Adam Stopek
IS 607 – Data Acquisition and Management

Project 4 – Comparative Study of three technologies: R, PostgreSQL, and MongoDB.

1. What is the use case addressed by your data set?

The Chicago Police Department places speed cameras in School Zones in order to protect our children and catch people violating traffic laws. However, it is commonly believed, that speed cameras are less effective than an actual police presence in reducing traffic violations and therefore they can be seen as an ineffective way of increasing safety. Many believe that installing cameras is a ploy by the local government to boost revenues without regard for actual safety. No matter what one believes, no one can argue the fact that the police force is a limited resource, whereas cameras are (more or less) unlimited. The question therefore becomes, how we enhance safety for our children in the most problematic of areas. The answer would be to find the least safe School Zones that are monitored by these cameras (the ones with the highest amount of traffic violations) and to augment the camera presence with a physical police presence. This project will do just that by finding the top 5 least safe locations each month where a camera is present in a School Zone, and to recommend an enhanced police presence there. An explanation of the data set is found in the appendix.

2. What are the advantages and disadvantages of each of the three technologies? Include both theoretical and practical considerations as appropriate.

R: Inserting into R is easy, it is a one line command. R is great for statistical analysis and visualization, but this project does not really require any of that. Importing was a single line of code. Ranking and aggregating is not too cumbersome to do in R. This was done in a few lines of code. First I aggregated the data by address and month, then I ranked it and kept only top 5. Pretty straight forward. The final dataset has the top 5 worst school zones per month. 20 rows of data with the address of the camera, the month, the number of violations, and its rank. The code and the final data can be found in the appendix.

SQL: Inserting into PostgreSQL requires quite a bit of setup and forethought, but in order to achieve the results that I want, it is a pretty straight forward query after the initial setup is complete. Because set up only needs to happen once, and querying could happen over and over (there are endless things I could ask of this data set) SQL is great choice. The query I ran took no more than a couple of minutes for me to

write. The setup required that I first insert all the data as is, even though the data is sub optimal for SQL. Once it was inserted, I separated out the metadata about each camera (coordinated, address, etc) from the violation data into two separate tables. After that all that was left to do was build the query. First step was to sum the violations by camera, and month. Once that was done, a window function was used for ranking. With that out of the way a join was performed with the metadata on the camera to get the address of the camera and only ranks of less than 5 were kept. The final dataset has the top 5 worst school zones per month. 20 rows of data with the address of the camera, the month, the number of violations, and its rank. The code and results are found in the appendix.

Mongo: Mongo requires no set up, just throw the data in using a mongoimport command, and you are ready to go. But the ease of use pretty much ends there. To get the desired results I used an aggregate pipeline for each month. The process is cumbersome and a bit unintuitive. Each pipeline went as follows. First project which fields we will want to keep and in the process create a violation month field. Then match only those with the month equal to the specific month we want. Then aggregate violations per address for that month. Then sort descending and limit 5. Finally write to a new collection that month's data. Do this again for each month. It may have been possible that a map reduce may have been easier, yet would have required some extra coding. The final result is 4 collections, 1 for each month with the top 5 worst zones (address, month, number of violations) sort by worst descending.

3. Which technology would you recommend for your use case?

Mongo is great for unstructured data, but my data is structured, and the structure will not change. R is great for statistical manipulations and visualizations, but with the type of analysis I am doing, there is no advantage to doing it in R. SQL requires a bit of set up, but once set up, it is pretty easy to query. On my dataset of 10k rows all methods were equally fast in terms of processing time. In the end I am most comfortable writing SQL queries and even though it has the most initial setup, it is much faster for me to write the SQL query and therefore I would choose SQL as the database of choice in the case.

Appendix:

A bit about the dataset:

The data set used has been obtained from the City of Chicago's website https://data.cityofchicago.org/Transportation/Speed-Camera-Violations/hhkd-xvj4. The dataset reflects the daily volume of violations that have occurred in Children's Safety Zones for each camera. The data reflects violations that occurred from July 1st, 2014 until present, minus the most recent 14 days (at the

time of analysis this was October 23rd, 2014). The reported violations are those that have been collected by the camera and radar system and reviewed by two separate City contractors. The fields in the data are: Address, Camera ID, Violation Date, Violations, X Coordinate, Y Coordinate, Latitude, Longitude, and Location. Notice that for each Camera ID there is only a single distinct value for each of the following fields: Address, X Coordinate, Y Coordinate, Latitude, Longitude, and Location. There were 9,986 rows of data when I downloaded it.

