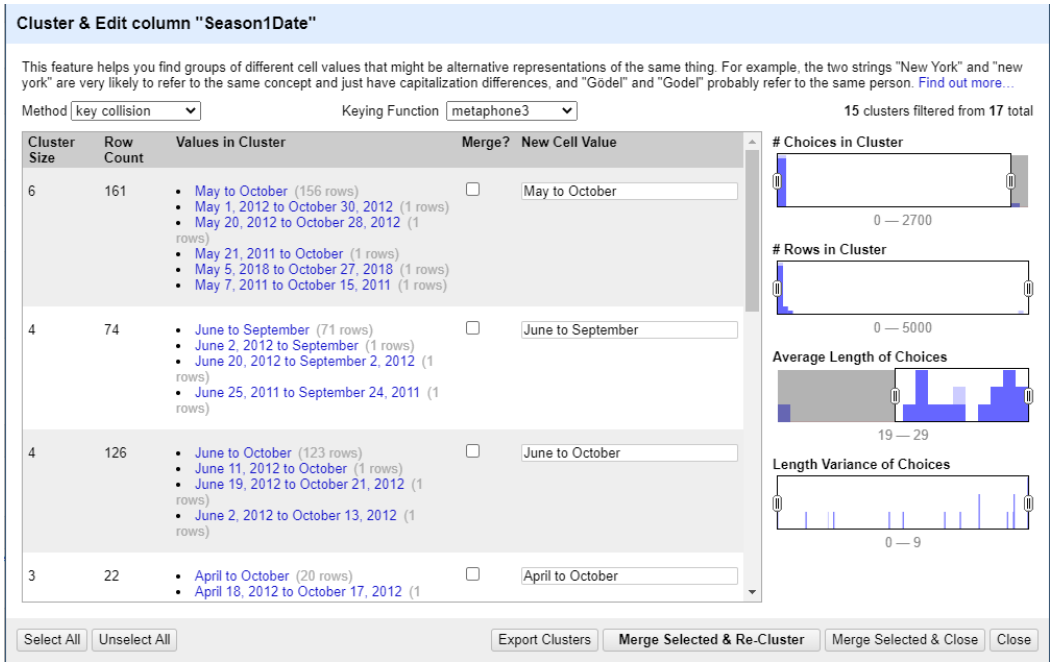
x	y	long	GoogleAPIValue	GoogleAPIResul	GoogleAPIResul
-81.73394	41.374802	41.374802	41.374802,-81.73394	44129	Parma
-105.073	40.395401	40.395401	40.395401,-105.073	80537	Loveland
-85.449944	31.232252	31.232252	31.232252,-85.449944	36305	Dothan
-85.250366	31.571272	31.571272	31.571272,-85.250366	36310	Abbeville
-93.262901	44.955601	44.955601	44.955601,-93.262901	55404	Minneapolis
-75.774597	41.424999	41.424999	41.424999,-75.774597	18411	S ABINGTN TWP
-106.56584	35.103989	35.103989	35.103989,-106.56584	87110	Albuquerque
-68.971603	44.590801	44.590801	44.590801,-68.971603	04438	Frankfort
-75.454102	43.209801	43.209801	43.209801,-75.454102	13440	Rome
-84.676697	34.065701	34.065701	34.065701,-84.676697	30101	Acworth

13. Season1Date, 15. Season2Date, 17. Season3Date, 19. Season4Date

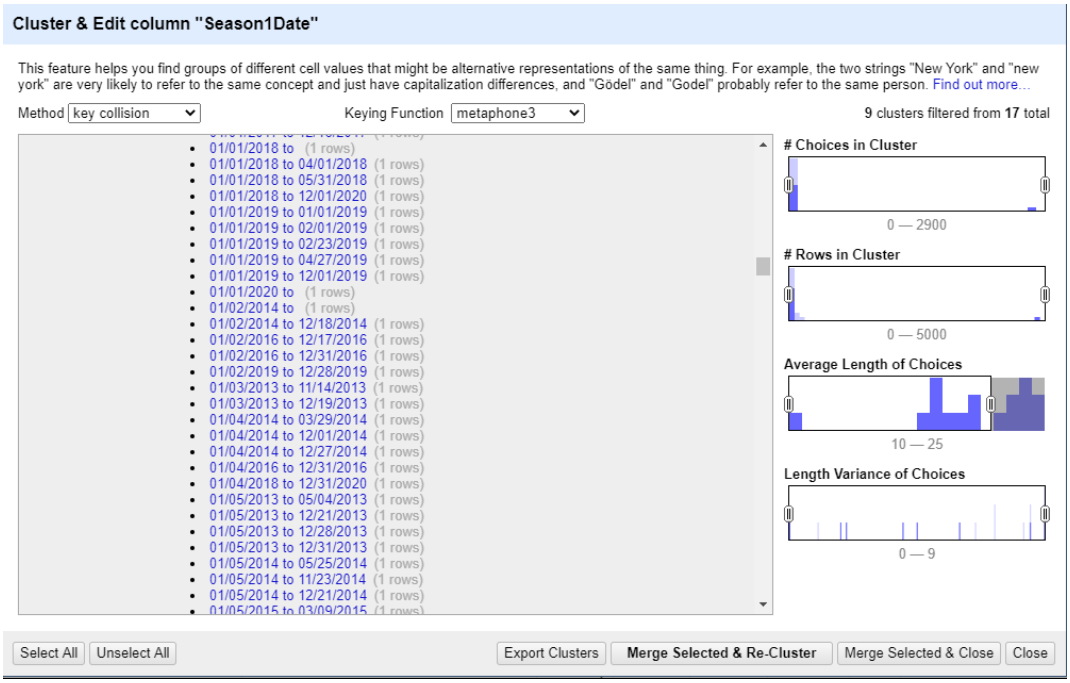
i) The season1Date column has season start and end dates, using a text cluster using keying function metaphone3 we identified 24 single value dates



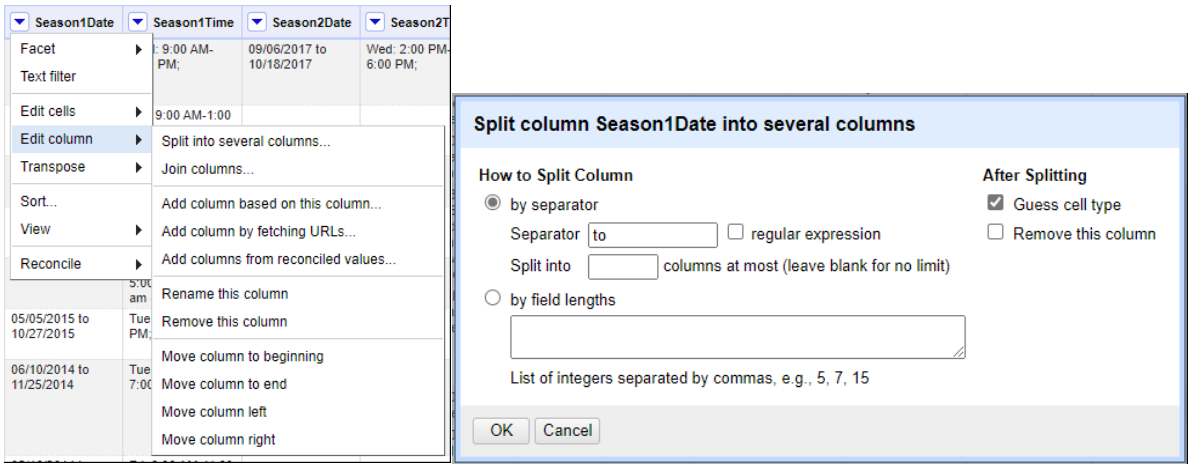
ii) We then used the average length of choices showing dates that are long format



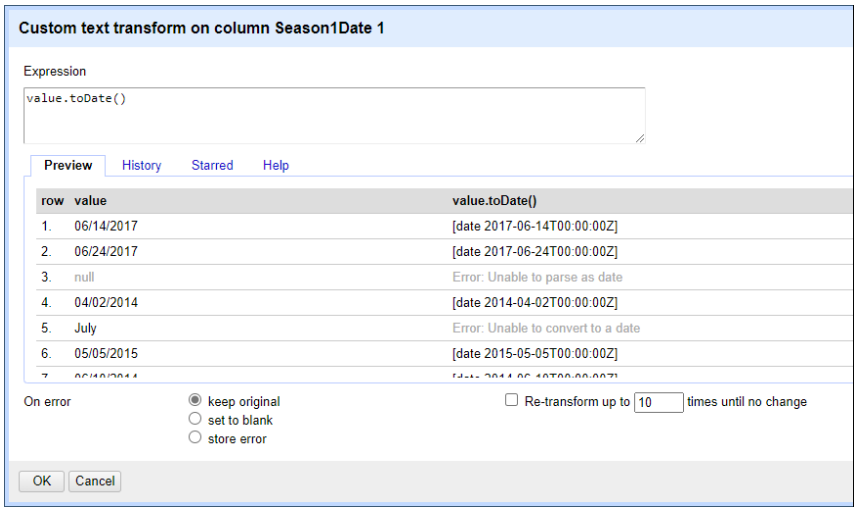
iii) Majority of the dates around 4916 rows had common format “mm-dd-yyyy to mm-dd-yyyy” with a common separator “to”

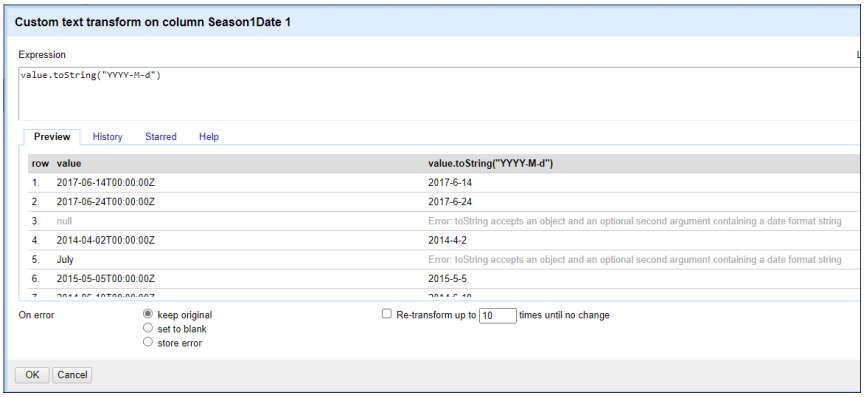


iv) Due the variations of patterns we decided to trim the Season1Date, Season2Date, Season3Date, Season4Date and used the split column transform to split values based on the separator “to”



v) The Season1Date, where separated into Season1Date 1, Season1Date 2 the two additional columns the Season1Date had 4940 rows with proper date formats (mm/dd/yyy) the two new columns were then converted to Date using transform and changed the date to ISO format"YYYY-M-d" using the GREL operation value.toDate() and value.toString("YYYY-M-d") around 4967 Season1FromDate and 4983 Season1ToDate was converted to proper date formats.





