Nicole Günther Guerrero

ID:158325

Hands-on MongoDB

Querying

Question 1: List all the restaurants in the collection, sorted in increasing order of names.

```
db.restaurants.find({"name":{$ne:""}},{"name": 1}).sort({"name": 1
})
```


```
{ _id: ObjectId("61ff1becc0fa9038a60603a9"),
  name: '#1 Garden Chinese' }
{ _id: ObjectId("61ff1bf0c0fa9038a6062bfd"),
  name: '#1 Me. Nick\'S' }
{ _id: ObjectId("61ff1be9c0fa9038a605f185"),
  name: '#1 Sabor Latino Restaurant' }
{ _id: ObjectId("61ff1bedc0fa9038a60611c2"),
  name: '$1.25 Pizza' }
```


Question 2: List all restaurants with "Italian" cuisine and display the name, zip code and geographic coordinates for each of them. Also, make sure the answer is ordered according to the sorting key (increasing postcode, decreasing name).

```
db.restaurants.find(
    {"cuisine" : "Italian"},  // Query
    {"name" : 1, "address.zipcode" : 1, "address.coord" : 1}// Pro
jection
).sort(
    {"address.zipcode" : 1, "name" : -1}  // Sorting
)
```


```
{ _id: ObjectId("61ff1be8c0fa9038a605e58f"),
  address: { coord: [ -73.9927809, 40.7451239 ], zipcode: '10001' },
  name: 'Tre Dici' }
{ _id: ObjectId("61ff1beec0fa9038a606165f"),
  address: { coord: [ -73.98952609999999, 40.7507917 ], zipcode: '10001' },
  name: 'Stella 34' }
{ _id: ObjectId("61ff1bebc0fa9038a606000d"),
  address: { coord: [ -73.990708, 40.750403 ], zipcode: '10001' },
  name: 'Spinelli\'S Pizza/Gyro Ii' }
{ _id: ObjectId("61ff1be4c0fa9038a605d295"),
  address: { coord: [ -73.99556, 40.748852 ], zipcode: '10001' },
  name: 'Salumeria Beillese/ Biricchino Rest' }
{ _id: ObjectId("61ff1be6c0fa9038a605da17"),
  address: { coord: [ -74.00370749999999, 40.7489719 ], zipcode: '10001' },
  name: 'Pepe Giallo' }
```

Question 3: List all Italian restaurants with the postcode "10075" for which the telephone number is provided in the database. Display their name, zip code and phone number.

```
***db.restaurants.find(
    {"cuisine" : "Italian", "address.zipcode": "10075"}, //Filter
the Italian restaurants and the postcode
    {"name" : 1, "address.zipcode" : 1} //Display the name and zip
code
)
```


```
{ _id: ObjectId("61ff1be4c0fa9038a605d193"),
  address: { zipcode: '10075' },
  name: 'Don Filippo Restaurant' }
{ _id: ObjectId("61ff1be4c0fa9038a605d1e1"),
  address: { zipcode: '10075' },
  name: 'Lusardi\'S Restaurant' }
{ _id: ObjectId("61ff1be4c0fa9038a605d2c0"),
  address: { zipcode: '10075' },
  name: 'Due' }
{ _id: ObjectId("61ff1be5c0fa9038a605d74f"),
  address: { zipcode: '10075' },
  name: 'Serafina Fabulous Pizza' }
```


Question 4: Find all restaurants with at least a score of 50.

```
db.Restaurants.find (
                    { "grades.score" : { $gte: 50 } },
                    { "name" : 1, "grades.score" : 1 }
)
```


```
{ _id: ObjectId("61ff1be4c0fa9038a605d198"),
  grades:
   [ { score: 11 },
     { score: 131 },
     { score: 11 },
     { score: 25 },
     { score: 11 },
     { score: 13 } ],
  name: 'Murals On 54/Randolphs\'S' }
```


