M3 Assignment - Setting Parameters And Extracting Data

Gino Murin

2022-05-22

## Defining Parameters

In R Markdown, we are able to define parameters within the header of our report. Defining parameters is extremely powerful when working with large, dynamic data sets, as the parameters can be modified to suit the users needs based on a variety of use cases. Predefined parameters will be used later in this report to load and manipulate data in MySQL.

To describe the utility of parameters, we can use our current data set as an example. We’re looking specifically at data from New York City between the years of 2016-2020. If the need arose within our organization to expand the results of this analysis to compare crash data from other cities, this script could be modified through the use of parameters to fit that use case as opposed to writing an entirely new script. In larger organizations, these modifications could be entirely automated and require little to no human intervention.

I’ve identified the following parameters that could prove useful in this particular use case:

print(params)

## $year\_from  
## [1] 2016  
##   
## $year\_to  
## [1] 2020  
##   
## $region  
## [1] "United States"  
##   
## $state  
## [1] "New York"  
##   
## $city  
## [1] "New York"  
##   
## $data  
## [1] "boro"  
##   
## $file  
## [1] "Z:\\SNHU Coursework\\DAT-375\\Week 3\\M3 Assignment\\boro.cnf"

## Loading the dataset

By passing the following commands into RMariaDB, we establish a connection with our mysql database, import our dataset from csv file into mysql, and write the data into a table within our database. Note the use of “params $ file” and “params $ data” which passes Z:\SNHU Coursework\DAT-375\Week 3\M3 Assignment\boro.cnf and boro to the compiler respectively.

# Giving R the correct directory for the mysql config data  
rmariadb.settingsfile<-params$file  
  
# Connecting to the tweet db from mysql and assigning to an object  
rmariadb.db<-params$data  
boroDb<-dbConnect(RMariaDB::MariaDB(),default.file=rmariadb.settingsfile,  
 group=rmariadb.db)  
  
#set the appropriate working directory to find our csv file  
#setwd(params$file)  
  
#read the csv data into a variable to be passed into mysql  
#boroData <- read.csv(file="boros.csv",  
# header=TRUE, sep=",")  
  
#write the csv data into mysql database  
#dbWriteTable(boroDb, value = boroData, row.names = FALSE,   
# name = "boro", append=TRUE)

## Determining NYC Boroughs With Highest Accidents

As part of the **Data Understanding Phase** of our data analysis pipeline, we need to collect and make sense of the data we’ve been provided. Tools such as MySQL Workbench and Microsoft Excel are useful for quickly and efficiently sorting and visualizing data.

### Using MySQL Workbench to Query the Dataset

Using MySQL Workbench gives us the ability to import/export data quickly and efficiently through the GUI in addition to performing basic queries.