vi) Steps i, ii, iii, iv, v where repeated for Season2Date, Season3Date, Season4Date resulting in additional columns Season2FromDate, Season2ToDate, Season3FromDate, Season3ToDate, Season4FromDate, Season4ToDate

14. Season1Time, 16. Season2Time, 18. Season3Time, 20. Season4Time

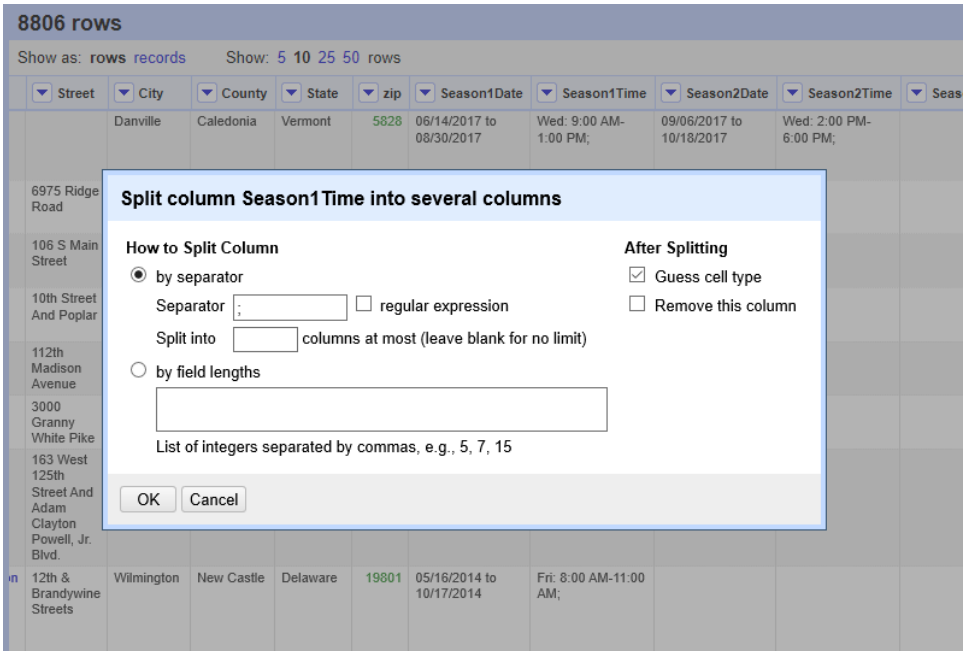
i) Trim leading and trailing whitespace – There were no cells affected by this

ii) To further make the Season1Time consistent, we convert Season1Time column to lower case. This affected 5868 cells

iii) Now we split the Season1Time column into several columns based on the separator ";" but ensure to keep the original column by unchecking the "remove this column" option

This results in 8 new columns namely Season1Time1, Season1Time2 up till Season1Time8

There were 5868 cells that got split into several columns by separator ";". Here ";" is the delimiter between the values such as "Wed: 3:00 Pm-6:00 Pm;sat: 8:00 Am-1:00 Pm;" of the Season1Time column.



iv) Similar steps – 14 - i, ii and ii are repeated for columns – Season2Time, Season3Time and Season4Time. And the number of split columns vary based on the maximum number of multi values in the SeasonTime column.

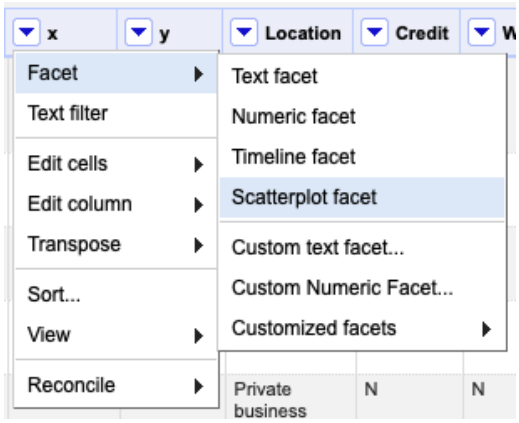
Season2Time – no cells affected by trim operation. 455 cells affected by lowercase () operation. And 455 cells were split into 8 columns.

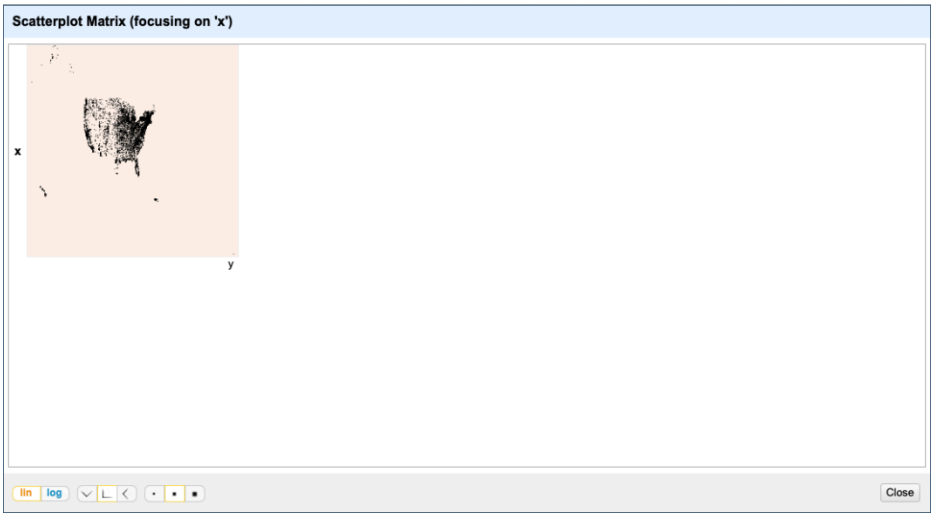
Season3Time - no cells affected by trim operation. 75 cells affected by lowercase () operation. And 75 cells were split into 7 columns.

Season4Time - no cells affected by trim operation. 5 cells affected by lowercase () operation. And 5 cells were split into 2 columns.

21. X

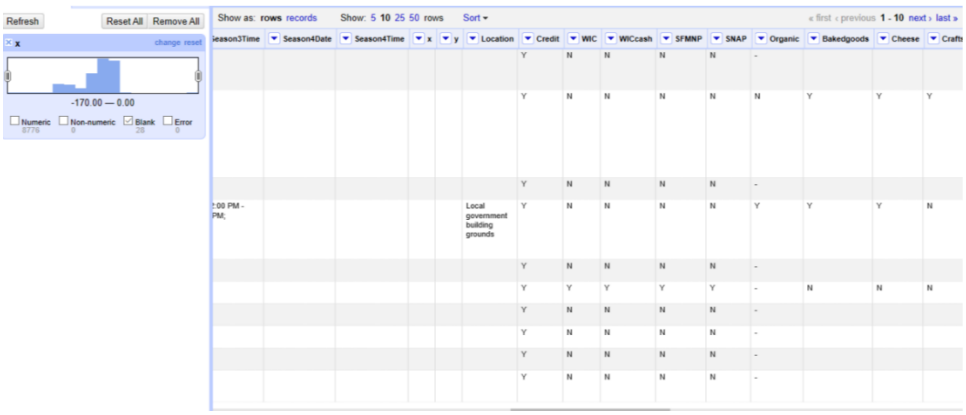
To verify the location accuracy, we did a Scatter Facet in OpenRefine to get a visual map of all the locations using the X and Y values to see an overview and observed that the data confirms to the shape of US with some outliers or errors



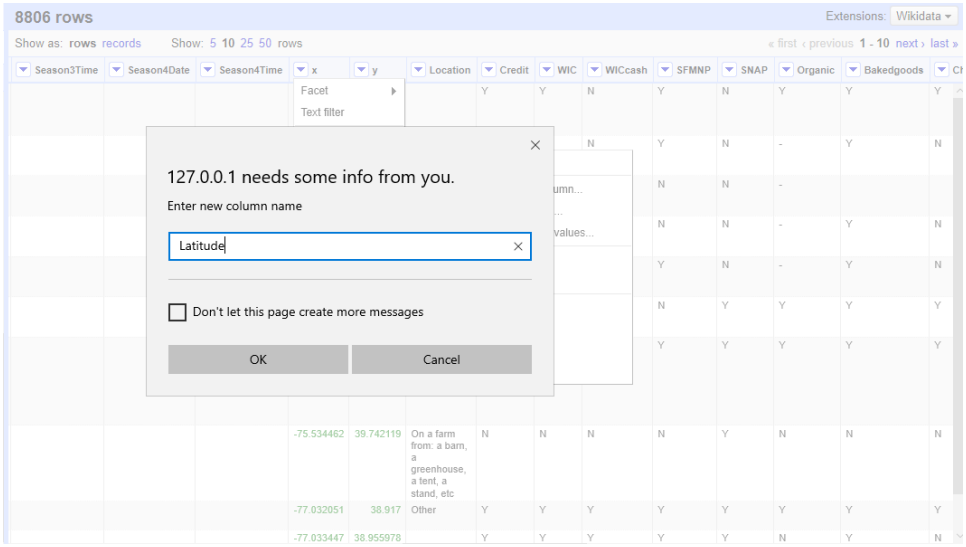


i) Convert to number – This column seems to have numeric values. To validate it, Convert to Number transformation was applied and 8778 cells were affected.

ii) Clustering using numeric facet shows there are 28 Blank values and rest are numeric.



