Assignment 2: NoSQL Data Arch with MongoDB and Neo4j

Report by Aaron Cashman and Brian Coveney

November 2017

Table of Contents

Setup: 2
MongoDB Queries: 6
PyMongo Queries 10
Neo4j Doc Manager 11
Neo4j Queries 12

Setup:

```
Select Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\brian> cd "C:\Users\brian\Desktop\mongo setup"
PS C:\Users\brian\Desktop\mongo setup> dir


    Directory: C:\Users\brian\Desktop\mongo setup


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       11/24/2017   8:37 AM           1198 1.create_nodes.bat
------       11/21/2017   8:16 AM            319 2.setup_cluster.bat
-a----       11/22/2017   9:53 AM            772 2.setup_cluster.js
-a----       11/22/2017  10:29 AM            349 3.insert_collection.bat
-a----       11/21/2017   8:16 AM        8728561 3.restaurants.json
-a----       11/22/2017  10:25 AM            306 4.shard_collection.bat
-a----       10/12/2017   9:16 AM           2130 4.shard_collection.js


PS C:\Users\brian\Desktop\mongo setup> .\1.create_nodes.bat

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>REM #-----------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #                                               #

C:\Users\brian\Desktop\mongo setup>REM # 0. Set myName to the name of the computer       #

C:\Users\brian\Desktop\mongo setup>REM #                                               #

C:\Users\brian\Desktop\mongo setup>REM #-----------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>SET myName=Desktop-bedroom

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>REM #-----------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #                                               #

C:\Users\brian\Desktop\mongo setup>REM # 1. Start the config server database instances   #

C:\Users\brian\Desktop\mongo setup>REM #                                               #

C:\Users\brian\Desktop\mongo setup>REM #-----------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>REM # 1.1. Create directories

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>mkdir \Users\brian\Desktop\NoSQL\Environment\dublin0

C:\Users\brian\Desktop\mongo setup>mkdir \Users\brian\Desktop\NoSQL\Environment\dublin1

C:\Users\brian\Desktop\mongo setup>mkdir \Users\brian\Desktop\NoSQL\Environment\dublin2

C:\Users\brian\Desktop\mongo setup>pause
Press any key to continue . . .
```

