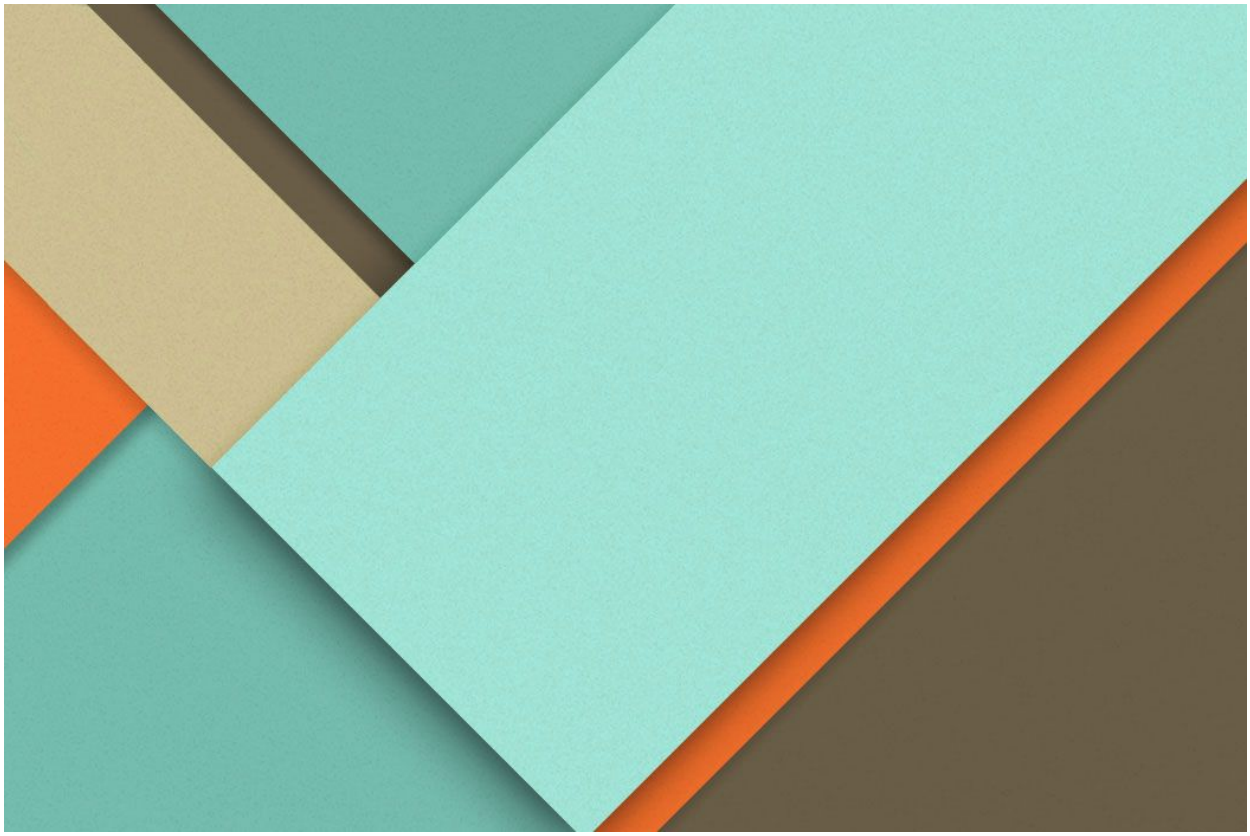


Animal Bites Analysis

03.06.2019

Arlen Navasartian

Hasan Ahmad

Overview

In this project we are going to use two different datasets.

1. Health_AnimalBites.csv from https://www.kaggle.com/rtatman/animal-bites includes information on over 9,000 animal bites which occurred near Louisville, Kentucky from 1985 to 2017 and includes information on whether the animal was quarantined after the bite occurred and whether that animal was rabid.
2. Census data from the Census API includes JSON access to all tables & variables as seen on US FactFinder.

Goal

Our goal is to see If there is a correlation between zip code of the victim and household income or population on the same zip code. By using year & zip as the primary index between both datasets, a collection can be formed for statistical analysis.