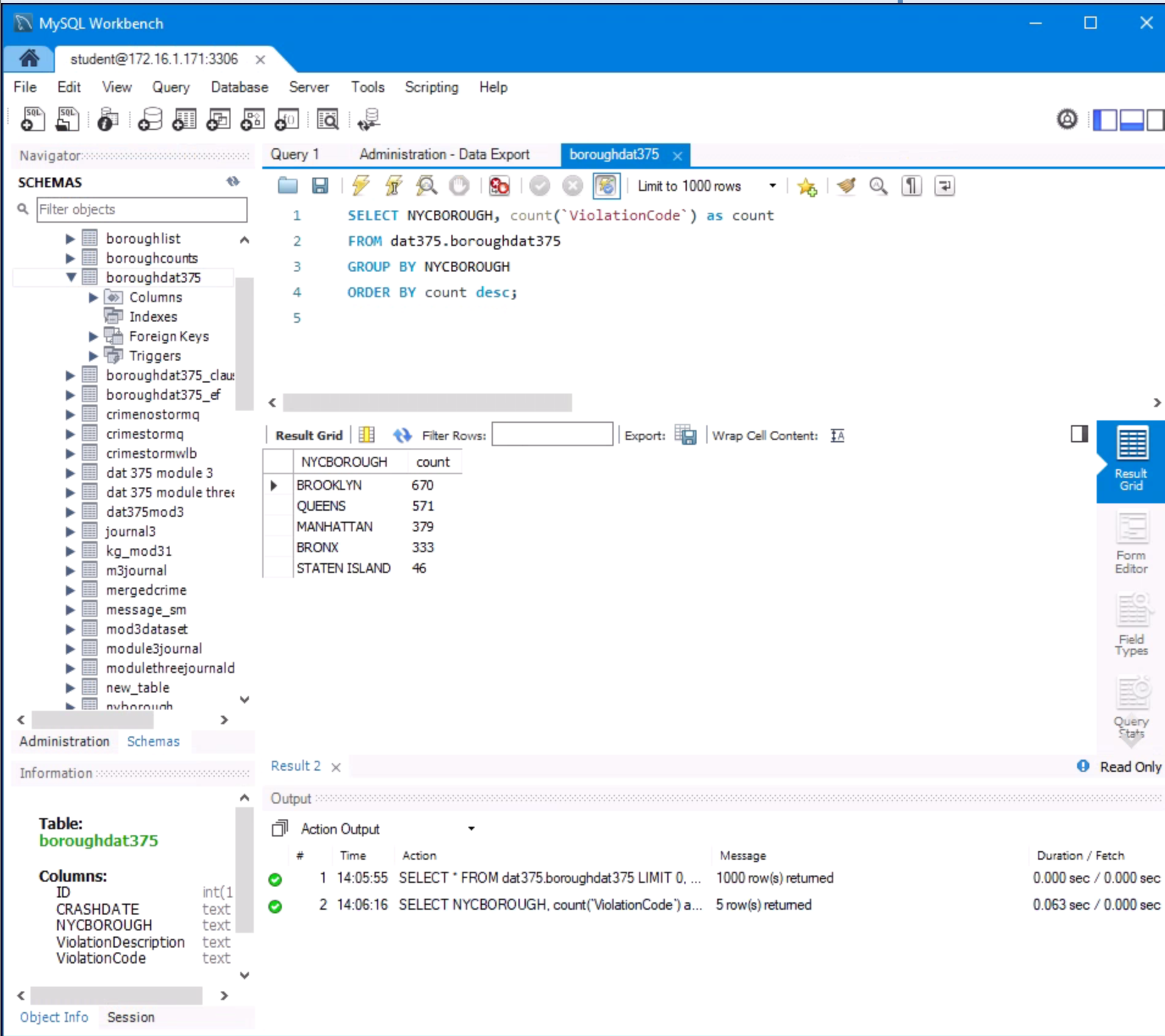


Figure 1: MySQL Workbench Query of Borough Dataset

### Using Microsoft Excel to Verify Findings

After analyzing the dataset and gaining a preliminary understanding, we can double check our work by exporting the data to excel and verifying that our queries are accurate by Excel’s built in filtering tools. Below is a pivot table that confirms our findings from the MySQL report:

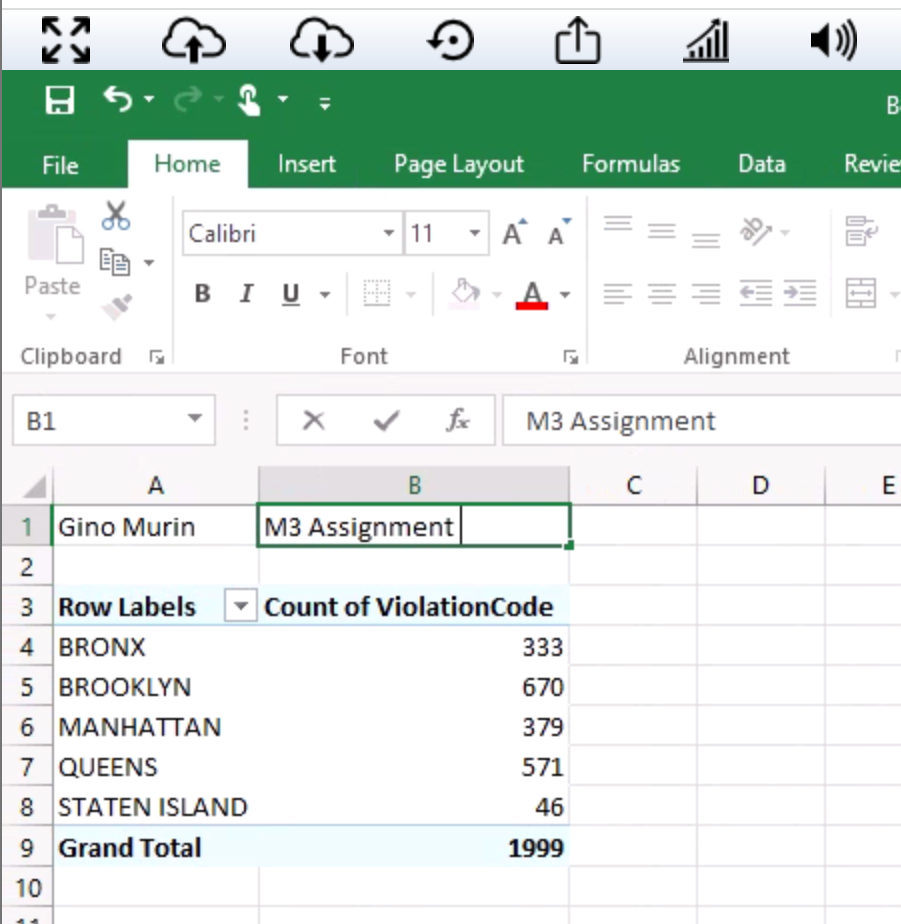


Figure 1: Excel Pivot Table Data

## Extracting the Data

In previous steps, we defined custom parameters, imported our data set, and made sense of our data set using data analysis tools. Our findings were verified and we are now ready to extract the data. We’ve been asked to find the 3 boroughs in the data set with the highest accident rate. Through the use of the RMariaDB library, we can create a query object, pass it to MySQL, and store the resulting data for later use.

#query to determine the number of tweets per user  
query<-paste("SELECT NYCBOROUGH, COUNT('ViolationCode')AS num\_crashes  
FROM boro  
GROUP BY NYCBOROUGH  
ORDER BY num\_crashes desc  
LIMIT 3;" , sep=",")  
  
#pass the query into sql and store result as a dataframe  
rs = dbSendQuery(boroDb, query)  
  
result <- dbFetch(rs)  
  
boros <- result  
print(result)

## NYCBOROUGH num\_crashes  
## 1 BROOKLYN 670  
## 2 QUEENS 571  
## 3 MANHATTAN 379

dbClearResult(rs)  
dbDisconnect(boroDb)