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. Open a terminal shell. Open your local terminal shell and optionally go to your *big-data-3* directory.

```
PS C:\Users\[redacted]\Desktop\coursera\big-data-3>
```

Step 2. Start Docker. Make sure to start Docker by opening Docker Desktop.

Once you have started Docker, go back to your terminal and run *docker pull pramonettivega/mongo-coursera:latest* to pull a Docker image for this activity.

```
docker pull pramonettivega/mongo-coursera:latest
```

Step 3. Run the container. Run *docker run --name my-mongo -d -p 27017:27017 pramonettivega/mongo-coursera:latest* to start the container. In this case, we are naming the container my-mongo, but the name is optional.

```
docker run --name my-mongo -d -p 27017:27017 pramonettivega/mongo-coursera:latest
```

Go to Docker Desktop, click on *Containers* and make sure the container is running

Step 4. Access the container. Run docker *docker exec -it my-mongo mongosh* to access the MongoDB shell from the terminal.

```
docker exec -it my-mongo mongosh
```

Step 5. Show Databases and Collections. Run the *show dbs* command to see the databases:

```
test> show dbs
admin        40.00 KiB
config       12.00 KiB
journaldev   44.00 KiB
local        72.00 KiB
sample        4.14 MiB
test         12.00 KiB
```

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

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

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

```
sample> show collections
collection
users
```

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

```
sample> db.users.countDocuments()
11188
```

Step 6. Look at document and find distinct values. We can examine the contents of one of the documents by running *db.users.findOne():*

```
sample> 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('2011-12-27T09:04:01.000Z'),
    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*:

```
sample> db.users.distinct("user_name")
[
  '007Shafy',         '04bike',           '07kennedyl',        '09dixonDixon',
  '0ASlS',            '0FootballDaily0',  '0nyeomachi',        '0riginalyst',
  '1001ptsUK',        '1010xl_T_Wigg',    '10PcLemonPepper',   '10_plus1',
  '10n_Meister',      '1122Mfmf',         '11alexburns',       '1480whbc',
  '16salazarjaime',   '178_sun',          '17albecker',        '1888jackie',
  '1986samc',         '1CookieCrumbles',  '1Easterbrook',      '1Flaco_P',
  '1Jamesoo',         '1PlayerWon',       '1TouchFootball_',   '1Wilke',
  '1andonlykiarra',   '1gangan',          '1holty23',          '1jasminesotox',
  '1kMike',           '1ldjy',            '1molarmike',        '1nzie',
  '1salmanjafri',     '2016FIFA17',       '21_tafolla',        '221_765975798',
  '22Quinn',          '234today',         '23jayslater_',      '25159050ronnie',
  '26harrywatling',   '2711Lesley',       '27eaf706f5c74a9',   '2ChainlezChainz',
  '2ClutchThaGawd',   '2Clutch_Island',   '2Girls1Richard',    '2RawChristian',
  '2bewell2',         '2le_Fuze',         '2official_john',    '2xAwesome',
  '31Reasonz',        '32viamcildasha',   '34bearcat',         '34cvc',
  '34intel12342a',    '360hsgaming',      '365FootballTN',     '37a63e38e38f4a2',
  '380_951',          '3HUNNAJ0',         '3NameSportsShow',   '3NovicesAfrican',
  '3NovicesAsian',    '3R1C_K3NDra',      '3ajrah',            '3badi_ALD',
  '3bdallahSindi',    '3la2_Hayek',       '3pointiscash',      '3rweh',
  '3z_al501',         '420ddy',           '433coach',          '465USD',
  '49ersHub',         '49ersSpyder',      '4Prina',            '4Real_P0litik',
  '4boysbrown',       '4yodeji',          '500KFlea',          '50ShadesOfGarza',
  '51nation_',        '52success',        '54321sunilkumar',   '55Dkt',
  '5808_623',         '5brittneytaylor',  '5mTKO',             '5nBOI',
  '666Beastism999',   '67fisherman',      '6Dariusglover',     '6Magazine',
  ... 9675 more items
]
```

Step 7. 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*:

```
sample> db.users.find({user_name:"ActionSportsJax"})
[
  {
    _id: ObjectId('579670bfc38159226b4c8e47'),
    user_name: 'ActionSportsJax',
    retweet_count: 0,
    tweet_followers_count: 3539,
    source: '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>',
    coordinates: null,
    tweet_mentioned_count: 2,
    tweet_ID: '757667800521531393',
    tweet_text: `RT @wwbrown19: I'm watching the great broadcast from @actionsportsjax on St. Augustine football and ask
ed myself "How on earth did we stop…`,
    user: {
      CreatedAt: ISODate('2009-04-27T20:52:10.000Z'),
      FavouritesCount: 120,
      FollowersCount: 3539,
      FriendsCount: 476,
      UserId: 35857042,
      Location: 'Jacksonville, FL'
    }
  }
]
```

Step 8. 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:

```
sample> 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:

```
sample> db.users.find({user_name:"ActionSportsJax"}, {tweet_ID:1, _id:0})
[ { tweet_ID: '757667800521531393' } ]
```

Step 9. 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:

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

sample>
```

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

```
sample> db.users.find({tweet_text:/FIFA/})
[
  {
    _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('2011-12-27T09:04:01.000Z'),
      FavouritesCount: 5223,
      FollowersCount: 461,
      FriendsCount: 619,
      UserId: 447818090,
      Location: '501'
    }
  },
  {
    _id: ObjectId('578ffa917eb9513f4f55a939'),
    user_name: 'Tonkatol',
    retweet_count: 0,
```

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

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

Step 10. 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():*

```
sample> db.users.createIndex({"tweet_text":"text"})
tweet_text_text
```

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:*

```
sample> 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:*

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

Step 11. **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:

```
sample> db.users.find({tweet_mentioned_count: {$gt : 6}})
[
  {
    _id: ObjectId('57966e26c3815920e1131b03'),
    user_name: 'marshallrupe',
    retweet_count: 0,
    tweet_followers_count: 215,
    source: '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>',
    coordinates: null,
    tweet_mentioned_count: 7,
    tweet_ID: '757665013817409536',
    tweet_text: 'RT @BrianBorrison: https://t.co/Z74VgYUUbB   Mepo Football Online Store  @marshallrupe @da_carlos30 @TrissyTim @BrockHiltonb @ZachZippe @ro…',
    user: {
      CreatedAt: ISODate('2013-11-29T16:00:22.000Z'),
      FavouritesCount: 326,
      FollowersCount: 215,
      FriendsCount: 165,
      UserId: -2073286695,
      Location: null
    }
  },
  {
    _id: ObjectId('57966ecfc3815920e1132053'),
    user_name: 'marvheys',
    retweet_count: 0,
    tweet_followers_count: 1215,
    source: '<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>',
```

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*:

```
sample> 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:

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

Step 12: Exiting the container. Once you've finished your analysis, run *exit* to exit the MongoDB shell. Then, go to Docker Desktop to stop the container. We recommend you not to delete it, as you will use it for the final project.

Go to next item   ✓ Completed

Like   Dislike   Report an issue