CWID: A20526972

CSP-554 Big Data Technologies

Bairi Rohith Reddy – Assignment 9

HBase

Start up a Hadoop cluster as previously, but instead of choosing the “Core Hadoop” configuration chose the “HBase” configuration

- **ssh -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer hadoop@ec2-34-200-220-199.compute-1.amazonaws.com**

Log on to the master Hadoop EC2 VM as per previous assignments and enter ‘hbase shell’ to start the HBase shell.

- **hbase shell**

Exercise 1)

Create an HBase table with the following characteristics

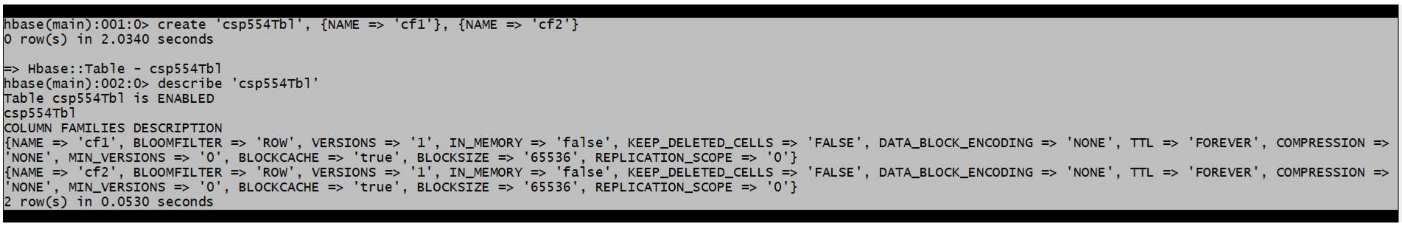
Table Name: csp554Tbl

First column family: cf1

Second column family: cf2

Then execute the **DESCRIBE** command on the table and return command you wrote and the output as the results of this exercise.

- **create 'csp554Tbl', {NAME => 'cf1'}, {NAME => 'cf2'}**
- **describe 'csp554Tbl'**



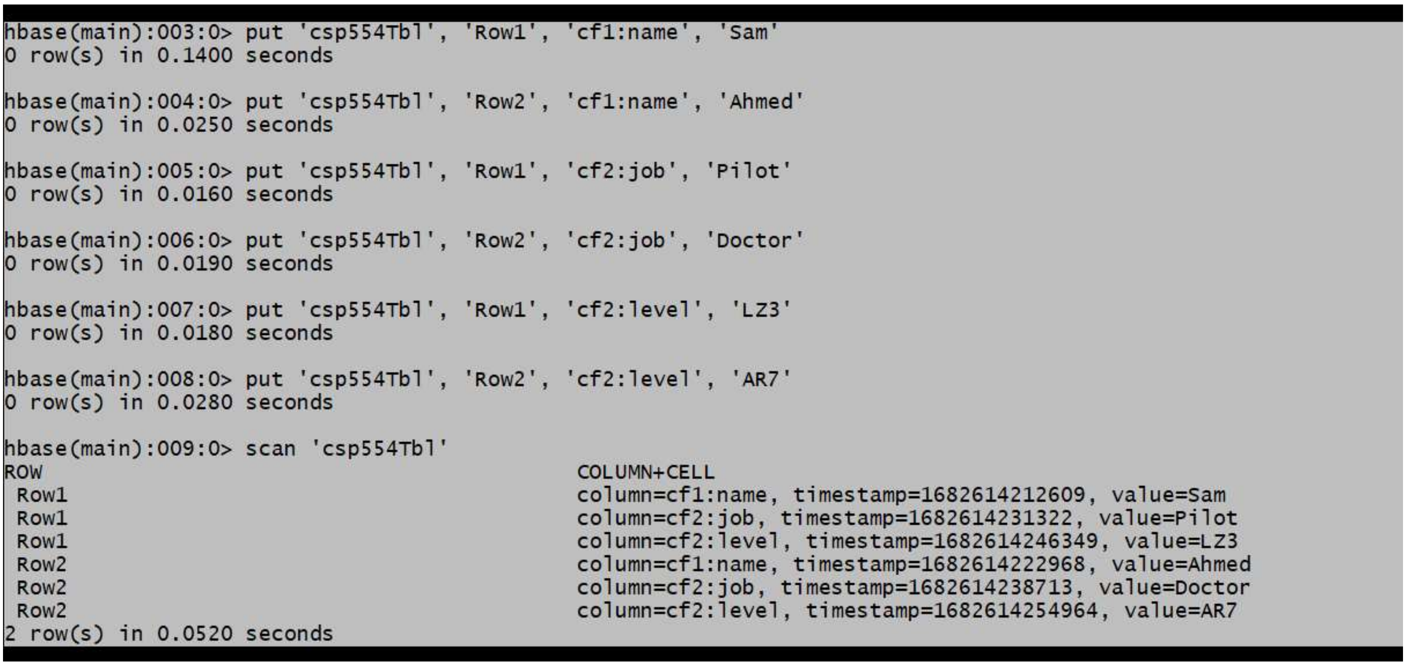
```
hbase(main):001:0> create 'csp554Tbl', {NAME => 'cf1'}, {NAME => 'cf2'}
0 row(s) in 2.0340 seconds

=> Hbase::Table - csp554Tbl
hbase(main):002:0> describe 'csp554Tbl'
Table csp554Tbl is ENABLED
csp554Tbl
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION =>
'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'cf2', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION =>
'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.0530 seconds
```