Fig1. Displays the initial set up

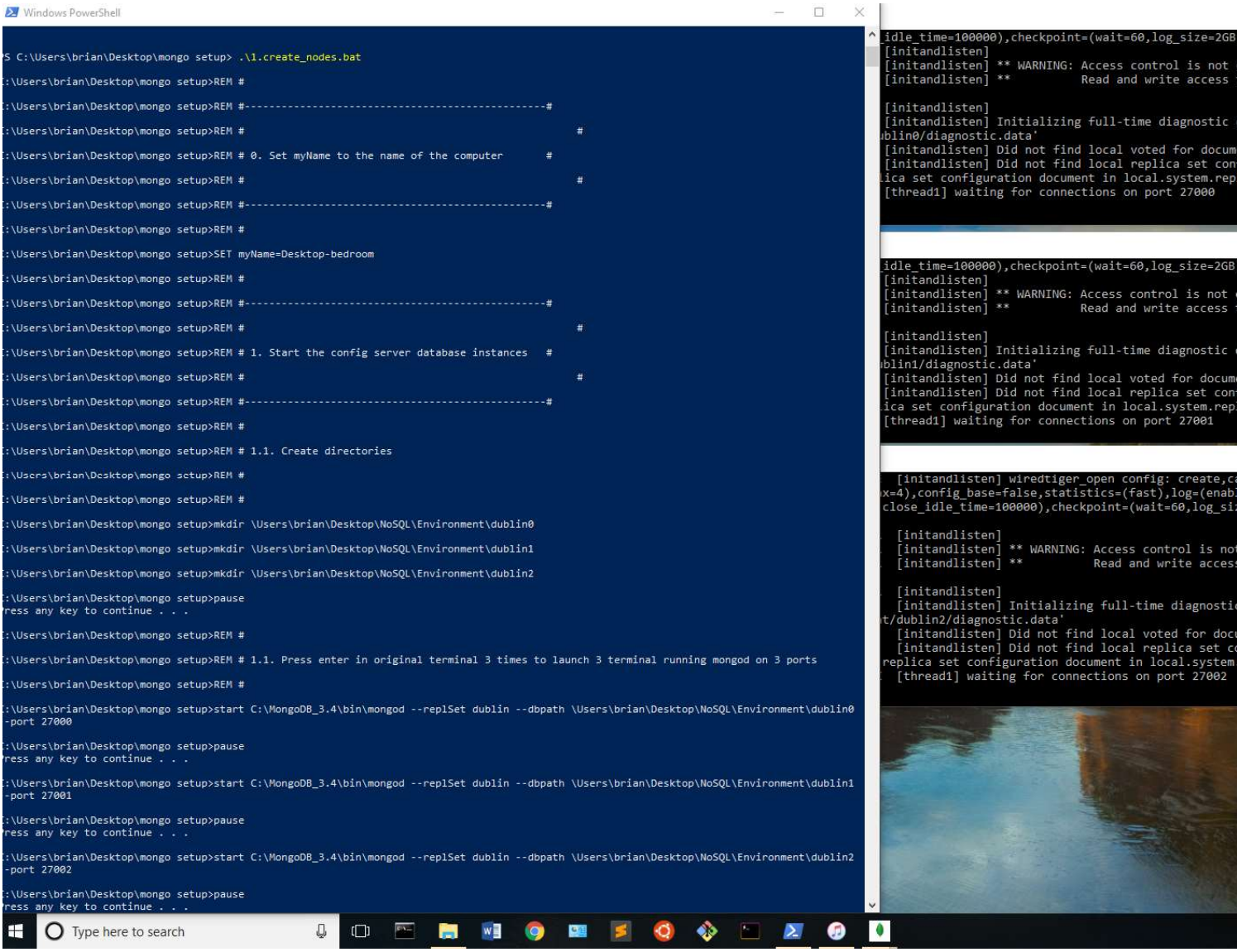

Fig1b. Displays the 3 mongo processes started on ports 27000, 27001 and 27002

```
Select Windows PowerShell

PS C:\Users\brian\Desktop\mongo setup> .\2.setup_cluster.bat
PS C:\Users\brian\Desktop\mongo setup>
C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>REM #-------------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #                                                 #

C:\Users\brian\Desktop\mongo setup>REM # 4. Set chunksize, Create Replica Sets and Shards  #

C:\Users\brian\Desktop\mongo setup>REM #                                                 #

C:\Users\brian\Desktop\mongo setup>REM #-------------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>C:\MongoDB_3.4\bin\mongo.exe --port 27000 --shell 2.setup_cluster.js
MongoDB shell version v3.4.10
connecting to: mongodb://127.0.0.1:27000/
MongoDB server version: 3.4.10
type "help" for help
connecting to: mongodb://Desktop-bedroom.local:27000/test
MongoDB server version: 3.4.10
Dublin Replica Set Created!
Dublin Replica Set Up!

C:\Users\brian\Desktop\mongo setup>REM #
PS C:\Users\brian\Desktop\mongo setup>
```

Fig2. Displays the setup of our cluster

```
Select Windows PowerShell

C:\Users\brian\Desktop\mongo setup>REM #
PS C:\Users\brian\Desktop\mongo setup> .\3.insert_collection.bat

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>REM #-------------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #                                                 #

C:\Users\brian\Desktop\mongo setup>REM # 5. Insert collection from File                 #

C:\Users\brian\Desktop\mongo setup>REM #                                                 #

C:\Users\brian\Desktop\mongo setup>REM #-------------------------------------------------#

C:\Users\brian\Desktop\mongo setup>REM #

C:\Users\brian\Desktop\mongo setup>C:\MongoDB_3.4\bin\mongoimport.exe --port 27000 --db test --collection restaur
ants --file 3.restaurants.json
2017-11-26T21:03:28.714+0000    connected to: localhost:27000
2017-11-26T21:03:30.476+0000    imported 25359 documents

C:\Users\brian\Desktop\mongo setup>REM #
PS C:\Users\brian\Desktop\mongo setup>
```

