



# MINIPROJECT 3

March 29, 2018

## ABSTRACT

- 1) MongoDB's querying capabilities using Yelp's datasets
- 2) To research MongoDB's data security capabilities.

SHILPA VERMA  
SRAVANI KOTHI  
SHREYA PATIL  
ANANNYA DAS

23

## Purpose 1

### Practice MongoDB's querying capabilities using Yelp's datasets

- Below are the Mongo Db queries as per requirement:

#### A. Overall total number of businesses per category:

- `db.business.aggregate([{$unwind: "$categories" },{$group: {_id:"$categories",count: {$sum:1}}})`

```
> db.business.aggregate([{$unwind: "$categories" },{$group: {_id:"$categories",count: {$sum:1}}})
{ "_id" : "Signature Cuisine", "count" : 1 }
{ "_id" : "Soba", "count" : 1 }
{ "_id" : "Makerspaces", "count" : 1 }
{ "_id" : "Luggage Storage", "count" : 1 }
{ "_id" : "Reunion", "count" : 2 }
{ "_id" : "Churros", "count" : 1 }
{ "_id" : "Dialysis Clinics", "count" : 1 }
{ "_id" : "Game Meat", "count" : 1 }
{ "_id" : "Friterie", "count" : 1 }
{ "_id" : "Otologists", "count" : 1 }
{ "_id" : "Neuropathologists", "count" : 3 }
{ "_id" : "Sledding", "count" : 2 }
{ "_id" : "Guamanian", "count" : 1 }
{ "_id" : "Market Stalls", "count" : 3 }
{ "_id" : "Homeless Shelters", "count" : 2 }
{ "_id" : "Halfway Houses", "count" : 1 }
{ "_id" : "Eastern European", "count" : 3 }
{ "_id" : "Undersea/Hyperbaric Medicine", "count" : 1 }
{ "_id" : "Bulgarian", "count" : 1 }
{ "_id" : "Hainan", "count" : 1 }
Type "it" for more
```

Summary: The query first unwinds all the categories using \$unwind operation as categories stored in array and then group them using \$group operation on basis of each category and counts the number of times it occurred using \$sum aggregator.

#### B. Overall total number of reviews per business:

- `db.review.aggregate([{$group: {_id:"$business_id", "count": { $sum: 1 }}}])`

```
> db.review.aggregate( [ { $group: { "_id": "$business_id", "count": { $sum: 1 } } } ] )
{ "_id" : "GQMnv10QTLcCpCHKLBcVUg", "count" : 3 }
{ "_id" : "ALJGJTsi2hxEH4f3_Fxo4g", "count" : 3 }
{ "_id" : "7yY0XAmi0TEDFvvDdG4y-A", "count" : 3 }
{ "_id" : "Q5frcnH11L_Wa-De2_-7QQ", "count" : 3 }
{ "_id" : "tZYFHe0R0Gux_2Qt7kf0vA", "count" : 3 }
{ "_id" : "pHq_h1Wq1B0QKexF7k02mg", "count" : 3 }
{ "_id" : "NvQJ0ZHRDSU245CuaygdwQ", "count" : 4 }
{ "_id" : "GrwDGSdhY4uNCFk9bPy_4A", "count" : 3 }
{ "_id" : "ITjzN0ImNSyS9Fbs0Z3IUA", "count" : 3 }
{ "_id" : "Nc5_iuwigzfu1DorLIXHdg", "count" : 3 }
{ "_id" : "gyprJHSEsxpvesxfKjYzA", "count" : 3 }
{ "_id" : "U-UnzdmX4WN0AwuSsQm2Ag", "count" : 3 }
{ "_id" : "pf911i3fT44SwNzEjy4V5A", "count" : 3 }
{ "_id" : "S8Wp6CALbFZb8G-if3-76g", "count" : 3 }
{ "_id" : "EfEa965NrYsNBtc6EFNSvA", "count" : 3 }
{ "_id" : "DkRDkF6waI-MrAegs8u2cw", "count" : 3 }
{ "_id" : "dvR-ImHgrDBCbKi0eDtBmg", "count" : 3 }
{ "_id" : "R1kN0-porVxkRCYeLowa2w", "count" : 3 }
{ "_id" : "N132xJtBa_tZHKA VPNevA", "count" : 3 }
{ "_id" : "BoZ_7fN8hH6_fdFHCqa9qw", "count" : 5 }
Type "it" for more
```

Summary: To find total number of reviews per business, we have to group by business\_id and count the number of records i.e., reviews in the review document.

