LIS590DCL Final Project

Jialu Wang

1 Initial assessment of the dataset

Microsoft excel and Rstudio is initially used to form an opinion on the initial assessment.

1.1 Structure

- There are 59 attributes and 8,665 records in the dataset

```
> dim(dataSet)
[1] 8665   59
```

- There are 185145 missing values in the dataset:

```
> sum(is.na(dataSet))
[1] 185145
```

- There are 3023 records that miss the values for baked goods to wild harvest.

1.2 Content

The attributes are:

```
> names(dataSet)
 [1] "FMID"         "MarketName"    "Website"
 [4] "Facebook"     "Twitter"       "Youtube"
 [7] "OtherMedia"   "street"        "city"
[10] "County"       "State"         "zip"
[13] "Season1Date"  "Season1Time"   "Season2Date"
[16] "Season2Time"  "Season3Date"   "Season3Time"
[19] "Season4Date"  "Season4Time"   "x"
[22] "y"            "Location"      "Credit"
[25] "WIC"          "WICcash"       "SFMNP"
[28] "SNAP"         "Organic"       "Bakedgoods"
[31] "Cheese"       "Crafts"        "Flowers"
[34] "Eggs"         "Seafood"       "Herbs"
[37] "Vegetables"   "Honey"         "Jams"
[40] "Maple"        "Meat"          "Nursery"
[43] "Nuts"         "Plants"        "Poultry"
[46] "Prepared"     "Soap"          "Trees"
[49] "Wine"         "Coffee"        "Beans"
[52] "Fruits"       "Grains"        "Juices"
[55] "Mushrooms"    "PetFood"       "Tofu"
[58] "WildHarvested" "updateTime"
```

Attributes	Description	Type	Errors	Missing Values	Incons-istency	Note
FMID	uniquely identify a farmer market(FM)	number	N	0	N	range from 1000001 - 2000036
MarketName	full name of the market.	string	Not know	0	Y	
Website	Link to the farmer market.	URL/ string	Y	3458	Y	Can be vaildated
Facebook, Twitter, Youtube, Other Media	Link to / account name of medias	URL/string	Y	Y	Y	Can be vaildated
Street, city, county, state, zip	Location information	strings	Not know	Y	Y	Hard to test
Season Dates &Times	Open date & time in four seasons	Dates & times	Not know	Y	Y	8500+ missing except season 1
x, y	Longitude and latitude	number		29	N	Can be vaildated
location	Location type	Strings	Not know	5729	Y	No standard form
Credit, WIC, WICcash, SFMNP, SNAP	Payment Methods	Booleans	Not know	0	N	
Organic	Whether products are organic	Booleans	Not know	5070	N	
Bakedgoods-WildHarvested	Product types	Booleans	Not know	3023	N	
UpdatedTime	Latest updated time	Dates & times	Not know	0	Y	

1.3 data cleaning goals

- Free of error – clean wrong values
- Assure Completeness – clean empty values
- Ensure Consistency - standardization
- Provide Feedback for Improvements

1.4 Quality

- Accessibility: the data is accessible online

- Volume of Data: large data set
- Completeness: contain missing values
- Representation: easy to understand
- Consistency: data are not standardized
- Relevancy: attributes are relevant to their uses
- Timeliness: dataset is up to date and continuously updated
- Free of error: there are errors, for instance, some websites are invalid

1.5 Uses

- Possible Uses:

a. Find nearby farmer markets (by longitude and latitude)
b. Find markets within a certain region level or the detailed address of a market (by city, county, state)
c. Search for which products are supported in a market or which markets sell certain product
d. Find which payment methods are supported in a market or which markets support certain payment methods
e. Find on which season/time a market is open or which markets open in a certain period
f. Find online media presentation of a certain market

- Uses without needs for cleaning

a. Looking for interesting social media contents of farmer markets
b. Looking for nearby farmer markets
c. Search for markets that contain certain products

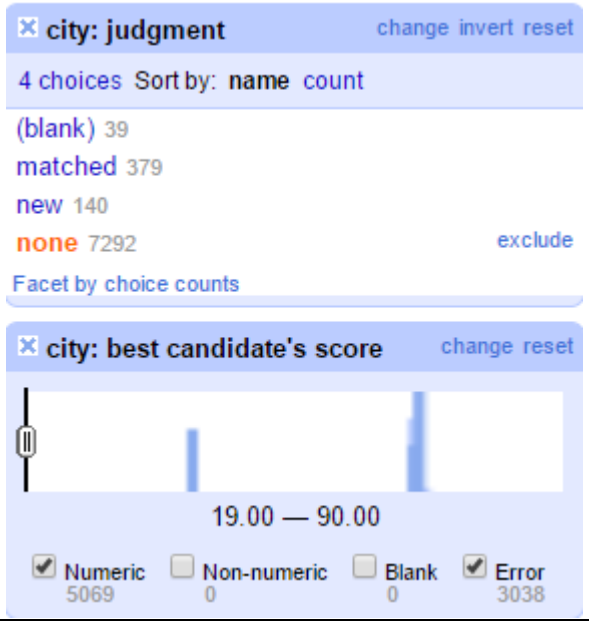
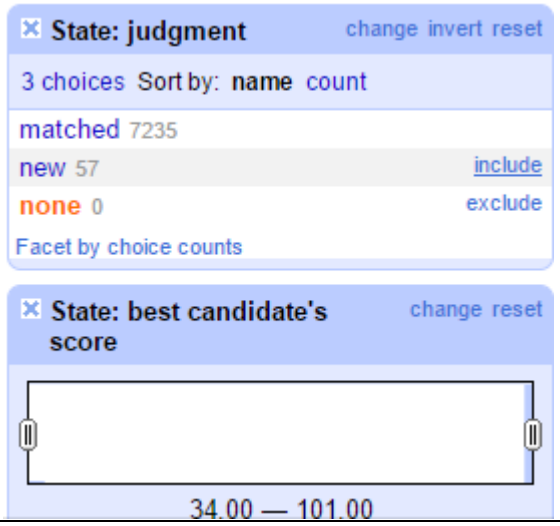
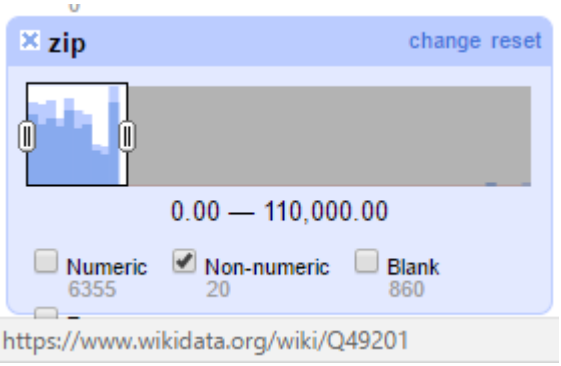
- Uses can never be clean enough

