

Homework 3



April 16, 2019

Xiang yu

CS 5513 Gia-Loi Gruenwald

Contents

[Query 1 Insert an (news or feature) article into the archive 3](#_Toc6309912)

[Query1.js file: 3](#_Toc6309913)

[Query1 output result 4](#_Toc6309914)

[Query 2 Retrieve the latest news from the archive. 5](#_Toc6309915)

[Query 2.js: 5](#_Toc6309916)

[Query 2 Output: 5](#_Toc6309917)

[Query 3 Retrieve the top 10 most read news in the past month 6](#_Toc6309918)

[Query3.js: 6](#_Toc6309919)

[Query3 output: 6](#_Toc6309920)

[Query 4 Add a comment to an article. 7](#_Toc6309921)

[Query4.js: 7](#_Toc6309922)

[Query 4 output: 8](#_Toc6309923)

[Query 5 Add a story to the ‘similar stories’ section of all articles that contain a given word and that have been published within the past 2 years. 9](#_Toc6309924)

[Query5.js : 9](#_Toc6309925)

[Query 5 output: 10](#_Toc6309926)

[Dataset Generation 11](#_Toc6309927)

[articleData.py: 11](#_Toc6309928)

[Dataset Generation Output: 14](#_Toc6309929)

[Data Insertion 15](#_Toc6309930)

[dataInsertion.js: (First five insertions display) 15](#_Toc6309931)

[Data Insertion Output: 16](#_Toc6309932)

[Txt parser 18](#_Toc6309933)

[Task f. Set Replicates 20](#_Toc6309934)

[rs.status() output 21](#_Toc6309935)

[Replication Design Explanation: 22](#_Toc6309936)

[Task G 23](#_Toc6309937)

[Immediately connect to another active MongoDB server in your replica set and run the command rs.status(). Take a screenshot that shows that no primary has yet been elected. 23](#_Toc6309938)

[Wait until a primary has been elected, and then run your queries one time each again with the remaining nodes. Provide screenshots for the output of each run 25](#_Toc6309939)

[Run query1 27](#_Toc6309940)

[Run Query 2: 27](#_Toc6309941)

[Run Query 3: 27](#_Toc6309942)

[Run Query 4: 28](#_Toc6309943)

[Run Query 5 28](#_Toc6309944)

[BONUS PROBLEM 29](#_Toc6309945)

[USE CASE 31](#_Toc6309946)

[REFERENCE 32](#_Toc6309947)

# Query 1 Insert an (news or feature) article into the archive

## Query1.js file:

db.article.insert({

"article\_type": "news article",

"article\_section": "Sports",

"article\_id": 75,

"post\_on\_date": {

"day": 31,

"month": 12,

"year": 2017,

"time": '23:59:59'

},

"reportor\_name": "Bridgette Nolan",

"similar\_stories": ["thedaily.com/sports/oklahomafootball-sooners-land-kentucky-graduate-transfer.html",

"thedaily.com/sports/oklahoma-football-sooners-look-tofill-void.html"],

"num\_times\_read": 11357,

"article\_text": "Despite the Cowboys last loss last season, they have several options to replace their linebacker…",

"comments": [

{

"comment\_id": 12,

"article\_id": 75,

"user\_id": "dtillman@gmail.com",

"post\_on\_date": {

"day": 31,

"month": 12,

"year": 2017,

"time": "09:29:31"

},

"comment\_text": "Keep up the work with your fake news!",

"score": -99

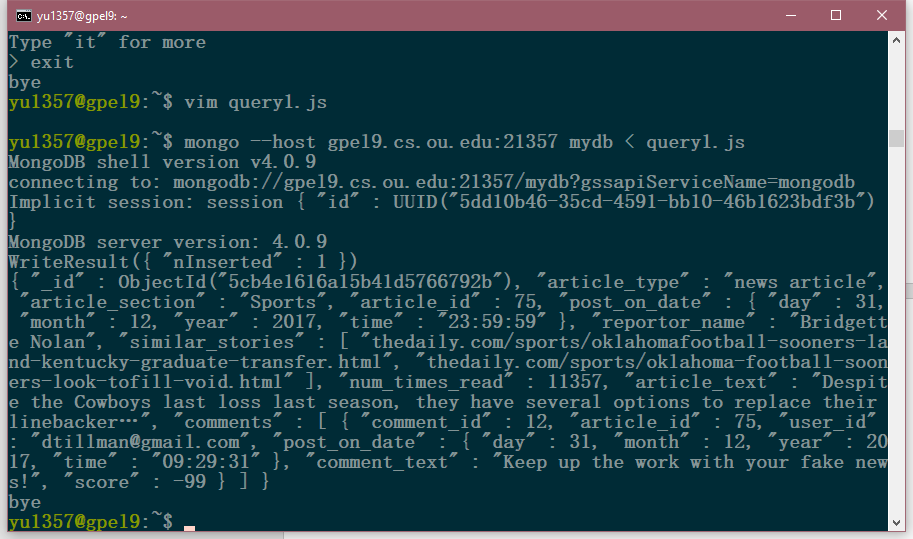
}

]

});

db.article.find({"article\_id": 75});