### C. Overall total number of reviewers per business:

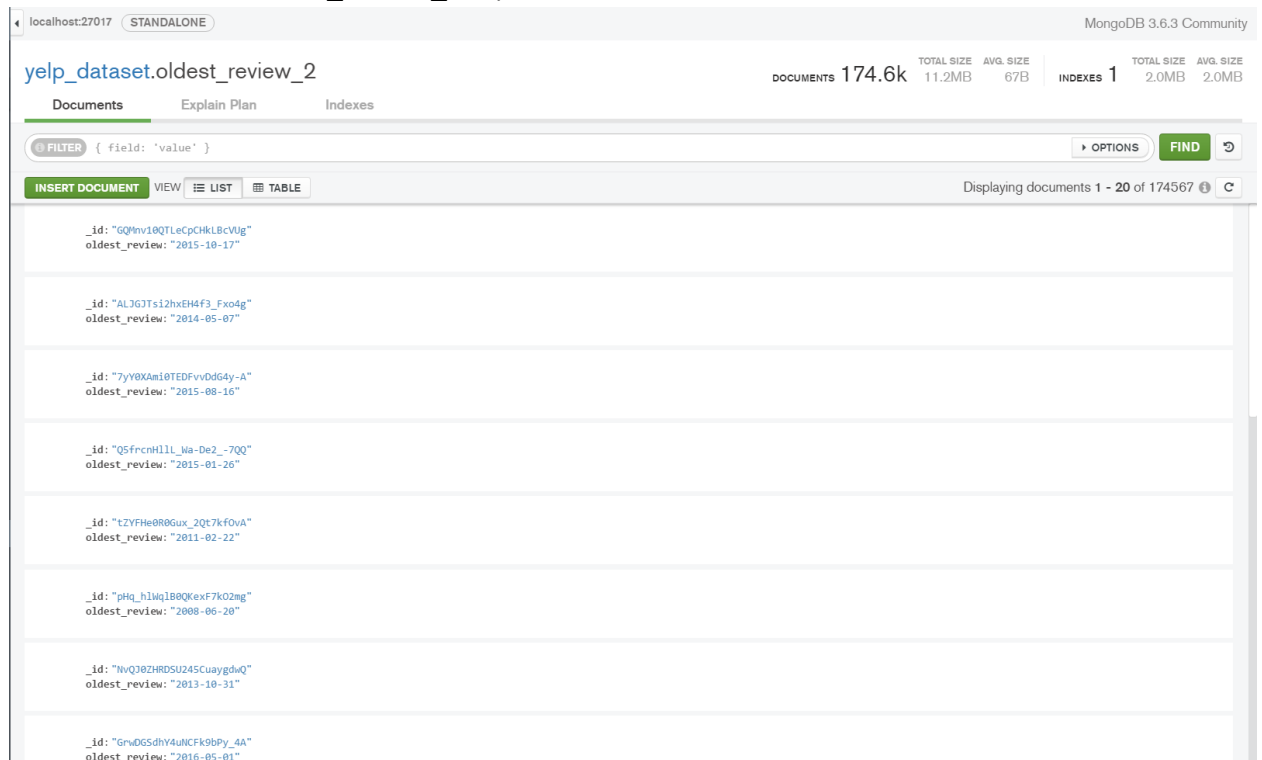
```
db.review.aggregate([ { "$group": { "_id": { "business": "$business_id", "reviewer": "$user_id", "reviewerCount": { "$sum": 1 } } }, { "$group": { "_id": "$_id.business", "count": { "$sum": "$reviewerCount" } } } ], { allowDiskUse: true, cursor: { } })
```

```
> db.review.aggregate([ { "$group": { "_id": { "business": "$business_id", "reviewer": "$user_id", "reviewerCount": { "$sum": 1 } } }, { "$group": { "_id": "$_id.business", "count": { "$sum": "$reviewerCount" } } } ], { allowDiskUse: true, cursor: { } })
{ "_id" : "zzzaIBwimxVej4tY6qFOUQ", "count" : 37 }
{ "_id" : "zzwicjPC9g246MK2M1ZFBA", "count" : 51 }
{ "_id" : "zzwaS0xn1MVEPEf0hNLjew", "count" : 64 }
{ "_id" : "zzsU528uoRB6qZUGhKDa6w", "count" : 7 }
{ "_id" : "zzmIMvqiBJ_-wVKg_OnGpw", "count" : 88 }
{ "_id" : "zz1ZJvKEh0zR2tJOLHcF2A", "count" : 38 }
{ "_id" : "zzjKekzQ6i4iR-qpo405Pw", "count" : 12 }
{ "_id" : "zzf3RkMI1Y2E1QaZqeU8yA", "count" : 33 }
{ "_id" : "zzaFwWri6yvvhvMsjs-16w", "count" : 3 }
{ "_id" : "zzYaAiC0rLNSDiFQ1MKOEQ", "count" : 5 }
{ "_id" : "zzXuJF6UUT1kgTyCsmgDmQ", "count" : 4 }
{ "_id" : "zzSbdY2Ztja3Htt8bsQ6Aw", "count" : 12 }
{ "_id" : "zzRgQ-Y1PKRMHVzXSHL1nA", "count" : 4 }
{ "_id" : "zzOo9n22fBbKAhbSpMzggA", "count" : 26 }
{ "_id" : "zz09QVUj-XvZ8trNX21qAg", "count" : 9 }
{ "_id" : "zzMu-6SmqhpvhXVRM6tx9g", "count" : 5 }
{ "_id" : "zzDJg87wUzGWPXesYEnFkQ", "count" : 5 }
{ "_id" : "zzDCRyti7enk9ZUUGowYNQ", "count" : 4 }
{ "_id" : "zzD64fKs3oEh2VTLYFu19w", "count" : 9 }
{ "_id" : "zzCsh8Rx40ZXcyDrzFnFOQ", "count" : 11 }
Type "it" for more
```

Summary: Here, we have aggregated the review collection by first grouping based on business\_id and user\_id and counting their occurrences and then eventually using this group to count the number of user\_ids reviewing a business using \$sum to display the number of reviewers a business has.

#### D. Date of the oldest review per business:

- `db.review.aggregate([ {$group: {_id: "$business_id", oldest_review: {$min: "$date"}}}, {$out: "oldest_review_2"}])`



The screenshot displays the MongoDB GUI for the 'yelp\_dataset.oldest\_review\_2' collection. The interface shows a filter bar with the filter '{ field: 'value' }'. Below the filter bar, there are tabs for 'Documents', 'Explain Plan', and 'Indexes'. The 'Documents' tab is active, showing a list of documents. The table view displays the following data:

_id	oldest_review
"6QWv10QTLcCpChkLbcVUg"	"2015-10-17"
"ALJG3Ts12hxEH4f3_Fxo4g"	"2014-05-07"
"7yY0XAmi0TEDFvvOd64y-A"	"2015-08-16"
"Q5frcnH1L_wa-De2_-7QQ"	"2015-01-26"
"tZYFHe0R6gux_2Qt7kFovA"	"2011-02-22"
"pHq_h1Wq1B0QcexF7kO2mg"	"2008-06-20"
"NvQ10ZHRDSU245CuaygdwQ"	"2013-10-31"
"GrwDGSdhY4uNCFk9bPy_4A"	"2016-05-01"

Summary: In this query, we have grouped the documents of Review collection using `business_id`, calculated the minimum date of review posted for the `business_ids` in the group and then stored the output of the above process in a collection called `"oldest_review_2"`. The screenshot is of the documents as seen in the GUI of MongoDB.