Question 5: List all restaurants that are either Italian or have the postcode "10075".

```
db.restaurants.find( {$or: [{"cuisine": "Italian"}, {"address.zipc
ode": "10075"}]})
```

```
{ _id: ObjectId("61ff1be4c0fa9038a605d072"),
  address:
   { building: '10004',
     coord: [ -74.03400479999999, 40.6127077 ],
     street: '4 Avenue',
     zipcode: '11209' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-02-25T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-06-27T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2012-12-03T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-11-09T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'Philadelhia Grille Express',
  restaurant_id: '40364305' }
```


Question 6: List all restaurants with a zip code of "10075" or "10098" with either Italian or American cuisine with a score of at least 50.

```
db.restaurants.find( {$and: [ {$or: [ {"address.zipcode" : "10075"
}, {"address.zipcode" : "10075"} ]}, {$or: [ {"cuisine" : "America
n"}, {"cuisine" : "italian"} ]}, {"grades.score" : {$gte : 50}}]})
```


```
{ _id: ObjectId("61ff1be8c0fa9038a605e580"),
  address:
   { building: '1462',
     coord: [ -73.953769, 40.77037 ],
     street: '1 Avenue',
     zipcode: '10075' },
  cuisine: 'American',
  grades:
   [ { date: 2015-01-16T00:00:00.000Z, grade: 'Z', score: 19 },
     { date: 2014-09-02T00:00:00.000Z, grade: 'C', score: 57 },
     { date: 2014-03-20T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-09-12T00:00:00.000Z, grade: 'B', score: 21 },
     { date: 2013-04-05T00:00:00.000Z, grade: 'B', score: 15 },
     { date: 2012-10-22T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2012-05-21T00:00:00.000Z, grade: 'B', score: 25 } ],
  name: 'Three Star Diner',
  restaurant_id: '41097286' }
```


Question 7: List all restaurants with at least one score concerning customer service *(C)*, *price (P)* or *quality (Q)*. Simply display the names, cuisine and zip code.

```
db.restaurants.find(
{ $or : [{"grades.grade" : "C"}, {"grades.grade" : "P"}, {"grades.
grade" : "Q"}]},
{"name" : 1, "cuisine" : 1, "address.zipcode" : 1})
```

Question 8: Change the type of cuisine in the restaurant "Juni" to "American (new)". In addition, record the date and time of the system in a "lastModified" field at the time the change is made. If there are several restaurants with the same name, only the first one must be modified.

```
db.restaurants.update( { "name": "juni" },
   {
    $set: { cousine: "American (New)" },
    $currentDate: { "lastModified": true }},
   {multi: true})
```

```
{ acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0 }
```

Question 9: Change the address of the restaurant whose id is "41156888" to "East 31st Street".

```
db.restaurants.update({id:"41156888"}, {$set: {address: "East 31st
Street"}});
```

```
{ acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0 }
```

Question 10: Change the type of cuisine of all restaurants with a postcode "10016" and the type of cuisine "Other" into "Cuisine to be determined". In addition, record the date and time of the system in a "lastModified" field at the time the change is made.

```
db.restaurants.update({ "address.zipcode": "10016", cuisine: "Othe
r" },
{
   $set: { cuisine: "Cuisine to be determined" },
   $currentDate: { "lastModified": true }},
```

```
  {multi: true}
)
```

```
{ acknowledged: true,
  insertedId: null,
  matchedCount: 20,
  modifiedCount: 20,
  upsertedCount: 0 }
```

Question 11: Replace all information about the restaurant with an ID "41154403" with the following information:

```
"name" : "Vella 2",
"address" : {

   "city" : "1480",
   "street" : "2 Avenue",
   "zipcode" : "10075"
}
```

```
db.restaurants.updateOne({restaurant_id: "41154403"},
       {$set : {"name" : "Vella 2", "address" : {"city" : "1480" ,
"street" : "2 Avenue", "zipcode" : "10075"}}});
```

```
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
```

Question 12: List the types of cuisine represented in the database. For each one, display the number of associated restaurants. Order the result by decreasing number of restaurants.

```
db.restaurants.aggregate([{ $group:
       {
              _id: "$cuisine",
              restaurants: {$sum: 1}}},
       {$sort: { restaurants: -1}}
])
```

```
{ _id: 'American', restaurants: 6181 }
{ _id: 'Chinese', restaurants: 2417 }
{ _id: 'Café/Coffee/Tea', restaurants: 1214 }
{ _id: 'Pizza', restaurants: 1162 }
{ _id: 'Italian', restaurants: 1070 }
{ _id: 'Other', restaurants: 990 }
{ _id: 'Latin (Cuban, Dominican, Puerto Rican, South & Central American)',
  restaurants: 854 }
{ _id: 'Japanese', restaurants: 759 }
{ _id: 'Mexican', restaurants: 756 }
{ _id: 'Bakery', restaurants: 691 }
{ _id: 'Caribbean', restaurants: 656 }
{ _id: 'Spanish', restaurants: 637 }
{ _id: 'Donuts', restaurants: 477 }
{ _id: 'Pizza/Italian', restaurants: 468 }
{ _id: 'Sandwiches', restaurants: 459 }
{ _id: 'Hamburgers', restaurants: 433 }
{ _id: 'Chicken', restaurants: 410 }
{ _id: 'Ice Cream, Gelato, Yogurt, Ices', restaurants: 348 }
{ _id: 'French', restaurants: 344 }
{ _id: 'Delicatessen', restaurants: 321 }
```

Question 13: Show, for each zip code, the number of Italian restaurants with this zip code. Order the result by decreasing number of restaurants.

```
db.restaurants.aggregate( [{$match: {"cuisine" : "Italian"}},
    {$group: {_id: "$address.zipcode", "cuisines": { $sum : 1}}},
    {$sort: "cuisines":-1}}
] )
```

```
{ _id: '10013', cuisines: 51 }
{ _id: '10012', cuisines: 45 }
{ _id: '10014', cuisines: 43 }
{ _id: '10019', cuisines: 42 }
{ _id: '10003', cuisines: 40 }
{ _id: '10011', cuisines: 37 }
{ _id: '10036', cuisines: 35 }
{ _id: '10022', cuisines: 32 }
{ _id: '10028', cuisines: 22 }
{ _id: '11215', cuisines: 22 }
{ _id: '10016', cuisines: 21 }
{ _id: '10017', cuisines: 20 }
{ _id: '10021', cuisines: 20 }
{ _id: '11201', cuisines: 19 }
{ _id: '10024', cuisines: 19 }
{ _id: '10023', cuisines: 18 }
{ _id: '10458', cuisines: 17 }
{ _id: '10065', cuisines: 17 }
{ _id: '10010', cuisines: 17 }
{ _id: '10009', cuisines: 17 }
```

Question 14: Consider Italian restaurants whose identifier (restaurant_id) is higher or equal to "41205309" and has an "averagePrice" attribute. Calculate the average of these average prices. Then repeat the same operation by calculating the average by zip code.

```
db.restaurants.aggregate (
    {$match: {"cuisine" : "Italian"}}, //Filter the Italian restau
rants
    {
      $group: {
```

```
                _id: {$gte: ["restaurant.id", 41205309]},
                averagePrice: {$avg: "$averagePrice"},
                zipcode: {$avg: "$zipcode"}
            }
    })
        { _id: true, averagePrice: 30.625, zipcode: null }
```

Question 15: Create a new collection called "comments" in the same database.

```
db.createCollection("comments")
```

```
{ ok: 1 }
```

Question 16: Insert three documents into the previously created collection, using the following pattern:

```
{
  "_id" : "----",
  "restaurant_id" : "----",
  "client_id" : "----",
  "comment" : "----",
  "date" : ISODate("----"),
  "type" : "----"
}
```

Some useful details:

- Restaurant identifiers must match existing restaurants in the collection `restaurants`.
- You need to provide feedback from different customers, and for different `restaurants`.
- The type `attribute` can take only the "positive" or "negative" values.

```
db.comments.insertMany([
    {restaurant_id: "30075445", client_id: "124781426", comment: "
Excellent service!", date: new Date("2014-12-03"), type: "positive
"},
    {restaurant_id: "30112340", client_id: "475732110", comment: "
The best food I ever had!", date: new Date("2015-01-21"), type: "p
ositive"},
    {restaurant_id: "30191841", client_id: "626141589", comment: "
I hated it, never returning.", date: new Date("2020-05-13"), type:
"negative"}
])
```

```
{ acknowledged: true,
  insertedIds:
   { '0': ObjectId("620d4dd8dc005c496f52d58c"),
     '1': ObjectId("620d4dd8dc005c496f52d58d"),
     '2': ObjectId("620d4dd8dc005c496f52d58e") } }
```

Question 17: List all the comments in your database. Each comment must also contain all the information about the restaurant to which it relates.

```
db.restaurants.aggregate([{
$lookup:{
       from: "comments",
       localField: "restaurant_id",
       foreignField: "restaurant_id",
       as: "restaurant_comments"}},
{$match: {"restaurant_comments": {$ne: []}}},
{$project: { restaurant_comments:{comment: 1}, name: 1, cuisine: 1
, restaurant_id: 1, address: 1}},
])
```

```
{ _id: ObjectId("61ff1be4c0fa9038a605d03b"),
  address:
   { building: '469',
     coord: [ -73.961704, 40.662942 ],
     street: 'Flatbush Avenue',
     zipcode: '11225' },
  cuisine: 'Hamburgers',
  name: 'Wendy\'S',
  restaurant_id: '30112340',
  restaurant_comments: [ { comment: 'The best food I ever had!' } ] }
{ _id: ObjectId("61ff1be4c0fa9038a605d03a"),
  address:
   { building: '1007',
     coord: [ -73.856077, 40.848447 ],
     street: 'Morris Park Ave',
     zipcode: '10462' },
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445',
  restaurant_comments: [ { comment: 'Excellent service!' } ] }
```

Question 18: Insert seven other documents into the comments collection, following the pattern described above, and following these rules:

- Restaurant identifiers must match existing restaurants in the restaurants collection.
- At least one of the restaurants must have several comments.
- At least one of the customers must have commented several times.
- At least one of the customers must have commented several times on the same restaurant.
- The type attribute can take only the "positive" or "negative" values.

```
db.comments.insertMany([
```

```
    {restaurant_id: "40367797", client_id: "475732110", comment: "
Good service!", date: new Date("2015-09-03"), type: "positive"},
    {restaurant_id: "40367797", client_id: "475732110", comment: "
The food was cold.", date: new Date("2015-10-03"), type: "negative
"},
    {restaurant_id: "40367797", client_id: "626141589", comment: "
Excellent service!", date: new Date("2015-11-03"), type: "positive
"},
    {restaurant_id: "40705931", client_id: "124781426", comment: "
There was a hair on my food.", date: new Date("2019-02-13"), type:
"negative"},
    {restaurant_id: "40705931", client_id: "475732110", comment: "
Love the food.", date: new Date("2021-03-13"), type: "positive"},
    {restaurant_id: "40705931", client_id: "475732110", comment: "
My tea was nice.", date: new Date("2022-04-13"), type: "positive"}
,
    {restaurant_id: "40705931", client_id: "475732110", comment: "
Love it, will come back.", date: new Date("2019-05-13"), type: "po
sitive"}
])
```

```
{ acknowledged: true,
  insertedIds:
   { '0': ObjectId("620e968c14857b5d95bfe00d"),
     '1': ObjectId("620e968c14857b5d95bfe00e"),
     '2': ObjectId("620e968c14857b5d95bfe00f"),
     '3': ObjectId("620e968c14857b5d95bfe010"),
     '4': ObjectId("620e968c14857b5d95bfe011"),
     '5': ObjectId("620e968c14857b5d95bfe012"),
     '6': ObjectId("620e968c14857b5d95bfe013") } }
```