Fig3. Displays insertion of the restaurant collection

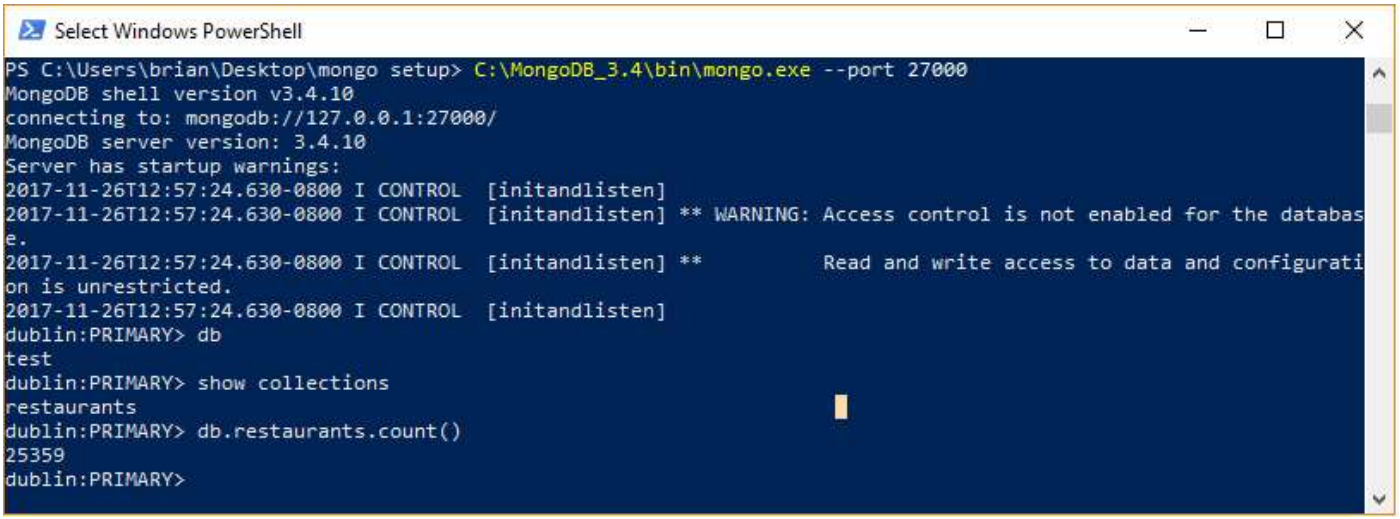

Fig4. Initial connection to our database

MongoDB Queries:

```
Select Windows PowerShell
dublin:PRIMARY> //
dublin:PRIMARY> var total_num_rest = 0;
dublin:PRIMARY> var cuisine_name = "";
dublin:PRIMARY> var ratio_cuisine = 0;
dublin:PRIMARY> var borough_name = "";
dublin:PRIMARY> var count_rest_boro = 0;
dublin:PRIMARY> var count_rest_with_cuisin = 0;
dublin:PRIMARY> var ratio_borough = 0;
dublin:PRIMARY> //
dublin:PRIMARY> //*********************************
dublin:PRIMARY> // 1. Most popular cuisine
dublin:PRIMARY> //*********************************
dublin:PRIMARY> //
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> // 1.1. Grouping by 'restaurant_id' and get a total count of restaurants
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> var aggRest = db.restaurants.aggregate([
...   { "$group" : { "_id" : "restaurant_id",
...                  "count_restaurants" : { "$sum" : 1 } } },
... ])
dublin:PRIMARY> //
dublin:PRIMARY> total_num_rest = aggRest.toArray()[0]["count_restaurants"]
25359
dublin:PRIMARY> //
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> // 1.2. Group the restaurants by their cuisine style, counting how many are there per group
dublin:PRIMARY> //    - Sort the documents by decreasing order
dublin:PRIMARY> //    - Filter the documents so as to get just the first document
dublin:PRIMARY> //    - Use var 'total_rest' in percentage calculation
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> var aggRest2 = db.restaurants.aggregate([
...   { "$group" : { "_id" : "$cuisine",
...                  "total" : { "$sum" : 1 } } },
...   { "$sort" : { "total" : -1 } },
...   { "$limit" : 1 },
...   { "$project" : { "count":1, "percentage":{ "$multiply":[{ "$divide":[100, total_num_rest]}, "$total"]}}}
... ])
dublin:PRIMARY> //
dublin:PRIMARY> cuisine_name = aggRest2.toArray()[0]["_id"]
American
dublin:PRIMARY> ratio_cuisine = aggRest2.toArray()[0]["percentage"]
24.381876256950193
dublin:PRIMARY> print("1. The kind of cuisine with more restaurants in the city is", cuisine_name, "(with a", ratio_cuisine, "percentage of restaurants of the city)")

1. The kind of cuisine with more restaurants in the city is American  (with a 24.381876256950193 percentage of restaurants of the city)
dublin:PRIMARY>
```