**E. Average review per business – display the results for 50 businesses only:**

- `db.review.aggregate([{$group: {_id:"$business_id", avg_review: {`  
`$avg:"$stars"}}},{ $limit: 50}])`

```
> db.review.aggregate([{$group: {_id:"$business_id", avg_review: { $avg:"$stars"}}},{ $limit: 50}])
{ "_id" : "GQMnv10QTLeCpCHkLBcVUg", "avg_review" : 2 }
{ "_id" : "ALJGJTsi2hxEH4f3_Fxo4g", "avg_review" : 4.666666666666667 }
{ "_id" : "7yY0XAmi0TEDFvvdG4y-A", "avg_review" : 5 }
{ "_id" : "Q5frcnH1lL_Wa-De2_-7QQ", "avg_review" : 5 }
{ "_id" : "tZYFHe0R0Gux_2Qt7kf0vA", "avg_review" : 3 }
{ "_id" : "pHq_hlWq1B0QKexF7k02mg", "avg_review" : 1.3333333333333333 }
{ "_id" : "NvQJ0ZHRDSU245CuaygdwQ", "avg_review" : 1 }
{ "_id" : "GrwDGSdhY4uNCFk9bPy_4A", "avg_review" : 5 }
{ "_id" : "ITjzN0lmNSyS9FbsOZ3IUA", "avg_review" : 3.6666666666666665 }
{ "_id" : "Nc5_iuwgzwfU1DorLIXHdg", "avg_review" : 3.6666666666666665 }
{ "_id" : "gyprJHSEsxpvesxfKjYzA", "avg_review" : 4.666666666666667 }
{ "_id" : "U-UnzdmX4WN0AwuSsQm2Ag", "avg_review" : 4.666666666666667 }
{ "_id" : "pf9l1i3fT44SwNzEjy4V5A", "avg_review" : 3.6666666666666665 }
{ "_id" : "S8Wp6CALbFZb8G-if3-76g", "avg_review" : 2 }
{ "_id" : "EFea965NrYsNBTc6EFNSvA", "avg_review" : 4.333333333333333 }
{ "_id" : "DkRDkF6waI-MrAegs8u2cw", "avg_review" : 3.6666666666666665 }
{ "_id" : "dvR-ImHgrDBCbKiOeDtBmg", "avg_review" : 3.6666666666666665 }
{ "_id" : "R1kN0-porVxkRCYeLowa2w", "avg_review" : 2 }
{ "_id" : "N132xJtBa_tZHKAVNPNevA", "avg_review" : 5 }
{ "_id" : "BoZ_7fN8hH6_fdFHcqa9qw", "avg_review" : 5 }
Type "it" for more
>
```

**Summary:**

The query first groups the review collection on basis of business\_id using \$group operation and calculates the average rating by calculating the average of stars associated with that business\_id using \$avg. Later using \$limit operation, it limits the result to 50 businesses only.

**F. Histogram of review ratings (5's, 4's, 3's, 2's and 1's) per business – display the results for 50 businesses only. Note that there is no need for a graphical histogram like the one displayed below, a tabular one is sufficient:**

- `db.review.aggregate([{"$group": {"_id": {"business": "$business_id", "star": "$stars"}, "starCount": {"$sum": 1}}}, {"$group": {"_id": "$_id.business", "stars": {"$push": {"star": "$_id.star", "count": "$starCount"}}}], {"$limit": 50}], {allowDiskUse:true, cursor:{}})`

```
> db.review.aggregate([{"$group": {"_id": {"business": "$business_id", "star": "$stars"}, "starCount": {"$sum": 1}}},
{"$group": {"_id": "$_id.business", "stars": {"$push": {"star": "$_id.star", "count": "$starCount"}}}], {"$limit": 50
}], {allowDiskUse:true, cursor:{}})
{"_id": "--6MefnULPED_I942VcFNA", "stars": [ {"star": 1, "count": 6 }, { "star": 2, "count": 6 }, { "star": 3,
"count": 8 }, { "star": 4, "count": 11 }, { "star": 5, "count": 6 } ] }
{"_id": "--7zmmkVg-IMGaXbuVd0SQ", "stars": [ {"star": 1, "count": 4 }, { "star": 2, "count": 3 }, { "star": 3,
"count": 6 }, { "star": 4, "count": 17 }, { "star": 5, "count": 17 } ] }
{"_id": "--8LPVSo5i00o61X01sV9A", "stars": [ {"star": 3, "count": 1 }, { "star": 5, "count": 2 } ] }
{"_id": "--9QQLMTbFzLJ_oT-ON3Xw", "stars": [ {"star": 1, "count": 3 }, { "star": 2, "count": 1 }, { "star": 3,
"count": 1 }, { "star": 4, "count": 1 }, { "star": 5, "count": 5 } ] }
{"_id": "--9e10NYQuAa-CB_Rrw7Tw", "stars": [ {"star": 1, "count": 45 }, { "star": 2, "count": 131 }, { "star":
3, "count": 171 }, { "star": 4, "count": 407 }, { "star": 5, "count": 697 } ] }
{"_id": "--DaPTJW3-tB1vP-PfdTEg", "stars": [ {"star": 1, "count": 4 }, { "star": 2, "count": 2 }, { "star": 3,
"count": 8 }, { "star": 4, "count": 14 }, { "star": 5, "count": 11 } ] }
{"_id": "--DdmeR16TRb3LsjG0ejrQ", "stars": [ {"star": 2, "count": 2 }, { "star": 4, "count": 3 } ] }
{"_id": "--EF5N7P70J_UYBTPypY1A", "stars": [ {"star": 2, "count": 2 }, { "star": 5, "count": 1 } ] }
{"_id": "--EX4rRznJrItyn-34Jz1w", "stars": [ {"star": 3, "count": 1 }, { "star": 4, "count": 2 }, { "star": 5,
"count": 1 } ] }
{"_id": "--FBCX-N37CMYDfs790Bnw", "stars": [ {"star": 1, "count": 19 }, { "star": 2, "count": 5 }, { "star": 3,
"count": 9 }, { "star": 4, "count": 29 }, { "star": 5, "count": 46 } ] }
{"_id": "--FLdgM0GnpXVMn74ppCGw", "stars": [ {"star": 1, "count": 2 }, { "star": 4, "count": 1 }, { "star": 5,
"count": 3 } ] }
{"_id": "--GM_ORV2cYS-h38DSaCLw", "stars": [ {"star": 1, "count": 1 }, { "star": 4, "count": 2 }, { "star": 5,
"count": 4 } ] }
{"_id": "--I7YYLada0tSLKORTHb5Q", "stars": [ {"star": 1, "count": 8 }, { "star": 2, "count": 10 }, { "star": 3,
"count": 13 }, { "star": 4, "count": 21 }, { "star": 5, "count": 20 } ] }
{"_id": "--KCl2FvVQpvjzmZSPyviA", "stars": [ {"star": 1, "count": 3 }, { "star": 2, "count": 2 }, { "star": 3,
"count": 3 }, { "star": 4, "count": 1 }, { "star": 5, "count": 3 } ] }
{"_id": "--KQsXc-clk07oHRqGzSzg", "stars": [ {"star": 1, "count": 9 }, { "star": 2, "count": 5 }, { "star": 3,
"count": 5 }, { "star": 4, "count": 7 }, { "star": 5, "count": 5 } ] }
{"_id": "--LY7PrnEegg1B7vnPCjQw", "stars": [ {"star": 1, "count": 1 }, { "star": 3, "count": 1 }, { "star": 4,
"count": 6 }, { "star": 5, "count": 2 } ] }
{"_id": "--Ni3oJ4V0qf0Eu7Sj2Vzg", "stars": [ {"star": 1, "count": 3 }, { "star": 2, "count": 2 }, { "star": 5,
"count": 1 } ] }
{"_id": "--R3uiY2dB43MpdwtG6jhQ", "stars": [ {"star": 1, "count": 1 }, { "star": 2, "count": 3 }, { "star": 3,
"count": 1 }, { "star": 4, "count": 2 }, { "star": 5, "count": 4 } ] }
{"_id": "--Rsj71PBe31h5Y1jVseKA", "stars": [ {"star": 1, "count": 1 }, { "star": 2, "count": 1 }, { "star": 4,
"count": 2 }, { "star": 5, "count": 3 } ] }
{"_id": "--S62v0QgkQaVUhfNhrw", "stars": [ {"star": 1, "count": 18 }, { "star": 2, "count": 1 }, { "star": 3,
"count": 1 }, { "star": 4, "count": 5 }, { "star": 5, "count": 3 } ] }
Type "it" for more
```