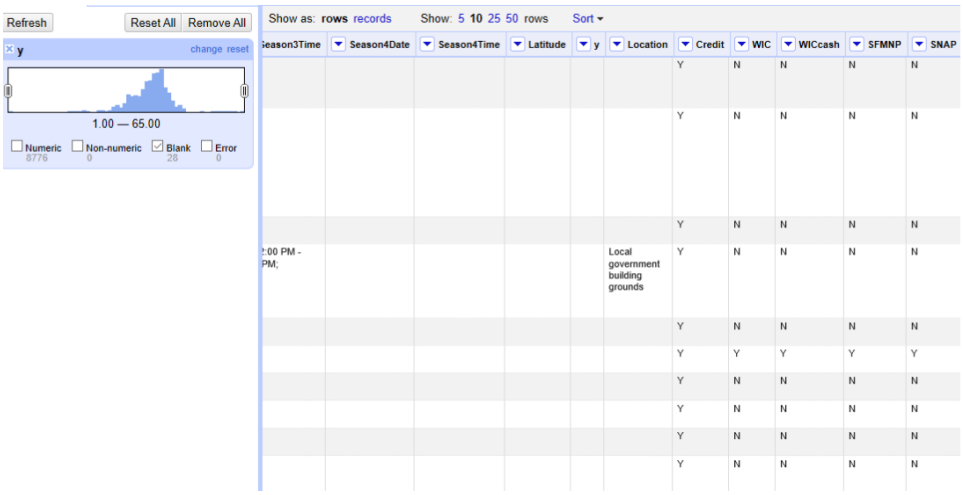
iii) Seeing the values of column x and column y, it can be inferred that column "x" means latitude values for any given market. So, this column "x" is renamed to "Latitude" to be meaningful.



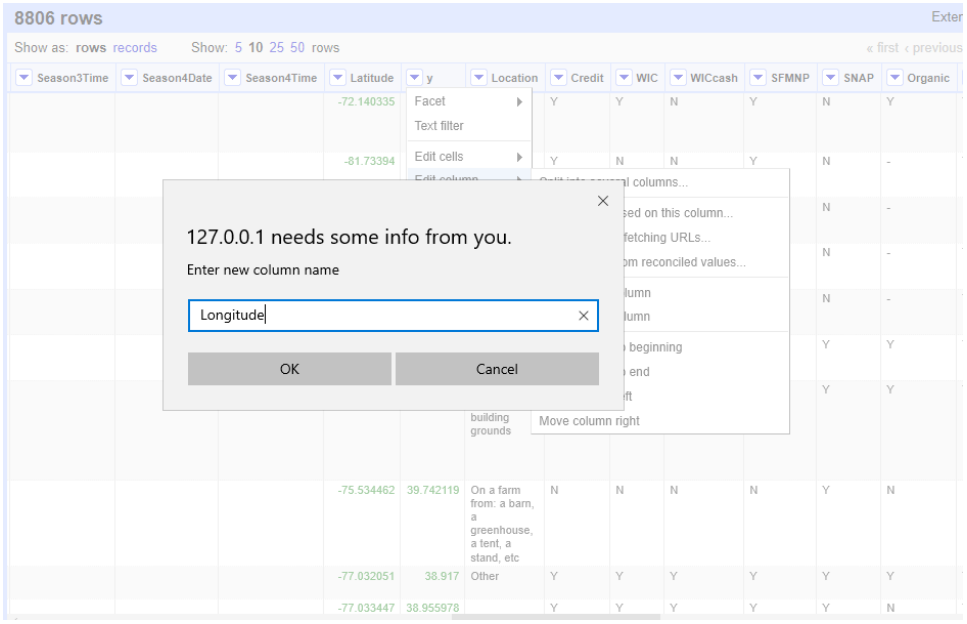
22. Y

i) Convert to number - This column seems to have numeric values. To validate it, Convert to Number transformation was applied and 8778 cells were affected.

ii) Clustering using numeric facet shows there are 28 Blank values and rest are numeric.

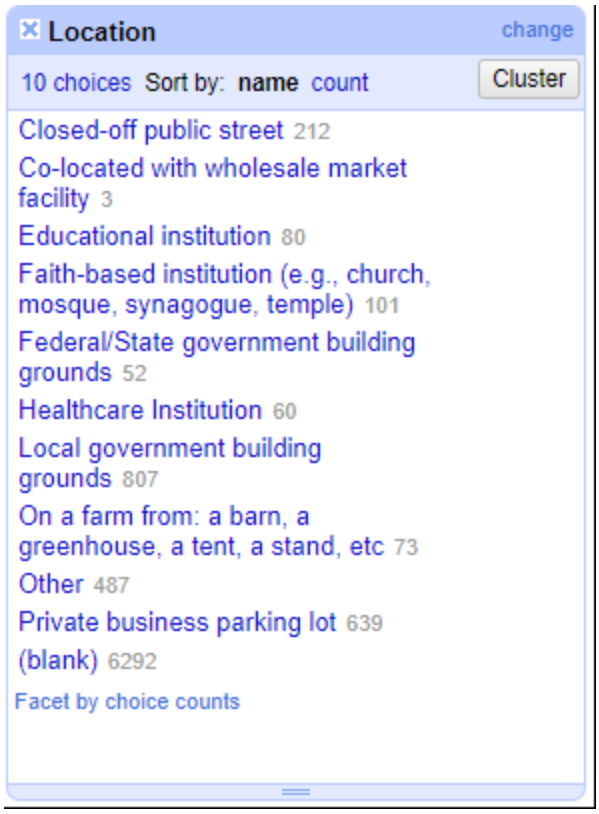


iii) Seeing the values of column x and column y, it can be inferred that column "y" means longitude values for any given market. So, this column "y" is renamed to "Longitude" to be meaningful.



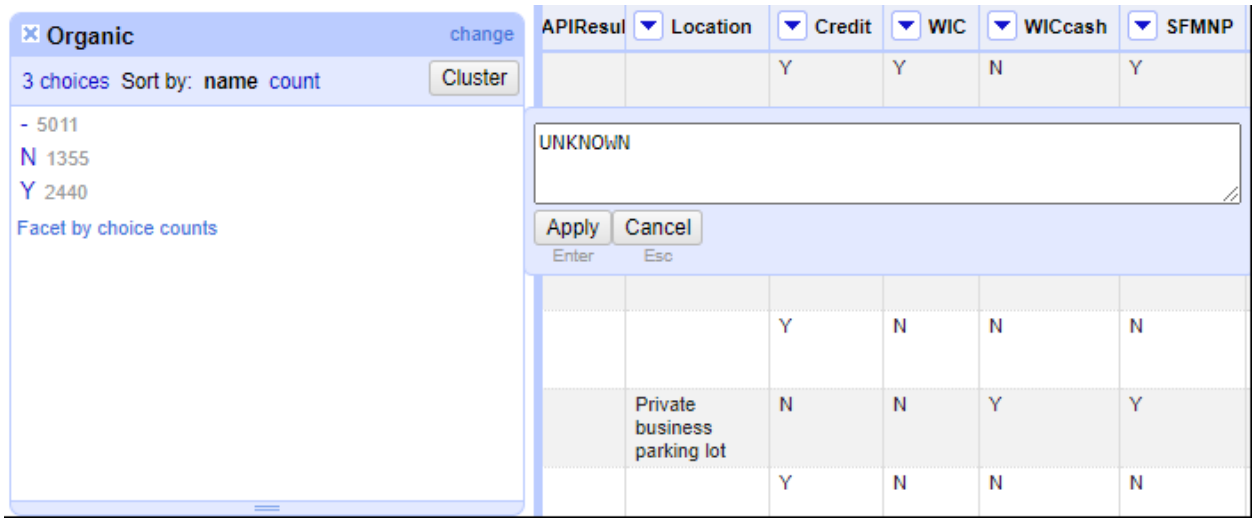
23. Location

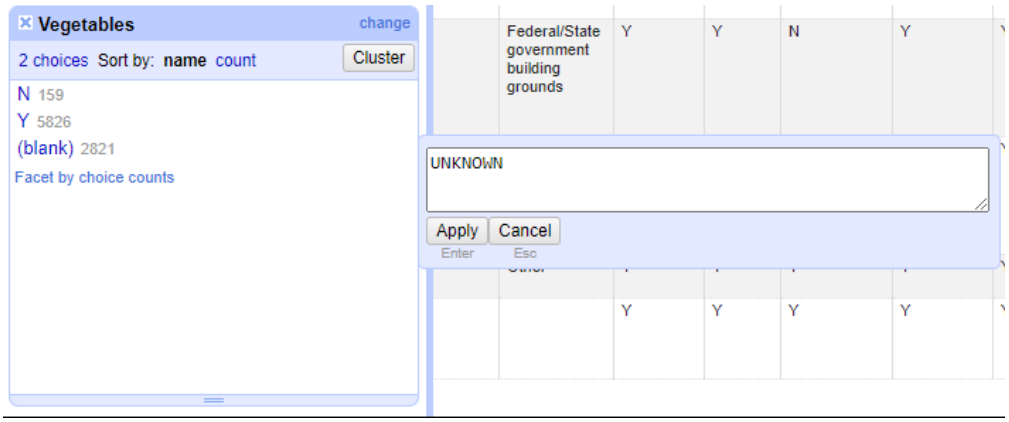
We performed a Transform > Trim leading and training spaces on the Location column resulting in cleaning any inconsistent spaces and also observed that there are 6592 blank values in the location field with 10 unique values was observed when performing a text facet.



24-58 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 columns had common theme of values having “Y”, “N” “-“ and blank values – we applied a transform to trim Leading Spaces and replaced all blank Values and “-“ using a Text Facet and edit command the below steps indicated are for Organic and Vegetables the same steps was repeated for all the 24-58 columns.