Fig5. Displays the MongoDB query No.1, with the following result:

*1. The kind of cuisine with more restaurants in the city is **American** (with **a 24.381876256950193** percentage of restaurants of the city)*

```
Select Windows PowerShell
dublin:PRIMARY> //*********************************
dublin:PRIMARY> //
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> // 2.1. Group the restaurants by their borough, with acount of each
dublin:PRIMARY> //    - Sort the documents by increasing order
dublin:PRIMARY> //    - skip the first borough named 'missing'
dublin:PRIMARY> //    - Filter the documents so as to get just the first relevant document
dublin:PRIMARY> //    - Use var 'total_rest' in percentage calculation
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> //
dublin:PRIMARY> var aggBoro1 = db.restaurants.aggregate([
...   { "$group" : { "_id" : "$borough",
...                  "count" : { "$sum" : 1 } } },
...   { "$sort" : { "count" : 1 } },
...   { "$skip" : 1 },
...   { "$limit" : 1 },
... ])
dublin:PRIMARY> //
dublin:PRIMARY> count_rest_borough = aggBoro1.toArray()[0]["count"];
969
dublin:PRIMARY> //
dublin:PRIMARY> //
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> // 2.2. Group the restaurants by their borough, with a count of the favoirite cuisine (returned from Q1)
dublin:PRIMARY> //    - Sort the documents by increasing order
dublin:PRIMARY> //    - skip the first borough, as it's an outlier named 'Missing'
dublin:PRIMARY> //    - Filter the documents so as to get just the first relevant document
dublin:PRIMARY> //    - Use var 'ratio_borough' as our percentage calculation
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> var aggBoro2 = db.restaurants.aggregate([
...   { "$project" : { "_id" : 0,
...                    "Borough" : "$borough",
...                    "Cuisine" : {"$cond" : [ {"$eq" : ["$cuisine", cuisine_name ] }, 1, 0]} } },
...   { "$group" : { "_id" : "$Borough", "count" : { "$sum" : "$Cuisine" } } },
...   { "$sort" : { "count" : 1 } },
...   { "$skip" : 1 },
...   { "$limit" : 1 },
...   { "$project": {"count":1,"percentage":{"$multiply":[{"$divide":[100, count_rest_borough]},"$count"]}}}
... ])
dublin:PRIMARY> //
dublin:PRIMARY> borough_name = aggBoro2.toArray()[0]["_id"];
Staten Island
dublin:PRIMARY> //
dublin:PRIMARY> ratio_borough = aggBoro2.toArray()[0]["percentage"];
25.180598555211557
dublin:PRIMARY> //
dublin:PRIMARY> //
dublin:PRIMARY> print ("2. The borough with smaller ratio of restaurants of this kind of cuisine is", borough_name, "(with a", ratio_borough, "percentage
 of restaurants of this kind)")
2. The borough with smaller ratio of restaurants of this kind of cuisine is Staten Island (with a 25.180598555211557 percentage of restaurants of this ki
nd)
dublin:PRIMARY>
```

Fig6. Displays the MongoDB query No.2, with the following result:

*2. The borough with smaller ratio of restaurants of this kind of cuisine is **Staten Island** (with a **25.180598555211557** percentage of restaurants of this kind)*