Summary: In this query, a group by on business\_id and stars is done and for each business\_id for multiple reviews, \$sum of each star rating is counted and pushing this array with star and starCount into business\_id and limiting to only 50 businesses.

**G. Top 10 most useful reviews per business – display the results for 50 businesses only (you will have to provide your own text-based definition of what helpful means and thus not use the Helpful field):**

- `db.review.createIndex({text: "text"})`
- `db.review.aggregate([{$match: {$text: {$search: "good awesome excellent nice amazing cool beautiful best love -bad -worst -horrible -trash -unfriendly -hate"}}}, {$group: {"_id": {"business": "$business_id", "review": "$_id", "date": "$date"}, "reviewCount": {"$sum": 1 }}}, {$group: {"_id": "$_id.business", "reviews": {"$push": {"review": "$_id.review"}}, "count": {"$sum": "$reviewCount"} }}, {$sort: { score: {$meta: "textScore"}, date: -1 } }, {$limit: 50 }, {$project: {"Useful reviews": {"$slice": [ "$reviews", 10] }}} ], {allowDiskUse: true, cursor: { }}).pretty()`

```
> db.review.aggregate([{$match: {$text: {$search: "good awesome excellent nice amazing cool beautiful best love -bad -worst -horrible -trash -unfriendly -hate"}}}, {$group: {"_id": {"business": "$business_id", "review": "$_id", "date": "$date"}, "reviewCount": {"$sum": 1 }}}, {$group: {"_id": "$_id.business", "reviews": {"$push": {"review": "$_id.review"}}, "count": {"$sum": "$reviewCount"} }}, {$sort: { score: {$meta: "textScore"}, date: -1 } }, {$limit: 50 }, {$project: {"Useful reviews": {"$slice": [ "$reviews", 10] }}} ], {allowDiskUse: true, cursor: { }}).pretty()
{
  "_id" : "--phjqoPSPa8sLmUVNby9w",
  "Useful reviews" : [
    {
      "review" : ObjectId("5ab3240113e54aef92d12604")
    },
    {
      "review" : ObjectId("5ab3240c13e54aef92d7821a")
    },
    {
      "review" : ObjectId("5ab3240f13e54aef92d97d6b")
    },
    {
      "review" : ObjectId("5ab3244d13e54aef92f4e860")
    }
  ]
}
{
  "_id" : "--z7PM8AGaJP0aBmGMY7RA",
  "Useful reviews" : [
    {
      "review" : ObjectId("5ab3240013e54aef92d1125f")
    },
    {
      "review" : ObjectId("5ab3241713e54aef92de1ebc")
    }
  ]
}
```

Summary: A useful review in our perception is which matches the text criteria of good, awesome, excellent, nice, amazing, cool, beautiful, best, love, and not containing bad, worst, horrible, trash, unfriendly, hate and with a recent review date. Therefore, in the query we have first created an index text for text search based on the reviews in the documents field called text, matched the text with the above criteria, grouped it by business\_id, sorted it on the textScore and date, and showing only 10 useful reviews from the useful reviews array and limiting the businesses to 50.