R Code:

```
library(plyr)
#Read data into R
Speed_Camera_Violations <- read.csv("C:/Users/adams/Downloads/Speed_Camera_Violations.csv",
na.strings="", stringsAsFactors=FALSE)

# Find the Month of the Violation
Speed_Camera_Violations$MONTH <- months.Date( as.Date(
Speed_Camera_Violations$VIOLATION.DATE , format = "%m/%d/%Y"))

# Function to calculate the locations with the most violations per month (top X)

top_x_per_month <- function(data, x) {
  # aggregate the data by month and camera and calculate VPM (violations per month)
  violations_per_month <- ddply(data, .(ADDRESS,MONTH), summarise,VPM = sum(VIOLATIONS, na.rm =
TRUE))
  # prior to ranking, order by month
  violations_per_month <- violations_per_month[order(violations_per_month$MONTH),]
  # rank the cameras by worst zones descending
  violations_per_month$RANK <- unlist(with(violations_per_month, tapply(VPM, MONTH, function(x)
rank(-x, ties.method= "first"))))
  # subset the data keeping top X zones
  worst_zones <- violations_per_month[ which(violations_per_month$RANK<=x), ]
  # organize the data
  worst_zones<- worst_zones[order(worst_zones$MONTH, worst_zones$RANK),]
  # remove previous row numbers
  row.names(worst_zones) <- NULL
  # return the data
  worst_zones
}

# call the function and save the data
x <- top_x_per_month(Speed_Camera_Violations, 5)
save(x, file='worst_zones.RData')
```

```
                ADDRESS      MONTH  VPM RANK
1           445 W 127TH    August 5419    1
2          2900 W OGDEN    August 5317    2
3  10318 S INDIANAPOLIS    August 5170    3
4       536 E MORGAN DR    August 4533    4
5    1142 W IRVING PARK    August 3923    5
6       536 E MORGAN DR      July 7108    1
7           445 W 127TH      July 5387    2
8          2900 W OGDEN      July 5357    3
9  10318 S INDIANAPOLIS      July 5094    4
10   1142 W IRVING PARK      July 4094    5
11 10318 S INDIANAPOLIS   October 3399    1
12          445 W 127TH   October 3176    2
13         2900 W OGDEN   October 3089    3
14      536 E MORGAN DR   October 2878    4
15   1142 W IRVING PARK   October 2725    5
16 10318 S INDIANAPOLIS September 5053    1
17          445 W 127TH September 4484    2
18         2900 W OGDEN September 4442    3
19      536 E MORGAN DR September 3799    4
20   1142 W IRVING PARK September 3526    5
> |
```

SQL Code:

```
-- First create raw dump table
CREATE TABLE raw_dump
(
 address        VARCHAR,
 camera_id      VARCHAR,
 violation_date  VARCHAR,
 violations     INT,
 x_coordinate    DOUBLE precision,
 y_coordinate    DOUBLE precision,
 latitude       DOUBLE precision,
 longitude      DOUBLE precision,
 lat_long       VARCHAR
);

-- Insert data into the table using copy command
COPY raw_dump
FROM 'C:/Program Files/PostgreSQL/9.0/Speed_Camera_Violations.csv' DELIMITER ',' CSV HEADER;

-- Create a table for the cameras and insert data into it from raw dump table
CREATE TABLE cameras
(
 camera_id     VARCHAR,
 address       VARCHAR,
 x_coordinate   DOUBLE precision,
 y_coordinate   DOUBLE precision,
 latitude      DOUBLE precision,
 longitude     DOUBLE precision,
 lat_long      VARCHAR
);
```

```
INSERT INTO cameras
SELECT DISTINCT camera_id,
      address,
      x_coordinate,
      y_coordinate,
      latitude,
      longitude,
      lat_long
FROM raw_dump;

-- Create a table for violations and insert data into it from raw dump table
CREATE TABLE violations
(
  camera_id       VARCHAR,
  violation_date   DATE,
  violations      INT
);

INSERT INTO violations
SELECT camera_id,
      to_date(violation_date,'MM/DD/YYYY') AS violation_date,
      violations
FROM raw_dump;

-- Query to get desired results
SELECT b.address,
      a.violation_month,
      a.violations,
      a.rank
FROM (SELECT camera_id,
           violation_month,
           violations,
           RANK() OVER (PARTITION BY violation_month ORDER BY violations DESC)
     FROM (SELECT camera_id,
                to_char(violation_date,'Month') AS violation_month,
                SUM(violations) AS violations
           FROM violations
           GROUP BY 1,
                 2) inner_query) a
  INNER JOIN cameras b
        ON a.camera_id = b.camera_id
        AND a.rank <= 5;
```

	address character varying	violation_month text	violations bigint	rank bigint
1	445 W 127TH	August	5419	1
2	2900 W OGDEN	August	5317	2
3	10318 S INDIANAPOLIS	August	5170	3
4	536 E MORGAN DR	August	4533	4
5	1142 W IRVING PARK	August	3923	5
6	536 E MORGAN DR	July	7108	1
7	445 W 127TH	July	5387	2
8	2900 W OGDEN	July	5357	3
9	10318 S INDIANAPOLIS	July	5094	4
10	1142 W IRVING PARK	July	4094	5
11	10318 S INDIANAPOLIS	October	3399	1
12	445 W 127TH	October	3176	2
13	2900 W OGDEN	October	3089	3
14	536 E MORGAN DR	October	2878	4
15	1142 W IRVING PARK	October	2725	5
16	10318 S INDIANAPOLIS	September	5053	1
17	445 W 127TH	September	4484	2
18	2900 W OGDEN	September	4442	3
19	536 E MORGAN DR	September	3799	4
20	1142 W IRVING PARK	September	3526	5