## Query1 output result

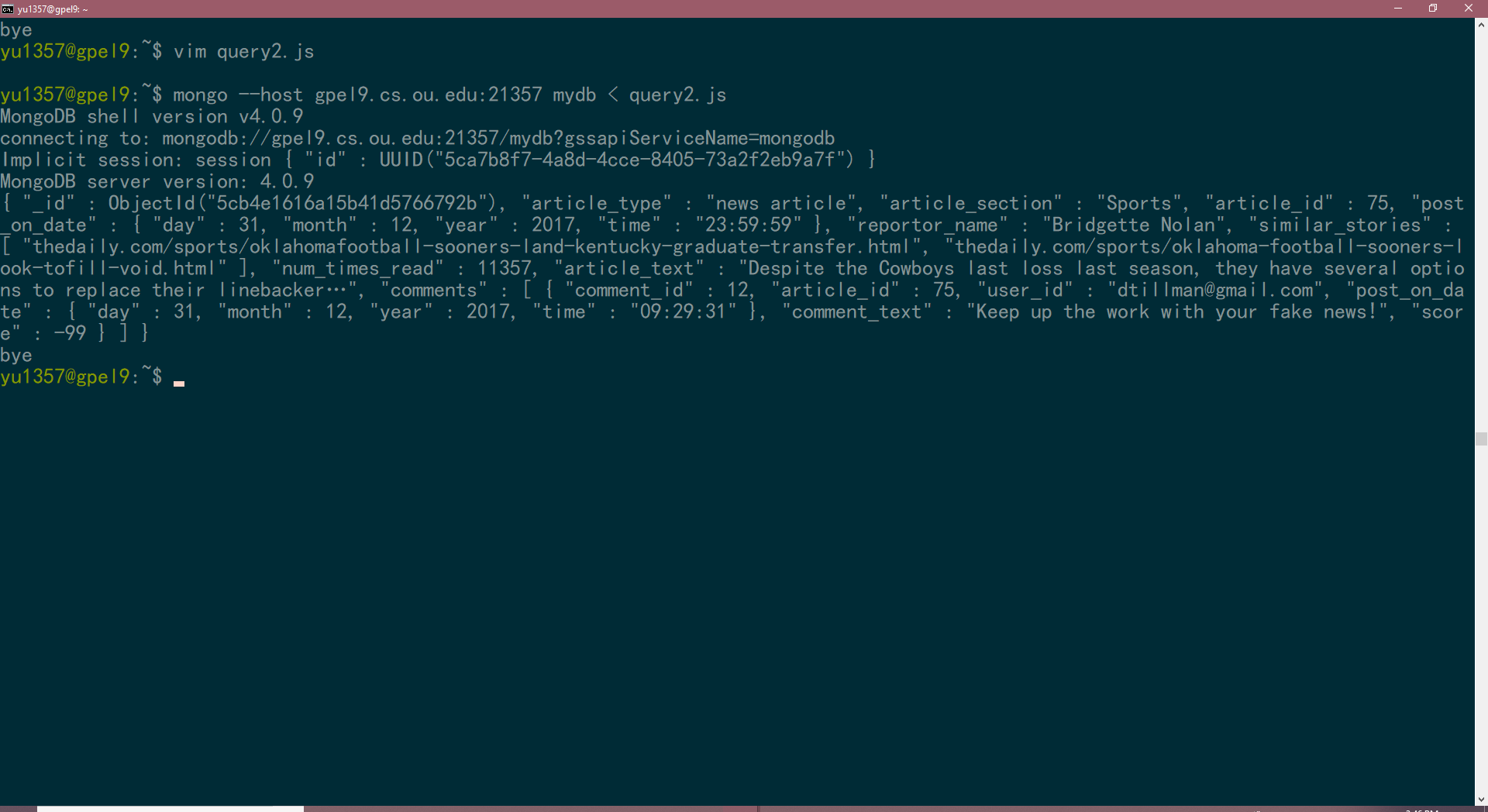


# Query 2 Retrieve the latest news from the archive.

## Query 2.js:

db.article.find({}).sort({"post\_on\_date.year":-1, "post\_on\_date.month":-1, "post\_on\_date.date":-1, "post\_on\_date.time":-1,}).limit(1)

## Query 2 Output:

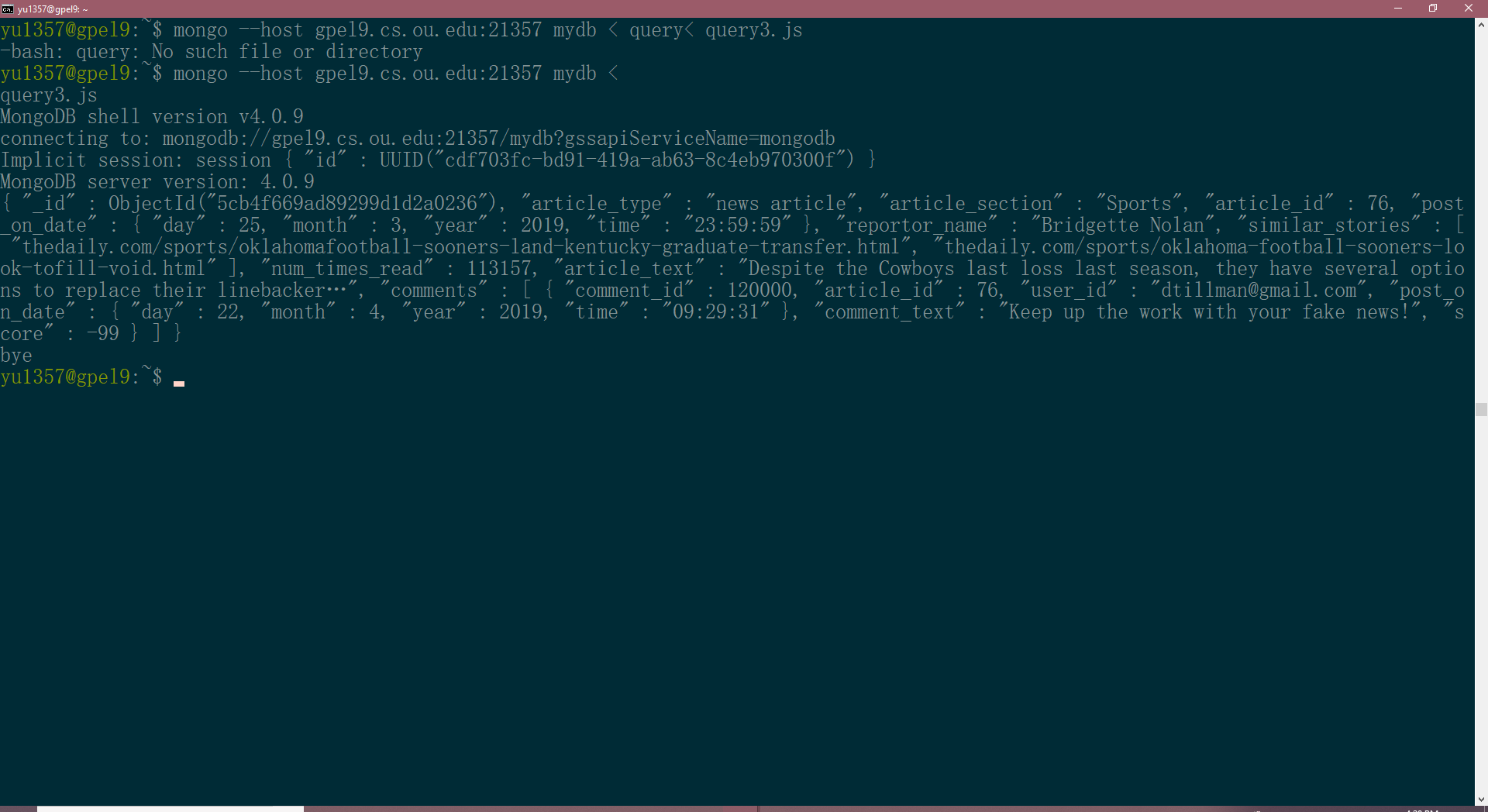


# Query 3 Retrieve the top 10 most read news in the past month

## Query3.js:

db.article.find({"post\_on\_date.year" : 2019, "post\_on\_date.month":3}).sort({"num\_times\_read":-1}).limit(10)