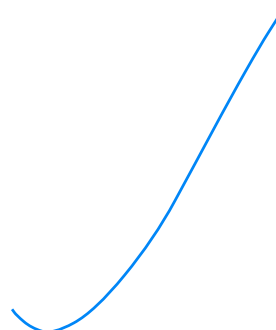


```
➤ db.review.aggregate([{"$group": {"_id": {"business": "$business_id", "review":
```

```

> db.review.aggregate([{"$group": {"_id": {"business": "$business_id", "review": "$review_id", "date": "$date"}, "reviewCount": {"$sum": 1}}}, {"$group": {"_id": "$_id.business", "reviews": {"$push": {"review": "$_id.review", "count": "$reviewCount"}, "count": {"$sum": "$reviewCount"}}}, {"$sort": {"_id.date": -1}}, {"$limit": 50}, {"$project": {"review": {"$slice": [ "$reviews", 10] }, "count": 1}}, {"allowDiskUse": true, cursor: {}})
{"_id": "--o5BoU7qYMALEvDK6mwVg", "count": 6, "reviews": [ {"review": "0UFGbvX83-0DdaPyj_ggBw", "count": 1 }, {
"review": "EVS0jYXTI40nSjNsrPF_JA", "count": 1 }, { "review": "FrRhsKK4V5oC9xGHSFVUMA", "count": 1 }, { "review": "d
3BCV3iKlVZBS6VT7ZV8w", "count": 1 }, { "review": "sOd7KjagYFKKuIdtu_4MrA", "count": 1 }, { "review": "ynnFNTzJa5ngT
iNz3Z2nkQ", "count": 1 } ] }
{"_id": "--ujyvoQ1lw0B6MytADiLA", "count": 68, "reviews": [ { "review": "-ikzPr1ZM3_EXUGHFZNzeA", "count": 1 }, {
"review": "-nx4S8G5HUKI-h-z-T60oQ", "count": 1 }, { "review": "060C7zHnScDINZFLq4doiQ", "count": 1 }, { "review": "1
Y2ZPR8dA867YtAeH2CXgw", "count": 1 }, { "review": "4mi5-2nKgRQ0u6K0hXA3vQ", "count": 1 }, { "review": "2EnQFpwyu98e
-S9HCzuga", "count": 1 }, { "review": "2y-lkb0eLPLjFAsM9YI06Q", "count": 1 }, { "review": "4-W5E55Z4nDM6vU1j-a5GA",
"count": 1 }, { "review": "41cydhKcENPMs2IHUYfi6Q", "count": 1 }, { "review": "4AOXqQAhY2CYFC79qYv0dg", "count": 1
} ] }
{"_id": "--U98MNIIdym2cLn36BBPGQ", "count": 4, "reviews": [ { "review": "IdbVo_V6aUWmaX4x_S_Tg", "count": 1 }, {
"review": "NT4BRvPEV4a1uNwOwp_ceg", "count": 1 }, { "review": "ag0cfqahx3mhcaXCTcewNg", "count": 1 }, { "review": "j
bOKThA2x3L16uQV7wDybA", "count": 1 } ] }
{"_id": "--i1tTcgB8i4cPkdh5hDg", "count": 19, "reviews": [ { "review": "40QMcShoqiB0oS1hui9AcQ", "count": 1 }, {
"review": "9DPpElTy2TeTzKr_jhk68A", "count": 1 }, { "review": "Asgw0Abe0omrWhoWJctyRQ", "count": 1 }, { "review": "
BYmhoMXce0j1BvZpZq-B5g", "count": 1 }, { "review": "BcdD7NcZ1PiCooixT0t6vQ", "count": 1 }, { "review": "B8xr7Nha6ch8D
KcemMYS2Xg", "count": 1 }, { "review": "EGBwT9d0ggjZ3GjNj0c8oJQ", "count": 1 }, { "review": "FZ1fyCMG9LgAs9bmpi50_g",
"count": 1 }, { "review": "FjxjJk_EVGuCz3JLpw2zBQ", "count": 1 }, { "review": "LXJInfEZqPV00-e7vzuZ9CA", "count": 1
} ] }
{"_id": "--Y7NhBKzLTbnliMUX_wfg", "count": 8, "reviews": [ { "review": "8lfr-ALfJ0kay3DgJjwyKg", "count": 1 }, {
"review": "EEERgynjBxLBT0H4ZPcrrmw", "count": 1 }, { "review": "G2ao2mNNGCB7yK18hv-JrA", "count": 1 }, { "review": "Z
zLOVggoZpmkMXvWCL_ERA", "count": 1 }, { "review": "a9mqDaqrAPQaXnWex625g", "count": 1 }, { "review": "c0bwdh42BDITw
m12wNHGjQ", "count": 1 }, { "review": "qDGhRvjNacdNnQPg5iEnUA", "count": 1 }, { "review": "sHaf_3_S1NDUA9pDskHgdg",
"count": 1 } ] }
{"_id": "--Rs7j1PB8e31h5YljVseKA", "count": 7, "reviews": [ { "review": "2DNxaJfpqjgJRCjVlt3Tw", "count": 1 }, {

```





ost:27017/yelp\_dataset.recent\_reviews\_per\_business

localhost:27017 STANDALONE MongoDB 3.6.3 Community

yelp\_dataset.recent\_reviews\_per\_business DOCUMENTS 50 TOTAL SIZE 25.4KB AVG. SIZE 520B INDEXES 1 TOTAL SIZE 16.0KB AVG. SIZE 16.0KB

Documents Explain Plan Indexes

FILTER { field: 'value' } OPTIONS FIND

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 20 of 50

```
{
  "_id": "--o5BoU7qYMALeVDK6mwVg",
  "count": 6,
  "reviews": Array
    > 0: Object
    > 1: Object
    > 2: Object
    > 3: Object
    > 4: Object
    > 5: Object
}

{
  "_id": "--ujyvoQlwVoBgMYtADiLA",
  "count": 68,
  "reviews": Array
    > 0: Object
    > 1: Object
    > 2: Object
    > 3: Object
    > 4: Object
    > 5: Object
    > 6: Object
    > 7: Object
    > 8: Object
    > 9: Object
}

{
  "_id": "--U98MNI1Dym2cln36BBPgQ",
  "count": 4,
  "reviews": Array
}

{
  "_id": "--11tTcgg8I4cPkd-h5hDg",
  "count": 19,
  "reviews": Array
}
```

Summary: In this query, we are aggregating the review collection by first creating a group called business that groups the documents using business\_id, review\_id and date and counting the number of reviews. Then we use the group business created to group again by pushing the review count into the new group and then sorting it by the date posted in descending manner and finally limiting it to 10 most recently posted reviews per business if number of reviews are greater than 10 or all posts in descending order of date posted.

## I. Top 10 most prolific reviewers (i.e. with most number of reviews):

### i. On basis of review count in user collection:

- `db.user.find({}, {user_id:1,review_count:1,_id:0}).sort({review_count: -1}).limit(10)`

```
> db.user.find({}, {user_id:1,review_count:1,_id:0}).sort({review_count: -1}).limit(10)
{ "user_id" : "8k3a0-mPeyhbR5HUucA5aA", "review_count" : 11954 }
{ "user_id" : "RtGqdDBvvBCjcu5dUqwFzA", "review_count" : 11323 }
{ "user_id" : "P5bUL3Engv-2z6kKohB6qQ", "review_count" : 9788 }
{ "user_id" : "hWdybu_KvYLSdEFzGrniTw", "review_count" : 8353 }
{ "user_id" : "8RcEwGrFIgkt9WQ35E6SnQ", "review_count" : 7764 }
{ "user_id" : "Xwnf20FKuikiHcSpcEbpKQ", "review_count" : 6653 }
{ "user_id" : "nmdkHL2JKFx55T3nq5VziA", "review_count" : 6314 }
{ "user_id" : "CxDOItnH8gp9KXzpBHJYXw", "review_count" : 5868 }
{ "user_id" : "HFECrzYDpgbS5EmTbtj2zQ", "review_count" : 5344 }
{ "user_id" : "kS1MQHYwIfD0462PE61IBw", "review_count" : 4634 }
>
```

### Summary:

The query finds 10 such records from user table whose review\_count in user collection is the most. Query first limits its projection to user\_id and review\_count, then sorts the records such that review\_count appears in descending order and then limits the result to 10 providing top 10 prolific reviewers.

### ii. On basis of actual reviews available in review collection for particular user:

- `db.review.aggregate([{$group: {_id:"$user_id", noOfReviews:{$sum:1}}},{ "$sort":{"noOfReviews":-1}},{$limit:10}], {allowDiskUse: true, cursor: {}})`

```
> db.review.aggregate([{$group: {_id:"$user_id", noOfReviews:{$sum:1}}},{ "$sort":{"noOfReviews":-1}},{$limit:10}], {allowDiskUse: true, cursor: {}})
{ "_id" : "CxDOItnH8gp9KXzpBHJYXw", "noOfReviews" : 3569 }
{ "_id" : "bLbSNkLggFnqWNNzzq-Ijw", "noOfReviews" : 2077 }
{ "_id" : "PKEzKwv_FktMm2mGPjwd0Q", "noOfReviews" : 1611 }
{ "_id" : "DK57YibC5ShBmqQ197CKog", "noOfReviews" : 1463 }
{ "_id" : "QJI90SEn6ujRCtrX06vslw", "noOfReviews" : 1322 }
{ "_id" : "d_TBs6J3twMy9GChqUEXkg", "noOfReviews" : 1184 }
{ "_id" : "ELcQD1f69kb-ihJfxZyL0A", "noOfReviews" : 1159 }
{ "_id" : "cMEtAiW60I5wE_vLFTxoJQ", "noOfReviews" : 1126 }
{ "_id" : "hWdybu_KvYLSdEFzGrniTw", "noOfReviews" : 1117 }
{ "_id" : "U4INQZOPSUaj8hmjL1Z3KA", "noOfReviews" : 1101 }
>
```

Summary: The query first groups the review collection on basis of user\_id using \$group and counts the number of its occurrence using \$sum which provides us the number of reviews provided by that customer. Later we sort this collection on basis of descending order of number of reviews obtained using \$sort and limits the result to 10 using \$limit. allowedDiskUse command in query enables writing to temporary files when memory limit exceeds in system.

#### J. Top 10 most verbose reviewers (i.e. that write the most text):

In this query, we can interpret verbose reviewers as the ones who write more text or the ones who write most the text w.r.t the number of reviews they write. There are 2 approaches for this query.

Ex: A user writes 10 reviews with total length of 100 and another user with 9 reviews with total length of 95, this shows discrepancy when taking average. So, we are showing 2 approaches with total text length and average text length.

- `db.review.aggregate([ { $group: { _id: "$user_id", "review_length": {$sum: {"$strLenCP": "$text"}} } }, { $sort: { review_length: -1 } }, { $limit: 10 } ], { allowDiskUse: true } )`

```
> db.review.aggregate([ { $group: { _id: "$user_id", "review_length": {$sum: {"$strLenCP": "$text"}} } }, { $sort: { review_length: -1 } }, { $limit: 10 } ], { allowDiskUse: true } )
{ "_id" : "CxDOIDnH8gp9KXzpBHJYXw", "review_length" : 2846762 }
{ "_id" : "U4INQZOPSUaj8hMjL1Z3KA", "review_length" : 2532699 }
{ "_id" : "bLbSNkLggFnqwNNzzq-Ijw", "review_length" : 2261445 }
{ "_id" : "PKEzKWv_FktMm2mGPjwd0Q", "review_length" : 1929929 }
{ "_id" : "rCWrxuRC8_pfagpchtHp6A", "review_length" : 1710151 }
{ "_id" : "0tvCcncfJnSs55iB6mqPk3w", "review_length" : 1648710 }
{ "_id" : "WeVkkF5L39888IPPlRhNpg", "review_length" : 1550064 }
{ "_id" : "n86B7IkbU20Akx1FX_5aew", "review_length" : 1539512 }
{ "_id" : "DK57YibC5ShBmqQ197CKog", "review_length" : 1412337 }
{ "_id" : "M9rRM6Eo5YbKlKM65QIIPA", "review_length" : 1370416 }
```

Summary: To find the reviewers that write most text, In the review collection, we have grouped by user\_id, and added the length of texts of reviews written by each user and sorted them.

- `db.review.aggregate([ { $group: { _id: "$user_id", "review_length": {$avg: {"$strLenCP": "$text"}} } }, { $sort: { review_length: -1 } }, { $limit: 10 } ], { allowDiskUse: true } )`

```
> db.review.aggregate([ { $group: { _id: "$user_id", "review_length": {$avg: {"$strLenCP": "$text"}} } }, { $sort: { review_length: -1 } }, { $limit: 10 } ], { allowDiskUse: true } )
{ "_id" : "-GR_rCzrKU2xa1_jHntQNw", "review_length" : 5000 }
{ "_id" : "54I2e_Z8dTjnGtT4ticgSA", "review_length" : 5000 }
{ "_id" : "5tXE_uTJXPdu7k6Tc40zCQ", "review_length" : 5000 }
{ "_id" : "1rXS8zB-e16rYcv7vLlFCg", "review_length" : 5000 }
{ "_id" : "0MwqON5z-9HNz0o_PNH1Lg", "review_length" : 5000 }
{ "_id" : "5Y4bQEYp_3HXDvy0pViJjg", "review_length" : 5000 }
{ "_id" : "C5Ak5YjzztEvY8FiHRQD7A", "review_length" : 5000 }
{ "_id" : "Ap6WwjPdpH_7PWc7H2sZ6g", "review_length" : 5000 }
{ "_id" : "8VURrvOI1BWLQmin2VfDFw", "review_length" : 5000 }
{ "_id" : "0x6xNj-v8M0pX_UoZai1Lg", "review_length" : 5000 }
```

Summary: To find the reviewers that write most text, In the review collection, we have grouped by user\_id, and found the average of the length of texts of reviews written by each user and sorted them.