Select Windows PowerShell

```
dublin:PRIMARY> //*********************************
dublin:PRIMARY> //  3. Zip Code
dublin:PRIMARY> //*********************************
dublin:PRIMARY> //
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> // 3.1. Get the biggest five zipcodes of the borough (Staten Island)
dublin:PRIMARY> //--------------------------------
dublin:PRIMARY> //
dublin:PRIMARY> db.restaurants.aggregate([
...   { "$match" : {  "borough" : borough_name}},
...   { "$group" : { "_id" :
...                  "$address.zipcode",
...                  "total" : { "$sum" : 1 } } },
...   { "$sort" : { "total" : -1 } },
...   { "$limit" : 5},
... ])
{ "_id" : "10314", "total" : 189 }
{ "_id" : "10306", "total" : 115 }
{ "_id" : "10301", "total" : 102 }
{ "_id" : "10305", "total" : 96 }
{ "_id" : "10312", "total" : 79 }
dublin:PRIMARY> var zipcode1 = "10314" // 29.100%
dublin:PRIMARY> var zipcode2 = "10306" // 20.869%
dublin:PRIMARY> var zipcode3 = "10301" // 30.392%
dublin:PRIMARY> var zipcode4 = "10305" // 19.791%
dublin:PRIMARY> var zipcode5 = "10312" // 27.848%
dublin:PRIMARY> var numRest = 0;
dublin:PRIMARY> var ratioZip5 = 0;
dublin:PRIMARY> //
dublin:PRIMARY> // change the zip code variables  to one of the above, to get that zipcode's ratio
dublin:PRIMARY> var aggBoroZip = db.restaurants.aggregate([
...   { "$match" : {  "borough" : borough_name, "address.zipcode" : zipcode4 } },
...   { "$group" : { "_id" : "$address.zipcode", "total" : { "$sum" : 1 } } },
...   { "$sort" : { "total" : -1 } },
...   { "$limit" : 5},
... ])
dublin:PRIMARY> numRest = aggBoroZip.toArray()[0]["total"];
96
dublin:PRIMARY> //
dublin:PRIMARY> var aggBoroZip2 = db.restaurants.aggregate([
...   { "$match" : {  "borough" : borough_name, "cuisine" : cuisine_name, "address.zipcode" : zipcode4 } },
...   { "$group" : { "_id" : "$address.zipcode", "total" : { "$sum" : 1 } } },
...   { "$sort" : { "total" : -1 } },
...   { "$project": {"count":1,"percentage":{"$multiply":[{"$divide":[100, numRest]},"$total"]}}}
... ])
dublin:PRIMARY> //
dublin:PRIMARY> //
dublin:PRIMARY> ratioZip = aggBoroZip2.toArray()[0]["percentage"];
19.79166666666668
dublin:PRIMARY> //
dublin:PRIMARY> //
dublin:PRIMARY> print ("3. The zipcode of the borough with smaller ratio of restaurants of this kind of cuisine is zipcode =", zipcode4, "(with a", ratioZip, "p
estaurants of this kind)")
3. The zipcode of the borough with smaller ratio of restaurants of this kind of cuisine is zipcode = 10305 (with a 19.791666666666668 percentage of restaurants
```

Fig7. Displays the MongoDB query No.3, with the following result:

3. The zipcode of the borough with smaller ratio of restaurants of this kind of cuisine is zipcode = ***10305*** *(with a* ***19.791666666666668*** *percentage of restaurants of this kind)*

```
Select Windows PowerShell
dublin:PRIMARY> //********************************
dublin:PRIMARY> //  4. Reviews
dublin:PRIMARY> //********************************/
dublin:PRIMARY> //
dublin:PRIMARY> db.restaurants.aggregate([
...   { "$match" : {
...       "borough" : borough_name,
...       "cuisine" : cuisine_name,
...       "address.zipcode" : zipcode4 } },
...   { "$project": {
...       "borough" : 1,
...       "cuisine" : 1,
...       "address.zipcode" : 1,
...       "grades.grade": 1,
...       "grades.score": 1,
...       "name" : 1 } },
...   { "$group" : {
...       "_id" : {
...        "Borough" : "$borough",
...        "Cuisine" : "$cuisine",
...        "Zip" : "$address.zipcode",
...        "Grades" : "$grades.grade",
...        "score" : "$grades.score",
...        "Name" : "$name" },
...   }},
...   { "$sort" : { "Points" : -1 } },
...   { "$limit" : 4},
... ])
{ "_id" : { "Borough" : "Staten Island", "Cuisine" : "American ", "Zip" : "10305", "Grades" : [ "A", "A", "A", "A" ], "score" : [ 10, 8, 10, 9 ], "Name" : "Gennaro
{ "_id" : { "Borough" : "Staten Island", "Cuisine" : "American ", "Zip" : "10305", "Grades" : [ "A", "A" ], "score" : [ 12, 7 ], "Name" : "Fire Grilled Burgers" } 
{ "_id" : { "Borough" : "Staten Island", "Cuisine" : "American ", "Zip" : "10305", "Grades" : [ "Not Yet Graded" ], "score" : [ 4 ], "Name" : "J'S On The Bay" } }
{ "_id" : { "Borough" : "Staten Island", "Cuisine" : "American ", "Zip" : "10305", "Grades" : [ "B", "B", "A", "B" ], "score" : [ 23, 25, 11, 18 ], "Name" : "D-Lish
dublin:PRIMARY>
dublin:PRIMARY> db.restaurants.aggregate([
...      { "$match" : {
...          "borough" : borough_name,
...          "cuisine" : cuisine_name,
...          "address.zipcode" : zipcode4
...        } },
...      {"$project" : {"_id" :0, "name" : 1,
...                  "AvgScore" : { "$avg" : "$grades.score" },
...                  "SizeGrades": {"$size": "$grades.grade"}
...      }},
...      { "$match": {"SizeGrades": {"$gt" : 4}}},
...      { "$sort" : { "AvgScore" : -1 } },
... ])
{ "name" : "Guys Community Store", "AvgScore" : 12.8, "SizeGrades" : 5 }
{ "name" : "Rosebank Tavern", "AvgScore" : 9.833333333333334, "SizeGrades" : 6 }
{ "name" : "Perkins Family Restaurant & Bakery", "AvgScore" : 8, "SizeGrades" : 5 }
dublin:PRIMARY>
dublin:PRIMARY>
```

Fig7. Displays the MongoDB query No.4, with the following result:

{ "name" : "Guys Community Store", "AvgScore" : 12.8, "SizeGrades" : 5 }

{ "name" : "Rosebank Tavern", "AvgScore" : 9.833333333333334, "SizeGrades" : 6 }

{ "name" : "Perkins Family Restaurant & Bakery", "AvgScore" : 8, "SizeGrades" : 5 }

PyMongo Queries

Next, we display the results for the PyMongo queries, but for brevity we will only show Function No. 4 of the PyMongo query code.

