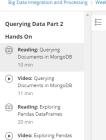
DataFrames 5 min

Quiz: Postgres, MongoDB, and Pandas 5 questions Prev | Next

Big Data Integration and Processing > Week 2 > Querying Documents in MongoDB



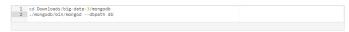
By the end of this activity, you will be able to:

- 1. Find documents in MongoDB with specific field values.
- 2. Filter the results returned by MongoDB queries.
- 3. Count documents in a MongoDB collection and returned by queries.

Step 1. Start MongoDB server and MongoDB shell. Open a terminal window by clicking on the square black box on the top left of the screen



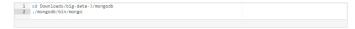
Next, change to the mongodb directory, and start the server



The arguments $-dbpath\,db$ specify that the directory db should be used for the MongoDB directory for datafiles. After starting the MongoDB server, you will see the following lines indicating that the server is running:

I FTDC [initandListen] Initializing full-time diagnostic data capture with directory 'db/diagnostic.data' I NETWORK [HostnameGanonicalizationWorker] Starting hostname canonicalization worker I NETWORK [initandListen] waiting for connections on port 27017

Next, let's run the MongoDB shell so that we can query the server. Open a new terminal shell window, change to the *mongodb* directory, and start the shell:



Step 2. Show Databases and Collections. Run the show dbs command to see the databases:

```
> show dbs
journaldev 0.000GB
local 0.000GB
sample 0.004GB
test 0.000GB
```

The database named sample has been created and loaded with Twitter JSON data. Let's switch to that database by running the use command:

```
> use sample 
switched to db sample
```

We can see the collections in the sample database by running show collections:

```
> show collections collection users
```

The Twitter data is stored in the users collection. We can run db.users.count() to count the number of documents:

```
> db.users.count()
11188
```

Step 3. Look at document and find distinct values. We can examine the contents of one of the documents by running

The document has several fields, e.g., user_name, retweet_count, tweet_ID, etc., and nested fields under user, e.g., CreatedAt,

We can find the distinct values for a specific field by using the distinct() command. For example, let's find the distinct values for

```
> db.users.distinct("user_name")
[
    "koteras",
    "Allieloves85_1D",
    "tonkatol",
    "aslet",
    "syaxmii",
    "camSteele_96",
```

Step 4. Search for specific field value. We can search for fields with a specific value using the find() command. For example, let's search for user_name with the value ActionSports[ax].

```
> db.users.find({user.name: "ActionSportsJax"})
{ ".id": 0bjectId('579670bfc38159226b4c8e47"), "user.name": "ActionSportsJax", "retweet
rce: "-«ahref-"http://witter.com/download/sphone\" rel=\"nofollow\">Twitter for iPhon
unt": 2, "tweet ID": "757667806921531393", "tweet text": "RT @wdbrownl5: I'm watching
daugustine football and asked myself \"How on earth did we stop..." "user": { "createdAt"
nt": 120, "FollowersCount": 3539, "FriendsCount": 476, "UserId": 35857042, "Location"
```

By appending .pretty() to the end of the find command, the results will be formatted:

```
> db.users.find({user_name: "ActionSportsJax*}).pretty()
{
    ".id": ObjectId(*579678bfc38159226b4c8e47*),
    "user_name": "ActionSportsJax",
    "retweet count": 0,
    "tweet followers count": 3539,
```

```
"source": "<a href=\"http://twitter.com/download/i
"coordinates": null,
"tweet mentioned count": 2,
"tweet_ID": "757667800521531393",
```

Step 5. Filter fields returned by query. We can specify a second argument to the find() command to only show specific field(s) in the result. Let's repeat the previous search, but only show the tweet_ID field:

```
> db.users.find({user_name: "ActionSportsJax"}, {tweet_ID: 1})
{ "_id" : ObjectId("579670bfc38159226b4c8e47"), "tweet_ID" : "757667809521531393" }
```

 $The _id \ field \ is \ primary \ key for \ every \ document, \ and \ we \ can \ remove \ it \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ with \ the \ following \ filter: \ from \ the \ results \ from \ the \ from \ the \ following \ filter: \ from \ the \ results \ from \ the \ the \ from \ the \ the \ from \ the \ the \ the \ from \ the \ t$

```
> db.users.find{{user_name: "ActionSportsJax"}, {tweet_ID: 1, _id: 0}) { "tweet_ID": "757667800521531393" }
```