**K. Top 10 most positive reviewers based on the ratings of their reviews:**

*i. On the basis of average\_stars field in user collection:*

➤ `db.user.find({average_stars: 5}, {user_id: 1, name: 1, average_stars: 1, _id: 0}).limit(10)`

```
> db.user.find({average_stars: 5}, {user_id: 1, name: 1, average_stars: 1, _id: 0}).limit(10)
{ "user_id" : "WRae-wZkpRoxMrgJdqwyxg", "name" : "Mike", "average_stars" : 5 }
{ "user_id" : "a_gKYQ5YMg39FHNyJLWRHg", "name" : "Joselyn", "average_stars" : 5 }
{ "user_id" : "JaTVvKsBl0bHHJEpESn4pQ", "name" : "Peter", "average_stars" : 5 }
{ "user_id" : "H54pA7YHfj18IjhHAfdXJA", "name" : "Chad", "average_stars" : 5 }
{ "user_id" : "eMBV7FugCJq7FIvGhARo2Q", "name" : "Jack", "average_stars" : 5 }
{ "user_id" : "b0JZW_hvGkVEIQVwhyqSuw", "name" : "Justin", "average_stars" : 5 }
{ "user_id" : "PASVfluTGQC5vpjd1-jinw", "name" : "Brian", "average_stars" : 5 }
{ "user_id" : "rH53dcL7YKjbtjyA45n0A", "name" : "Juliana", "average_stars" : 5 }
{ "user_id" : "8wxTBgvIKcGZrNfTc1ewQ", "name" : "mike", "average_stars" : 5 }
{ "user_id" : "ddhNJ-nbjwjHoStH6qXJ8g", "name" : "Ryan", "average_stars" : 5 }
```

Summary: In this part, we have considered a user having average\_stars = 5 as the most positive reviewer. Hence, we have used the find function to first filter out users that have an average\_stars of 5 and then display their user\_id, name and their average\_stars value. We have used limit() to limit the output to 10 users.

*ii. On the basis of average of stars in Review collection*

➤ `db.review.aggregate([ { $group: { _id: "$user_id", "review_avg": { $avg: "$stars" } } }, { $sort: { review_avg: -1 } }, { $limit: 10 } ], { allowDiskUse: true } )`

```
> db.review.aggregate([ { $group: { _id: "$user_id", "review_avg": { $avg: "$stars" } } }, { $sort: { review_avg: -1 } }, { $limit: 10 } ], { allowDiskUse: true } )
{ "_id" : "---94vtJ_5o_nikEs6hUjg", "review_avg" : 5 }
{ "_id" : "--32DuayFIUrx-cre2TGpQ", "review_avg" : 5 }
{ "_id" : "-4t44Ti0HQ2rhqsrtBTuQ", "review_avg" : 5 }
{ "_id" : "-0WZ5gk10fbUIodJuKfaQ", "review_avg" : 5 }
{ "_id" : "-26jc8nCJBy4-7r3ZtmiQ", "review_avg" : 5 }
{ "_id" : "--0sXNBv6IizZXuV-nl0Aw", "review_avg" : 5 }
{ "_id" : "--5-MQZPsn2CJkyMFQk0gQ", "review_avg" : 5 }
{ "_id" : "--4ww39MLTS1SBRmCrSmww", "review_avg" : 5 }
{ "_id" : "--44NNdtngXMzsyN7ju6Q", "review_avg" : 5 }
{ "_id" : "-1mPJZdSY9KluaBYAGboQ", "review_avg" : 5 }
```

Summary: Here, we have aggregated the review collection by grouping the documents based on user\_id and then calculating the average of the stars of each review of that user. With this group, we sorted them in descending order of the review average and then displayed the user\_id and their corresponding review average of those who fall in top 10 of the result.

## Purpose 2

### Technical review of data security with MongoDB

- **Introduction:**

MongoDB is a free and an open source document database written in C++ language which runs on various platforms including Amazon Linux, Debian, Ubuntu and Windows system. It is a NoSQL database and uses JSON-like documents to store information. MongoDB declared itself as the leading NoSQL database on basis of their statistics and is claimed as much faster than schema-oriented databases. Its usage is also very fast growing with millions of downloads.

As MongoDB is becoming so popular, its usage is being increased widely and more and more sensitive information is being stored in it. However, the integrity and confidentiality of information stored in it is not guaranteed. There are many inbuilt security issues with MongoDB and is even prone to injection attacks. Also, it was the victim of the recent ransomware attack which lead to exposure of over 680TB data in MongoDB databases. Thus, security issues are becoming major concerns with MongoDB database.

In this review, we discuss more about the data security with MongoDB, its security features and its comparison with relational database.

There are various types of security attacks which are given below:

- a) Unauthenticated or unauthorized access to the database data by intruders.
- b) Malware infections causing unauthorized access to data, deletion or editing the data stored in database.
- c) Data integrity not maintained.
- d) Data confidentiality and copyright or ownership preservation from intruders or attackers.
- e) Data editing, updating, deletion by the authenticated persons.

#### **DATA SECURITY: Comparison with Relational Database:**

**MongoDB does not use SQL, so it's not vulnerable to SQL injection.**

MongoDB avoids the potential for problems by not parsing.

Any API, anywhere, that involves encoding user data in formatted text that gets parsed has the potential for the caller and callee to disagree on how that text should be parsed. These disagreements can be security issues when data is misinterpreted as metadata. This is true whether you're talking about printf format strings, including user generated content in HTML, or generating SQL.

Why are you talking about APIs here?