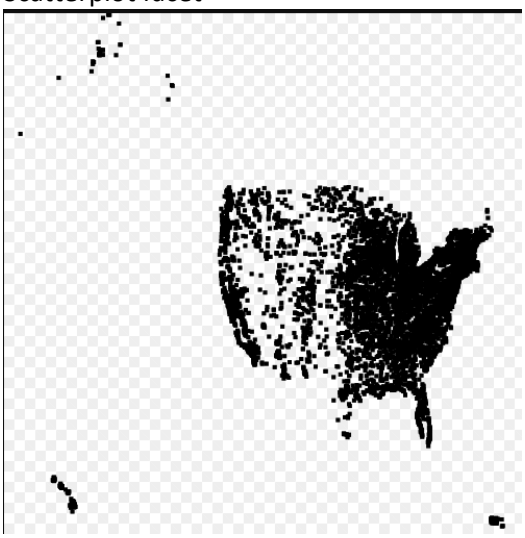
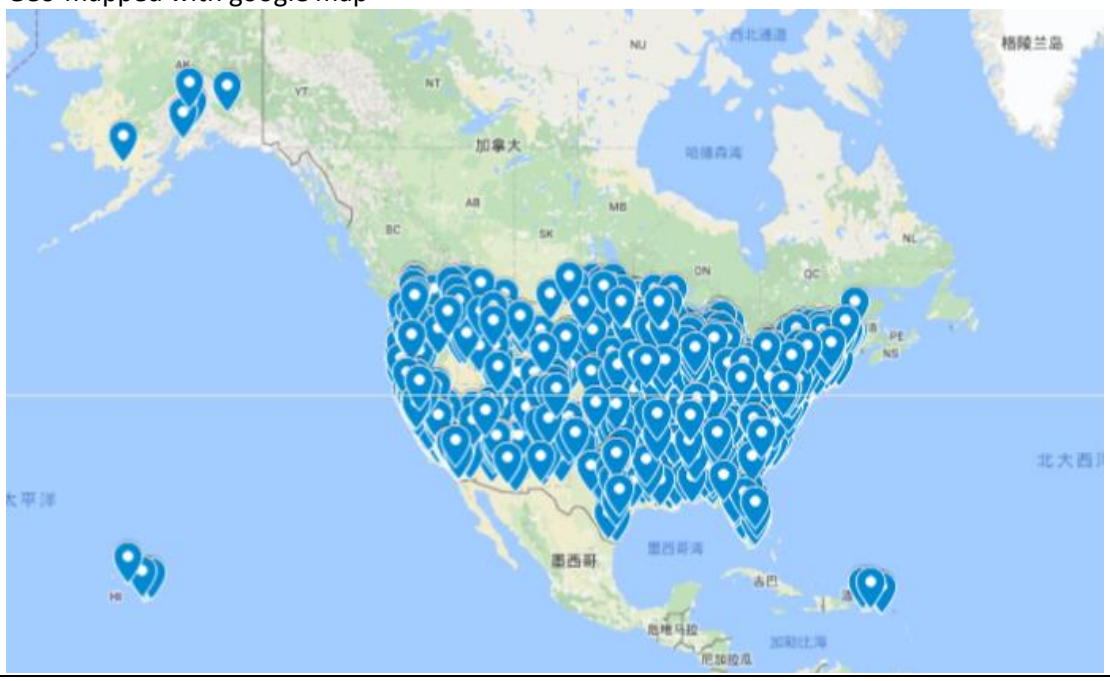
a. Count how many markets that fits a certain condition
b. Study distributions of opening times

2 DC with OpenRefine (quantify narratives + supplemental info)

We deal with the dataset in the order of its attributes.

Attributes	Actions
FMID	Sort as numeric values
MarketName	Clustering
Website	Hard to refine
Facebook	Hard to refine
Twitter	Hard to refine
Youtube	Hard to refine
OtherMedia	Hard to refine
street	Reconcile to text

city	Reconcile with cities in the United Stated, result is not good 
County	Reconcile cells in column State to type Q35657, result Is not good
State	Reconcile with states in the US (new: DC, Virgin Island, Puerto Rico) 
zip	Numeric Facet 

Season1Date Season1Time Season2Date Season2Time Season3Date Season3Time Season4Date Season4Time	Convert to date
X, y	Scatterplot facet  Geo-mapped with google map 
Location	Not really relevent

Credit WIC WICcash SFMNP SNAP	No need to clean
Organic	Clean null values and empty values
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	Remove empty values when analyzing product types.
updateTime	

3 Alternatives

3.1 R language

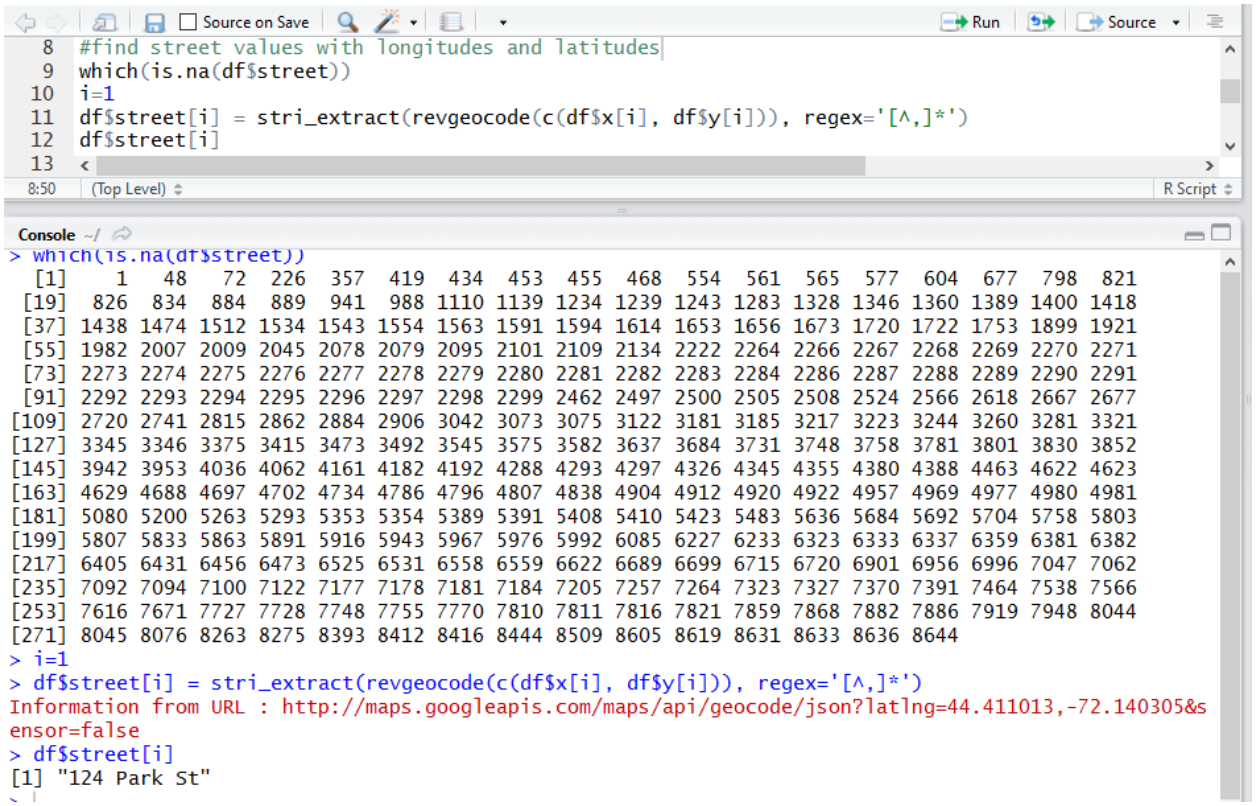
- Advantages:
  - Can remove/replace values with certain characteristics with a single command, without selecting every attribute once a time

```
3 df[df=="no" | df=="none" |df=="N/A" |df=="n/a" |df=="None" |df=="NA" |df=="-" ] <- NA
```

  - Can get statistics with single command

```
4 summary(df)
```

  - Can get street/city values from longitudes and latitudes



- Cleaning of URLS:

```
# check if URL exists
library(RCurl)
for (i in length(df$Website)){
  if(url.exists(df$Website[i])== FALSE){
    df$Website[i]<-NA}}
df$Website

for (i in length(df$Facebook)){
  if(url.exists(df$Facebook[i])== FALSE){
    df$Facebook[i]<-NA}}
summary(df$Facebook)
```

```
library(RCurl)
a={}
for i in df$MarketName:
  if url.exists(df[i,3]) is TRUE:
    df[i,3]=df[i,3]
  if url.exists(df[i,3]) is FALSE:
    df[i,3]=NA
  print(df[,3])
```

- Seeing the geographic distance matrix between long/lat points (R explode =o=):

```r
deg2rad <- function(deg) return(deg*pi/180)

gcd.hf <- function(long1, lat1, long2, lat2) {
  R <- 6371 # Earth mean radius [km]
  delta.long <- (long2 - long1)
  delta.lat <- (lat2 - lat1)
  a <- sin(delta.lat/2)^2 + cos(lat1) * cos(lat2) * sin(delta.long/2)^2
  c <- 2 * asin(min(1,sqrt(a)))
  d = (R * c)*1000
  return(d) }

CalcDists <- function(longlats) {
  name <- list(rownames(longlats), rownames(longlats))
  n <- nrow(longlats)
  z <- matrix(0, n, n, dimnames = name)
  for (i in 1:n) {
    for (j in 1:n) z[i, j] <- gcd.hf(long1 = deg2rad(longlats[i, 2]),
                                     lat1 = deg2rad(longlats[i, 1]), long2
                                     lat2 = deg2rad(longlats[j, 1]))
  }
  z <- as.dist(z)
  return(z)
}
# E.g.
longlats <- data.frame(long = df$x, lat = df$y)
dists <- CalcDists(longlats)
```

