MongoDB

mongodb

curl -fsSL https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list

导入数据集：

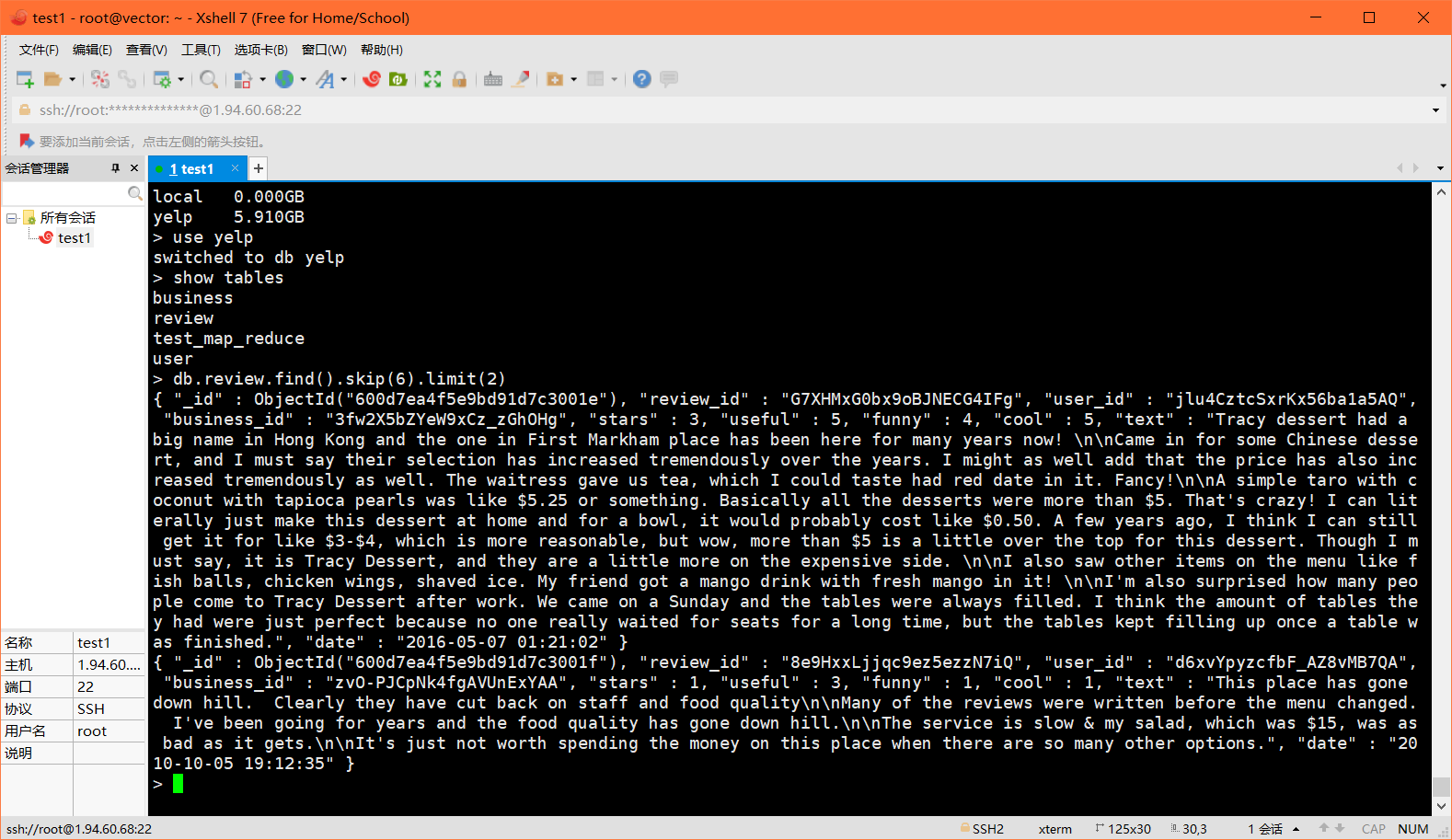
mongorestore -h localhost -d yelp –dir mongo

启动mongo服务：

mongod --dbpath /var/lib/mongodb/ --logpath /var/log/mongodb/mongodb.log --logappend &

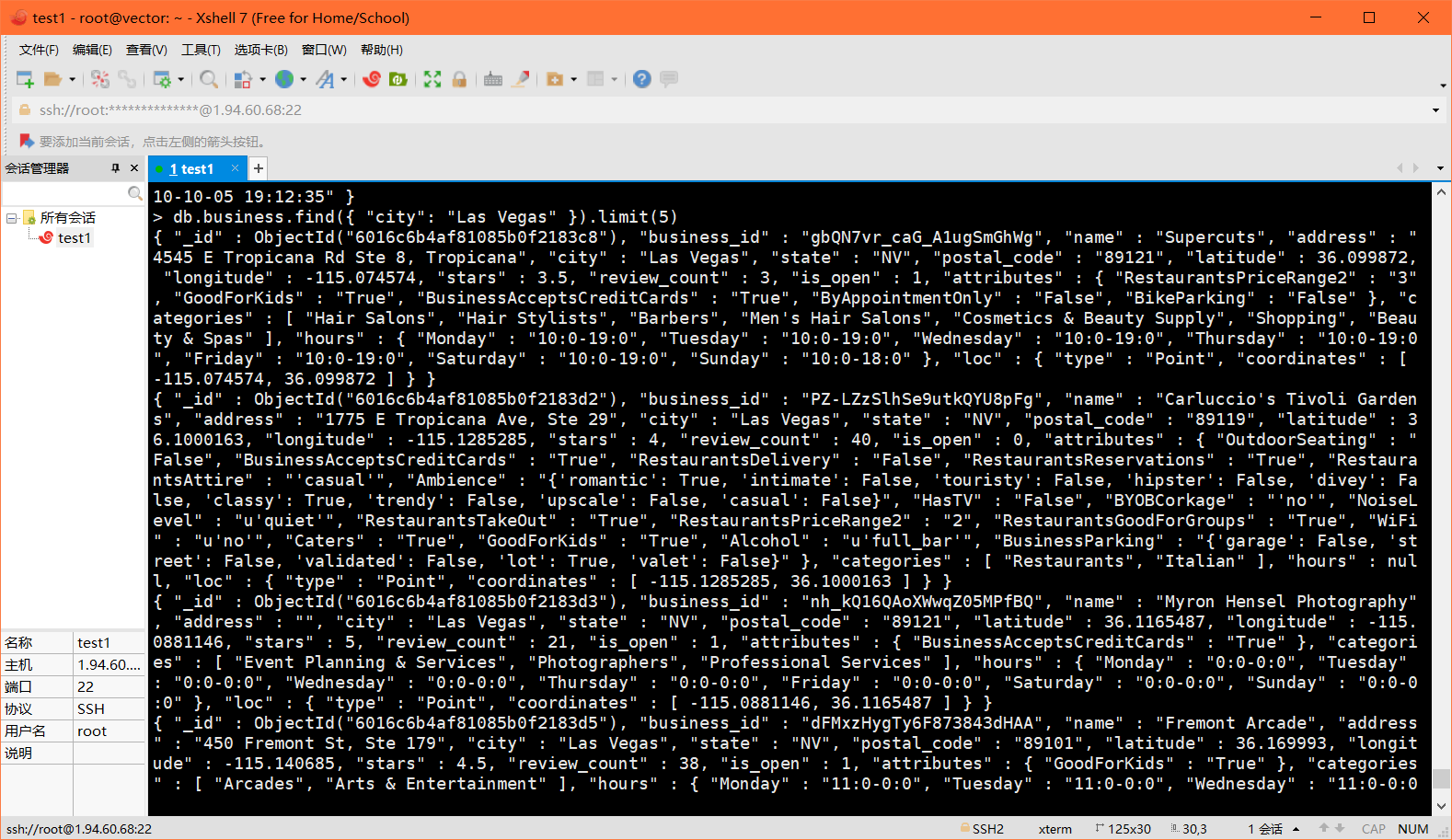
输入mongo即可启动

1.查询review集合的2条数据，跳过前6条。db.review.find().skip(6).limit(2)



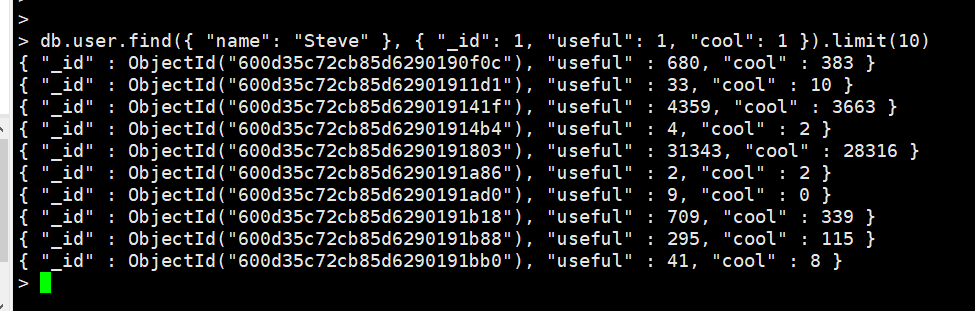
2.查询business集合中city是Las Vegas的5条数据。

db.business.find({ "city": "Las Vegas" }).limit(5)



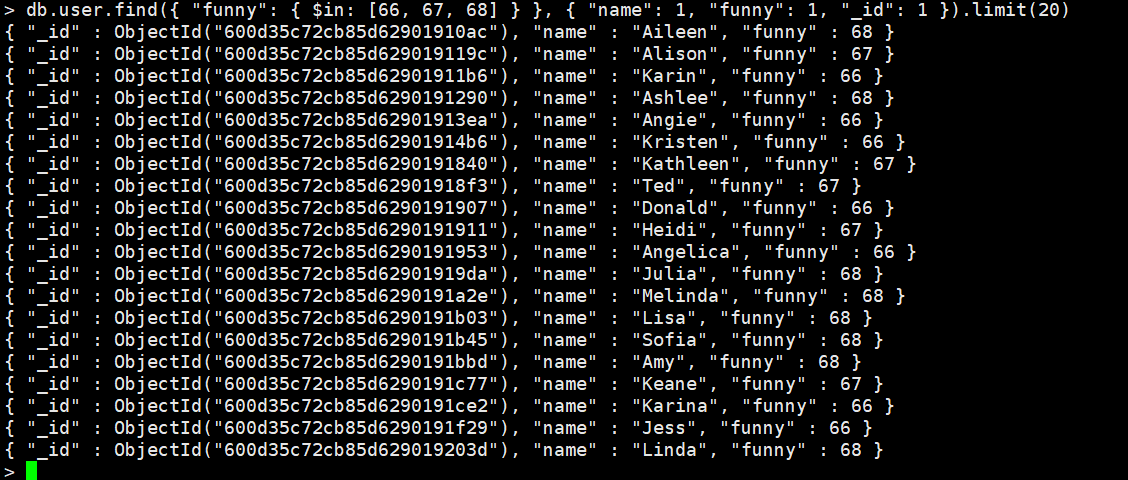
3. 查询user集合中name是Steve的user，只需要返回useful和cool,限制10条数据。

db.user.find({ "name": "Steve" }, { "\_id": 1, "useful": 1, "cool": 1 }).limit(10)