Mongo Code:

.\mongoimport --db project9 --collection data --type csv --headerline --file C:/Users/adams/Downloads/Speed_Camera_Violations.csv --upsert

I then need to convert the date ("yyyy-mm-dd") from a string to a date

```
db.data.find().forEach(function(doc) {
   doc.violation_date=new Date(doc.violation_date);
   db.data.save(doc);
   });
```

Once I do that, I can run an aggregate for each month's top 5 and write them to a new collection, 1 for each month

```
db.data.aggregate(
        { $project: {address: "$address", violation_month: {$month: "$violation_date"}, violations: "$violations"}},
        { $match: {violation_month: 7}},
        { $group: {_id: {address: "$address", violation_month: "July"}, vpm: {$sum: "$violations"}}},
        { $sort: { vpm: -1 } },
        { $limit: 5},
        { $out: "July"}
);
db.data.aggregate(
        { $project: {address: "$address", violation_month: {$month: "$violation_date"}, violations: "$violations"}},
        { $match: {violation_month: 8}},
        { $group: {_id: {address: "$address", violation_month: "August"}, vpm: {$sum: "$violations"}}},
        { $sort: { vpm: -1 } },
        { $limit: 5},
        { $out: "August"}
```

```
);
db.data.aggregate(
	{ $project: {address: "$address", violation_month: {$month: "$violation_date"}, violations:
"$violations"}},
	{ $match: {violation_month: 9}},
	{ $group: {_id: {address: "$address", violation_month: "September"}, vpm: {$sum:
"$violations"}}},
	{ $sort: { vpm: -1 } },
	{ $limit: 5},
	{ $out: "September"}
);
db.data.aggregate(
	{ $project: {address: "$address", violation_month: {$month: "$violation_date"}, violations:
"$violations"}},
	{ $match: {violation_month: 10}},
	{ $group: {_id: {address: "$address", violation_month: "October"}, vpm: {$sum: "$violations"}}},
	{ $sort: { vpm: -1 } },
	{ $limit: 5},
	{ $out: "October"}
);
```

```
> db.July.find()
{ "_id" : { "address" : "536 E MORGAN DR", "violation_month" : "July" }, "vpm" : 7108 }
{ "_id" : { "address" : "445 W 127TH", "violation_month" : "July" }, "vpm" : 5387 }
{ "_id" : { "address" : "2900 W OGDEN", "violation_month" : "July" }, "vpm" : 5357 }
{ "_id" : { "address" : "10318 S INDIANAPOLIS", "violation_month" : "July" }, "vpm" : 5094 }
{ "_id" : { "address" : "1142 W IRVING PARK", "violation_month" : "July" }, "vpm" : 4094 }
> db.August.find()
{ "_id" : { "address" : "445 W 127TH", "violation_month" : "August" }, "vpm" : 5419 }
{ "_id" : { "address" : "2900 W OGDEN", "violation_month" : "August" }, "vpm" : 5317 }
{ "_id" : { "address" : "10318 S INDIANAPOLIS", "violation_month" : "August" }, "vpm" : 5170 }
{ "_id" : { "address" : "536 E MORGAN DR", "violation_month" : "August" }, "vpm" : 4533 }
{ "_id" : { "address" : "1142 W IRVING PARK", "violation_month" : "August" }, "vpm" : 3923 }
> db.September.find()
{ "_id" : { "address" : "10318 S INDIANAPOLIS", "violation_month" : "September" }, "vpm" : 5053 }
{ "_id" : { "address" : "445 W 127TH", "violation_month" : "September" }, "vpm" : 4484 }
{ "_id" : { "address" : "2900 W OGDEN", "violation_month" : "September" }, "vpm" : 4442 }
{ "_id" : { "address" : "536 E MORGAN DR", "violation_month" : "September" }, "vpm" : 3799 }
{ "_id" : { "address" : "1142 W IRVING PARK", "violation_month" : "September" }, "vpm" : 3526 }
> db.October.find()
{ "_id" : { "address" : "10318 S INDIANAPOLIS", "violation_month" : "October" }, "vpm" : 3399 }
{ "_id" : { "address" : "445 W 127TH", "violation_month" : "October" }, "vpm" : 3176 }
{ "_id" : { "address" : "2900 W OGDEN", "violation_month" : "October" }, "vpm" : 3089 }
{ "_id" : { "address" : "536 E MORGAN DR", "violation_month" : "October" }, "vpm" : 2878 }
{ "_id" : { "address" : "1142 W IRVING PARK", "violation_month" : "October" }, "vpm" : 2725 }
>
```