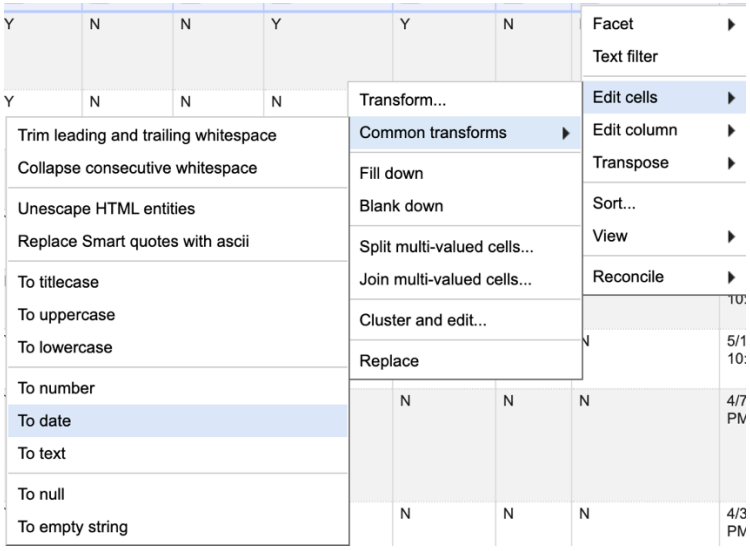
The breakdown of all columns that had "-" or "blank" Values:

Credit, WIC, WICcash, SFMNP, SNAP, Organic[5011 "-" values] , Bakedgoods[2821 "blank" values], Cheese[2821 "blank" values], Crafts[2821 "blank" values], Flowers[2821 "blank" values], Eggs[2821 "blank" values], Seafood[2821 "blank" values], Herbs[2821 "blank" values], Vegetables[2821 "blank" values], Honey[2821 "blank" values], Jams[2821 "blank" values], Maple[2821 "blank" values], Meat[2821 "blank" values], Nursery[2821 "blank" values], Nuts[2821 "blank" values], Plants[2821 "blank" values], Poultry[2821 "blank" values], Prepared[2821 "blank" values], Soap[2821 "blank" values], Trees[2821 "blank" values], Wine[2821 "blank" values], Coffee[2821 "blank" values], Beans[2821 "blank" values], Fruits, Grains[2821 "blank" values], Juices[2821 "blank" values], Mushrooms[2821 "blank" values], PetFood[2821 "blank" values], Tofu[2821 "blank" values], WildHarvested[2821 "blank" values]

59. updateTime

i) Trim leading and trailing whitespace – There were no leading and trailing space in any of the value of this column.

ii) We converted this column into date column such that we have data in ISO format. This transformation affected 8251 cells.



c. Quantifying the results of cleaning (e.g., provide a table of changes along with appropriate quantification) [5%]

ID	Column Name	Steps	Cells Affected
1	FMID	1.1 Convert to Numbers	8806
2	Market Name	2.1 Remove trailing and leading spaces including consecutive spaces	461
		2.2 Convert special character '&' to 'and'	192
		2.3 Remove duplicates using clustering	691
3	Website	3.1 Remove trailing and leading spaces including consecutive spaces	27
		3.2 Convert to all URL to lowercase	606
		3.3 Remove duplicates using clustering	168
4	Facebook	4.1 Remove trailing and leading spaces including consecutive spaces	43
		4.2 Remove all duplicates using clustering	117
5	Twitter	5.1 Remove trailing and leading space including consecutive spaces	10
		5.2 Remove all duplicates using clustering	33
6	Youtube	6.1 Remove trailing and leading spaces including consecutive spaces	4
		6.2 Remove duplicates using clustering	17
7	OtherMedia	7.1 Remove trailing and leading spaces including consecutive spaces	40
		7.2 Remove duplicates using clustering	63
8	street	8.1 Remove trailing and leading spaces including consecutive spaces	407
		8.2 Change street names case sensitive to title case	2507
		8.3 Remove all duplicates using clustering	179
9	city	9.1 Remove trailing and leading spaces including consecutive spaces	969
		9.2 Change city names case sensitive to title case	342
		9.3 Remove duplicates using clustering	104
10	County	10.1 Remove trailing and leading spaces including consecutive spaces	0
		10.2 Change county names case sensitive to title case	232
		10.3 Remove duplicates using clustering	13
11	State	11.1 Remove trailing and leading spaces including consecutive spaces	0
		11.2 Change state names case sensitive to title case	60
		11.3 Remove duplicates using clustering	0
12	zip	12.1 Convert to Numbers	7838
		12.2 Remove values other than numbers (10 nonnumeric value like FL, IL, OR and 947 blank value)	21
		12.3. Retrieve zip using Latitude and Longitude values	584
13	Season1Date	13.1 Remove trailing and leading spaces on Season1Date	110

		13.2 Create new columns From and To based on separator value “to”	5703
		13.3 Remove trailing and leading spaces Season1FromDate	5673
		13.4 Remove trailing and leading spaces Season1ToDate	5571
		13.5 Convert Season1FromDate to Date format and convert to ISO standard “YYYY-M-d” format	4966
		13.6 Convert Season1ToDate to date format and convert to ISO standard “YYYY-M-d” format	4823
		13.7 Convert to ISO Standard Date (or only Month) and remove default timestamp	4940
14	Season1Time	14.1 Remove trailing and leading spaces	0
		14.2 Convert to lowercase	5868
		14.3 Split into several columns based on the separator “;”	5868
15	Season2Date	15.1 Remove trailing and leading spaces Season2Date	14
		15.2 Create new columns From and To based on separator value “to”	464
		15.3 Remove trailing and leading spaces Season2FromDate	459
		15.4 Remove trailing and leading spaces Season2ToDate	446
		15.5 Convert Season2FromDate to Date format and convert to ISO standard “YYYY-M-d” format	442
		15.6 Convert Season2ToDate to Date format and convert to ISO standard “YYYY-M-d” format	423
		15.7 Convert to ISO Standard Date (or only Month) and remove default timestamp	442
16	Season2Time	16.1 Remove trailing and leading spaces	0
		16.2 Convert to lowercase	455
		16.3 Split into several columns based on the separator “;”	455
17	Season3Date	17.1 Remove trailing and leading spaces including consecutive spaces	2
		17.2 Create new columns From and To	78
		17.3 Remove trailing and leading spaces Season3FromDate	77
		17.4 Remove trailing and leading spaces Season3ToDate	76
		17.5 Convert Season3FromDate to Date format and convert to ISO standard “YYYY-M-d”	73
		17.6 Convert Season3ToDate to Date format and convert to ISO standard “YYYY-M-d” format	71
		17.7 Convert to ISO Standard Date (or only Month) and remove default timestamp	73
18	Season3Time	18.1 Remove trailing and leading spaces	0
		18.2 Convert to lowercase	75
		18.3 Split into several columns based on the separator “;”	75
19	Season4Date	19.1 Remove trailing and leading spaces	0
		19.2 Create new columns From and To	5

		19.3 Remove trailing and leading spaces Season3FromDate	5
		19.4 Remove trailing and leading spaces Season3ToDate	5
		19.5 Convert Season3FromDate to Date format and convert to ISO standard “YYYY-M-d”	5
		19.6 Convert Season3ToDate to Date format and convert to ISO standard “YYYY-M-d”	5
		19.7 Convert to ISO Standard Date and remove default timestamp	5
20	Season4Time	20.1 Remove trailing and leading spaces	0
		20.2 Convert to lowercase	5
		20.3 Split into several columns based on the separator “;”	5
21	x	21.1 Convert to Numbers	8778
22	y	22.1 Convert to Numbers	8778
23	Location	23.1 Remove trailing and leading spaces including consecutive spaces	0
24	Credit	24.1 Remove trailing and leading spaces	0
		24.2 Replace values to NULL if other than “Y” or “N”	0
25	WIC	25.1 Remove trailing and leading spaces	0
		25.2 Replace values to NULL if other than “Y” or “N”	0
26	WICCash	26.1 Remove trailing and leading spaces	0
		26.2 Replace values to NULL if other than “Y” or “N”	0
27	SFMNP	27.1 Remove trailing and leading spaces	0
		27.2 Replace values to NULL if other than “Y” or “N”	0
28	SNAP	28.1 Remove trailing and leading spaces	0
		28.2 Replace values to NULL if other than “Y” or “N”	0
29	Organic	29.1 Remove trailing and leading spaces	0
		29.2 Replace values to NULL if other than “Y” or “N”	5011
30	Bakedgoods	30.1 Remove trailing and leading spaces	0
		30.2 Replace values to NULL if other than “Y” or “N”	2821
31	Cheese	31.1 Remove trailing and leading spaces	0
		31.2 Replace values to NULL if other than “Y” or “N”	2821
32	Crafts	32.1 Remove trailing and leading spaces	0
		24.2 Replace values to NULL if other than “Y” or “N”	2821
33	Flowers	33.1 Remove trailing and leading spaces	0
		33.2 Replace values to NULL if other than “Y” or “N”	2821
34	Eggs	34.1 Remove trailing and leading spaces	0
		34.2 Replace values to NULL if other than “Y” or “N”	2821
35	Seafood	35.1 Remove trailing and leading spaces	0
		35.2 Replace values to NULL if other than “Y” or “N”	2821
36	Herbs	36.1 Remove trailing and leading spaces	0
		36.2 Replace values to NULL if other than “Y” or “N”	2821