ETL Workflow

Extract

We used pandas to load and explore **Health_AnimalBites.csv** file and for **Census data** we made API requests to https://api.census.gov/data/ to get the required data. The Census Developer's Page shows how to make a JSON call to the different tables. Notice the addition of `subject?` to the subject table calls. This distinct break in the API link branches the API data sources into two, one for each respective API.

Transform

We cleaned the dataset by first removing empty rows, extracting the year from bite_date (The date the bite occurred), filtering data for years 2014 to 2017 and selecting the required columns.

	bite_year	victim_zip	species	breed	color	gender	wherebitten
0	2014	40205	DOG	PIT BULL	WHITE	NaN	BODY
1	2014	40211	DOG	PIT BULL	GRAY/WHITE	NaN	BODY
2	2014	40208	DOG	BEAGLE	TRI	FEMALE	HEAD
3	2015	40220	DOG	AAUST. TERR.	BROWN	FEMALE	BODY
4	2015	40047	DOG	LABRADOR RETRIV	GOLDEN	FEMALE	UNKNOWN

Each Census API url was broken into fixed & looped variables. By using list comprehension, the API can be manipulated to pass whatever report, or year desired for analysis into the lists [[reportsf], [reportsd], [years]].

```
# urls of detailed & subject tables with example (not used)
url = 'https://api.census.gov/data/2016/acs/acs5/subject?get=NAME,S1902_C01_001E,S2409_C05_006E&for=zip%20code%20tabulation%
urlg = 'https://api.census.gov/data/2017/acs/acs5?get=NAME,group(B01001)&for=us:1'

#url fixed variables
urlbeg = 'https://api.census.gov/data/'

urlmid = '/acs/acs5/subject?get=NAME,'
urlmidd = '/acs/acs5?get=NAME,'

urlend = '&for=zip%20code%20tabulation%20area:*'

# reports, while independently stored in list by name
#, are a single string within the call, notice the reports & reports names are in inverse order

reportsf = 'S1902_C01_001E,S2409_C05_006E,S1251_C01_001E,S2301_C01_001E,S1701_C01_001E'
reportsd = 'B01003_001E,B19013_001E,B19301_001E,B25075_001E'
reportnames = ['PopInPoverty','PopEmployed','MarriedPastYr','PercentFemaleEmployed','EstMeanIncAll']
reportnamesd = ['PropVal', 'Per Capita Income','Household Income','Population']



# The looped variable in the calls are the years
years = ['2014', '2015', '2016', '2017']
```

The result is two data frames each cast by the requested years & zips by each requested report.

	zip	PropVal	Per Capita Income	Household Income	Population	year
0	00601	3307	7229	10833	18088	2014
1	00602	10143	9048	16353	40859	2014
2	00603	11384	9888	16323	53162	2014
3	00606	1334	6385	14138	6415	2014
4	00610	7165	8197	17265	28805	2014

Load

For the final step we decided to load our dataset into non-relational database MongoDB.

First we discussed If we are going to have more normalized data schema, we better create two collections in our database, one for **Animal Bites data** and one for **Census data** but after further investigation we ended up creating one collection as Document-oriented databases such as MongoDB are designed to store denormalized data. Ideally, there should be no relationship between collections. If the same data is required in two or more documents, it must be repeated.

We chose MongoDB because documents do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

This eliminates the prerequisites of defining a fixed structure like mySQL.

In future If we get more related datasets which may not fit with our current schema we can easily load and merge without worrying about the current schema.

