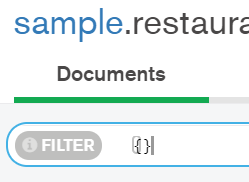
**Guided Study: MongoDB Basic Queries in Compass**

Database Exploration

1. In the FILTER query bar in Compass, type *{ }* and select FIND.



2. How many documents are returned?

25359

3. What are the first level keys in the first document? By first level, I mean the names of keys that you can see in Compass without expanding the values. Include *\_id* in your list.

\_id

address

borough

cuisine

grades

name

restaurant\_id

4. a. How many first level arrays are there?

1

b. Which keys are arrays?

grades

c. What is the type of the values in the array?

They are objects

d. Expand the first object (its key is 0). What are the keys and types of the object?

date – type date

grade – type string

score – type int32

e. Do all of the array entries (objects 0-4) have the same structure (i.e., same fields and types)?

Yes

5. a. How many first level objects are there?

1

b. Which keys are objects?

Address

c. How many second level keys are in the object?

4 total

d. What are the second level keys and their types?

Building – String

Coord – Array

Street – String

Zipcode – String

e. What is the key and type of the third level of nesting in this document?

In Coord, there are the keys 0 and 1 which are int32. Their values are

f. How many values are in the third level? What are their types?

2, they are doubles

0: -73.856077

1: 40.848447

6. Try each of the different view buttons:



a. How does the second one display the documents?

The second displays the JSON in its naturally built form

b. How does the third one display the documents?

In a tabular format.

c. Which do you prefer? Why?

The first, it reads more like the actual format, and it’s cleaner

7. Select the *SCHEMA* tab and then *ANALYZE*.

a. Scroll over the bar graph for *borough*. What are the top three boroughs and what percentage of the documents have these values?

Manhattan – 48%

Queens - 21%

Brooklyn – 19%

b. Scroll over the bar graph for *cuisine*. What are the top three cuisine types and what percentage of the documents have these values?

American – 20%

Chinese – 10%

Café/Coffee/Tea – 8%

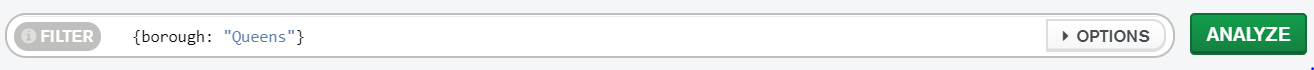
c. Expand the *grades* array. What is the most frequent day of the week (and its frequency) that occurs in the array values?

Tuesday – 23%

d. What is the most frequent grade value, and how frequent is it?

A – 82%

8. In the FILTER bar on the SCHEMA screen, type in *{borough: “Queens”}* and select ANALYZE.



a. How many documents are in the result?

5656

b. What percentage of documents is the analysis based on?

17.68%

c. What restaurant name occurs most frequently in Queens in the sampled data, and how frequently does it occur?

Mcdonald’S – 4%

Basic Querying

Write and execute queries for the following data demands.

1. a. Give documents where the borough is “Staten Island” and the cuisine type is “African” (2 documents). Write the filtering expression here:

{borough:"Staten Island", cuisine: "African"}

b. Give documents where the borough is “Queens” or the cuisine type is “Italian” (6594 documents). Write the filtering expression here:

{$or: [{borough:"Queens"}, {cuisine: "Italian"}]}

c. Give documents where the borough is “Manhattan” and the cuisine type is “Mexican” or “Jewish/Kosher”. Use $in for filtering cuisine type. There should be 346 documents. Write the filtering expression here:

{borough: "Manhattan", cuisine: {$in: ["Mexican", "Jewish/Kosher"]}}

2. Write filtering conditions for the following data demands using */pattern/* syntax. Here are some examples:

* {name: /^abc/} returns documents where the name starts with “abc”
* {name: /abc$/} returns documents where the name ends with “abc”
* {name: /^abc/i} returns documents that start with “ABC” or “abc” (the *i* option makes the pattern case insensitive)

a. Give documents where the restaurant name starts with “Bro” (8424 documents).

{borough: /^Bro/}

b. Give documents where the cuisine ends with “ican” and is not “American” (822 documents).

{$and: [{cuisine: /ican$/}, {cuisine: /^(?!American)/}]}

c. Give documents where the name contains starts or ends with “Indian” (10 documents).

{$or: [{name:/^Indian/}, {name: /Indian$/}]}

3. Values of embedded objects can be accessed using dot notation. For example, {“a.b”: 10} returns documents where the *b* key nested under the *a* key has the value 10. Note that double quotes are required.

a. Give the documents that have the zipcode value “11432” (179 documents).

{"address.zipcode":"11432"}

b. Give the documents where the street starts with “W” and the cuisine type is “Pizza” (90 documents).

{"address.street": /^W/, cuisine: "Pizza"}

4. Searching for items in arrays uses $all and $in operators. For example,

* {mylist: {$all: [42, “kitty”]}} finds documents containing an array *mylist* with values of 42 and “kitty”
* {mylist: {$in: [42, “kitty”]}} finds documents containing an array *mylist* with values of 42 or “kitty”

Here is an example of how to reference an object that is an array element:

* { “inventory.qty”: 5} where *inventory* is the array and *qty* is a key of an object in the array

a. Give the documents where the restaurant has a grade of “C” (2708 documents).

{"grades.grade": "C"}

b. Give the documents where the restaurant has a score of 11 and score of 25 (199 documents).

{"grades.score": {$all: [11, 25]}}

5. The PROJECT query bar specifies keys to include/omit in the result. For example,

* {x: 0} includes all keys except *x*
* {x: 1} includes *x* and the *\_id* key
* {x: 1, \_id: 0} includes *x* only

a. Modify your query in (1a) to show only the restaurant names. Give the project expression and results here:

{borough:"Staten Island", cuisine: "African"}

b. Modify your query in (2c) to give only names and zipcodes. Give the project expression and show the first three results here:

6. The SORT query bar specifies keys and sort orders (1 is for ascending and -1 is for descending order).

Modify your query in (3b) to give cuisine types in descending order and restaurant names in ascending order (show only cuisine and restaurant names). Give the sort expression here and show the first three results here: