Lab 8

Student information

- Full name: Qicheng Hu
- E-mail: qhu027@cs.ucr.edu
- UCR NetID: qhu027
- Student ID: X675102

Answers

- (Q1) Insert the sample JSON file into a new collection named contacts.

  Command:

```
use admin
db.createCollection("contacts")
db.contacts.save([
  {
    "Name": "Craft Parks",
    "Address": {
      "StreetNumber": 397,
      "streetName": "Hudson Avenue",
      "city": "Glenbrook",
      "state": "UT",
      "ZIPCode": 96867
    },
    /*...CopyContentFromTheJSONFile...*/
])
```

  The result shows that 10 records are inserted:

```
BulkWriteResult([
"writeErrors" : [],
"writeConcernErrors" : [],
"nInserted" : 10,
/*..OtherInformation..*/
])
```

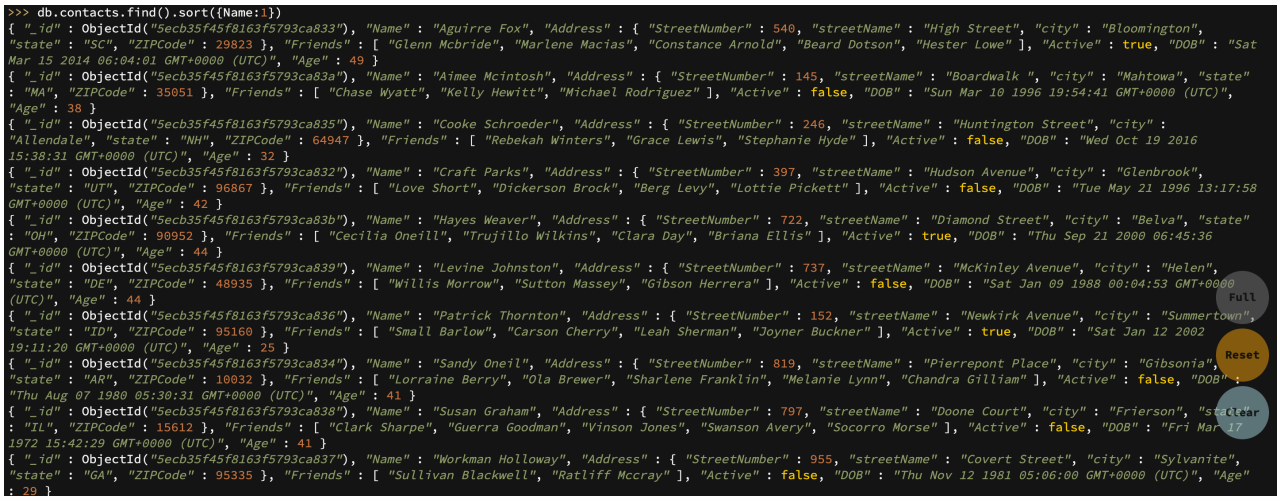
÷

- (Q2) Retrieve all the users sorted by name.

  The command is:

  ```
  db.contacts.find().sort({Name:1})
  ```

  the result is:



- (Q3) List only the `id` and `names` sorted in reverse alphabetical order by `name` (Z-to-A).

  The command is:

  ```
  db.contacts.find({},{_id:1,Name:1}).sort({Name:-1})
  ```

  the result is :

```
. 29 }
>>> db.contacts.find({},{_id:1,Name:1}).sort({Name:-1})
{ "_id" : ObjectId("5ecb35f45f8163f5793ca837"), "Name" : "Workman Holloway" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca838"), "Name" : "Susan Graham" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca834"), "Name" : "Sandy Oneil" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca836"), "Name" : "Patrick Thornton" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca839"), "Name" : "Levine Johnston" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca83b"), "Name" : "Hayes Weaver" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca832"), "Name" : "Craft Parks" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca835"), "Name" : "Cooke Schroeder" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca83a"), "Name" : "Aimee Mcintosh" }
{ "_id" : ObjectId("5ecb35f45f8163f5793ca833"), "Name" : "Aguirre Fox" }
```

- (Q4) Is the comparison of the attribute `name` case-sensitive? Show how you try this with the previous query and include your answer.

