LAB4

Mongo Shell

Contents

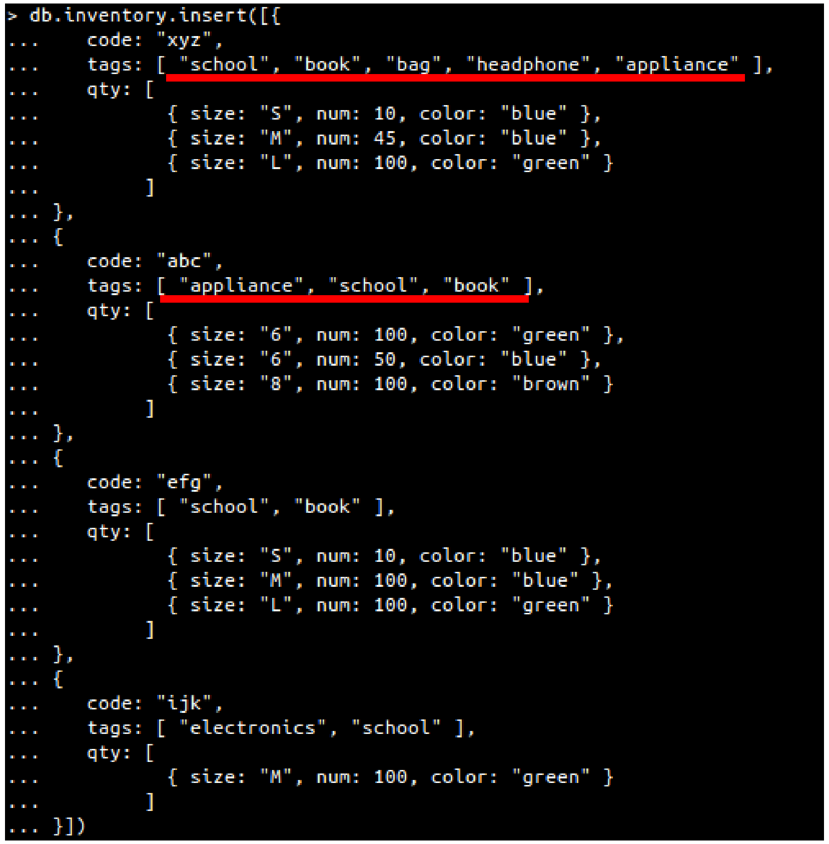
Query: $all

- Syntax
  - { <field>: { $all: [<value1>, <value2>, ...] } }
- Select the documents where the value of a field is an array that contains all the specified elements
- It is equivalent to $and

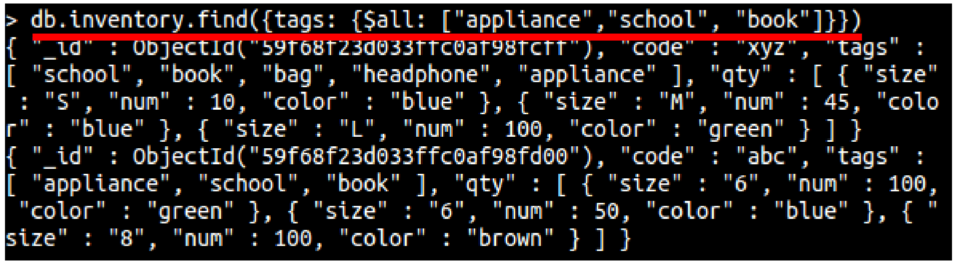
  Example)
  - { tags: { $all: ["ssl" , "security"] } }
  - { $and: [{ tags: "ssl" }, { tags: "security" }] }

Query: $all

- Example



```
> db.inventory.insert([{
...     code: "xyz",
...     tags: [ "school", "book", "bag", "headphone", "appliance" ],
...     qty: [
...           { size: "S", num: 10, color: "blue" },
...           { size: "M", num: 45, color: "blue" },
...           { size: "L", num: 100, color: "green" }
...         ]
... },
... {
...     code: "abc",
...     tags: [ "appliance", "school", "book" ],
...     qty: [
...           { size: "6", num: 100, color: "green" },
...           { size: "6", num: 50, color: "blue" },
...           { size: "8", num: 100, color: "brown" }
...         ]
... },
... {
...     code: "efg",
...     tags: [ "school", "book" ],
...     qty: [
...           { size: "S", num: 10, color: "blue" },
...           { size: "M", num: 100, color: "blue" },
...           { size: "L", num: 100, color: "green" }
...         ]
... },
... {
...     code: "ijk",
...     tags: [ "electronics", "school" ],
...     qty: [
...           { size: "M", num: 100, color: "green" }
...         ]
... }])
```