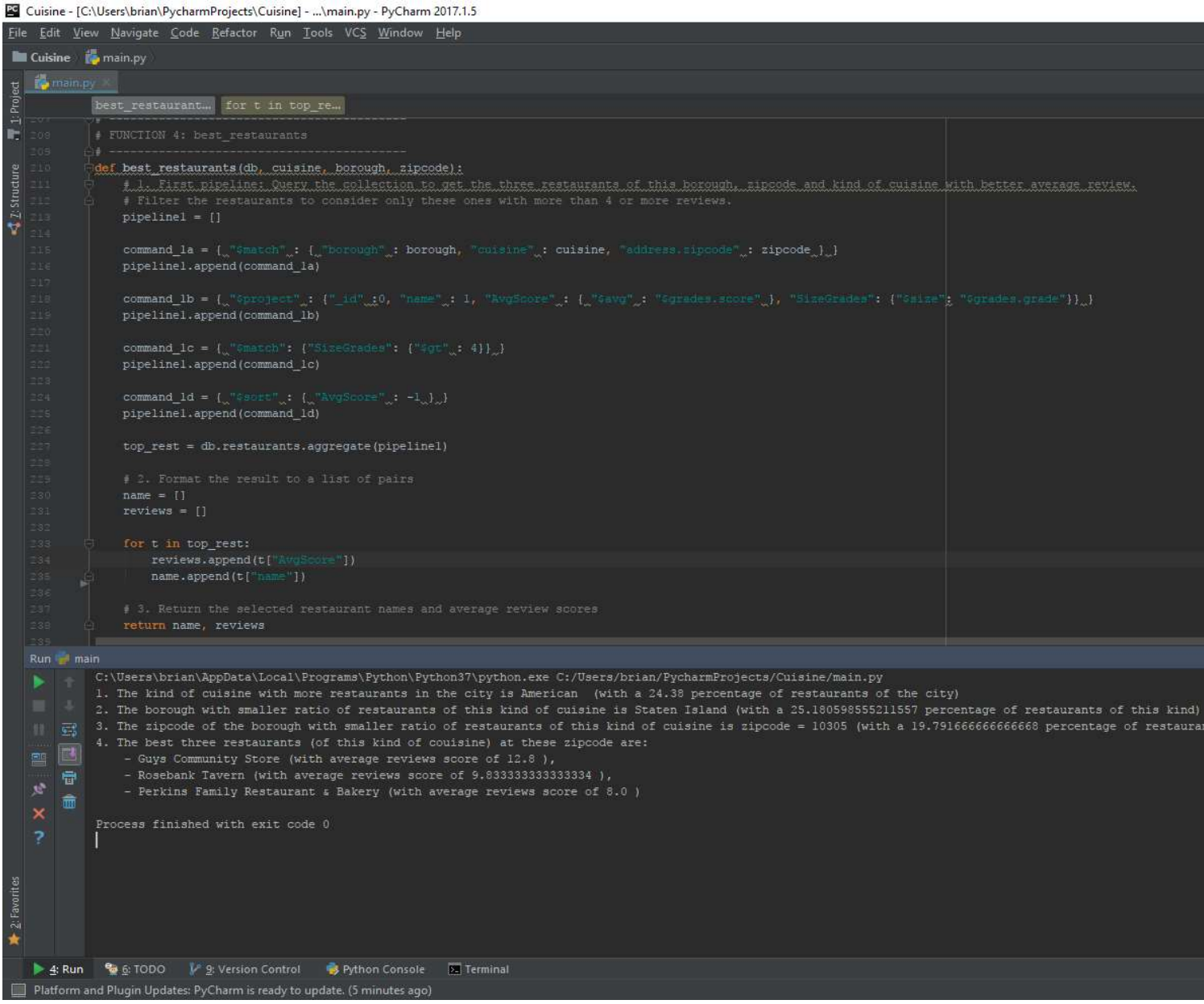

Fig8. Displays the PyMongo query No.4. All queries are displayed with the following result:

1. The kind of cuisine with more restaurants in the city is American (with a 24.38 percentage of restaurants of the city)

2. The borough with smaller ratio of restaurants of this kind of cuisine is Staten Island (with a 25.180598555211557 percentage of restaurants of this kind)

3. The zipcode of the borough with smaller ratio of restaurants of this kind of cuisine is zipcode = 10305 (with a 19.791666666666668 percentage of restaurants of this kind)

4. The best three restaurants (of this kind of couisine) at these zipcode are:

- Guys Community Store (with average reviews score of 12.8),
- Rosebank Tavern (with average reviews score of 9.833333333333334),
- Perkins Family Restaurant & Bakery (with average reviews score of 8.0)