Exercise 2)

Put the following data into the table created in exercise 1:

- **put 'csp554Tbl', 'Row1', 'cf1:name', 'Sam'**
- **put 'csp554Tbl','Row2','cf1:name','Ahmed'**
- **put 'csp554Tbl','Row1','cf2:job','Pilot'**
- **put 'csp554Tbl','Row2','cf2:job','Doctor'**
- **put 'csp554Tbl','Row1','cf2:level','LZ3'**
- **put 'csp554Tbl','Row2','cf2:level','AR7'**

- **scan 'csp554Tbl'**

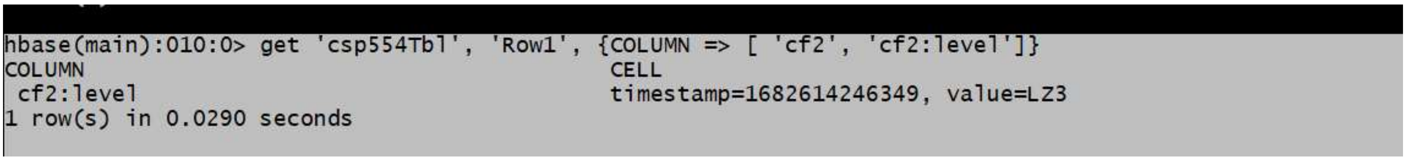


```
hbase(main):003:0> put 'csp554Tbl', 'Row1', 'cf1:name', 'Sam'
0 row(s) in 0.1400 seconds

hbase(main):004:0> put 'csp554Tbl', 'Row2', 'cf1:name', 'Ahmed'
0 row(s) in 0.0250 seconds

hbase(main):005:0> put 'csp554Tbl', 'Row1', 'cf2:job', 'Pilot'
0 row(s) in 0.0160 seconds

hbase(main):006:0> put 'csp554Tbl', 'Row2', 'cf2:job', 'Doctor'
0 row(s) in 0.0190 seconds

hbase(main):007:0> put 'csp554Tbl', 'Row1', 'cf2:level', 'LZ3'
0 row(s) in 0.0180 seconds

hbase(main):008:0> put 'csp554Tbl', 'Row2', 'cf2:level', 'AR7'
0 row(s) in 0.0280 seconds

hbase(main):009:0> scan 'csp554Tbl'
ROW                                          COLUMN+CELL
 Row1                                        column=cf1:name, timestamp=1682614212609, value=Sam
 Row1                                        column=cf2:job, timestamp=1682614231322, value=Pilot
 Row1                                        column=cf2:level, timestamp=1682614246349, value=LZ3
 Row2                                        column=cf1:name, timestamp=1682614222968, value=Ahmed
 Row2                                        column=cf2:job, timestamp=1682614238713, value=Doctor
 Row2                                        column=cf2:level, timestamp=1682614254964, value=AR7
2 row(s) in 0.0520 seconds
```

Exercise 3) (1 point)

Using the above table write a command that will get the value associated with row (Row1), column family (cf2) and column/qualifier (level). Provide the command and its result as the output of this exercise.