## Query3 output:



# Query 4 Add a comment to an article.

## Query4.js:

db.article.update({"article\_id": 76},{$push: {"comments": {

"comment\_id": 2840,

"article\_id":75,

"user\_id": "haha@qq.com",

"post\_on\_date":{

"day":22,

"month": 11,

"year": 2018,

"time": "09:29:31"

},

"comment\_text": "nothings special",

"score": 10

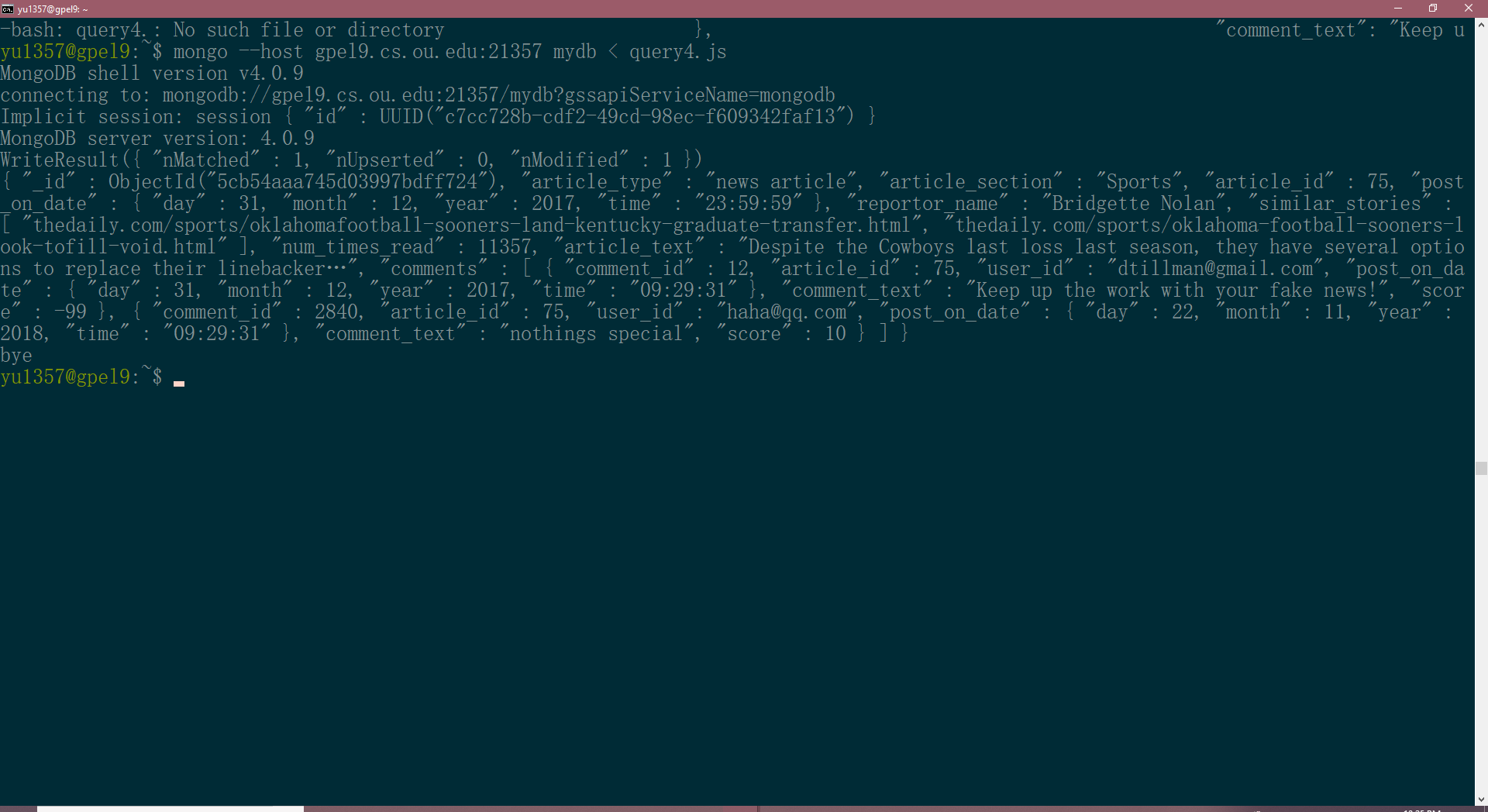
}

}

});

db.article.find({"article\_id": 76});

## Query 4 output:



# Query 5 Add a story to the ‘similar stories’ section of all articles that contain a given word and that have been published within the past 2 years.

## Query5.js :

Give a word: “house”. **Add story “*How to build* a house”** **to all article that contain a word “house” and that published from 2017-2019 (in past 2 years)**

db.article.update(

{

"post\_on\_date.year" : {$gt: 2016},

"article\_text" : {$regex : ".\*house.\*"}

},

{

$addToSet : {

"similar\_stories" : {

$each: ["How to build a house"]

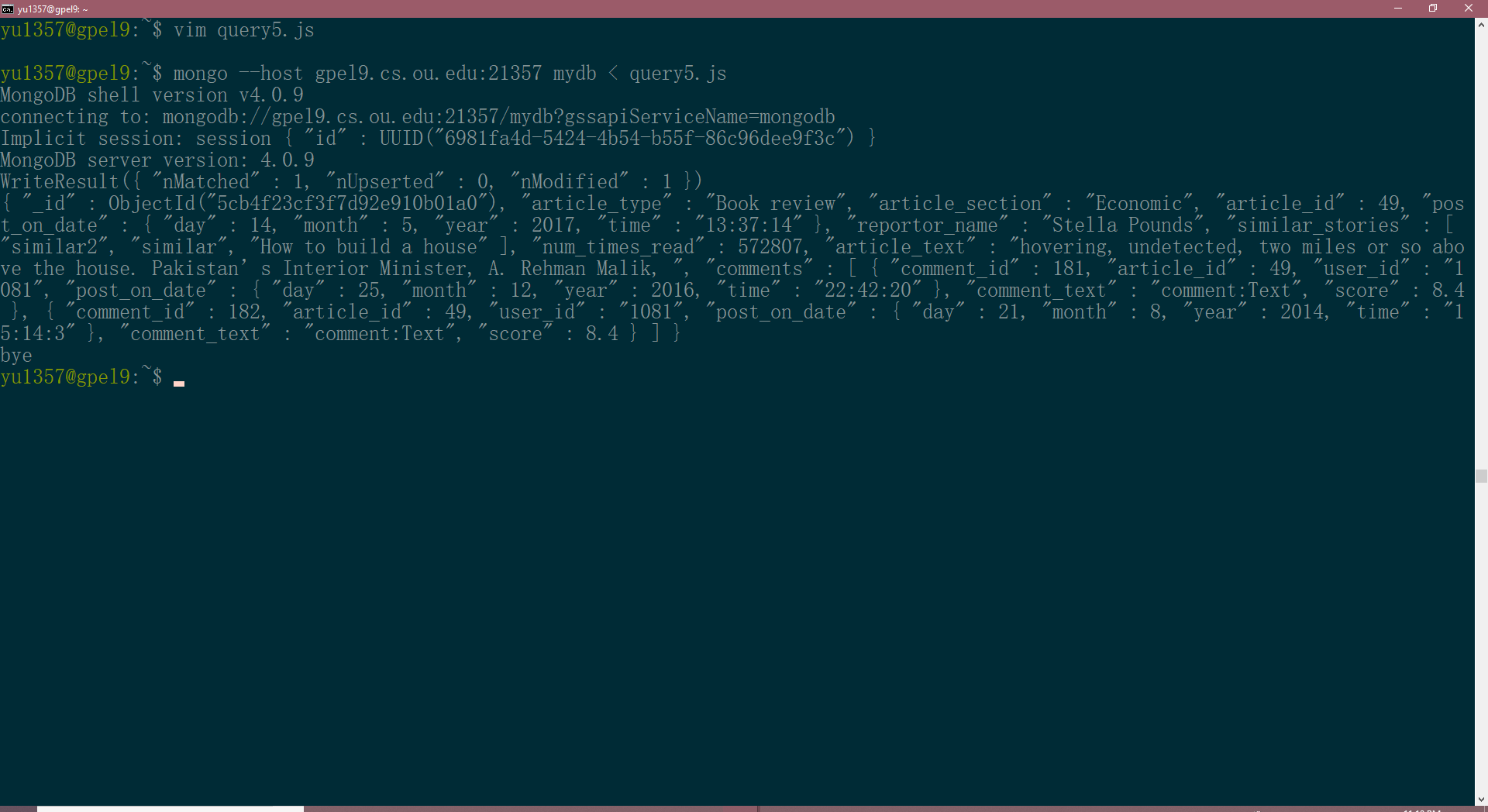
}

}

});

db.article.find({"post\_on\_date.year" : {$gt: 2016} ,"article\_text" : {$regex : ".\*house.\*"}});

## Query 5 output:



# Dataset Generation

## articleData.py:

import random

import names

from requests.exceptions import HTTPError

import codecs

article\_type = ['Original research', 'Review article', 'Clinical case study', 'Clinical case study', 'Clinical trial', 'Perspective opinion and commentary', 'Book review']

article\_section = ['Sports', 'Political', 'Economic', 'Entertainment', 'World', 'Technology', 'Faith & Values','Autos', 'Travel + Outdoors', 'Food + Drink','House + Home', 'Fitness + Well-being']

def new\_article\_data(number):

data\_list = []

j = 1

for i in range(number):

data = {}

while True:

try:

j = j + 1

break

except HTTPError:

j = j + 1

data['article\_type'] = random.choice(article\_type)

data['article\_section'] = random.choice(article\_section)

data['article\_id'] = i

data['post\_time'] = (random.randrange(1, 31), random.randrange(1, 13), random.randrange(2000, 2019), (str(random.randrange(0, 24)) + ':' + str(random.randrange(0, 60)) + ':' + str(random.randrange(0,60))))

data['reporter\_name'] = names.get\_full\_name()

data['similar article'] = []

data['similar article'].append('similar2')

data['similar article'].append('similar')

data['number\_times\_read'] = random.randrange(0,1000000)

lines = open('file.txt').read().splitlines()

data['article\_text'] = random.choice(lines)

        #comments

temp\_list = []

for k in range(2):

temp = ()

commentID = 100 + j + k

articleID = data['article\_id']

userID = 1000 + j

postedDate = (random.randrange(1, 30), random.randrange(1, 13), random.randrange(2000, 2019),)

time = (str(random.randrange(0, 24)) + ':' + str(random.randrange(0, 60)) + ':' + str(random.randrange(0,60)),)

commentText = 'comment:Text'

score = random.randrange(2, 100)/10

temp += (commentID, articleID, userID) + (postedDate + time, ) + (commentText,) + (score,)

temp\_list.append(temp)

data['comments'] = temp\_list

data\_list.append(data)

# write to file: articleData.txt

with codecs.open('./articleData.txt', 'a') as f:

for data in data\_list:

keys = list(data.keys())

for i in range(len(keys)):

value = data[keys[i]]

f.write(str(value))

if i != len(keys) - 1:

f.write(', ')

else:

pass

f.write('\n')

# a) print the first 5 entries

with codecs.open('articleData.txt', 'r') as f:

for i in range(5):

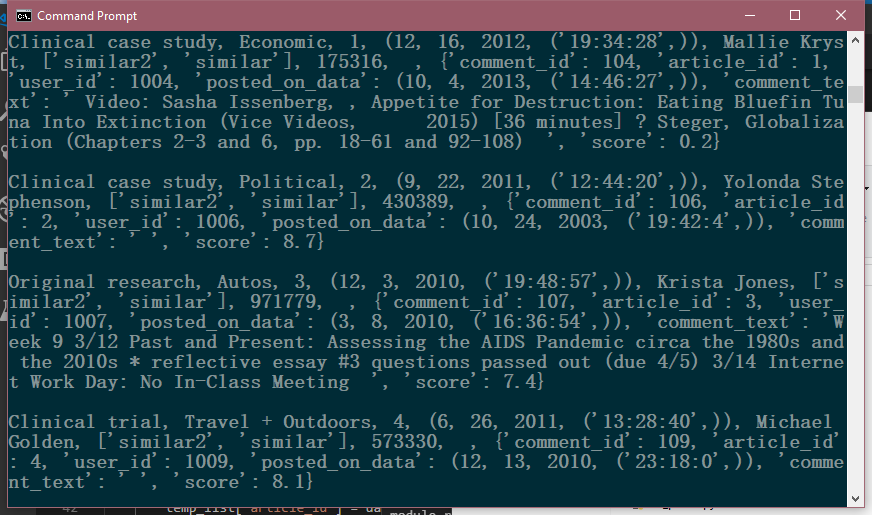
print(f.readline())

if \_\_name\_\_ == "\_\_main\_\_":

# create 50 article datas

new\_article\_data(50)

## Dataset Generation Output:



# Data Insertion

## dataInsertion.js: (First five insertions display)

connection = new Mongo()

db = connection.getDB('mydb')

db.article.insert({"article\_type": "Clinical case study", "article\_section": "Entertainment", "article\_id": 0, "post\_on\_date": {"day": 15, "month": 4, "year": 2011, "time": '4:17:15'}, "reportor\_name": "Robert Cox", "similar\_stories": ['similar2', 'similar'], "num\_times\_read": 641088, "article\_text": "father-in-law, his motherin-law, a lieutenant, and seven bodyguard", "comments": [{"comment\_id": 103, "article\_id": 0, "user\_id": "1003", "post\_on\_date": {"day": 26, "month": 5, "year": 2014, "time": '2:34:0'}, "comment\_text": 'comment:Text', "score": 6.1}, {"comment\_id": 104, "article\_id": 0, "user\_id": "1003", "post\_on\_date": {"day": 8, "month": 10, "year": 2006, "time": '2:53:32'}, "comment\_text": 'comment:Text', "score": 6.6}]})

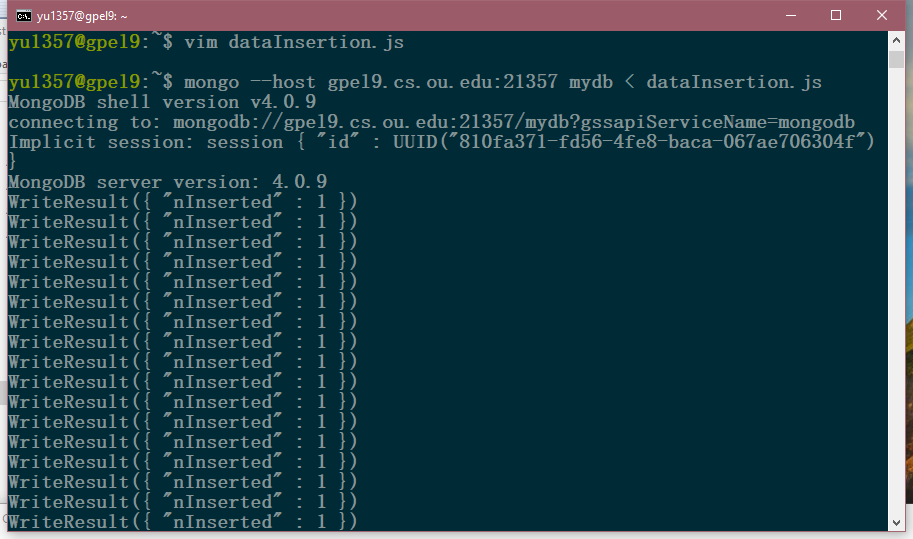
db.article.insert({"article\_type": "Clinical case study", "article\_section": "Travel + Outdoors", "article\_id": 1, "post\_on\_date": {"day": 28, "month": 5, "year": 2004, "time": '16:46:26'}, "reportor\_name": "Crystal Dole", "similar\_stories": ['similar2', 'similar'], "num\_times\_read": 22995, "article\_text": "told me recently that Mehsud was resting on his back. Malik, using his hands to make a picture frame, explained that ", "comments": [{"comment\_id": 104, "article\_id": 1, "user\_id": "1004", "post\_on\_date": {"day": 18, "month": 11, "year": 2006, "time": '12:49:44'}, "comment\_text": 'comment:Text', "score": 2.0}, {"comment\_id": 105, "article\_id": 1, "user\_id": "1004", "post\_on\_date": {"day": 19, "month": 5, "year": 2014, "time": '6:35:11'}, "comment\_text": 'comment:Text', "score": 9.7}]})

db.article.insert({"article\_type": "Perspective opinion and commentary", "article\_section": "Entertainment", "article\_id": 2, "post\_on\_date": {"day": 13, "month": 3, "year": 2008, "time": '10:11:6'}, "reportor\_name": "Katherine Ruiz", "similar\_stories": ['similar2', 'similar'], "num\_times\_read": 46472, "article\_text": "Mehsud\u2019s death is instant. Nor, described unambiguously as a terrorist, does he seem undeserving ", "comments": [{"comment\_id": 106, "article\_id": 2, "user\_id": "1006", "post\_on\_date": {"day": 13, "month": 1, "year": 2015, "time": '14:16:25'}, "comment\_text": 'comment:Text', "score": 7.7}, {"comment\_id": 107, "article\_id": 2, "user\_id": "1006", "post\_on\_date": {"day": 4, "month": 4, "year": 2006, "time": '23:7:58'}, "comment\_text": 'comment:Text', "score": 1.7}]})

db.article.insert({"article\_type": "Perspective opinion and commentary", "article\_section": "Fitness + Well-being", "article\_id": 3, "post\_on\_date": {"day": 7, "month": 10, "year": 2014, "time": '3:45:23'}, "reportor\_name": "Carrie Robinson", "similar\_stories": ['similar2', 'similar'], "num\_times\_read": 252900, "article\_text": "Study. It is hard to imagine an environment that is more stimulating or more congenial", "comments": [{"comment\_id": 107, "article\_id": 3, "user\_id": "1007", "post\_on\_date": {"day": 7, "month": 9, "year": 2001, "time": '21:16:46'}, "comment\_text": 'comment:Text', "score": 5.1}, {"comment\_id": 108, "article\_id": 3, "user\_id": "1007", "post\_on\_date": {"day": 2, "month": 12, "year": 2011, "time": '13:41:12'}, "comment\_text": 'comment:Text', "score": 9.3}]})

db.article.insert({"article\_type": "Perspective opinion and commentary", "article\_section": "Entertainment", "article\_id": 4, "post\_on\_date": {"day": 11, "month": 5, "year": 2006, "time": '2:59:16'}, "reportor\_name": "Lucy Harper", "similar\_stories": ['similar2', 'similar'], "num\_times\_read": 411832, "article\_text": "This book was written when I spent a year at the Princeton Institute for Advanced", "comments": [{"comment\_id": 109, "article\_id": 4, "user\_id": "1009", "post\_on\_date": {"day": 20, "month": 5, "year": 2003, "time": '18:54:32'}, "comment\_text": 'comment:Text', "score": 7.6}, {"comment\_id": 110, "article\_id": 4, "user\_id": "1009", "post\_on\_date": {"day": 22, "month": 11, "year": 2014, "time": '4:15:50'}, "comment\_text": 'comment:Text', "score": 3.4}]})