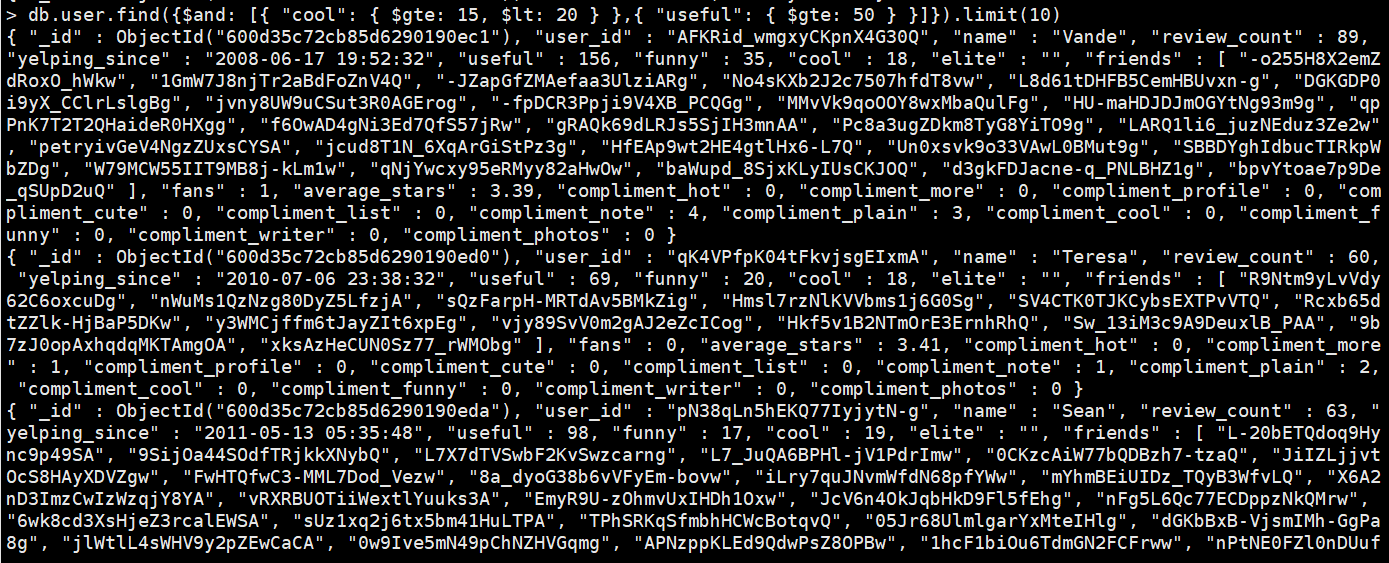
4.查询user集合中funny位于[66，67，68]的user，只需返回name和funny，限 制20条数据。

db.user.find({ "funny": { $in: [66, 67, 68] } }, { "name": 1, "funny": 1, "\_id": 1 }).limit(20)



5.查询user集合中15≤cool

db.user.find({$and: [{ "cool": { $gte: 15, $lt: 20 } },{ "useful": { $gte: 50 } }]}).limit(10)



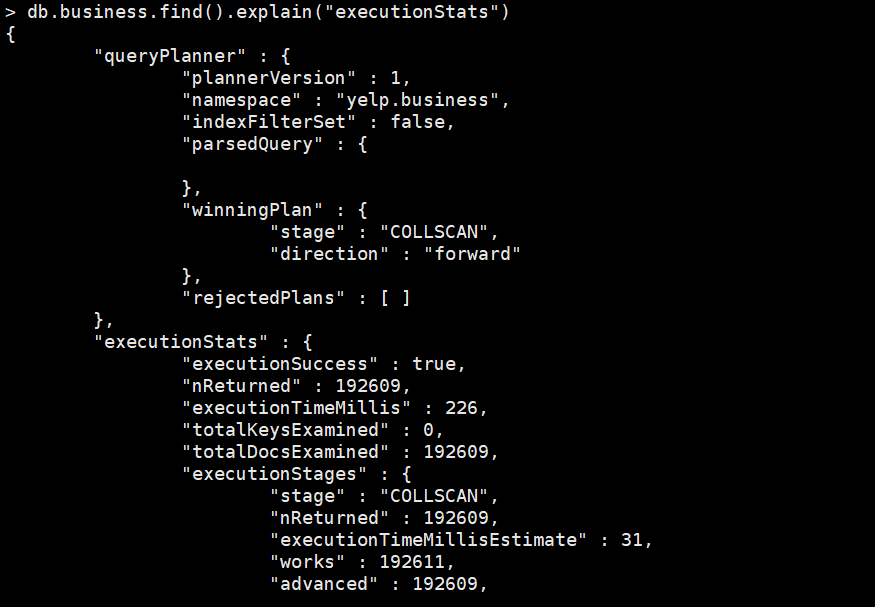
6. 统计business一共有多少条数据，并使用explain查询执行计划，了解MongoDB对集函数的执行方式。

db.business.count()



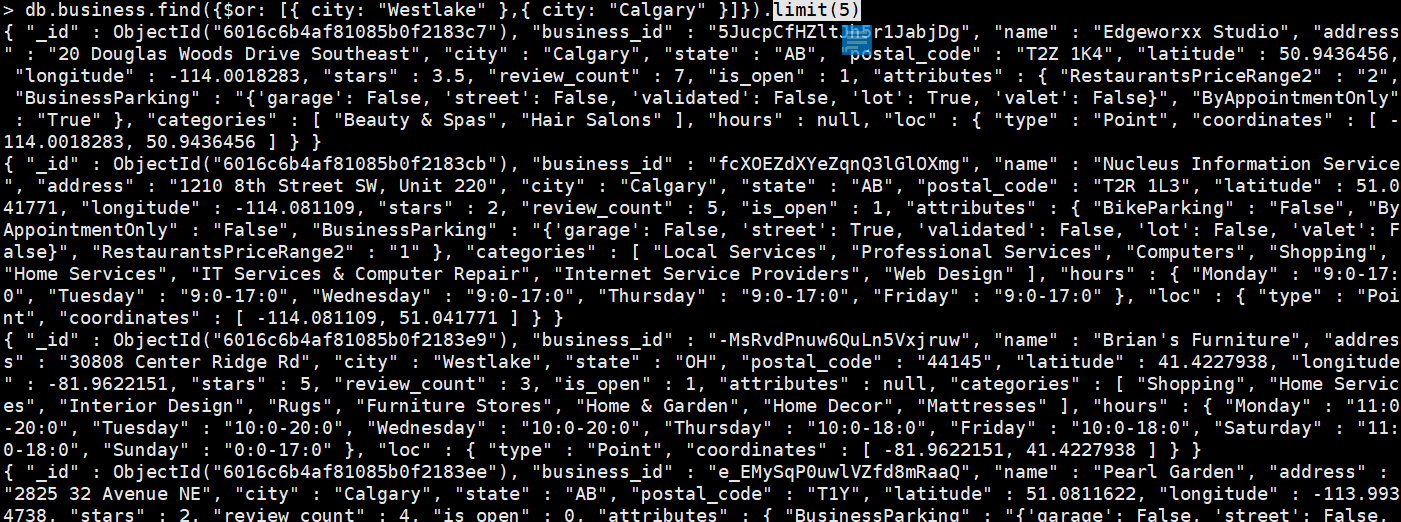
db.business.find().explain("executionStats")

根据提供的执行计划信息，可以看出查询db.business.find().explain("executionStats")使用了COLLSCAN（全表扫描）的方式来执行查询，因为totalKeysExamined和totalDocsExamined的值都是非常接近文档总数（192609），这表明它在没有使用索引的情况下对整个集合进行了扫描。虽然这个查询成功执行，但全表扫描通常不是最有效的方式，特别是对于大型集合。为了提高查询性能，你可以考虑以下几种方式：索引优化：为经常查询的字段创建索引，以减少扫描的文档数量。在business集合中，可能需要为一些经常作为查询条件的字段创建索引，比如city或其他常见过滤条件。数据分区：如果数据量很大，可以考虑将数据分区，以减少单个查询需要扫描的文档数量。使用适当的查询条件：确保查询条件是准确的，以避免不必要的全表扫描。升级硬件和优化配置：如果可能，可以考虑升级硬件或优化MongoDB的配置，以提高性能。



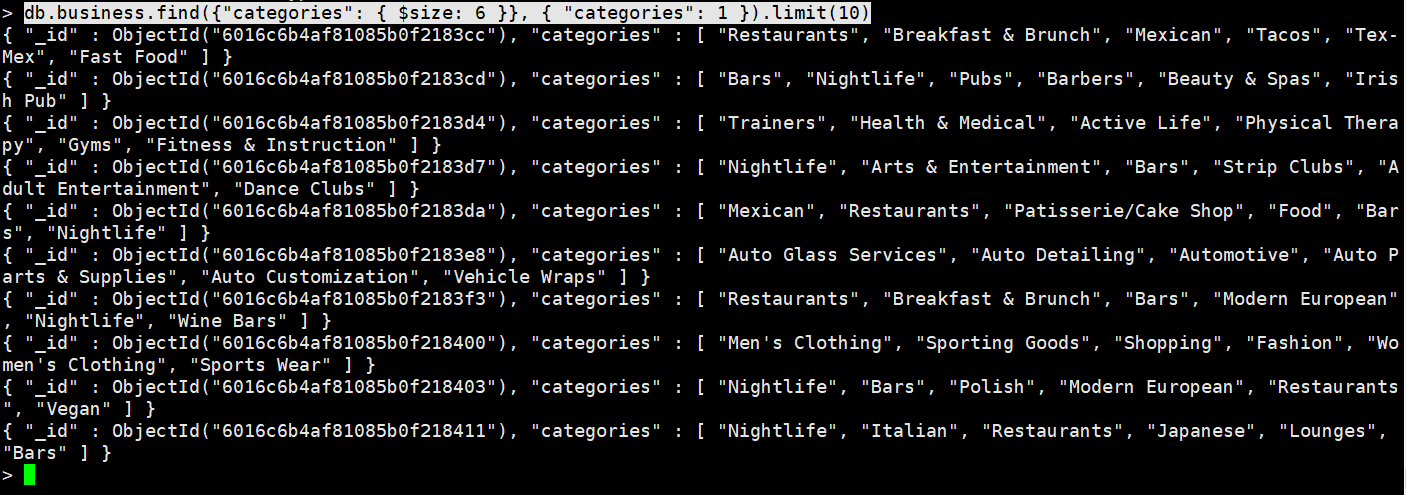
7. 查询business集合city为Westlake或者Calgary的数据。

db.business.find({$or: [{ city: "Westlake" },{ city: "Calgary" }]})



8.查询business集合中，类别为6种的商户信息，显示这6种类别，限制10条。

db.business.find({"categories": { $size: 6 }}, { "categories": 1 }).limit(10)



9.使用explain看 db.business.find({business\_id: "5JucpCfHZltJh5r1JabjDg"}) 的执行计划，了解该查询的执行计划及查询执 行时间，并给出物理优化手段，以提高查询性能，通过优化前后的性能对比展现优 化程度。

**建立索引之前**：db.business.find({ business\_id: "5JucpCfHZltJh5r1JabjDg" }).explain("executionStats")：

结果：

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "yelp.business",

"indexFilterSet" : false,

"parsedQuery" : {

"business\_id" : {

"$eq" : "5JucpCfHZltJh5r1JabjDg"

}

},

"winningPlan" : {

"stage" : "COLLSCAN",

"filter" : {

"business\_id" : {

"$eq" : "5JucpCfHZltJh5r1JabjDg"

}

},

"direction" : "forward"

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 1,

"executionTimeMillis" : 99,

"totalKeysExamined" : 0,

"totalDocsExamined" : 192609,

"executionStages" : {

"stage" : "COLLSCAN",

"filter" : {

"business\_id" : {

"$eq" : "5JucpCfHZltJh5r1JabjDg"

}

},

"nReturned" : 1,

"executionTimeMillisEstimate" : 8,

"works" : 192611,

"advanced" : 1,

"needTime" : 192609,

"needYield" : 0,

"saveState" : 192,

"restoreState" : 192,

"isEOF" : 1,

"direction" : "forward",

"docsExamined" : 192609

}

},

"serverInfo" : {

"host" : "vector",

"port" : 27017,

"version" : "4.4.25",

"gitVersion" : "3e18c4c56048ddf22a6872edc111b542521ad1d5"

},

"ok" : 1

}

**建立索引后：db.business.createIndex({ business\_id: 1 })**

**结果：**

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

db.business.find({ business\_id: "5JucpCfHZltJh5r1JabjDg" }).explain("executionStats")：

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "yelp.business",

"indexFilterSet" : false,

"parsedQuery" : {

"business\_id" : {

"$eq" : "5JucpCfHZltJh5r1JabjDg"

}

},

"winningPlan" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"business\_id" : 1

},

"indexName" : "business\_id\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"business\_id" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"business\_id" : [

"[\"5JucpCfHZltJh5r1JabjDg\", \"5JucpCfHZltJh5r1JabjDg\"]"

]

}

}

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 1,

"executionTimeMillis" : 2,

"totalKeysExamined" : 1,

"totalDocsExamined" : 1,

"executionStages" : {

"stage" : "FETCH",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"docsExamined" : 1,

"alreadyHasObj" : 0,

"inputStage" : {

"stage" : "IXSCAN",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"keyPattern" : {

"business\_id" : 1

},

"indexName" : "business\_id\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"business\_id" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"business\_id" : [

"[\"5JucpCfHZltJh5r1JabjDg\", \"5JucpCfHZltJh5r1JabjDg\"]"

]

},

"keysExamined" : 1,

"seeks" : 1,

"dupsTested" : 0,

"dupsDropped" : 0

}

}

},

"serverInfo" : {

"host" : "vector",

"port" : 27017,

"version" : "4.4.25",

"gitVersion" : "3e18c4c56048ddf22a6872edc111b542521ad1d5"

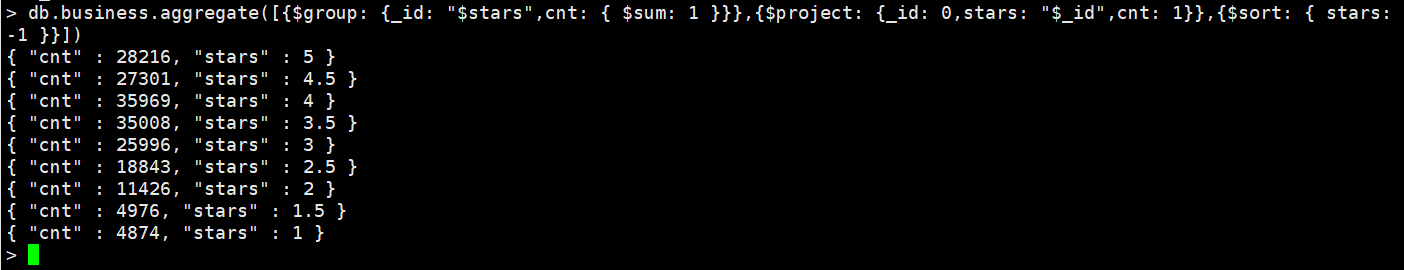
},

"ok" : 1

}

10. 统计各个星级的商店的个数，返回星级数和商家总数，按照星级降序排列。

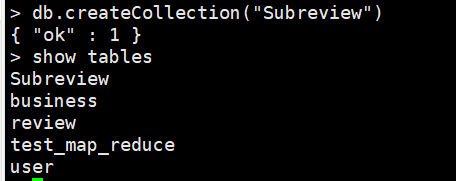
db.business.aggregate([{$group: {\_id: "$stars",cnt: { $sum: 1 }}},{$project: {\_id: 0,stars: "$\_id",cnt: 1}},{$sort: { stars: -1 }}])



11.创建一个review的子集合Subreview(取review的前五十万条数据)，分别对评 论的内容建立全文索引，对useful建立升序索引，然后查询评价的内容中包含关键 词delicious且useful大于9的评价。

// 创建Subreview子集合

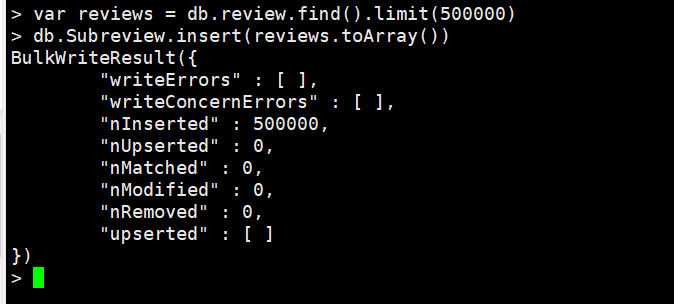
db.createCollection("Subreview")