37	Vegetables	37.1 Remove trailing and leading spaces	0
		37.2 Replace values to NULL if other than “Y” or “N”	2821
38	Honey	38.1 Remove trailing and leading spaces	0
		38.2 Replace values to NULL if other than “Y” or “N”	2821
39	Jams	39.1 Remove trailing and leading spaces	0
		39.2 Replace values to NULL if other than “Y” or “N”	2821
40	Maple	40.1 Remove trailing and leading spaces	0
		40.2 Replace values to NULL if other than “Y” or “N”	2821
41	Meat	41.1 Remove trailing and leading spaces	0
		41.2 Replace values to NULL if other than “Y” or “N”	2821
42	Nursey	42.1 Remove trailing and leading spaces	0
		42.2 Replace values to NULL if other than “Y” or “N”	2821
43	Nuts	43.1 Remove trailing and leading spaces	0
		43.2 Replace values to NULL if other than “Y” or “N”	2821
44	Plants	44.1 Remove trailing and leading spaces	0
		44.2 Replace values to NULL if other than “Y” or “N”	2821
45	Poultry	45.1 Remove trailing and leading spaces	0
		45.2 Replace values to NULL if other than “Y” or “N”	2821
46	Prepared	46.1 Remove trailing and leading spaces	0
		46.2 Replace values to NULL if other than “Y” or “N”	2821
47	Soap	47.1 Remove trailing and leading spaces	0
		47.2 Replace values to NULL if other than “Y” or “N”	2821
48	Trees	48.1 Remove trailing and leading spaces	0
		48.2 Replace values to NULL if other than “Y” or “N”	2821
49	Wine	49.1 Remove trailing and leading spaces	0
		49.2 Replace values to NULL if other than “Y” or “N”	2821
50	Coffee	50.1 Remove trailing and leading spaces	0
		50.2 Replace values to NULL if other than “Y” or “N”	2821
51	Beans	51.1 Remove trailing and leading spaces	0
		51.2 Replace values to NULL if other than “Y” or “N”	2821
52	Fruits	52.1 Remove trailing and leading spaces	0
		52.2 Replace values to NULL if other than “Y” or “N”	2821
53	Grains	53.1 Remove trailing and leading spaces	0
		53.2 Replace values to NULL if other than “Y” or “N”	2821
54	Juices	54.1 Remove trailing and leading spaces	0
		54.2 Replace values to NULL if other than “Y” or “N”	2821
55	Mushrooms	55.1 Remove trailing and leading spaces	0
		56.2 Replace values to NULL if other than “Y” or “N”	2821
56	PetFood	56.1 Remove trailing and leading spaces	0

		56.2 Replace values to NULL if other than “Y” or “N”	2821
57	Tofu	57.1 Remove trailing and leading spaces	0
		57.2 Replace values to NULL if other than “Y” or “N”	2821
58	WildHarvested	58.1 Remove trailing and leading spaces	0
		58.2 Replace values to NULL if other than “Y” or “N”	2821
59	updateTime	59.1 Remove trailing and leading spaces	0
		59.2 Convert to ISO Standard Date	8251

3. Developing a relational schema [15%]

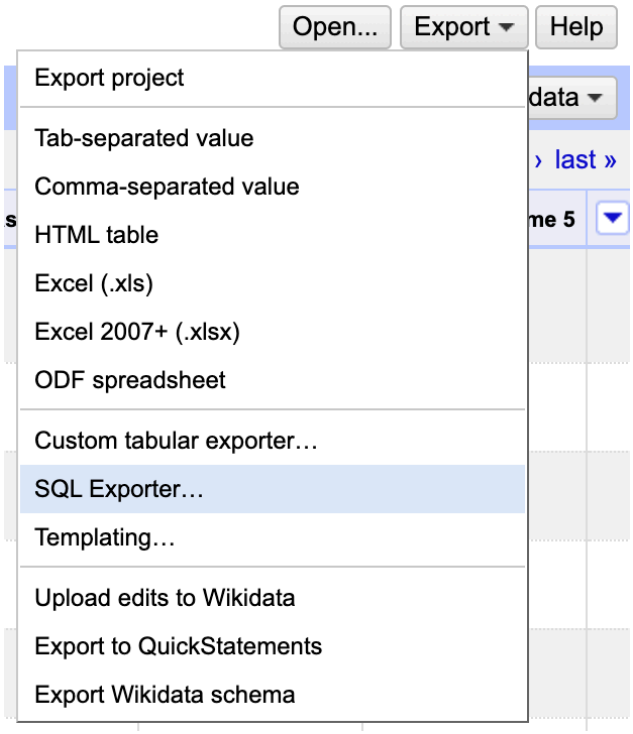
a. Identifying the appropriate integrity constraints [5%]

1. FMID: Integrity constraint check to ensure all FMIDs are unique with no blank values. To ensure this FMID is the primary key of the table STORE_DETAILS. FMID is not duplicated in the table STORE_DETAILS.
2. Market name: All stores should have a valid market name. Integrity constraint check to ensure that no FMID should be associated with NULL market name.
3. Product: Each farmers market should have at least one product available. Integrity constraint is to ensure no farmer market have all the products unavailable. There are 30 products in total provided in PRODUCTS table having PRODUCT_ID from 6 to 35.
4. Payment: Each farmers market should have at least one product method available Integrity constraint is to ensure no farmer market have all the payment options as unavailable I.e. marked as “N” or “UNKOWN”. There are 5 payment options – Credit, WIC, WICCash, SFMNP and SNAP; provided in PRODUCTS table having PRODUCT_ID from 1 to 5.
5. Season From and End Date and Time: Every farmers market should have at least one time period meaning at least one season when it is open - both date/month and time.
6. Latitude and Longitude: As per geographical standards, Latitudes must range from -90 to 90 and Longitudes must range from -180 to 180. Integrity constraint check is to ensure that Latitude and Longitude column of STORE_DETAILS table fall between the relevant ranges only.
7. Zip: Integrity constraint check to ensure valid format for zip having numeric values of length 4, 5 or of the format comprising 5 numeric values I.e. XXXXX followed by a hyphen and then followed by a 4-digit numeric value I.e. XXXXX-XXXX. Only Puerto Rico has an exception of zip with length 3.
8. Combination of City, County and State: Integrity constraint to check the combination against actual data from US Gov data. 28889 rows from US Gov data was loaded into the table CITY_COUNTY_STATE_US_GOV. Columns from table STATES, COUNTIES, CITIES are compared with US GOV table to check for the data accuracy and validity.
9. Season From And To Dates: For every season I.e. Season1/2/3/4, integrity check to ensure Season From Date appears before Season End Date if the values represent date of the form “YYYY-MM-DD". Note that if the values represent Month then the ordering of the From and To Date doesn’t matter.
10. Store timings: To ensure that Store timings are valid and meaningful, integrity constraint check to ensure all the starting 3 digits of the Time column of table STORE_SEASONAL_HOURS represent valid weekday values such as “mon” for Monday, “tue” for Tuesday and so on.

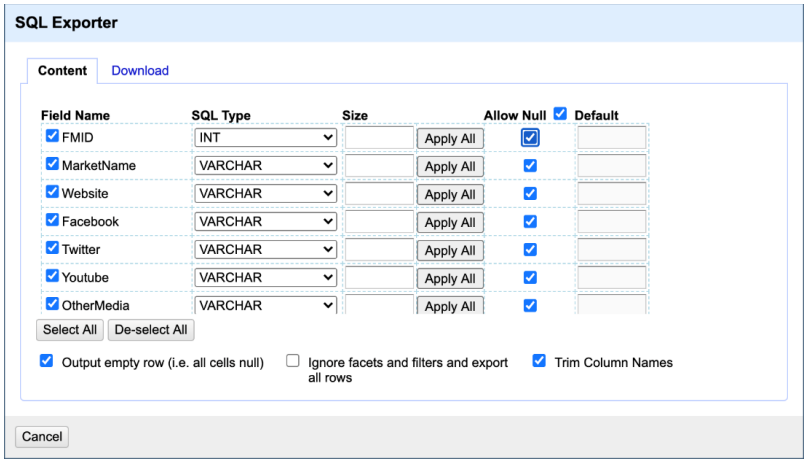
11. Website: Integrity constraint check to ensure all the non-blank website values have a valid format of the website that starts either with "http://" or "https://". Though we cannot validate the website, meaning if the URL is correct or invalid as such, but these integrity check to ensure the format is correct for the non-blank website.

b. Loading data into a database with proper schema [3%]

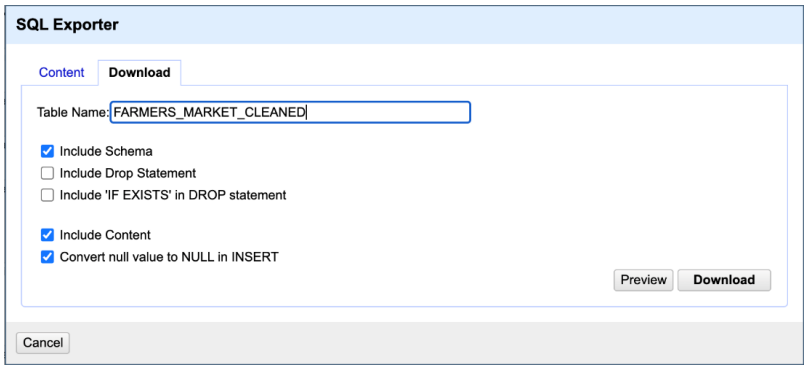
After the data has been cleaned in Open Refine, we used SQL Exporter functionality of Open Refine.



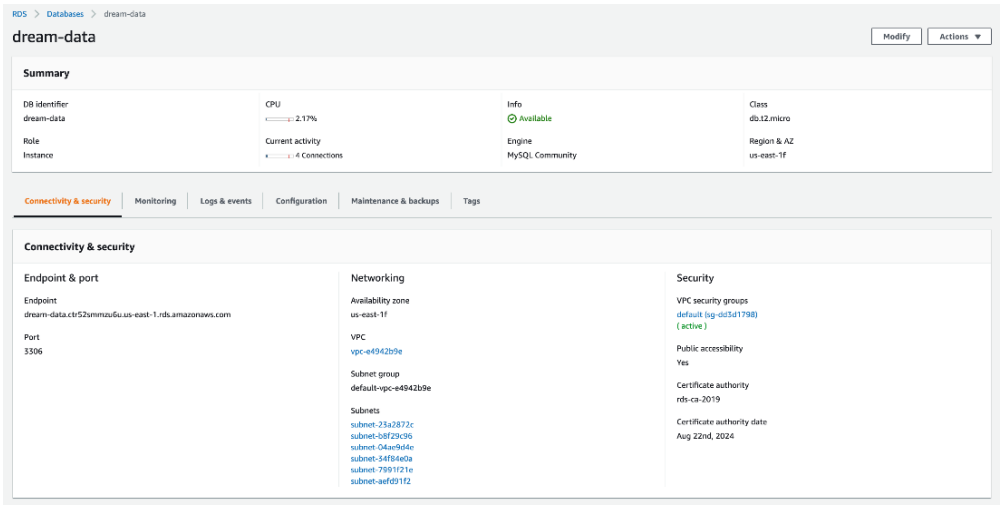
After selection SQL Exporter, select all columns which you would like to be exported, making sure all data type of columns is appropriate.