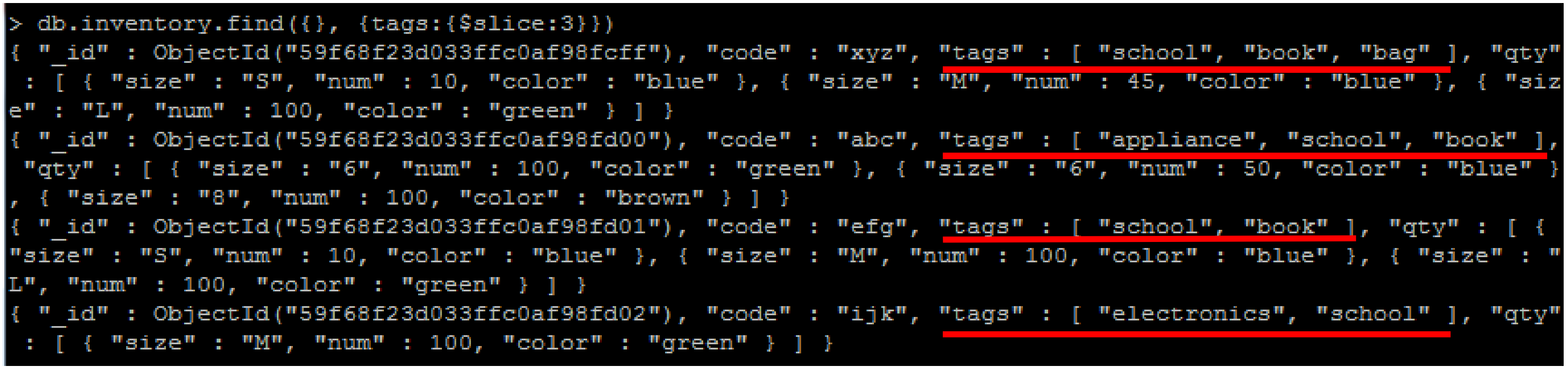
```
> db.inventory.find({tags: {$all: ["appliance","school", "book"]}})
{ "_id" : ObjectId("59f68f23d033ffc0af98fcff"), "code" : "xyz", "tags" :
[ "school", "book", "bag", "headphone", "appliance" ], "qty" : [ { "size"
 : "S", "num" : 10, "color" : "blue" }, { "size" : "M", "num" : 45, "colo
r" : "blue" }, { "size" : "L", "num" : 100, "color" : "green" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd00"), "code" : "abc", "tags" :
[ "appliance", "school", "book" ], "qty" : [ { "size" : "6", "num" : 100,
 "color" : "green" }, { "size" : "6", "num" : 50, "color" : "blue" }, { "
size" : "8", "num" : 100, "color" : "brown" } ] }
```

Query: $slice

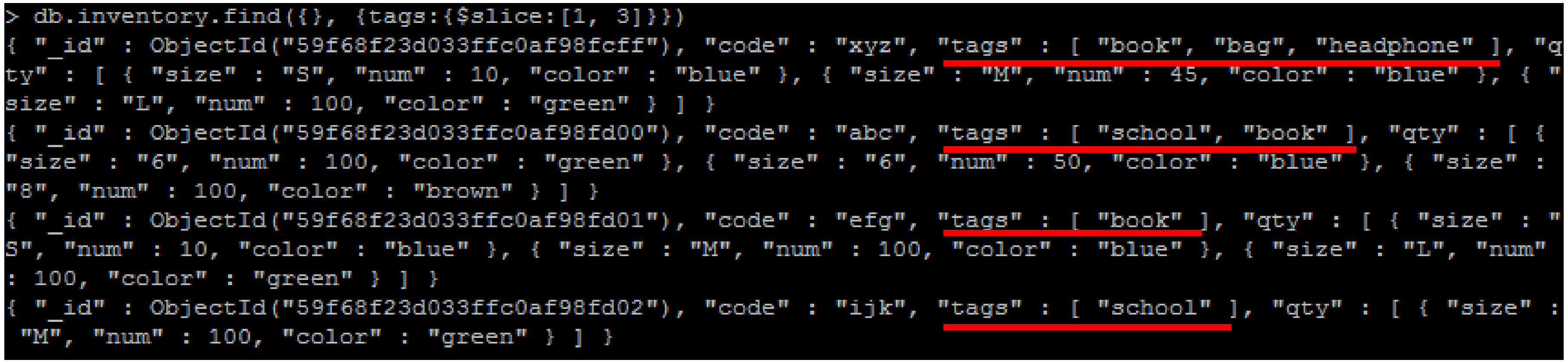
- Syntax
  - db.collection.find({field: value}, {array: {$slice: *count*}})
  - db.collection.find({field: value}, {array: {$slice: [*skip* , *limit*]}})
- Control the number of items in an array that a query returns
- Accept arguments in a number of formats, including negative values and arrays

Query: $slice

- Example



```
> db.inventory.find({}, {tags:{$slice:3}})
{ "_id" : ObjectId("59f68f23d033ffc0af98fcff"), "code" : "xyz", "tags" : [ "school", "book", "bag" ], "qty"
 : [ { "size" : "S", "num" : 10, "color" : "blue" }, { "size" : "M", "num" : 45, "color" : "blue" }, { "siz
e" : "L", "num" : 100, "color" : "green" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd00"), "code" : "abc", "tags" : [ "appliance", "school", "book" ],
 "qty" : [ { "size" : "6", "num" : 100, "color" : "green" }, { "size" : "6", "num" : 50, "color" : "blue" }
, { "size" : "8", "num" : 100, "color" : "brown" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd01"), "code" : "efg", "tags" : [ "school", "book" ], "qty" : [ {
"size" : "S", "num" : 10, "color" : "blue" }, { "size" : "M", "num" : 100, "color" : "blue" }, { "size" : "
L", "num" : 100, "color" : "green" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd02"), "code" : "ijk", "tags" : [ "electronics", "school" ], "qty"
 : [ { "size" : "M", "num" : 100, "color" : "green" } ] }
```



```
> db.inventory.find({}, {tags:{$slice:[1, 3]}})
{ "_id" : ObjectId("59f68f23d033ffc0af98fcff"), "code" : "xyz", "tags" : [ "book", "bag", "headphone" ], "q
ty" : [ { "size" : "S", "num" : 10, "color" : "blue" }, { "size" : "M", "num" : 45, "color" : "blue" }, { "
size" : "L", "num" : 100, "color" : "green" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd00"), "code" : "abc", "tags" : [ "school", "book" ], "qty" : [ {
"size" : "6", "num" : 100, "color" : "green" }, { "size" : "6", "num" : 50, "color" : "blue" }, { "size" :
"8", "num" : 100, "color" : "brown" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd01"), "code" : "efg", "tags" : [ "book" ], "qty" : [ { "size" : "
S", "num" : 10, "color" : "blue" }, { "size" : "M", "num" : 100, "color" : "blue" }, { "size" : "L", "num"
: 100, "color" : "green" } ] }
{ "_id" : ObjectId("59f68f23d033ffc0af98fd02"), "code" : "ijk", "tags" : [ "school" ], "qty" : [ { "size" :
 "M", "num" : 100, "color" : "green" } ] }
```

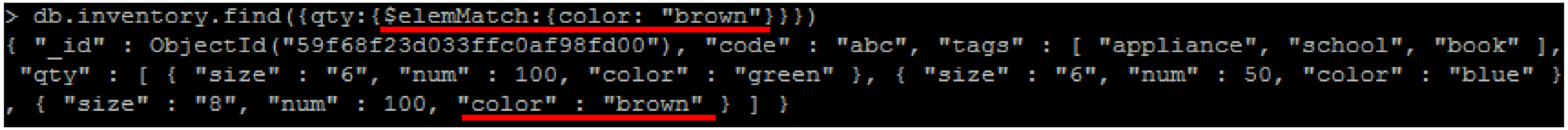
Exercise 1

Import blog.json into the *blog* collection in *lab4*

- Find documents
  - The language includes both *Korean* and *English*
- Display a document slicing two comments
  - The document's title is *Big data*

Query: $elemMatch

- Syntax
  - { <field>: { $elemMatch: { <query1>, <query2>, ... } } }
- Matches documents that contain an array field with at least one element that matches all the specified query criteria
- Example



```
> db.inventory.find({qty:{$elemMatch:{color: "brown"}}})
{ "_id" : ObjectId("59f68f23d033ffc0af98fd00"), "code" : "abc", "tags" : [ "appliance", "school", "book" ],
 "qty" : [ { "size" : "6", "num" : 100, "color" : "green" }, { "size" : "6", "num" : 50, "color" : "blue" }
, { "size" : "8", "num" : 100, "color" : "brown" } ] }
```

Exercise 2

Use the *blog* collection in *lab4*

- Find a document as following conditions
  - The writer's first name is *Ki*
  - The language is *Korean*
- Find documents that have *Kim*'s comment
  1. Only for an array
  2. Regardless of an array

Cursor

- The db.collection.find() method returns a cursor
- The cursor method
  - next() to access the documents
    - var myCursor = db.users.find({ type: 2 });
    - while (myCursor.hasNext()) { print(tojson(myCursor.next())); }
  - forEach() to iterate the cursor and access the documents
    - var myCursor = db.users.find({ type: 2 });
    - myCursor.forEach(printjson);

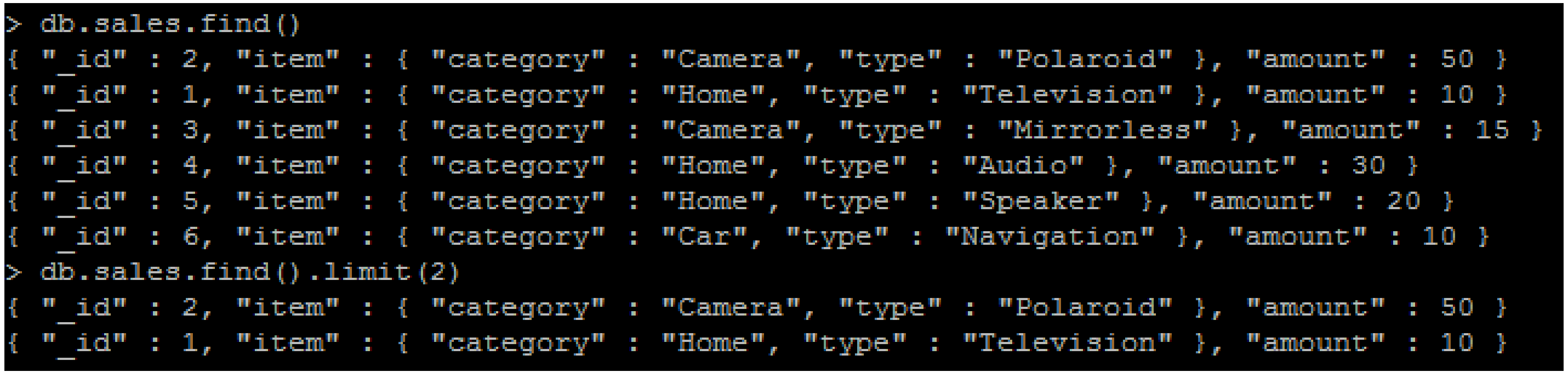
Note:

```
forEach(func) {
    while (this.hasNext())
        func(this.next());
}
```

```
printjson (x) {
    print(tojson(x));
}
```

Cursor: limit

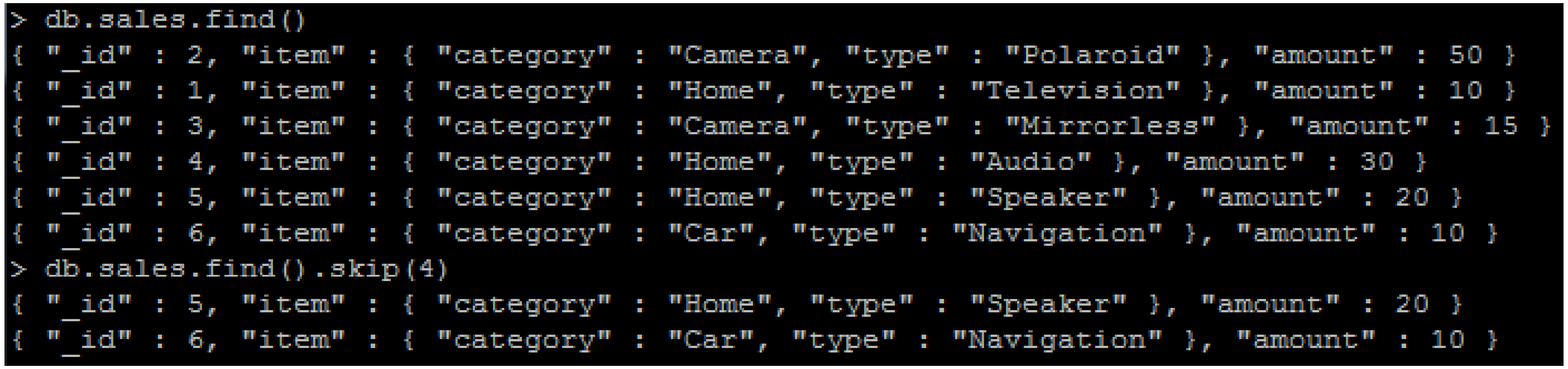
- Syntax
  - db.collection.find(<query>).limit(<number>)
- Specify the maximum number of documents the cursor will return
- Example



```
> db.sales.find()
{ "_id" : 2, "item" : { "category" : "Camera", "type" : "Polaroid" }, "amount" : 50 }
{ "_id" : 1, "item" : { "category" : "Home", "type" : "Television" }, "amount" : 10 }
{ "_id" : 3, "item" : { "category" : "Camera", "type" : "Mirrorless" }, "amount" : 15 }
{ "_id" : 4, "item" : { "category" : "Home", "type" : "Audio" }, "amount" : 30 }
{ "_id" : 5, "item" : { "category" : "Home", "type" : "Speaker" }, "amount" : 20 }
{ "_id" : 6, "item" : { "category" : "Car", "type" : "Navigation" }, "amount" : 10 }
> db.sales.find().limit(2)
{ "_id" : 2, "item" : { "category" : "Camera", "type" : "Polaroid" }, "amount" : 50 }
{ "_id" : 1, "item" : { "category" : "Home", "type" : "Television" }, "amount" : 10 }
```

Cursor: skip

- Syntax
  - db.collection.find(<query>).skip(<number>)
- Control where MongoDB begins returning results
- Example