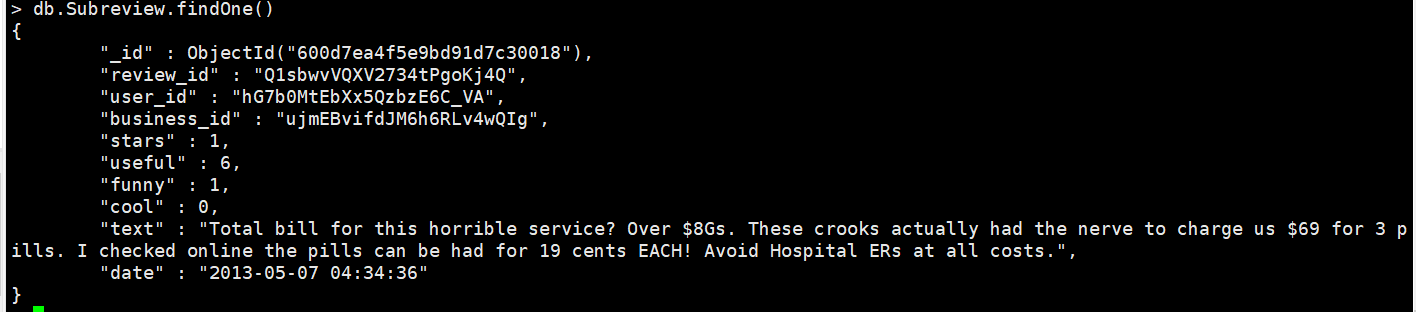
// 导入前五十万条review数据

var reviews = db.review.find().limit(500000)

db.Subreview.insert(reviews.toArray())

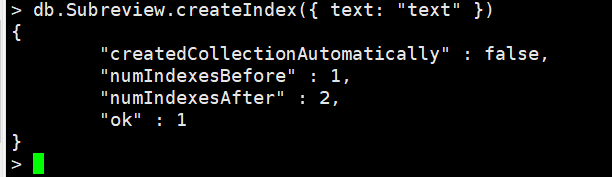


**db.Subreview.findOne()进行检查**

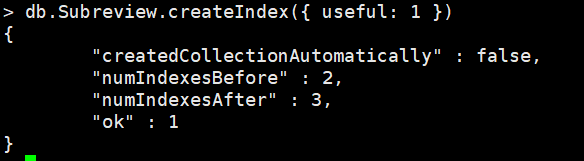


建立索引：

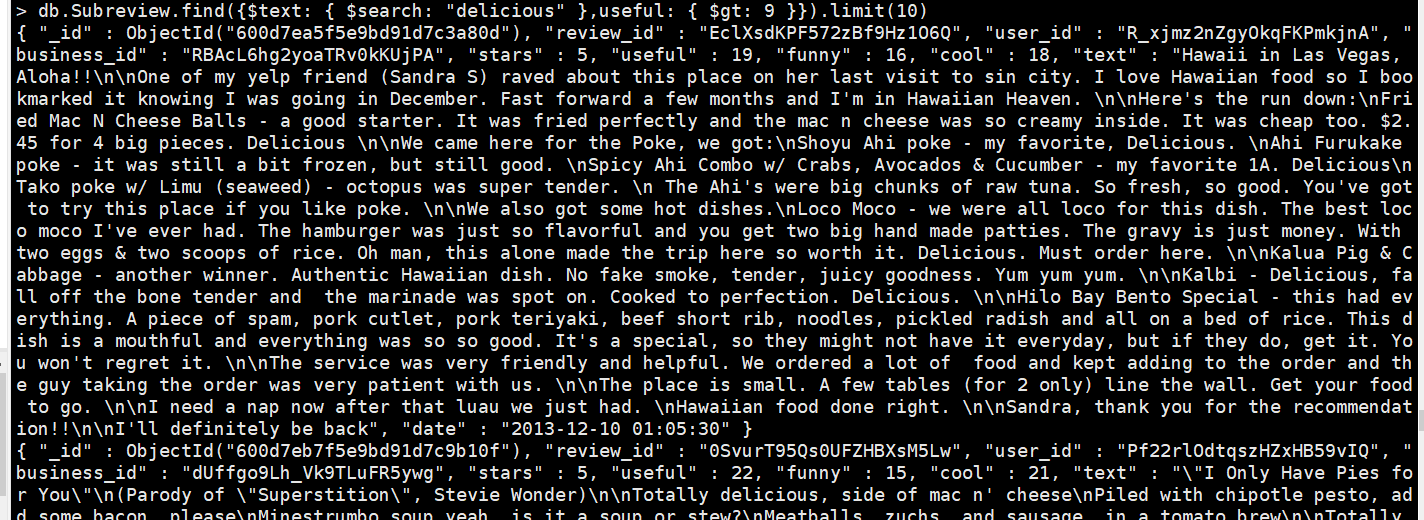
db.Subreview.createIndex({ text: "text" })



db.Subreview.createIndex({ useful: 1 })



查询：db.Subreview.find({$text: { $search: "delicious" },useful: { $gt: 9 }}).limit(10)