Then, go to Download tab, provide appropriate table name and download SQL file.



After getting SQL script, we loaded in MySQL database. We have hosted MySQL on Amazon Web Services Relational Database Service (RDS).



The tables are in 3NF form to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management. This schema allows adaptability and scalability for using this data for developing any application:

- **Farmers Market Details:**

States: All states have been stored.
Counties: All counties with foreign key from States table has been stored.
Cities: All cities with foreign key from Counties table has been stored.
Store Details: This table stores data related to Farmers Market – name, website, social media pages, address, zip, latitude, longitude, and foreign keys from States, Counties and City table.

- **Farmers Market Product Availability:**

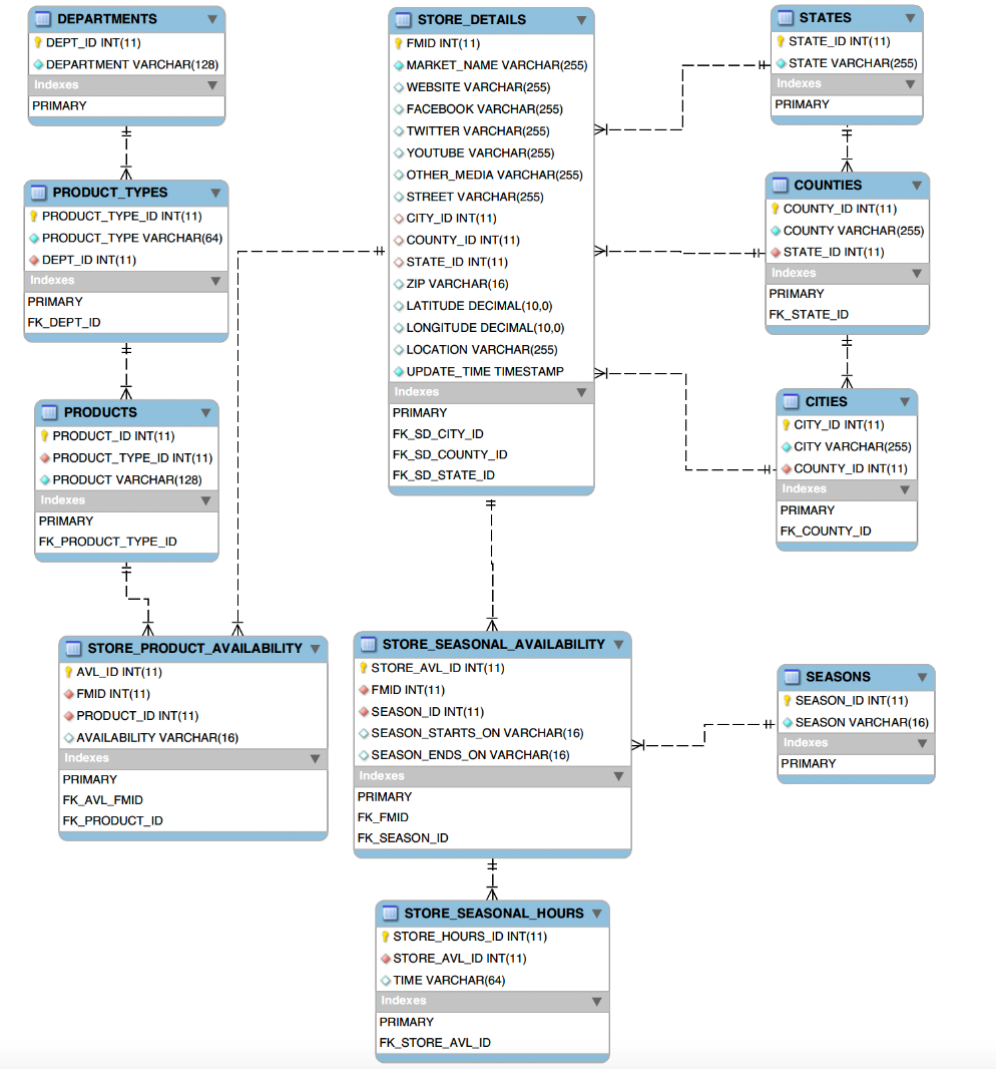
Departments: This table contains different departments of Farmers Markets. Departments have been formed by grouping similar product types.
Product Types: This table contains product types grouping similar products. This table contains foreign key from Departments table.
Products: All products including different payment types and products. This table foreign key from Product Types table.

Store Product Availability: This table stores all products' availability with foreign key from Store Details and Products table.

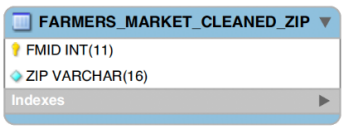
- **Farmers Market Availability:**

Seasons: Farmers Markets have availability based on seasons. This table stores all possible seasons.
Store Seasonal Availability: This table stores season start and end date for each market for each season. It has foreign key from Seasons and Store Details table.
Store Seasonal Hours: This table stores hours of store. This table references Store Seasonal Availability's primary key as foreign key.

The following ER diagram represents the relational schema for storing cleaned data:



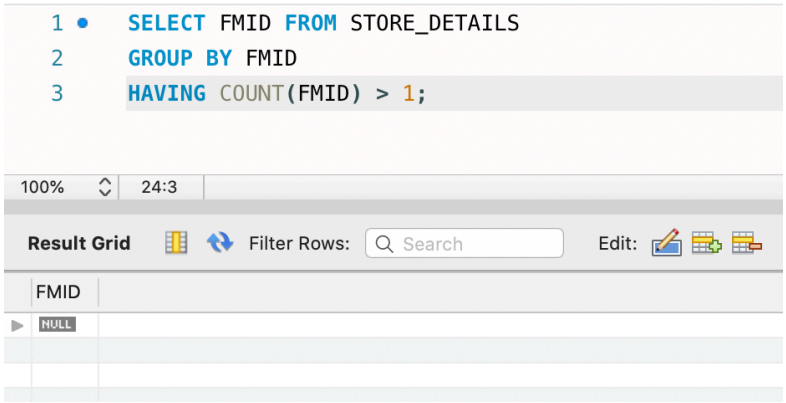
Data Load Process – After downloading SQL file from Open Refine with table definition (DDL) and content (DML), we loaded in MySQL table (FARMERS_MARKET_CLEANED). We also enriched zip column by using Google API based on latitude and longitude data. This data was stored in different table (FARMERS_MARKET_CLEANED_ZIP).



Based, on these two tables, we loaded relational tables by executing SQL queries.
i.e.
INSERT INTO TABLE (<COLUMN_1, COLUMN_2, …. COLUMN_N>)
SELECT COLUMN 1, COLUMN_2, …. COLUMN_N FROM FARMERS_MARKET_CLEANED;
You can refer to appendix section for detailed SQL queries for loading each relational table.

c. Writing queries to check the integrity constraints [7%]

1. FMID



All FMID values are unique nonblank values in the table STORE_DETAILS.

2. Market Name

```
SELECT FMID,MARKET_NAME FROM STORE_DETAILS
WHERE MARKET_NAME IS NULL;
```

100% 27:2

Result Grid | Filter Rows: Search | Edit:

FMID	MARKET_NAME
NULL	NULL

All stores have a corresponding market name. No FMID found without a market name.

3. Product

```
SELECT COUNT(*) FROM
(SELECT SD.FMID,SD.MARKET_NAME  FROM STORE_DETAILS SD
JOIN STORE_PRODUCT_AVAILABILITY PRD_AVL  ON SD.FMID = PRD_AVL.FMID
JOIN PRODUCTS PRD  ON PRD_AVL.PRODUCT_ID = PRD.PRODUCT_ID
JOIN PRODUCT_TYPES PRD_TYP  ON PRD.PRODUCT_TYPE_ID = PRD_TYP.PRODUCT_TYPE_ID
JOIN DEPARTMENTS DEPT  ON PRD_TYP.DEPT_ID = DEPT.DEPT_ID
WHERE DEPT.DEPARTMENT != 'Payment' AND PRD_AVL.AVAILABILITY = 'UNKNOWN'
GROUP BY SD.FMID,SD.MARKET_NAME  HAVING COUNT(PRD_AVL.AVAILABILITY) = 30) A;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)
2784

There were 2784 market names out of 8806 market names found which had all its product as “UNKNOWN” meaning all their products were not known if available or not. Inner SELECT query will give the list of all such FMID and farmer market names that violate the integrity constraint.

4. Payment

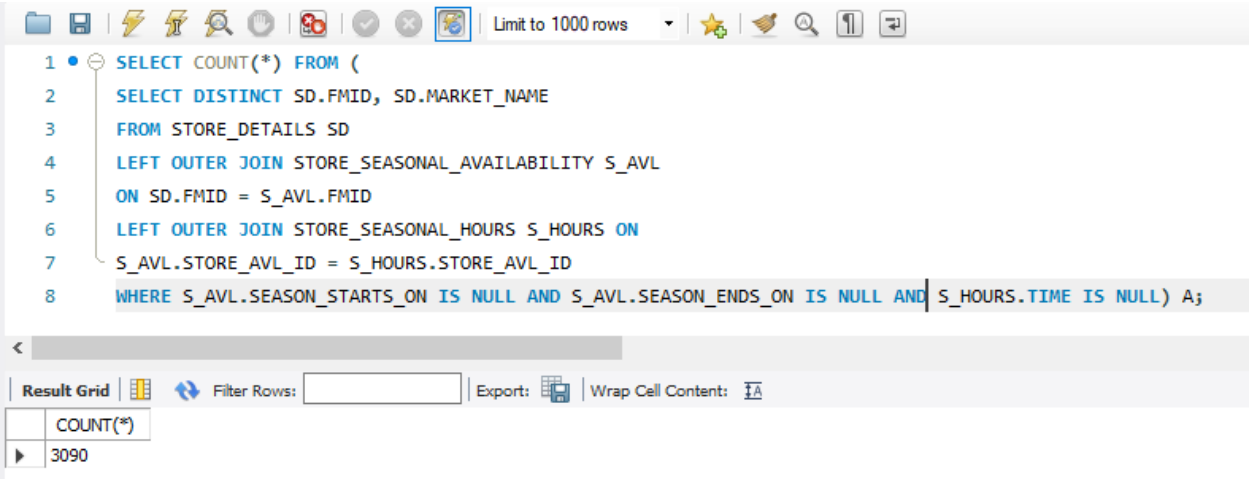
```
SELECT SD.FMID,SD.MARKET_NAME
FROM STORE_DETAILS SD
JOIN STORE_PRODUCT_AVAILABILITY PRD_AVL
ON SD.FMID = PRD_AVL.FMID
JOIN PRODUCTS PRD
ON PRD_AVL.PRODUCT_ID = PRD.PRODUCT_ID
JOIN PRODUCT_TYPES PRD_TYP
ON PRD.PRODUCT_TYPE_ID = PRD_TYP.PRODUCT_TYPE_ID
JOIN DEPARTMENTS DEPT
ON PRD_TYP.DEPT_ID = DEPT.DEPT_ID
WHERE DEPT.DEPARTMENT = 'Payment' AND PRD_AVL.AVAILABILITY IN ('UNKNOWN','N')
GROUP BY SD.FMID,SD.MARKET_NAME
HAVING COUNT(PRD_AVL.AVAILABILITY) = 5) A;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)
2854

There were 2854 market names found that didn't have any payment options available. Inner SELECT query provides the list of FMID and market names violating the integrity constraint check.