Question 19: Find the list of restaurants with reviews and display only the name and comment list for each restaurant. Several strategies are possible.

```
db.restaurants.aggregate([
   {$lookup:
   {from: "comments", localField: "restaurant_id",
       foreignField: "restaurant_id",
       as: "restaurants_n_comments"}},
```

```
  {$unwind: "$restaurants_n_comments"},
  {$match: {"restaurants_n_comments.comment": { $exists: true, $ne
: ""}},
  {$project:
      {"name": 1, comments": "$restaurants_n_comments.comment"}}
])
```

```
{ _id: ObjectId("61ff1be4c0fa9038a605d03b"),
  name: 'Wendy\'S',
  comments: 'The best food I ever had!' }
{ _id: ObjectId("61ff1be4c0fa9038a605d03a"),
  name: 'Morris Park Bake Shop',
  comments: 'Excellent service!' }
{ _id: ObjectId("61ff1be4c0fa9038a605d03c"),
  name: 'Dj Reynolds Pub And Restaurant',
  comments: 'I hated it, never returning.' }
```

Question 20: Create an increasing index on the cuisine attribute of the restaurants collection.

```
db.restaurants.createIndex({cuisine:1})
```

```
< 'cuisine_1'
```

Question 21: Create another index for the restaurants collection, consisting of the cuisine attribute (increasing) and the zipcode attribute (decreasing).

```
db.restaurants.createIndex({cuisine:1, zipcode: -1})
```

```
'cuisine_1_zipcode_-1'
```

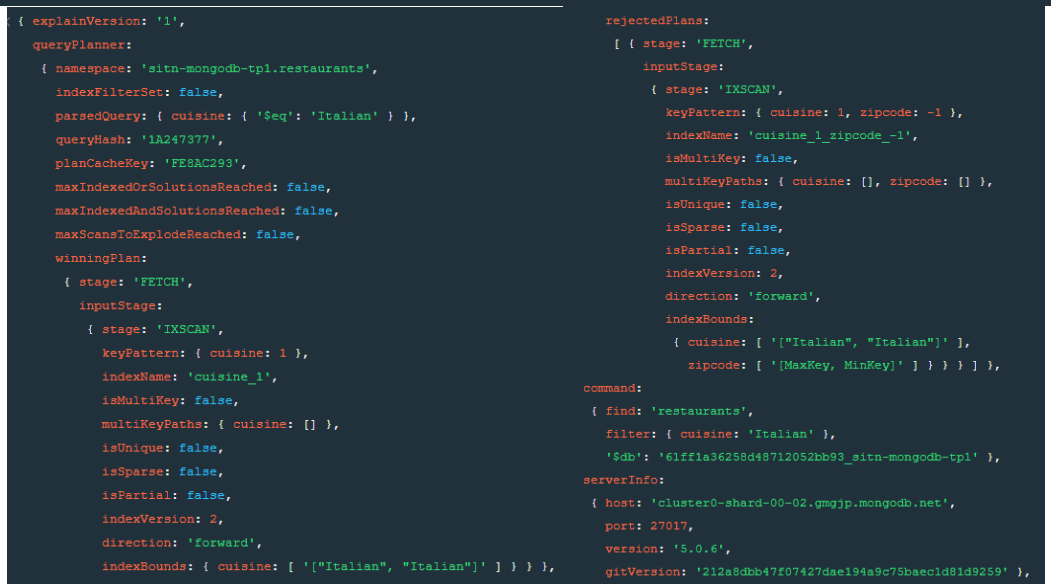
Question 22: List all indexes created on the restaurants collection.

```
db.restaurants.getIndexes({"listindexes": "restaurants"})
```

```
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { cuisine: 1 }, name: 'cuisine_1' },
  {
    v: 2,
    key: { cuisine: 1, zipcode: -1 },
    name: 'cuisine_1_zipcode_-1'
  }
]
```