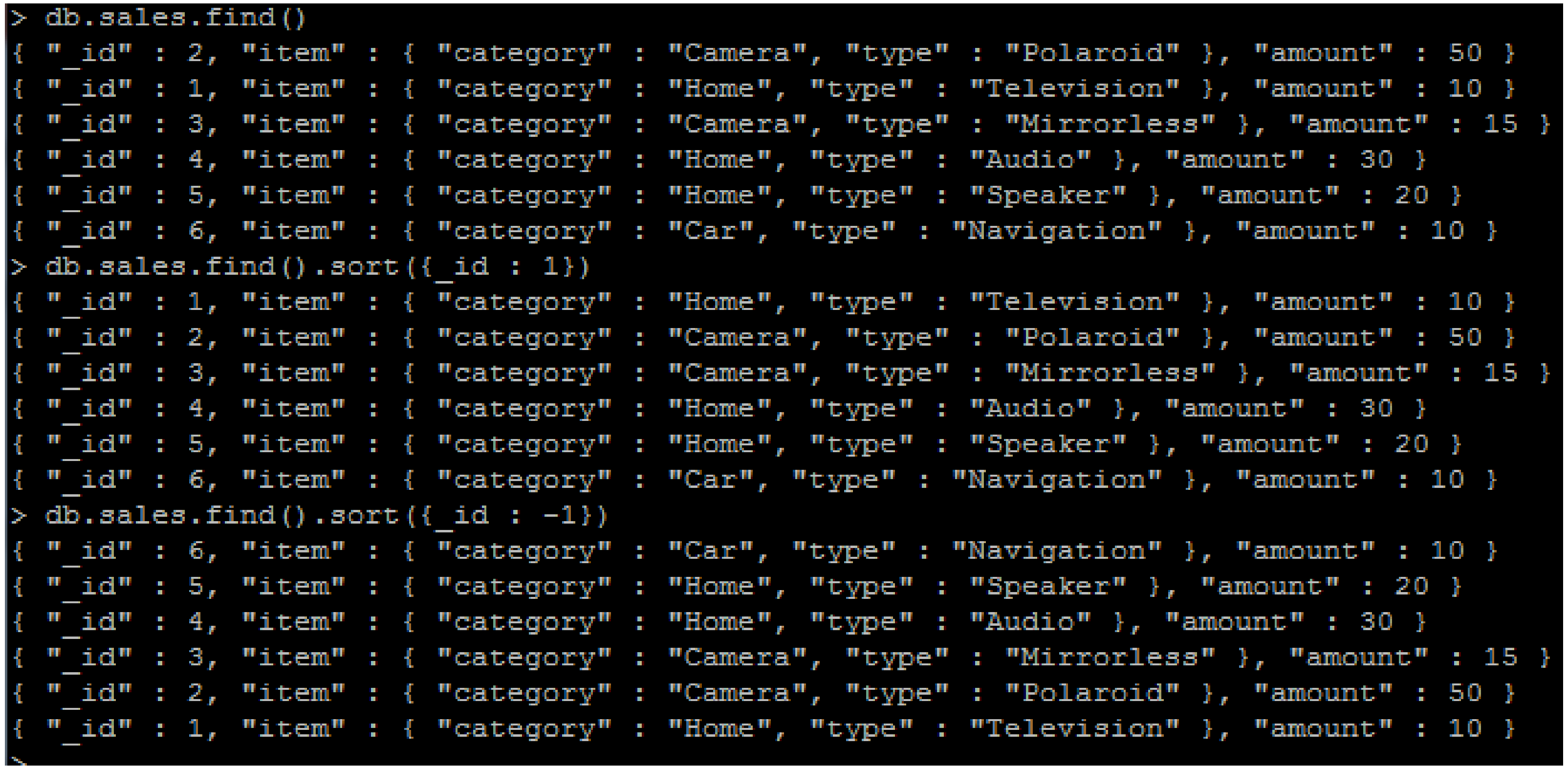
```
> db.sales.find()
{ "_id" : 2, "item" : { "category" : "Camera", "type" : "Polaroid" }, "amount" : 50 }
{ "_id" : 1, "item" : { "category" : "Home", "type" : "Television" }, "amount" : 10 }
{ "_id" : 3, "item" : { "category" : "Camera", "type" : "Mirrorless" }, "amount" : 15 }
{ "_id" : 4, "item" : { "category" : "Home", "type" : "Audio" }, "amount" : 30 }
{ "_id" : 5, "item" : { "category" : "Home", "type" : "Speaker" }, "amount" : 20 }
{ "_id" : 6, "item" : { "category" : "Car", "type" : "Navigation" }, "amount" : 10 }
> db.sales.find().skip(4)
{ "_id" : 5, "item" : { "category" : "Home", "type" : "Speaker" }, "amount" : 20 }
{ "_id" : 6, "item" : { "category" : "Car", "type" : "Navigation" }, "amount" : 10 }
```

Cursor: sort

- Syntax
  - db.collection.find(<query>).sort(<document>)
- Specifies the order in which the query returns matching documents
  - 1: ascending order
  - -1: descending order

Cursor: $sort

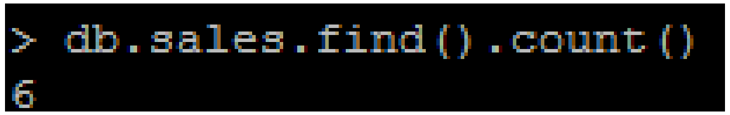
- Example



```
> db.sales.find()
{ "_id" : 2, "item" : { "category" : "Camera", "type" : "Polaroid" }, "amount" : 50 }
{ "_id" : 1, "item" : { "category" : "Home", "type" : "Television" }, "amount" : 10 }
{ "_id" : 3, "item" : { "category" : "Camera", "type" : "Mirrorless" }, "amount" : 15 }
{ "_id" : 4, "item" : { "category" : "Home", "type" : "Audio" }, "amount" : 30 }
{ "_id" : 5, "item" : { "category" : "Home", "type" : "Speaker" }, "amount" : 20 }
{ "_id" : 6, "item" : { "category" : "Car", "type" : "Navigation" }, "amount" : 10 }
> db.sales.find().sort({_id : 1})
{ "_id" : 1, "item" : { "category" : "Home", "type" : "Television" }, "amount" : 10 }
{ "_id" : 2, "item" : { "category" : "Camera", "type" : "Polaroid" }, "amount" : 50 }
{ "_id" : 3, "item" : { "category" : "Camera", "type" : "Mirrorless" }, "amount" : 15 }
{ "_id" : 4, "item" : { "category" : "Home", "type" : "Audio" }, "amount" : 30 }
{ "_id" : 5, "item" : { "category" : "Home", "type" : "Speaker" }, "amount" : 20 }
{ "_id" : 6, "item" : { "category" : "Car", "type" : "Navigation" }, "amount" : 10 }
> db.sales.find().sort({_id : -1})
{ "_id" : 6, "item" : { "category" : "Car", "type" : "Navigation" }, "amount" : 10 }
{ "_id" : 5, "item" : { "category" : "Home", "type" : "Speaker" }, "amount" : 20 }
{ "_id" : 4, "item" : { "category" : "Home", "type" : "Audio" }, "amount" : 30 }
{ "_id" : 3, "item" : { "category" : "Camera", "type" : "Mirrorless" }, "amount" : 15 }
{ "_id" : 2, "item" : { "category" : "Camera", "type" : "Polaroid" }, "amount" : 50 }
{ "_id" : 1, "item" : { "category" : "Home", "type" : "Television" }, "amount" : 10 }
>
```

Cursor: count

- Syntax
  - db.collection.find(<query>).count()
  - Specify the maximum number of documents the cursor will return
- Example



```
> db.sales.find().count()
6
```

Exercise 3

Import sales.json into the *sales* collection in *lab4*

- Print all documents using cursor method
- Sorting *amount* values in ascending order and then *_id* in descending order
- Display three documents whose amount is less than 40
- Skip two documents sorted by *amount* in descending order
- Counting documents whose *category* of *item* is *Home*

PyMongo

Try…except

- **try:** … **except** exceptions:
- Example

```
try:
    4 / 0
except ZeroDivisionError as e:
    print(e)
```

  - Output: division by zero
- Refer to
  - http://api.mongodb.com/python/current/api/pymongo/errors.html

sort

- Syntax
  - db.collection.find().sort(field, order)
- Example
  - db.Account.find().sort("UserName",1)
  - db.Account.find().sort("UserName",-1)

Exercise 4

- Using try except statements when you insert documents with duplicated *_id*
  - Use the *sales* collection in *lab4*
  - { "_id": 4, "item": { "category": “Camera", "type": “Digital" }, "amount": 15 }
- Print error message
  - *E11000 duplicate key error collection: …*
  - Using the *except Exception* statement

Exercise 5

Import grade.json into *grade* in *lab4*

- the grade documents are composed of three types:
  - quiz, exam, and homework
  - Student id (*sid*) is from 0 to 99
- A quiz score of a student is lost
  - Print the student the student whose quiz score is lost
  - Insert the quiz score (80) with sid

Exercise 6

Import grade.json into *grade* in *lab4*

- Print three documents sorted by *sid* in ascending order when you insert the number.
  - def showgroup(number)

  Hint: sort, limit and skip

Exercise 7

Grading students' work

- Compute the total score and give the letter grade to each student
  - Total score = quiz * 0.2 + homework*0.3 + exam * 0.5
  - >=90: A, >=80: B, >=70: C, >=60: D, and others: F
  - Insert the letter grade with sid into the *letter* collection
  - Print the average of total scores

Social Networking

Post

- Create the post schema
- Insert and remove user’s text

Post

- Open post.py
- Read the comment carefully and write the code in the file.
- Add the item to enter the posting page in the user.py

Checklist

- The functions are executed correctly
- User's texts are stored in your database
- Exception cases are considered
  - try…catch