5. Season From and End Date and Time



There were 3090 FMIDs with market names that don't have any information about Season From and End Date and Time. Inner SELECT query provides the list of FMID and Market Names violating the integrity constraint check.

6. Latitude and Longitude

All Latitude and Longitude values were found to be valid.

7. Zip

Below query was used and it found 4 invalid zip codes. FMID 1012563 had nonnumeric zip as "726o1", FMID 1009876 had shorter zip as "33" of length 2, FMID 1005771 had longer zip as "513338" of length 6 and FMID 1006166 had longer zip as "550424" of length 6.

```
SELECT FMID,MARKET_NAME,ZIP
FROM (SELECT SD.FMID,SD.MARKET_NAME,SD.ZIP
        FROM STORE_DETAILS SD
        JOIN STATES ST
        ON SD.STATE_ID = ST.STATE_ID
        WHERE SD.ZIP IS NOT NULL AND ST.STATE != 'Puerto Rico') TEMP
WHERE ZIP NOT REGEXP '^[0-9][0-9][0-9][0-9][0-9]$'
AND ZIP NOT REGEXP '^[0-9][0-9][0-9][0-9]$'
AND ZIP NOT REGEXP '^[0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$'
UNION
SELECT FMID,MARKET_NAME,ZIP
FROM (SELECT SD.FMID,SD.MARKET_NAME,SD.ZIP
        FROM STORE_DETAILS SD
        JOIN STATES ST
        ON SD.STATE_ID = ST.STATE_ID
        WHERE SD.ZIP IS NOT NULL AND ST.STATE = 'Puerto Rico') TEMP
WHERE ZIP NOT REGEXP '^[0-9][0-9][0-9]$'
AND ZIP NOT REGEXP '^[0][0-9][0-9][0-9][0-9]$'
AND ZIP NOT REGEXP '^[0][0][0-9][0-9][0-9]$';
```

100% 43:18

Result Grid Filter Rows: Search Export:

FMID	MARKET_NAME	ZIP
1005771	Everly Farmers Market	513338
1006166	Buffalo Center Farmers Market	550424
1012563	Harrison Farmer's Market	726o1
1009876	Bradford Farmers Market	33

8. Combination of City, County and State:

```
SELECT DISTINCT CT.CITY AS CITY,CNT.COUNTY AS COUNTY,ST.STATE AS STATE
FROM STATES ST
JOIN COUNTIES CNT
ON ST.STATE_ID = CNT.STATE_ID
JOIN CITIES CT
ON CNT.COUNTY_ID = CT.COUNTY_ID
WHERE NOT EXISTS (SELECT DISTINCT CITY,COUNTY,STATE
                  FROM CITY_COUNTY_STATE_US_GOV);
```

100% 37:8

Result Grid Filter Rows: Search Export:

CITY	COUNTY	STATE

All the values of State, County and City were found to be valid and accurate

9. Season From And To Dates:

Season From and To dates have been validated in three steps as from and to values are not consistent:

1) Using Dates

```
SELECT SD.FMID,SD.MARKET_NAME,S_AVL.SEASON_STARTS_ON,S_AVL.SEASON_ENDS_ON
FROM STORE_DETAILS SD
JOIN STORE_SEASONAL_AVAILABILITY S_AVL
ON SD.FMID = S_AVL.FMID
WHERE STR_TO_DATE(S_AVL.SEASON_STARTS_ON,'%Y-%m-%d') > STR_TO_DATE(S_AVL.SEASON_ENDS_ON,'%Y-%m-%d');
```

100% 101:5

Result Grid Filter Rows: Search Export:

FMID	MARKET_NAME	SEASON_STARTS_ON	SEASON_ENDS_ON
1011959	Clark Park Farmer's Market	2016-10-1	2016-5-7
1018773	Sedalia Area Farmers' Market	2017-11-2	2017-4-27

There were 2 FMIDs found which failed the above integrity constraint check

2) Using Months

```
SELECT * FROM (SELECT SD.FMID,SD.MARKET_NAME,
        CASE WHEN S_AVL.SEASON_STARTS_ON = 'January' THEN 1 WHEN S_AVL.SEASON_STARTS_ON = 'February' THEN 2 WHEN S_AVL.SEASON_STARTS_ON = 'March' THEN 3
             WHEN S_AVL.SEASON_STARTS_ON = 'April' THEN 4 WHEN S_AVL.SEASON_STARTS_ON = 'May' THEN 5 WHEN S_AVL.SEASON_STARTS_ON = 'June' THEN 6
             WHEN S_AVL.SEASON_STARTS_ON = 'July' THEN 7 WHEN S_AVL.SEASON_STARTS_ON = 'August' THEN 8 WHEN S_AVL.SEASON_STARTS_ON = 'September' THEN 9
             WHEN S_AVL.SEASON_STARTS_ON = 'October' THEN 10 WHEN S_AVL.SEASON_STARTS_ON = 'November' THEN 11 WHEN S_AVL.SEASON_STARTS_ON = 'December' THEN 12
        END AS SEASON_STARTS_ON_MON,
        CASE WHEN S_AVL.SEASON_ENDS_ON = 'January' THEN 1 WHEN S_AVL.SEASON_ENDS_ON = 'February' THEN 2 WHEN S_AVL.SEASON_ENDS_ON = 'March' THEN 3
             WHEN S_AVL.SEASON_ENDS_ON = 'April' THEN 4 WHEN S_AVL.SEASON_ENDS_ON = 'May' THEN 5 WHEN S_AVL.SEASON_ENDS_ON = 'June' THEN 6
             WHEN S_AVL.SEASON_ENDS_ON = 'July' THEN 7 WHEN S_AVL.SEASON_ENDS_ON = 'August' THEN 8 WHEN S_AVL.SEASON_ENDS_ON = 'September' THEN 9
             WHEN S_AVL.SEASON_ENDS_ON = 'October' THEN 10 WHEN S_AVL.SEASON_ENDS_ON = 'November' THEN 11 WHEN S_AVL.SEASON_ENDS_ON = 'December' THEN 12
        END AS SEASON_ENDS_ON_MON,S_AVL.SEASON_STARTS_ON,S_AVL.SEASON_ENDS_ON
    FROM STORE_DETAILS SD
    JOIN STORE_SEASONAL_AVAILABILITY S_AVL ON SD.FMID = S_AVL.FMID
    WHERE (S_AVL.SEASON_STARTS_ON IS NOT NULL AND S_AVL.SEASON_ENDS_ON IS NOT NULL)
        AND (S_AVL.SEASON_STARTS_ON REGEXP '^January|February|March|April|May|June|July|August|September|October|November|December$'
        AND S_AVL.SEASON_ENDS_ON REGEXP '^January|February|March|April|May|June|July|August|September|October|November|December$')) A
WHERE SEASON_STARTS_ON_MON > SEASON_ENDS_ON_MON;
```

100% 49:17

Result Grid Filter Rows: Search Export:

FMID	MARKET_NAME	SEASON_STARTS_ON_MON	SEASON_ENDS_ON_MON	SEASON_STARTS_ON	SEASON_ENDS_ON
1000108	Ely Farmers Market	11	3	November	March
1000865	Columbus Winter Farmers Market	11	3	November	March
1001511	Marco Island Farmers Market	11	4	November	April
1002027	Oceanside Farmer's market at Lake Worth Beach	11	5	November	May
1002273	Green Spring Station Winter Farmers Market	12	5	December	May
1003898	LMU Farmers Market	9	4	September	April
1003932	Roots in the City Urban Farmers' Market	11	5	November	May
1004929	Certified SC Winter Farmers Market	11	3	November	March
1005022	Manhattan Beach Farmers Market	9	5	September	May
1005379	Norwich Winter Farmers Market	11	4	November	April
1006039	Bronson Winter Farmers' Market	11	4	November	April
1006056	Waterbury Farmers Market	11	4	November	April
1006508	Winter Sun Farms Indoor Winter Market	12	4	December	April
1000107	Ely Farmers Market	11	4	November	April
1000176	North Raleigh Farmers' Market	11	3	November	March
1002854	East Goshen Farmers Market	11	4	November	April
1003022	Chesterfield Farmers Market	11	4	November	April
1004007	Marshall County Farmers Market	11	4	November	April
1004024	Farmer's Market @ Woburn's Spence Farm	11	5	November	May
1006078	Santa Fe Farmers' Market	12	3	December	March

Here no conflicting values found where season starting month is before season ending month. The print screen represents where starting month number is higher than ending month, but this is acceptable as a store can be open between November of current year to May of next year.

3) Using combination of dates and months

```
SELECT * FROM
    (SELECT
        SD.FMID,SD.MARKET_NAME,S_AVL.SEASON_STARTS_ON,S_AVL.SEASON_ENDS_ON
    FROM STORE_DETAILS SD
    JOIN STORE_SEASONAL_AVAILABILITY S_AVL ON SD.FMID = S_AVL.FMID
    WHERE S_AVL.SEASON_STARTS_ON IS NOT NULL
          AND S_AVL.SEASON_ENDS_ON IS NOT NULL
          AND (S_AVL.SEASON_STARTS_ON NOT REGEXP '^(January|February|March|April|May|June|July|August|September|October|November|December)$'
          OR S_AVL.SEASON_ENDS_ON NOT REGEXP '^(January|February|March|April|May|June|July|August|September|October|November|December)$')
          AND (S_AVL.SEASON_STARTS_ON NOT REGEXP '^[0-9]'
          OR S_AVL.SEASON_ENDS_ON NOT REGEXP '^[0-9]')) A;
```

100% 61:11

Result Grid Filter Rows: Search Export:

FMID	MARKET_NAME	SEASON_STARTS_ON	SEASON_ENDS_...
1001137	Hill Country Farmers Market Association	April	2011-11-4
1001139	Hill Country Farmers Market Association	April	2011-9-24
1001963	Garden Shop Nursery Farmers Market	October	ber
1002517	Fairhaven Winter Farmers' Market	October	ber
1002724	Seminole Heights Sunday Morning Ma...	October	ber
1002777	Main Street Calumet Market	October	ber
1001431	Marlboro County Farmers Market	October	ber
1002222	Alcona Farmers Market	October	ber
1004736	Lansing City Market	2012-7-1	December
1008935	water canyon farmers market	2013-1-1	End Date 12/31/1
1004248	Cadillac Area Farmers Market	2012-6-15	June
1004753	The Market at Bissel Gardens	2012-7-11	November
1000788	Weirton Farmers Market	2012-7-9	October
1000789	Weirton Events Center	2012-7-12	October
1002312	Highland Farmers Market	2012-6-2	October
1002313	South Pearl Street Farmers' Market	2012-5-20	October
1002713	Troostwood Youth Garden Market	2011-5-21	October
1003469	Acton-Boxborough Farmers Market	2012-6-19	October
1003827	Ellsworth Farmers Market	2012-6-11	October
1004072	Downtown San Leandro Certified Farm...	2012-4-18	October
1004268	Emmett Farmers Market	2012-4-28	October
1005772	Silverton Farmers Market	2011-5-7	October
1007647	Andover Farmers Market	2012-5-1	October
1018884	Downtown Bloomington Association Fa...	2018-5-5	October
1011963	Wauconda Farmers Market	June 23	Sept. 8
1001722	Greenville Farmers Market	2012-6-2	September

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Here we found 6 rows with invalid season end month “ber” and valid end date with “End Date” text.

10. Store Timings

```
1 •  SELECT DISTINCT SUBSTR(TIME,1,3) AS WEEK_DAY
2    FROM STORE_SEASONAL_HOURS;
```

100% 27:2

Result Grid Filter Rows: Search Export:

WEEK_DAY
tue
sat
wed
thu
fri
mon
sun

All weekday values were found valid

12. Website

```
1 •  SELECT * FROM STORE_DETAILS
2    WHERE LOWER(SUBSTR(WEBSITE,1,7)) NOT LIKE "http://"
3    AND LOWER(SUBSTR(WEBSITE,1,8)) NOT LIKE "https://"
4    AND WEBSITE IS NOT NULL;
```

100% 25:4

Result Grid Filter Rows: Search Edit: Export/Import:

FMID	MARKET_NAME	WEBSITE	FACEBOOK	TWITTER	YOUTUBE	OTHER_MEDIA	STREET	CITY_ID	COUNTY_ID	STATE_ID	ZIP	LATITUDE	LONGITUDE	LOCATION	UPDATE_TIME
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

There we no nonblank Website values found that did not start with either "http://" or "https://"

4. Creating a workflow model [10%]

a. Identifying the key inputs, outputs of your workflow along with the dependencies [3%]

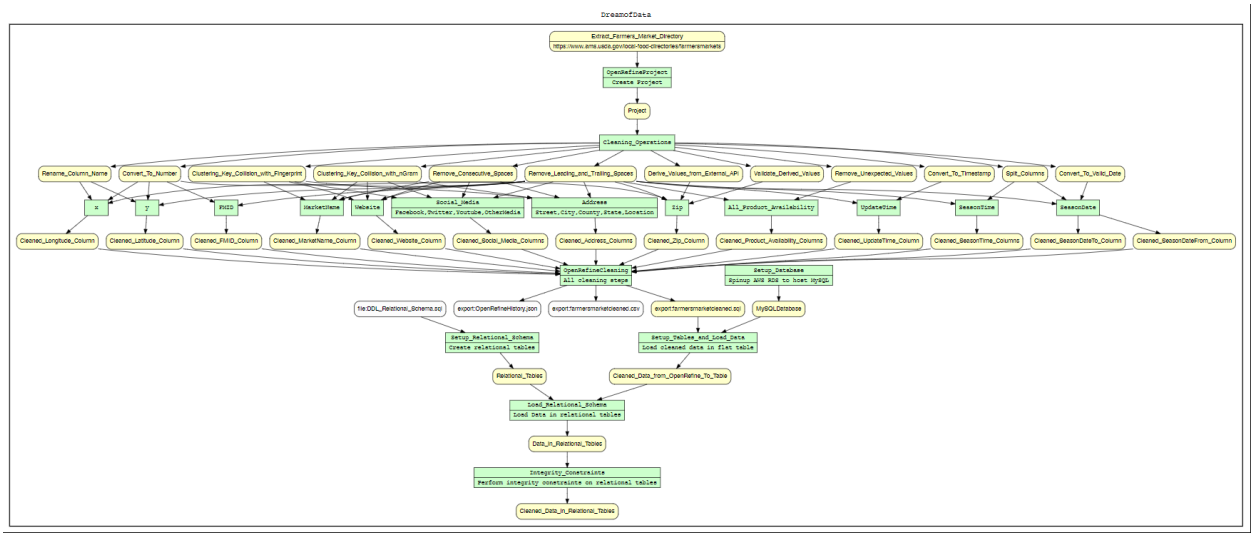
Step	Begin	In	Out	Params	URI
1	Extract Farmers Market Directory	Link to the data source	Farmers Market csv file		https://www.ams.usda.gov/local-food-directories/farmersmarkets
2	Create Open Refine Project	Farmers Market csv file	Open Refine Project		
3	Open Refine Project				

3.1	Cleaning Steps				
3.1.1	Remove Leading and Trailing Spaces	All Columns			
3.1.2	Remove consecutive spaces	Selected Columns			
3.1.3	Clustering				
3.1.3.1	Key-Collision with fingerprint	MarketName, Website, Facebook, Twitter, Youtube, OtherMedia, street, city, County, State, Location	MarketName, Website, Facebook, Twitter, Youtube, OtherMedia, street, city, County, State, Location		
3.1.3.2	Key-Collision with n-gram	MarketName, Website, Facebook, Twitter, Youtube, OtherMedia, street, city, County, State, Location	MarketName, Website, Facebook, Twitter, Youtube, OtherMedia, street, city, County, State, Location		
3.1.4	Convert to Number	FMID, x, y, zip	FMID, x, y, zip		
3.1.5	Rename Column Names	x, y	longitude, latitude		
3.1.6	Remove values except "Y" and "N" and replace with "UNKNOWN"	All products	All Products		
3.1.7	Convert to Timestamp	updateTime	updateTime		
3.1.8	Split Columns and converted to valid date formats	SeasonDates and SeasonTimes	Season{season_no}FromDate, Season{season_no}ToDate, Season{season_no}Time{count}	Season{season_no}Date, seperator("to"), Season{season_no}Time	

3.1.9	Derive values using external API	Zip, x, y	GoogleAPI_ZIP	x,y	
3.1.10	Validate derived values	GoogleAPI_ZIP	Cleaned CSV File		
3.2	Export DDL and DML from Open Refine		SQL Files		
3.3	Setup Database instance		DB Instance		
3.4	Create Tables for Cleaned Data	SQL Files, DB Instance	Cleaned Data in Table from Open Refine		
3.5	Create Relational Schema and Tables		Tables, Schema		
3.6	Load data to relational schema from cleaned data	Schema, Cleaned Table Data as SQL	Cleaned Master Data in Relational Tables		
3.7	Perform and Identify Integrity Constrains on data which comply and doesn't comply with Integrity Constraints	Cleaned Master Data in Relational Tables	Cleaned Data for Application use and further analysis		

b. A visual representation of your overall workflow, e.g., using YesWorkflow [4%]

See attachment: Overall_YW_Diagram.pdf



c. A visual representation of your OpenRefine workflow, e.g., using OR2YWTool [3%]

See attachment:

1. output_graph_all_openrefine_steps.pdf
2. output_graph_openrefine_steps_zip.pdf

5. Other factors [10%]

a. Further analysis/takeaways/challenges [5%] (In Progress)

Further Analysis

- After cleaning steps, identified Data which doesn't comply Integrity Constraints could be deleted as required
- Validation of Websites could be done using external API's to check for domain existence

Challenges:

- ZIP codes, all address fields majority of the data is blank and inaccurate
- Normalization is not ideal but could be improved

REFERENCES:

(Farmers Market, 2020)

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

https://www.listplanit.com/list-of-categories-for-an-organized-grocery-list/

https://developers.google.com/maps/documentation/geocoding/intro?hl=en_US

http://try.yesworkflow.org/

APPENDIX

Folder and File Details:

1. Project Report: This folder contains project report.
2. Source Data: This folder contains raw farmers market csv file used for cleaning.
3. SQL Scripts: This folder contains all SQL files used to create table, load tables and perform integrity constraints.
4. OpenRefine Cleaning: This folder contains json file representing all cleaning steps performed in OpenRefine.
5. Relational Schema Diagram: This folder contains relational schema diagram.
6. Cleaned Data: This folder contains cleaned farmers market csv file exported from OpenRefine.
7. YesWorkflow: This folder contains scripts used to generate YesWorkflow and diagrams for both – Overall cleaning process and OpenRefine cleaning steps.