## Data Insertion Output:



# Txt parser

import codecs

import json

def txt\_parser():

with codecs.open('./articleData.txt', 'r') as f1:

with codecs.open('./dataInsertion.js', 'a') as f2:

f2.write('connection = new Mongo()\n')

f2.write('db = connection.getDB(\'mydb\')\n')

lines = f1.readlines()

for line in lines:

lbrackets = [pos for pos, char in enumerate(line) if char == '[']

rbrackets = [pos for pos, char in enumerate(line) if char == ']']

post\_on\_date = {}

left = line.find(', (')

right = line[left:].find(')') + left

post\_on\_date\_list = [word.strip() for word in line[left + 3: right].split(',')]

post\_on\_date['day'] = int(post\_on\_date\_list[0])

post\_on\_date['month'] = int(post\_on\_date\_list[1])

post\_on\_date['year'] = int(post\_on\_date\_list[2])

post\_on\_date['time'] = post\_on\_date\_list[3]

similars = line[lbrackets[0]: rbrackets[0] + 1]

commentRaw = line[lbrackets[1] : rbrackets[1]]

commentRaw = commentRaw.replace('[', '')

commentRaw = commentRaw.replace(']', '')

commentRaw = commentRaw.replace('(', '')

commentRaw = commentRaw.replace(')', '')

comments\_list = [word.strip() for word in commentRaw.split(',')]

comments = []

numberOfcomments = len(comments\_list)//9

for j in range(numberOfcomments):

comment = {}

comment['comment\_id'] = int(comments\_list[j\*9 + 0])

comment['article\_id'] = int(comments\_list[j\*9 + 1])

comment['user\_id'] = comments\_list[j\*9 + 2]

comment['post\_on\_date'] = {}

comment['post\_on\_date']['day'] = int(comments\_list[j\*9 + 3])

comment['post\_on\_date']['month'] = int(comments\_list[j\*9 + 4])

comment['post\_on\_date']['year'] = int(comments\_list[j\*9 + 5])

comment['post\_on\_date']['time'] = comments\_list[j\*9 + 6]

comment['comment\_text'] = comments\_list[j\*9 + 7]

comment['score'] = float(comments\_list[j\*9 + 8])

comments.append(comment)

lbrackets = [pos for pos, char in enumerate(line) if char == '[']

rbrackets = [pos for pos, char in enumerate(line) if char == ']']

restString = line[0: left]

restString += line[right : lbrackets[0] - 2]

restString = restString.replace(')','')

restString = restString.replace('(','')

words = [word.strip() for word in restString.split(',')]

readStartIndex = rbrackets[0] + 3

readEndIndex = line[readStartIndex:].find(',') + readStartIndex

record = {}

record['article\_type'] = words[0]

record['article\_section'] = words[1]

record['article\_id'] = int(words[2])

record['post\_on\_date'] = post\_on\_date

record['reportor\_name'] = words[3]

record['similar\_stories'] = similars

record['num\_times\_read'] = int(line[readStartIndex:readEndIndex])

record['article\_text'] = line[readEndIndex + 2 : lbrackets[1] - 2]

record['comments'] = comments

recordRaw = json.dumps(record)

recordRaw = recordRaw.replace('"[', '[')

recordRaw = recordRaw.replace(']"', ']')

recordRaw = recordRaw.replace('"\'', '\'')

recordRaw = recordRaw.replace('\'"', '\'')

record = 'db.article.insert(' + recordRaw + ')\n'

f2.write(record)

if \_\_name\_\_ == '\_\_main\_\_':

txt\_parser()

# Task f. Set Replicates

mkdir -p /home/yu1357/data/gpel8

mongod --fork --dbpath ~/data/gpel8/ --port 21357 --logpath mongodb.log --replSet myDBReplicaSet --bind\_ip\_all

mkdir -p /home/yu1357/data/gpel9

mongod --fork --dbpath ~/data/gpel9/ --port 21357 --logpath mongodb.log --replSet myDBReplicaSet --bind\_ip\_all

mkdir -p /home/yu1357/data/gpel11

mongod --fork --dbpath ~/data/gpel11/ --port 21357 --logpath mongodb.log --replSet myDBReplicaSet --bind\_ip\_all

HOST:

mongo –host gpel9.cs.ou.edu --port 21357 mydb

rs.initiate()

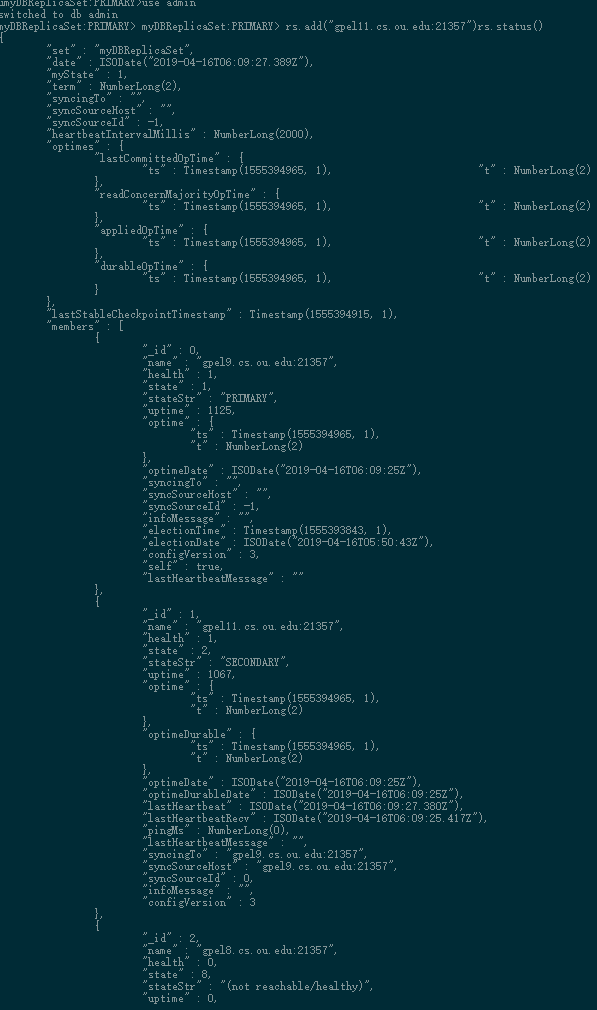
myDBReplicaSet:PRIMARY> rs.add("gpel8.cs.ou.edu:21357")

myDBReplicaSet:PRIMARY> rs.add("gpel11.cs.ou.edu:21357")

use admin

rs.status()