Not quite accurate

Need clarification

Need to be specific and mention when

## How does MongoDB address SQL or Query injection?

How is that relevant to security?  
Need to clarify

Fortunately, you can express most queries in MongoDB without JavaScript and for queries that require JavaScript, you can mix JavaScript and non-JavaScript in a single query. Place all the user-supplied fields directly in a BSON field and pass JavaScript code to the \$where field.

If you need to pass user-supplied values in a \$where clause, you may escape these values with the CodeWScope mechanism. When you set user-submitted values as variables in the scope document, you can avoid evaluating them on the database server.

Since MongoDB doesn't parse structured text to figure out what to do, there is no possibility of misinterpreting user input as instructions, and hence no possible security hole.

**Conclusion:** NoSQL infers it's safer than RDBMS's which are vulnerable to SQL Injection – Therefore it's being deployed with this inference, assuming more security.

## Advanced Security features of MongoDB:

MongoDB, the leading NoSQL database, offers Enterprise Server, the commercial version of MongoDB with advanced security features. The Enterprise version meets strict security and compliance standards with Kerberos and LDAP authentication, Red Hat Identity Management Certification, and auditing.

With these advanced security features, you can defend, detect, and control access to your data. MongoDB's comprehensive security framework features:

- a) **Authentication** with integration with external security mechanisms including LDAP, Windows Active Directory, Kerberos and x.509 PKI certificates.
- b) **Authorization.** User-defined roles means you can configure granular permissions for a user or application, based on the privileges they need to do their job.
- c) **Auditing.** A native audit log lets you track access and operations performed on the database which works for regulatory compliance.
- d) **Encryption.** MongoDB data can be encrypted on the network and on disk. Protection of data at-rest is an integral feature within the database thanks to the introduction of MongoDB's Encrypted storage engine.

In relational databases there are various techniques to handle all above security threats. There is strong authentication and authorization techniques, data encryption techniques, audit log, database watermarking and many other techniques have been developed to handle security issues. But document-oriented databases are developing and they have lack of security techniques which can handle the security threats to the databases.

These statements are quite generic.

Some security issues and lacks in document-oriented databases like MongoDB are listed below:

- a) Data files are unencrypted and no method to encrypt automatically. All data in MongoDB is stored as plain text and there is no encryption mechanism to encrypt data files. This means that any malicious user with access to the file system can extract the information from the files.
- b) No method to protect data ownership and copyright protection. Like in relational databases watermarking is used for ownership and copyright protection, there should be such a technique which provides copyright preservation.
- c) Authentication is simple based on user name and password. In sharded mode MongoDB does not support authentication, but in standalone and replica set mode authentication can be enabled. So, there should be a certificate-based authentication for more security.

It uses SSL with X.509 certificates for secure communication between user and MongoDB cluster and intra-cluster authentication but it does not support authentication and authorization when running in sharded mode. The passwords are encrypted by MD5 hash algorithm and MD5 algorithm is not a very secure algorithm. Since mongo uses JavaScript as an internal scripting language, it is potential for scripting injection attack.

- d) MongoDB supports binary wire level protocol using port 27017. This protocol is the most efficient way to communicate with MongoDB. This port is neither encrypt nor compressed. Binary client port 28017 is used as HTTP server. The default distribution of MongoDB does not support SSL. It must be enabled separately. If any attacker finds any of these ports open then he has the access. So, the above security issues cause problems related to security in document-oriented databases which should be handled.
- e) MongoDB is not immune from injection attacks. As noted in the same documentation, injection attacks are still possible as MongoDB operations allow arbitrary JavaScript expressions to be executed directly on the server.
- f) To store information securely, a database needs to provide confidentiality, integrity and availability (CIA). Enterprise RDBMS databases provide CIA through integrated security features such as role-based security, encrypted communications, support for row and field access control, as well as access control through user-level permissions on stored procedures.
- g) RDBMS databases also have ACID (atomicity, consistency, isolation, durability) properties that guarantee database transactions are processed reliably; data replication and logging ensure durability and data integrity. These features increase the time it takes to retrieve large amounts of data, so they are not implemented in NoSQL databases.  
*How does that impact security?*
- h) NoSQL databases also lack confidentiality and integrity. As NoSQL databases don't have a schema, permissions on a table, column or row can't be segregated. This can also lead to



multiple copies of the same data. This can make it hard to keep data consistent, particularly as changes to multiple tables can't be wrapped in a transaction where a logical unit of insert, update or delete operations is executed.

- i) MongoDB may lose events if the server terminates before it commits the events to the audit log. The client may receive confirmation of the event before MongoDB commits to the audit log. For example, while auditing an aggregation operation, the server might crash after returning the result but before the audit log flushes.

Which ones? You were mostly listing disadvantages.

**Given the advantages of MongoDB, we can ensure hardening of security by following these steps:**

- Disabling the default status page – using the 'nohttpinterface' option to turn off the 28017 port.
- Use a different port – using the 'port' option
- Do not enable REST in production environments – don't use 'rest' option
- Bind the MongoDB process to only one interface/IP – using the 'bind\_ip'
- Don't run MongoDB daemon as root
- Disable anonymous access – using the 'auth' option
- Encrypt data - "To support audit requirements, you may need to encrypt data stored in MongoDB. For best results you can encrypt this data in the application layer, by encrypting the content of fields that hold secure data."
- Encrypt communication – Recommended to use SSL  
SSL helps ensure the security of your data over unsecured networks. If you employ a database that interacts with the internet, you should use SSL.

There are two very good reasons to use SSL to secure MongoDB: privacy and authentication.

Without SSL, your data can be accessed, copied, and used for illegal or harmful ends. With authentication, you have a secondary level of safety. SSL's private key infrastructure (PKI) guarantees that only users with the correct CA certificate can access MongoDB.

- **References:**

- a) Harpreet kaur et al, International Journal of Advanced Research in Computer Science, 4 (8), May–June 2013,227-228
- b) A Security Comparison between MySQL and MongoDB Omar Al-Ithawi
- c) A Survey on Security Issues in Big Data and NoSQL Ebrahim Sahafizadeh1, Mohammad Ali Nematbakhsh
- d) <https://www.trustwave.com/Resources/SpiderLabs-Blog/Mongodb---Security-Weaknesses-in-a-typical-NoSQL-database/>