Below you can see our final data schema in MongoDB:

Sample document:

{"_id":"5c808b9b4151573c88675186",

"Year":"2015",

"Zip":"40220",

"species":"DOG",

"breed":"AAUST.TERR.",

"color":"BROWN",

"gender":"FEMALE",

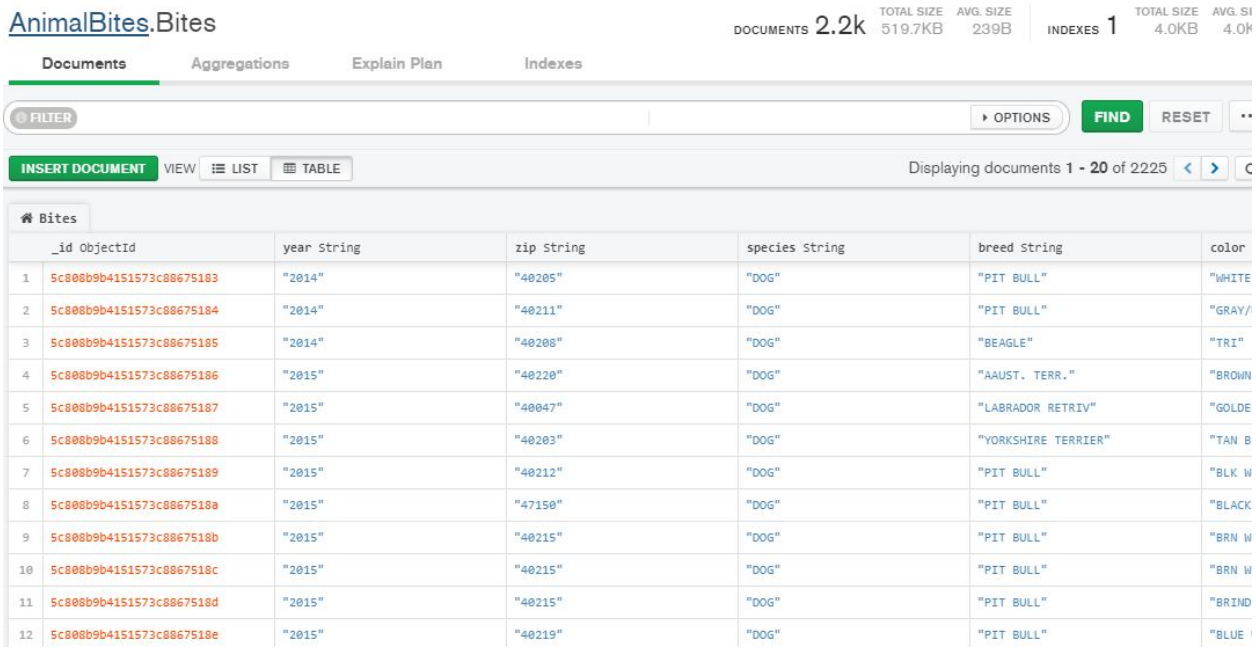
"wherebitten":"BODY",

"PopInPoverty":32474,

"PopEmployed":27282,

"Household Income":"52155",

"Population":33009}



	_id ObjectId	year String	zip String	species String	breed String	color
1	5c808b9b4151573c88675183	"2014"	"40205"	"DOG"	"PIT BULL"	"WHITE
2	5c808b9b4151573c88675184	"2014"	"40211"	"DOG"	"PIT BULL"	"GRAY/
3	5c808b9b4151573c88675185	"2014"	"40208"	"DOG"	"BEAGLE"	"TRI"
4	5c808b9b4151573c88675186	"2015"	"40220"	"DOG"	"AAUST. TERR."	"BROWN
5	5c808b9b4151573c88675187	"2015"	"40047"	"DOG"	"LABRADOR RETRIV"	"GOLDE
6	5c808b9b4151573c88675188	"2015"	"40203"	"DOG"	"YORKSHIRE TERRIER"	"TAN B
7	5c808b9b4151573c88675189	"2015"	"40212"	"DOG"	"PIT BULL"	"BLK W
8	5c808b9b4151573c8867518a	"2015"	"47150"	"DOG"	"PIT BULL"	"BLACK
9	5c808b9b4151573c8867518b	"2015"	"40215"	"DOG"	"PIT BULL"	"BRN W
10	5c808b9b4151573c8867518c	"2015"	"40215"	"DOG"	"PIT BULL"	"BRN W
11	5c808b9b4151573c8867518d	"2015"	"40215"	"DOG"	"PIT BULL"	"BRIND
12	5c808b9b4151573c8867518e	"2015"	"40219"	"DOG"	"PIT BULL"	"BLUE