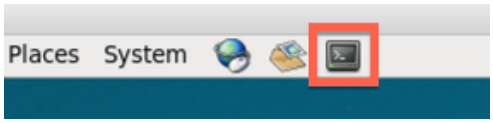
By the end of this activity, you will be able to:

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:

```
1  cd Downloads/big-data-3/mongodb
2  ./mongodb/bin/mongod --dbpath db
```

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  [HostnameCanonicalizationWorker] 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:

```
1  cd Downloads/big-data-3/mongodb
2  ./mongodb/bin/mongo
```

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 *db.users.findOne():*

```
> db.users.findOne()
{
        "_id" : ObjectId("578ffa8e7eb9513f4f55a935"),
        "user_name" : "koteras",
        "retweet_count" : 0,
        "tweet_followers_count" : 461,
        "source" : "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhone</a>",
        "coordinates" : null,
        "tweet_mentioned_count" : 1,
        "tweet_ID" : "755891629932675072",
        "tweet_text" : "RT @ochocinco: I beat them all for 10 straight hours #FIFA16KING  https://t.co/BFnV6jfkBL",
        "user" : {
                "CreatedAt" : ISODate("2011-12-27T09:04:01Z"),
                "FavouritesCount" : 5223,
                "FollowersCount" : 461,
                "FriendsCount" : 619,
                "UserId" : 447818090,
                "Location" : "501"
        }
}
```

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

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

```
> db.users.distinct("user_name")
[
        "koteras",
        "AllieLovesR5_1D",
        "Tonkatol",
        "Gaslet",
        "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 *ActionSportsJax*:

```
> db.users.find({user_name : "ActionSportsJax"})
{ "_id" : ObjectId("579670bfc38159226b4c8e47"), "user_name" : "ActionSportsJax", "retweet
rce" : "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhon
unt" : 2, "tweet_ID" : "757667800521531393", "tweet_text" : "RT @wwbrown19: I'm watching
Augustine 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("579670bfc38159226b4c8e47"),
        "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" : "757667800521531393" }
```

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

```
> 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://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhon
 "tweet_ID" : "755891629932675072", "tweet_text" : "RT @ochocinco: I beat them a
kBL", "user" : { "CreatedAt" : ISODate("2011-12-27T09:04:01Z"), "FavouritesCount
serId" : 447818090, "Location" : "501" } }
{ "_id" : ObjectId("578ffa917eb9513f4f55a939"), "user_name" : "Tonkatol", "retwe
<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>", "coordi
755891638921232384", "tweet_text" : "RT @GameSeek: Follow & Retweet for your cha
our choice) in our #giveaway! https://…", "user" : { "CreatedAt" : ISODate("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 *$text*. 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 *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({$text : {$search : "FIFA"}}).count()
4031
```

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

```
> 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: {$gt : 6}})
{ "_id" : ObjectId("57966e26c3815920e1131b03"), "user_name" : "marshallrupe",
 : "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter f
 : 7, "tweet_ID" : "757665013817409536", "tweet_text" : "RT @BrianBorrison: h
lrupe @da_carlos30 @TrissyTim @BrockHiltonb @ZachZippe @ro…", "user" : { "Cre
: 326, "FollowersCount" : 215, "FriendsCount" : 165, "UserId" : -2073286695,
{ "_id" : ObjectId("57966ecfc3815920e1132053"), "user_name" : "marvheys", "re
"<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for
: 7, "tweet_ID" : "757665717948977152", "tweet_text" : "RT @philhayton11: Ple
1882 @marvheys @Matlalor  https://t.co/yOs8MxjA5d", "user" : { "CreatedAt" :
ollowersCount" : 1215, "FriendsCount" : 850, "UserId" : 477884041, "Location"
{ "_id" : ObjectId("5796706bc38159226b4c8af1"), "user_name" : "HearAllAbtIt",
 "<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