## rs.status() output





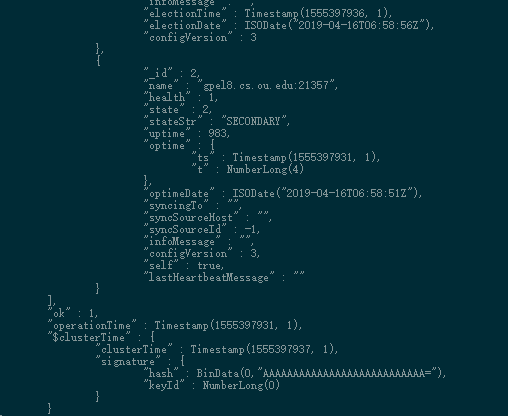
## Replication Design Explanation:

I have created multiple mongod processes sharing same data. If there is one node fail, for example if primary node gpel9 fails, and then the replica set will detect this situation and elect a new primary node, therefore, user could continue access data he/she needs. What is more, if secondary node fails like gpel8, the database still can be accessible. This replica design could increase availability of database.

# Task G

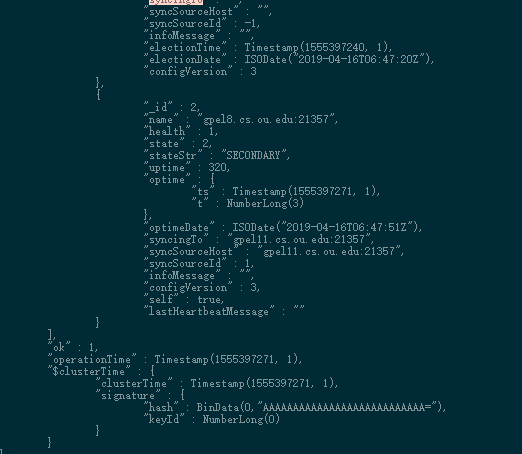
## Immediately connect to another active MongoDB server in your replica set and run the command rs.status(). Take a screenshot that shows that no primary has yet been elected.



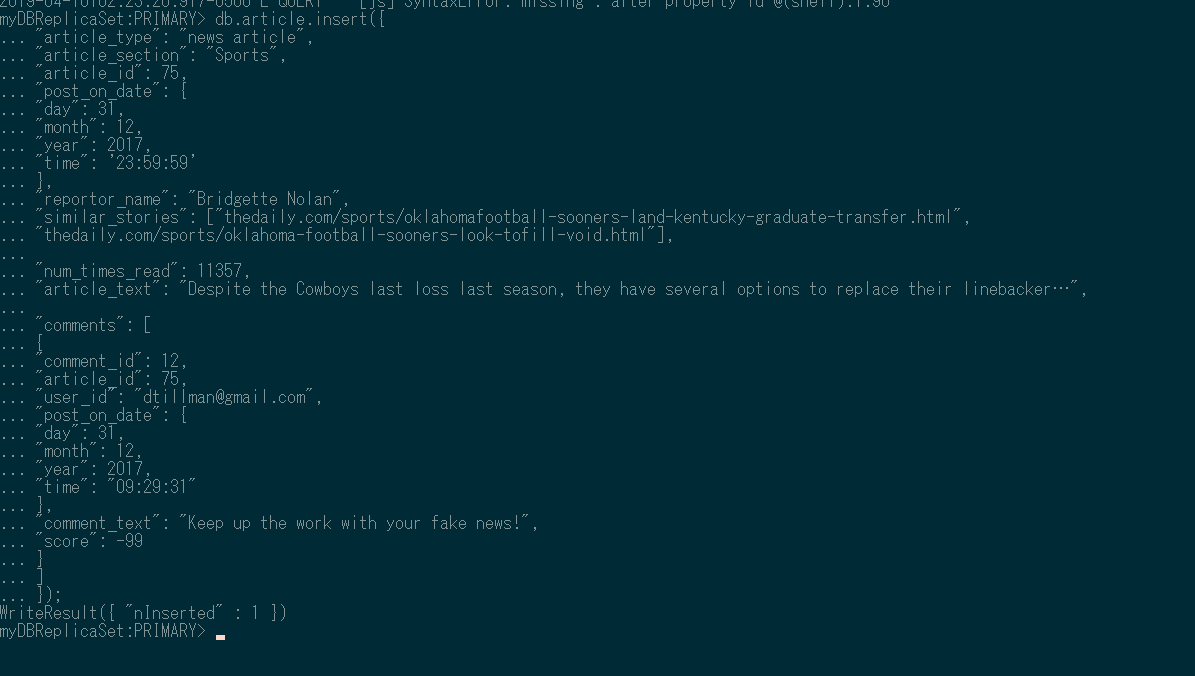


## Wait until a primary has been elected, and then run your queries one time each again with the remaining nodes. Provide screenshots for the output of each run

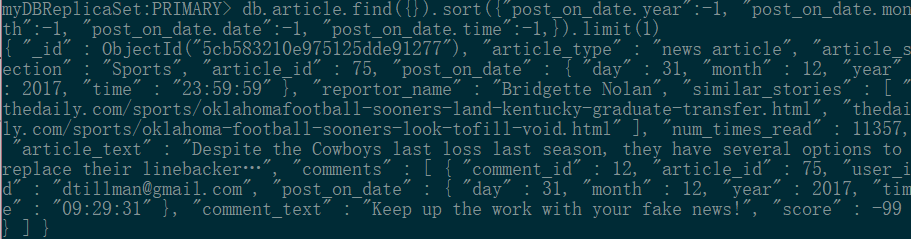




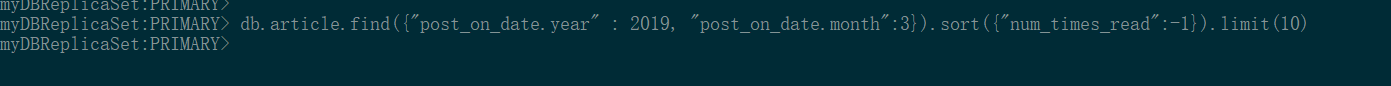
## Run query1



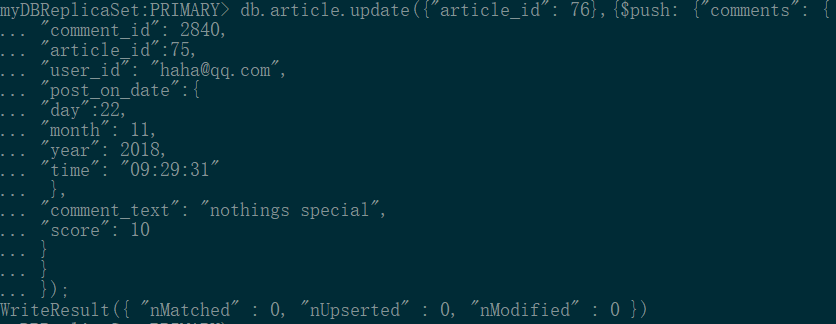
## Run Query 2:



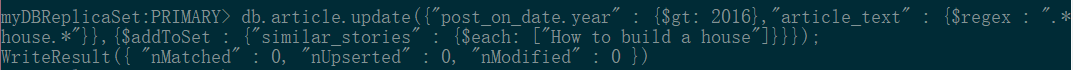
## Run Query 3:



## Run Query 4:



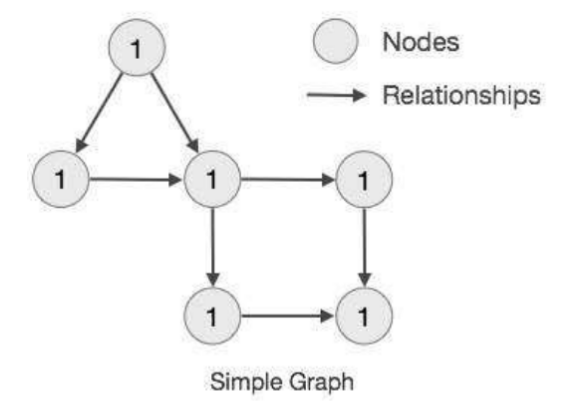
## Run Query 5



# BONUS PROBLEM

**Neo4j graph database system:**

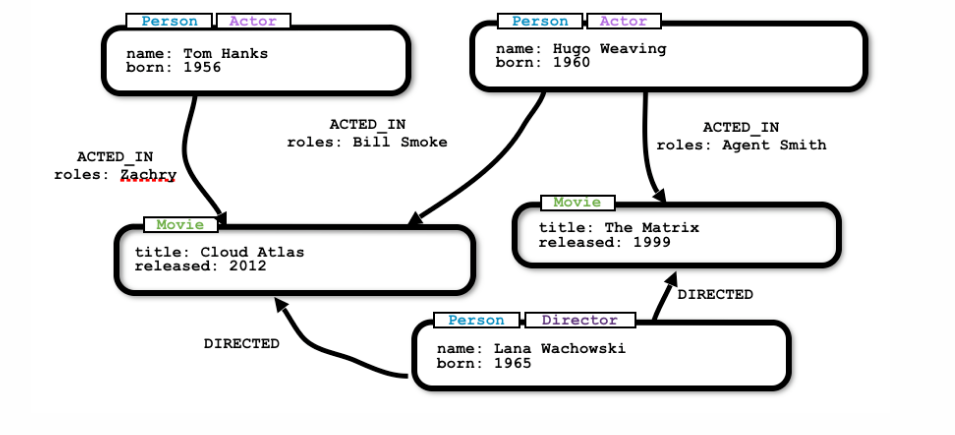
Neo4j graph database system data model is consisted by ***Nodes, Label, Relationships*** and ***Properties***.



From above, the ***nodes*** are represented by graph circles and relationships are represented by arrows. What is more, because using arrows to represent ***relationship***, it means ***relationship*** in this graph database are directional. And ***properties*** (key-value) are represented by ***node’s*** data.

**Graph Data Model = Whiteboard-Friendly**

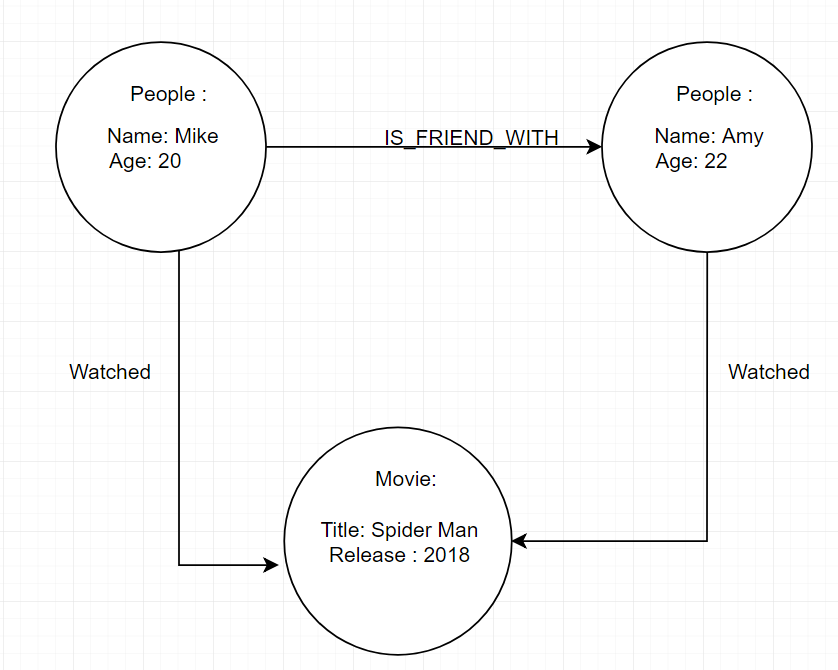
The Whiteboard-Friendly means that when people design data model, they usually create nodes, and then using arrow relationships to connect them instead of modifying the data model to fit a normalized table structure.



Above is the move graph data. ***Label*** is the type of nodes such as Person Actor. The database using directional edges to represents relationships. The set of edges are store in the systems. Once we need to change data, for example, (from above figure) the director Lana Wachowski direct a new movie called “666”, we just need to create a movie ”666” node and use arrow from Lana to “666” to connect them and put DIRECTED on the arrow.

## USE CASE

Assume that Mike and Amy are friends, and they have watched movie “Spider Man”. In the graph database, the data model is shown below.



Once we want to add more people who has watched this people, only thing we need to is that create new people nodes, and put the arrow to movie. We also could use people node to represents more relationship, such as that new node people John is teaching of Amy, we could use arrow to represent this relationship. We don’t have to worry about one-one, many to many or one to many those relationship in schema. We also could use graph travel to list people that Mike may know in the database. And relational database may not use friend may know recommendation function. In relational database, if you want to delete a record whose attribute are foreign key of other record, you cannot delete this record. In graph database, you could delete the relationship and nodes to represents remove instead of modify data model.

**Bonus Problems referenced from [1] Neo4j documentation**

## REFERENCE

[1] “Data Modeling Concepts and Techniques | Neo4j.” *Neo4j Graph Database Platform*, neo4j.com/developer/guide-data-modeling/.