Step 6. **Perform regular expression search**. MongoDB also supports searching documents with regular expressions. If we search for the value FIFA in the tweet_text field, there are no results:

```
> db.users.find({tweet_text: "FIFA"})
> ||
```

However, if we search using a regular expression, there are many results:

```
> db.users.find({tweet text: /FIFA/}) {
    " id" : ObjectId('578ffa8e7eb9513f4f55a935"), "user name" : "koteras", "retwee a href=\"http://tvaiter.com/download/jphone\"rela\"nofollon\">Twitter for iPhone "tweet_lD": "75881629932675972", 'tweet_text" : RT @ochocinco: I beat them a' kBL , "user" : { "CreatedAt" : ISDOate("2011-12-27709:04:012"), "FavouritesCount serid" : 44738090, "toaction" : 501    "} {
    " id" : ObjectId('578ffa917eb9513f4f5sa939"), "user name" : "Tonkatol*, "retwee a href=\"http://twitter.com\" rela\"nofollon\">Twitter web Cliente/a>", "coordin 755891638921232384", "tweet_text" : "RT @GameSeek: Follow & Retweet for your challour choicel in our #giveawly inttps://m", "user" : ("CreatedAt" : ISDOate("2012 nt" : 616, "FriendSCount" : 2675, "UserId" : 722815650, "Location" : null } }
```

The difference between the queries is that the first searched for where the tweet_text field value was exactly equal to FIFA, and the second searched for where the field value contained FIFA.

We can append .count() to the command to count the number of results:

```
> db.users.find({tweet_text: /FIFA/}).count()
3697
```

Step 7. Search using text index. A text index can be created to speed up searches and allows advanced searches with Stext Let's create the index using createIndex():

```
> db.users.createIndex({"tweet_text": "text"})
{
    "createdCollectionAutomatically": false,
    "numIndexesBefore": 2,
    "numIndexesAfter": 2,
    "note": "all indexes already exist",
    "ok": 1
```

The argument $\mathit{tweet_text}$ specifies the field on which to create the index.

Next, we can use the \$text operator to search the collection. We can perform the previous query to find the documents containing FIFA:

```
> db.users.find({\text{stext : \{ssearch : "FIFA"\}}}).count() 4031
```

We can also search for documents not containing a specific value. For example, let's search for documents containing FIFA, but

```
> db.users.find({$text : {$search : "FIFA -Texas"}}).count()
4022
```

Step 8. Search using operators. MongoDB can also search for field values matching a specific criteria. For example, we can find where the tweet mentioned count is greater than six:

```
> db.users.find({tweeT mentioned count: (Sgt : 6})}
{ ".id" : 0bjectId(*57966e26c3815920e1131b03"), "user name" : "marshallrupe", : "<a href=\"http://twitter.com/download/jibnone\" rel=\"nofollom\">Twitter for townload.iphone\" rel=\"nofollom\">Twitter for townload.iphone\" rel=\"nofollom\">Twitter for townload.iphone\" rel=\"nofollom\">Twitter for townload.iphone\" rel=\"nofollom\">Twitter for s236, "FollowerSount": 165, "user'al : "237286655, " id" : 0bjectId(*37960e6c63815920e1132053"), "user name": "marvheys", "re "a href=\"http://twitter.com/download/mdoriod\" rel=\"nofollom\">Twitter for : 7, "tweet ID" : "757665717948971752", "tweet text" : "Rf @philhaytonl1: Ple 1820 @marvheys @Matlalor https://t.co/y058Wtj365", "user" : { "Createdat" : 0llowerSount" : 1215, "FriendsCount" : 859, "UserId" : 477884041, "Location" (" "id" : 0bjectId(*57997606031395220b6463f"), "user name" : "HeartAlbhitt", "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>>", "c
```

The \$gt operator search for values greater than a specific value. We can use the \$where command to compare between fields in the same document. For example, the following searches for tweet_mentioned_count is greater than tweet_followers_count.

```
> db.users.find({$where : "this.tweet_mentioned_count > this.tweet_followers_count"}).count() 18
```

Note that the field names for \$where are required to be prefixed with this, which represent the document

We can combine multiple searches by using \$and. For example, let's search for $tweet_text$ containing FIFA and $tweet_mentioned_count$ greater than four:

```
> db.users.find({$and} : [ {tweet_text} : /FIFA/}, {tweet_mentioned_count: {$gt} : 4}}]).count()    1
```

When you are done querying MongoDB, run exit in the MongoDB shell, and Control-C in the terminal window running the server.

Mark as completed