- Disadvantages:
  - Can be slow with large data
  - Need coding technique (not new-user friendly)
  - Visualization not good

3.2 Python (only used to find near points because it is super slow in R)

```python
from math import radians, cos, sin, asin, sqrt
import csv
import pandas
def main():
    data = pandas.read_csv("farmers-markets.csv",usecols=[0,20,21])
    sim_list = []
    for i in range(len(data.x)):
        for j in range(i,len(data.y)):
            if data.x[i]:
                temp_dis = haversine(data.x[i],data.y[i],data.x[j],data.y[j])
            if temp_dis < 0.1 and i != j:
                sim_list.append((data.FMID[i],data.FMID[j]))
    print(sim_list)
    sim_data = pandas.DataFrame(sim_list)
    sim_data.to_csv('sim.csv')
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6367 * c
    return km

if __name__ == "__main__":
    main()
```

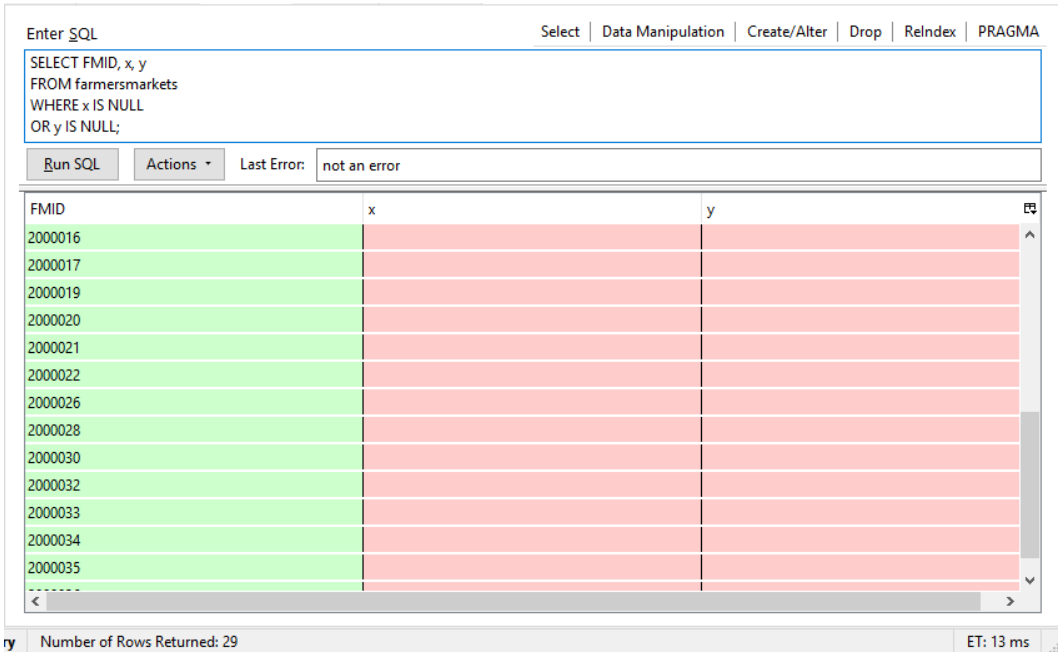
	A	B
286	1003445	1005574
287	1007693	1011997
288	1010236	1006120
289	1007771	1009450
290	1003881	1009848
291	1009910	1006324
292	1011764	1002246
293	1008429	1012771
294	1011499	1006550
295	1012325	1012329
296	1010478	1010339
297	1003769	1003768
298	1012016	1003473
299	1001794	1001796
300	1001794	1005795
301	1001796	1005795
302	1009270	1009074
303	1006893	1008931
304	1005969	1011301
305	1006422	1006423

There are 305 pairs of results.

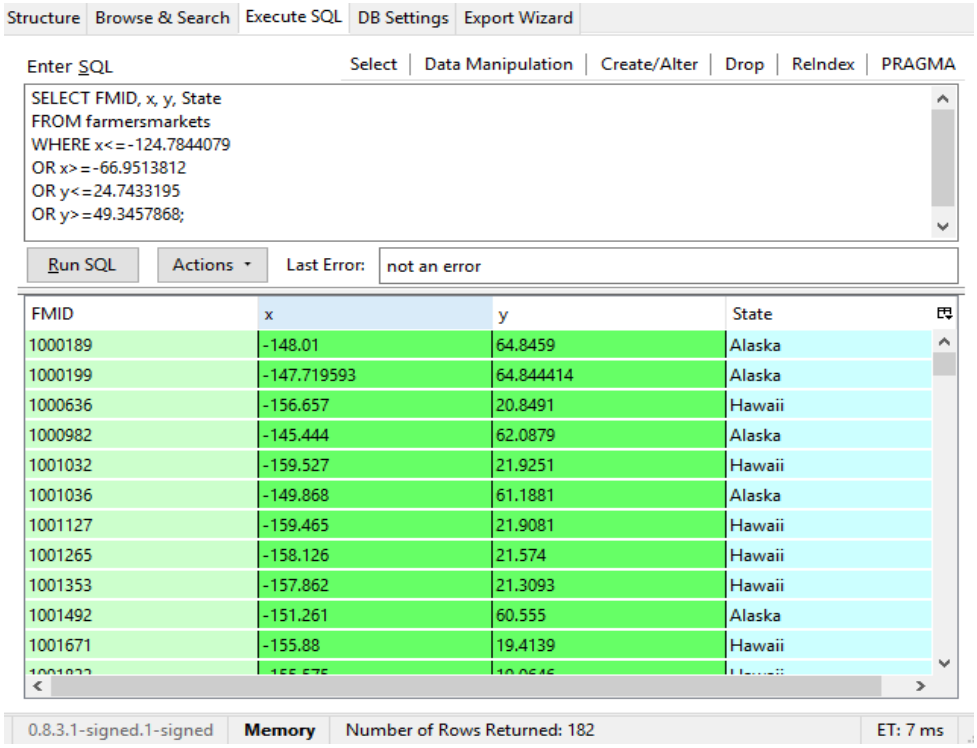
3.3 Excel (used to view the CSV file and find values such as “no" "none" "N/A" "n/a" "None" "NA" "-")