Question 23: Use the explain method to display the execution plan for the query that returns all Italian restaurants. What information is provided by the system?

```
db.restaurants.explain().find({"cuisine" : "Italian"})
```




```
{ explainVersion: '1',
  queryPlanner:
   { namespace: 'sitn-mongodb-tp1.restaurants',
     indexFilterSet: false,
     parsedQuery: { cuisine: { '$eq': 'Italian' } },
     queryHash: '1A247377',
     planCacheKey: 'FE8AC293',
     maxIndexedOrSolutionsReached: false,
     maxIndexedAndSolutionsReached: false,
     maxScansToExplodeReached: false,
     winningPlan:
      { stage: 'FETCH',
        inputStage:
         { stage: 'IXSCAN',
           keyPattern: { cuisine: 1 },
           indexName: 'cuisine_1',
           isMultiKey: false,
           multiKeyPaths: { cuisine: [] },
           isUnique: false,
           isSparse: false,
           isPartial: false,
           indexVersion: 2,
           direction: 'forward',
           indexBounds: { cuisine: [ '["Italian", "Italian"]' ] } } },
     rejectedPlans:
      [ { stage: 'FETCH',
          inputStage:
           { stage: 'IXSCAN',
             keyPattern: { cuisine: 1, zipcode: -1 },
             indexName: 'cuisine_1_zipcode_-1',
             isMultiKey: false,
             multiKeyPaths: { cuisine: [], zipcode: [] },
             isUnique: false,
             isSparse: false,
             isPartial: false,
             indexVersion: 2,
             direction: 'forward',
             indexBounds:
              { cuisine: [ '["Italian", "Italian"]' ],
                zipcode: [ '[MaxKey, MinKey]' ] } } } ] },
  command:
   { find: 'restaurants',
     filter: { cuisine: 'Italian' },
     '$db': '61ff1a36258d48712052bb93_sitn-mongodb-tp1' },
  serverInfo:
   { host: 'cluster0-shard-00-02.gmgjp.mongodb.net',
     port: 27017,
     version: '5.0.6',
     gitVersion: '212a8dbb47f07427dae194a9c75baec1d81d9259' },
  serverParameters:
   { internalQueryFacetBufferSizeBytes: 104857600,
     internalQueryFacetMaxOutputDocSizeBytes: 104857600,
     internalLookupStageIntermediateDocumentMaxSizeBytes: 16793600,
     internalDocumentSourceGroupMaxMemoryBytes: 104857600,
     internalQueryMaxBlockingSortMemoryUsageBytes: 33554432,
     internalQueryProhibitBlockingMergeOnMongoS: 0,
     internalQueryMaxAddToSetBytes: 104857600,
     internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600 },
  ok: 1,
  '$clusterTime':
   { clusterTime: Timestamp({ t: 1645222778, i: 6 }),
     signature:
      { hash: Binary(Buffer.from("1a196ac880ce59e54766772bc00ba7bf70b4b057", "hex"), 0),
        keyId: 7027133973104951000 } },
  operationTime: Timestamp({ t: 1645222778, i: 6 }) }
```


Question 24: Same question but adding the parameter "executionStats" in the explain method.

```
db.restaurants.explain("executionStats").find()
```

```
executionStats:
 { executionSuccess: true,
   nReturned: 25356,
   executionTimeMillis: 21,
   totalKeysExamined: 0,
   totalDocsExamined: 25356,
   executionStages:
    { stage: 'COLLSCAN',
      nReturned: 25356,
      executionTimeMillisEstimate: 4,
      works: 25358,
      advanced: 25356,
      needTime: 1,
      needYield: 0,
      saveState: 25,
      restoreState: 25,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 25356 } },
```

Question 25: Drop the two indexes you previously created, and then re-display the statistics on the query execution plan that returns all Italian restaurants. What do you see?

```
db.restaurants.dropIndexes("cuisine_1", "cuisine_1_zipcode_-1")
```

```
{
  nIndexesWas: 3,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1645223691, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("5769eb4b933405863606950b14cc69b56a8a426a", "hex"), 0),
      keyId: Long("7027133973104951299")
    }
  },
  operationTime: Timestamp({ t: 1645223691, i: 1 })
}
```