Neo4j Doc Manager

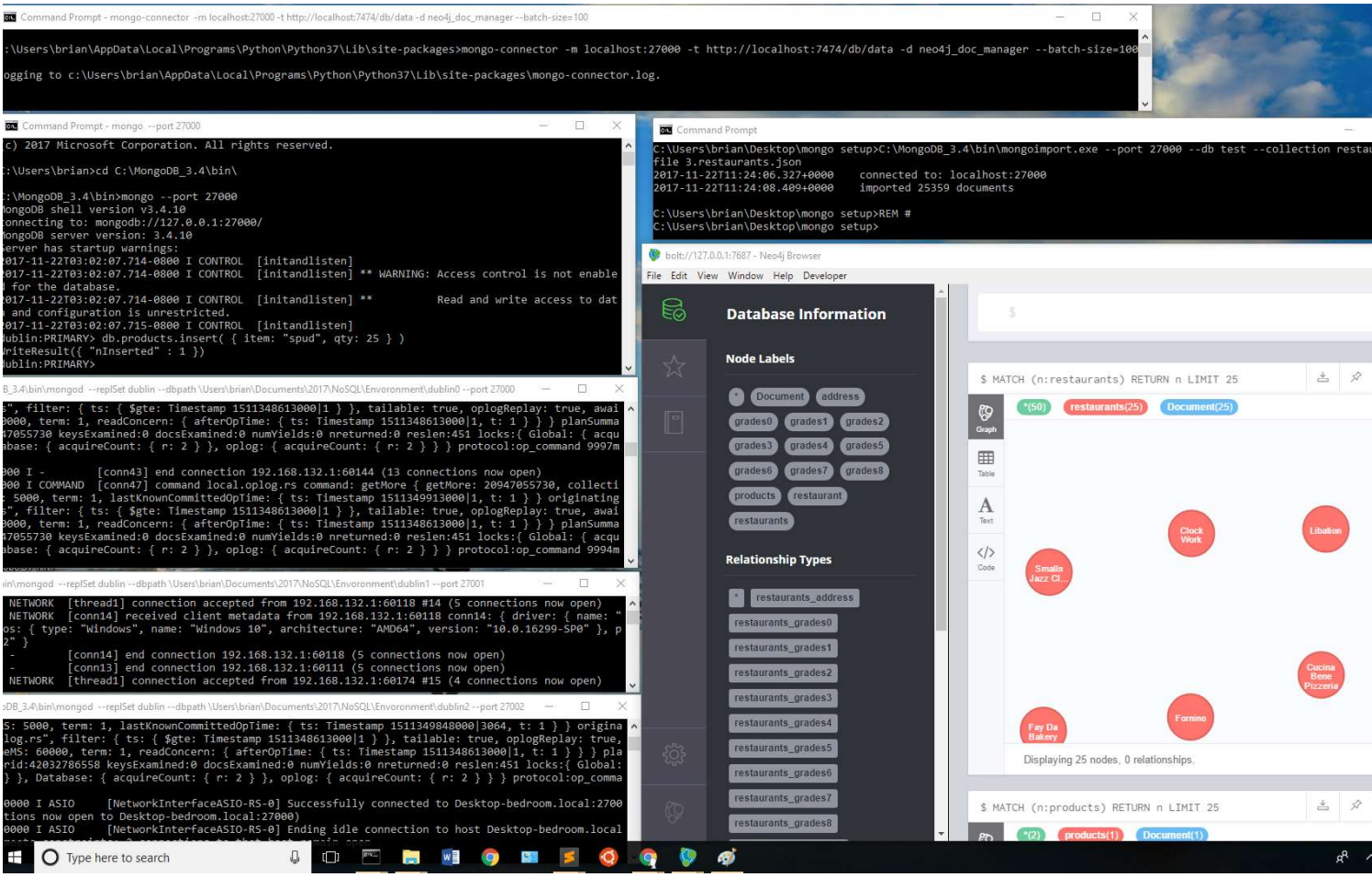

Fig8. Displays neo4j-doc-manager, mongo instances on ports 27000 – 27002, and Neo4j Browser

Neo4j Queries

Find the most_popular_cuisine:
Returns the name of the kind of cuisine with higher number of restaurants in New York and its ratio (percentage).

match (n:restaurants) with count(n.cuisine) as total match (n:restaurants) return (n.cuisine),count (n.cuisine),((100.0((count(n.cuisine))))/total) as percent order by count(n.cuisine) DESC Limit 1*

"(n.cuisine)"	"count (n.cuisine)"	"percent"
"American "	6183	24.381876256950196

Find the ratio_per_borough_and_cuisine :
Returns the name of the borough with smaller percentage of restaurants of the kind of cuisine from (i). It also returns the proper percentage.

match (n:restaurants) with count(n.cuisine) as total match (n:restaurants) where (n.cuisine) = "American " return (n.borough),count (n.cuisine), ((100.0((count(n.cuisine))))/total) as percent order by n.borough DESC Limit 1*

"(n.borough)"	"count (n.cuisine)"	"percent"
"Staten Island"	244	0.9621830513821523

Find the ratio_per_zipcode:
Returns the name of the zipcode with smaller percentage of restaurants of a particular kind of cuisine from (i) and (ii). It also returns the proper percentage

match (n:restaurants) with count(n.cuisine) as total match (n:restaurants)-[:restaurants_address]->(a:address) where n.cuisine = "American " and n.borough = "Staten Island" return a.zipcode,Count(n.cuisine),((100.0((count(n.cuisine))))/total) as percent order by a.zipcode LIMIT 1*

"(n.borough)"	"count (n.cuisine)"	"percent"
"Staten Island"	244	0.9621830513821523