12.在Subreview集合中统计评价中useful、funny和cool都大于6的商家，返回商 家id及平均打星，并按商家id降序排列。

db.Subreview.aggregate([

{

$match: {

useful: { $gt: 6 },

funny: { $gt: 6 },

cool: { $gt: 6 }

}

},

{

$group: {

\_id: "$business\_id",

avg\_stars: { $avg: "$stars" }

}

},

{

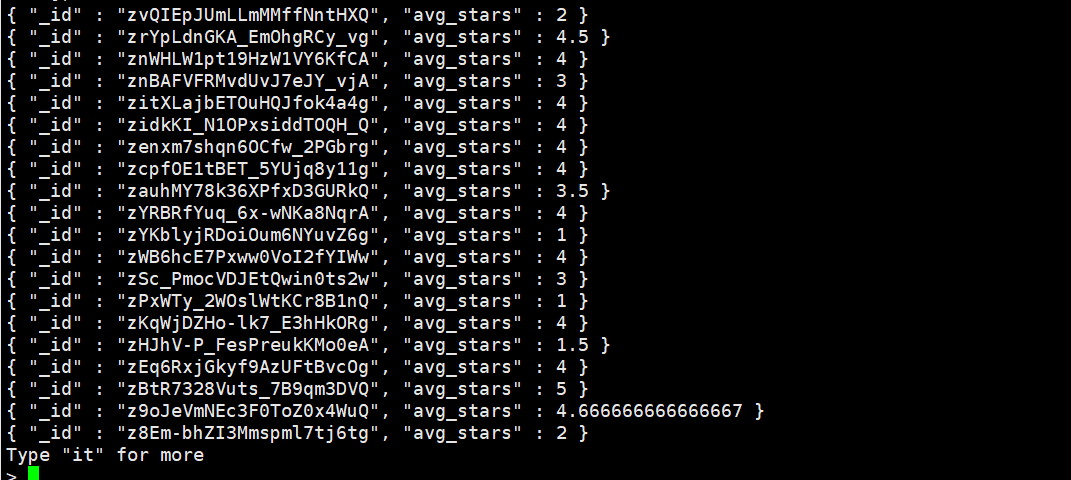
$sort: {

\_id: -1

}

}

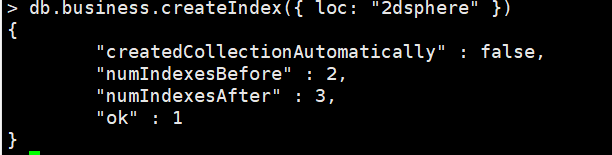
])



13.查询距离商家xvX2CttrVhyG2z1dFg\_0xw(business\_ id) 100米以内的商家， 只需要返回商家名字，地址和星级。

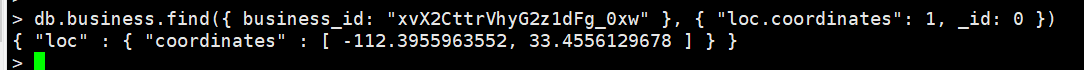
使用2dsphere创建索引：

db.business.createIndex({ loc: "2dsphere" })



查询：

先查询business\_id: "xvX2CttrVhyG2z1dFg\_0xw"的商家的经度和纬度坐标。db.business.find({ business\_id: "xvX2CttrVhyG2z1dFg\_0xw" }, { "loc.coordinates": 1, \_id: 0 })



db.business.find({

business\_id: "xvX2CttrVhyG2z1dFg\_0xw",

loc: {

$near: {

$geometry: {

type: "Point",

coordinates: [-112.3955963552, 33.4556129678]

},

$maxDistance: 100

}

}

}, {

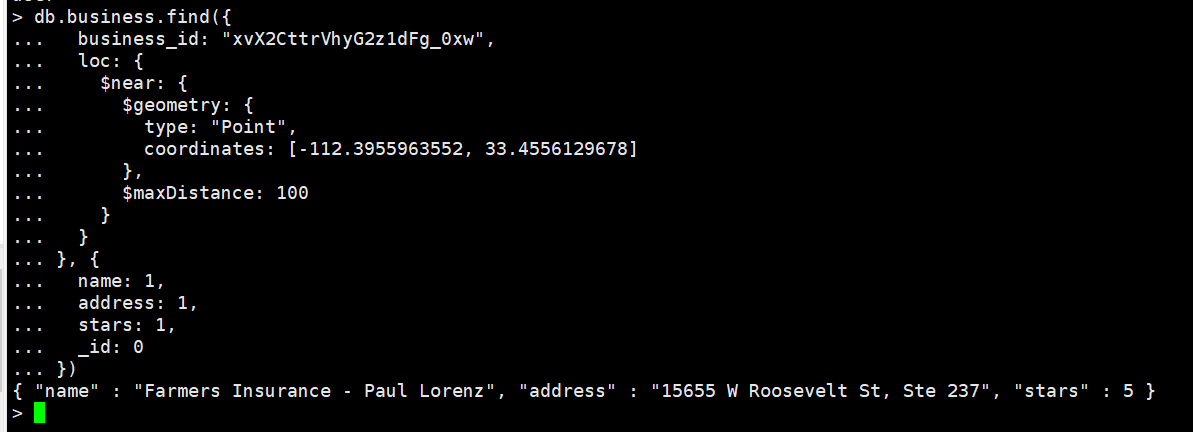
name: 1,

address: 1,

stars: 1,

\_id: 0

})



14 在集合Subreview上建立索引，统计出用户从2017年开始发出的评价有多少， 按照评价次数降序排序，需要返回用户id和评价总次数，只显示前20条结果。

先为 Subreview 集合建立日期字段 date 的索引：

db.Subreview.createIndex({ date: 1 })