  The comparison is case-sensitive. I tried with the following command:

```
db.createCollection("test")
db.test.save([{"Name":"Alpha"},{"Name":"abpha"}])
db.test.find().sort({Name:-1})
```

the result is:

```
>>> db.test.save([{"Name":"Alpha"},{"Name":"abpha"}])
BulkWriteResult({
    "writeErrors" : [ ],
    "writeConcernErrors" : [ ],
    "nInserted" : 2,
    "nUpserted" : 0,
    "nMatched" : 0,
    "nModified" : 0,
    "nRemoved" : 0,
    "upserted" : [ ]
})
>>> db.test.find().sort({Name:-1})
{ "_id" : ObjectId("5ecb3d2e5f8163f5793ca83e"), "Name" : "abpha" }
{ "_id" : ObjectId("5ecb3d2e5f8163f5793ca83d"), "Name" : "Alpha" }
```

it is sorted by reversed alphabetical order, if it is not case senitive then "Alpha" should be followed by "abpha" but it is not.

- (Q5) Repeat Q3 above but do not show the _id field. The command is:

```
db.contacts.find({},{_id:0,Name:1}).sort({Name:-1})
```

the result is :

- (Q6) Insert the following document to the collection. Does MongoDB accept this document while the `name` field has a different type than other records? The MangoDB accepts this record, as the following picture shows.

```
>>> db.contacts.save({"Name": {"First": "David", "Last": "Bark"}})
WriteResult({ "nInserted" : 1 })
```

- (Q7) Where do you expect the new record to be located in the sort order? Verify the answer and explain.

  The new record appears at the first row of the result as the following figure shows:

  The new record is treated as a new type so it is separated from previous ones.

- (Q8) Where do you expect the new document to appear in the sort order. Verify your answer and explain after running the query.

The new record appears before "Craft Parks", as the following figure shows:

```
>>> db.contacts.find({},{_id:1,Name:1}).sort({Name:-1})
{ "_id" : ObjectId("5ecb42475fcf64402c8970d5"), "Name" : { "First" : "David", "Last" : "Bark" } }
{ "_id" : ObjectId("5ecb42165fcf64402c8970d0"), "Name" : "Workman Holloway" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970d1"), "Name" : "Susan Graham" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970cd"), "Name" : "Sandy Oneil" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970cf"), "Name" : "Patrick Thornton" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970d2"), "Name" : "Levine Johnston" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970d4"), "Name" : "Hayes Weaver" }
{ "_id" : ObjectId("5ecb43895fcf64402c8970d6"), "Name" : [ "David", "Bark" ] }
{ "_id" : ObjectId("5ecb42165fcf64402c8970cb"), "Name" : "Craft Parks" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970ce"), "Name" : "Cooke Schroeder" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970d3"), "Name" : "Aimee Mcintosh" }
{ "_id" : ObjectId("5ecb42165fcf64402c8970cc"), "Name" : "Aguirre Fox" }
```

The new record is treated as the same type as the previous ones and is sorted using its first element of array.

- (Q9) Where do you expect the last inserted record, `{Name: [“David”, “Bark”]}` to appear this time? Does it appear in the same position relative to the other records? Explain why or why not.

  This record appears after "A" names before "C" names, it seems that it uses the second element "Bark" as key to sort.

```
>>> db.contacts.find({},{_id:1,Name:1}).sort({Name:1})
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443db"), "Name" : "Aguirre Fox" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443e2"), "Name" : "Aimee Mcintosh" }
{ "_id" : ObjectId("5ecb459b2b8ca3571d0443e5"), "Name" : [ "David", "Bark" ] }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443dd"), "Name" : "Cooke Schroeder" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443da"), "Name" : "Craft Parks" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443e3"), "Name" : "Hayes Weaver" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443e1"), "Name" : "Levine Johnston" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443de"), "Name" : "Patrick Thornton" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443dc"), "Name" : "Sandy Oneil" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443e0"), "Name" : "Susan Graham" }
{ "_id" : ObjectId("5ecb455b2b8ca3571d0443df"), "Name" : "Workman Holloway" }
{ "_id" : ObjectId("5ecb45862b8ca3571d0443e4"), "Name" : { "First" : "David", "Last" : "Bark" } }
>>>
```

- (Q10) Build an index on the Name field for the users collection. Is MongoDB able to build the index on that field with the different value types stored in the Name field?

  The index is built successfully as the following screenshot shows:

```
>>> db.contacts.ensureIndex({Name:1})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
```