4 Integrity constraints with SQLite

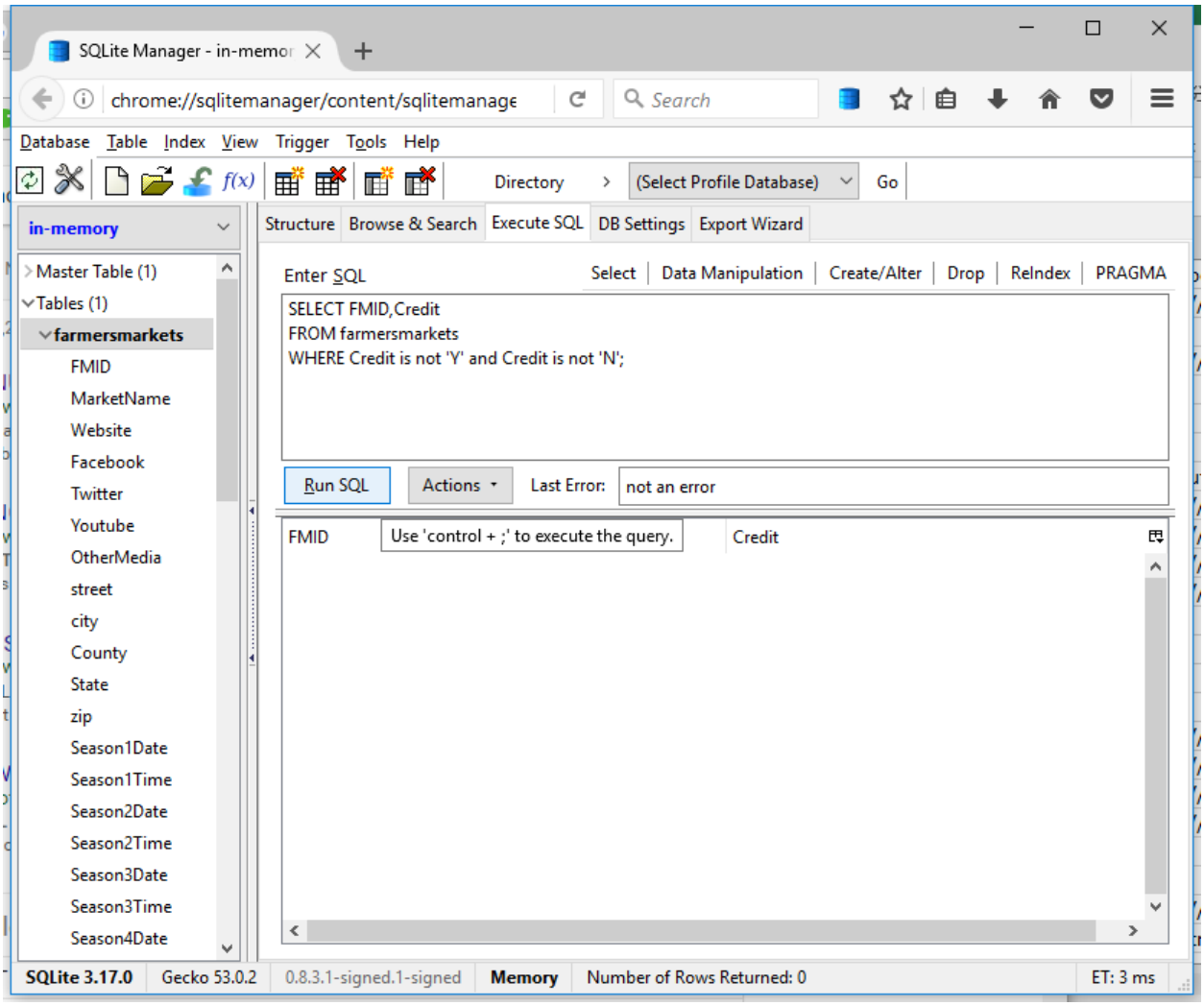
4.1 [column name] cannot be null



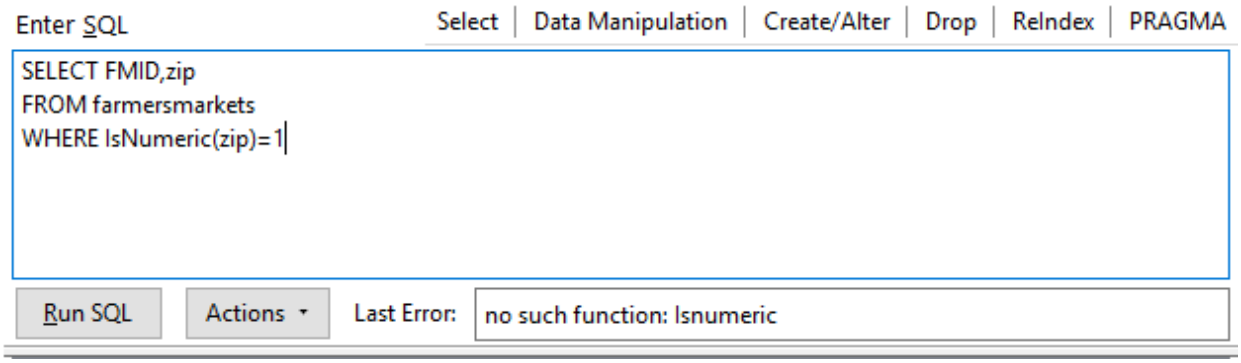
4.2 X, y within US boundaries (outliers are from Hawaii and Alaska and Virgin Islands)



4.3 Product values/payment methods cannot be values except Y or N



4.4 Zip cannot be non-numeric values (have error)



5 Workflow model

5.1 what are the key inputs and outputs of your workflow?

Step 1

Input: database

Output: summary result

Step 2

Input: original data

Output: cleaned data (half cleaned)

Step 3

Input: integrity constraints

Output: result of check of ICs