- **get 'csp554Tbl', 'Row1', {COLUMN => ['cf2', 'cf2:level']}**

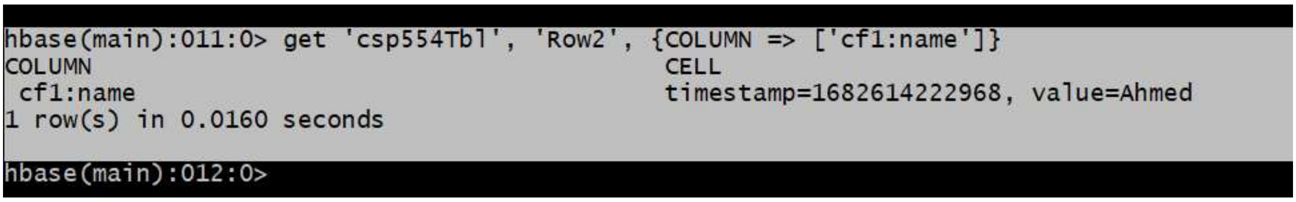


```
hbase(main):010:0> get 'csp554Tbl', 'Row1', {COLUMN => [ 'cf2', 'cf2:level']}
COLUMN                                        CELL
 cf2:level                                    timestamp=1682614246349, value=LZ3
1 row(s) in 0.0290 seconds
```

Exercise 4) (1 point)

Using the above table write command that will get the value associated with row (Row2), column family (cf1) and column/qualifier (name). Provide the command and its result as the output of this exercise.

- **get 'csp554Tbl', 'Row2',{COLUMN=> 'cf1:name'}**

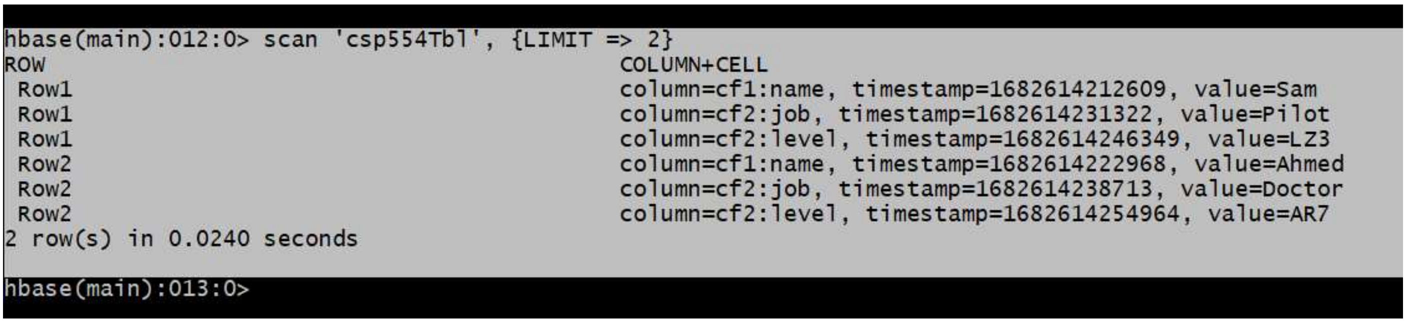


```
hbase(main):011:0> get 'csp554Tbl', 'Row2', {COLUMN => ['cf1:name']}
COLUMN                                        CELL
 cf1:name                                     timestamp=1682614222968, value=Ahmed
1 row(s) in 0.0160 seconds

hbase(main):012:0>
```

Exercise 5) (1 point)

Using the above table write a SCAN command that will return information about only two rows using the LIMIT modifier. Provide the command and its result as the output of this exercise.