然后进行查询：

db.Subreview.aggregate([

{

$match: {

date: { $gte: "2017-01-01" }

}

},

{

$group: {

\_id: "$user\_id",

totalReviews: { $sum: 1 }

}

},

{

$sort: { totalReviews: -1 }

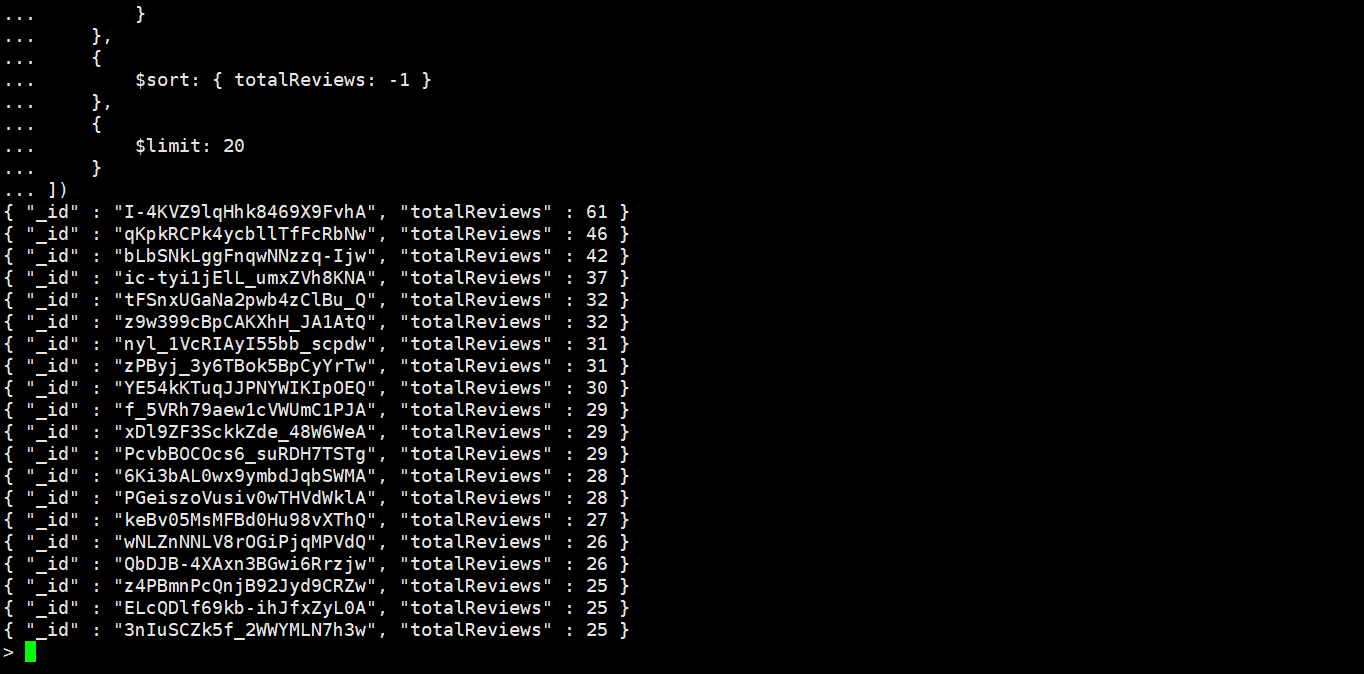
},

{

$limit: 20

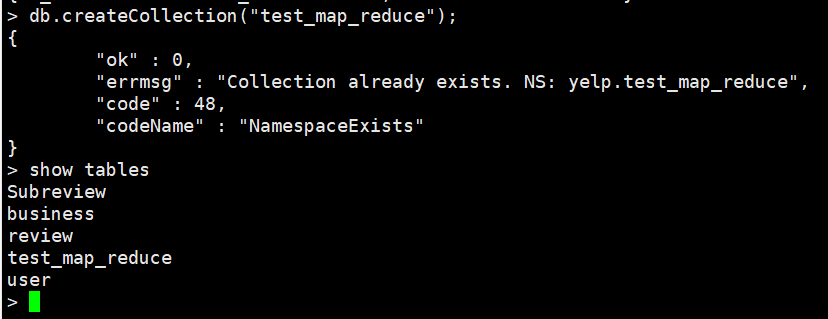
}

])



15.

创建test\_map\_reduce的集合



db.Subreview.mapReduce(

function () {

emit(this.business\_id, { totalStars: this.stars, count: 1 });

},

function (key, values) {

var reducedVal = { totalStars: 0, count: 0 };

values.forEach(function (value) {

reducedVal.totalStars += value.totalStars;

reducedVal.count += value.count;

});

return reducedVal;

},

{

out: "test\_map\_reduce",

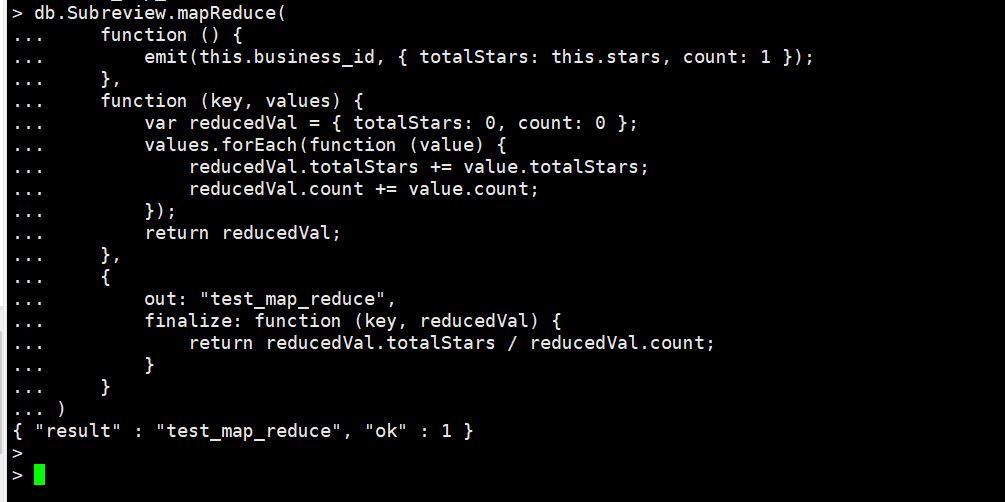
finalize: function (key, reducedVal) {

return reducedVal.totalStars / reducedVal.count;

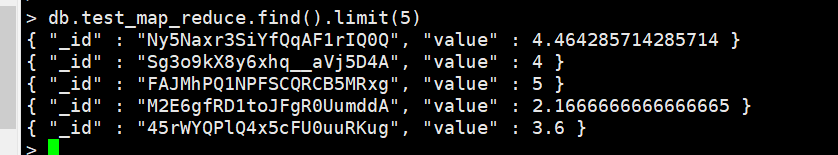
}

}

)



查询前五个：



查询题目里的

db.test\_map\_reduce.findOne({ "\_id": "--Gc998IMjLn8yr-HTzGUg" })