- **scan 'csp554Tbl' ,{LIMIT=> 2}**



```
hbase(main):012:0> scan 'csp554Tbl', {LIMIT => 2}
ROW                                          COLUMN+CELL
 Row1                                        column=cf1:name, timestamp=1682614212609, value=Sam
 Row1                                        column=cf2:job, timestamp=1682614231322, value=Pilot
 Row1                                        column=cf2:level, timestamp=1682614246349, value=LZ3
 Row2                                        column=cf1:name, timestamp=1682614222968, value=Ahmed
 Row2                                        column=cf2:job, timestamp=1682614238713, value=Doctor
 Row2                                        column=cf2:level, timestamp=1682614254964, value=AR7
2 row(s) in 0.0240 seconds

hbase(main):013:0>
```

Cassandra

Exercise 1) (1 point)

Read the article "A Big Data Modeling Methodology for Apache Cassandra" and provide a ½ page summary including your comments and impressions.

The article presents a comprehensive methodology for designing data models in Apache Cassandra, a popular distributed NoSQL database system designed to handle large amounts of data across multiple commodity servers. The methodology includes three stages: conceptual modeling, logical modeling, and physical modeling. Each stage is described in detail, providing practical tips and examples to illustrate the concepts.

The paper "A Big Data Modeling Methodology for Apache Cassandra" offers a comprehensive approach to modeling data in Apache Cassandra, a distributed database management system designed to handle large amounts of data across multiple servers. The authors propose a three-step process for modeling data in Cassandra: conceptual modeling, logical modeling, and physical modeling. They stress the importance of considering performance when designing data models for Cassandra and offer guidelines for achieving high write and read throughput while minimizing query latency.

The Cassandra data model is based on tables, where each table is a collection of partitions that contain rows with similar structures. A table schema includes columns with primitive, complex, or counter data types. A primary key is a combination of a partition key and a clustering key that uniquely identifies a row in a table. CQL, a syntax similar to SQL, is used to express queries over tables. CQL doesn't support binary operations like joins, instead relying on query predicate rules for efficiency and scalability.

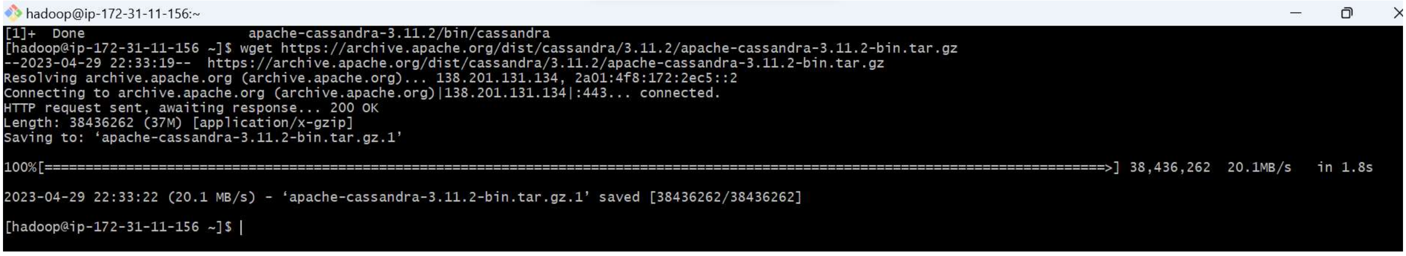
Conceptual modeling defines entities and their relationships in the domain being modeled. Logical modeling maps the conceptual model into tables, columns, and primary keys in Cassandra. Physical modeling describes how data will be physically stored on disk and how queries can efficiently retrieve that data from the cluster. The authors propose a query-driven approach to mapping from a conceptual to a logical data model, where entities and relationships map to table rows, and attributes map to columns in a table. They also propose four data modeling principles: know your data, know your questions, data nesting, and data duplication. Mapping patterns are introduced to automate Cassandra database schema design.

In summary, the paper provides a practical and informative guide to modeling data in Cassandra. The three-step process, query-driven mapping, and data modeling principles provide a structured approach to designing effective data models. The guidelines and best practices for optimizing performance offer useful insights for developers and data architects working with Cassandra.

Exercise 2) (1 point)

Step A – Start an EMR cluster

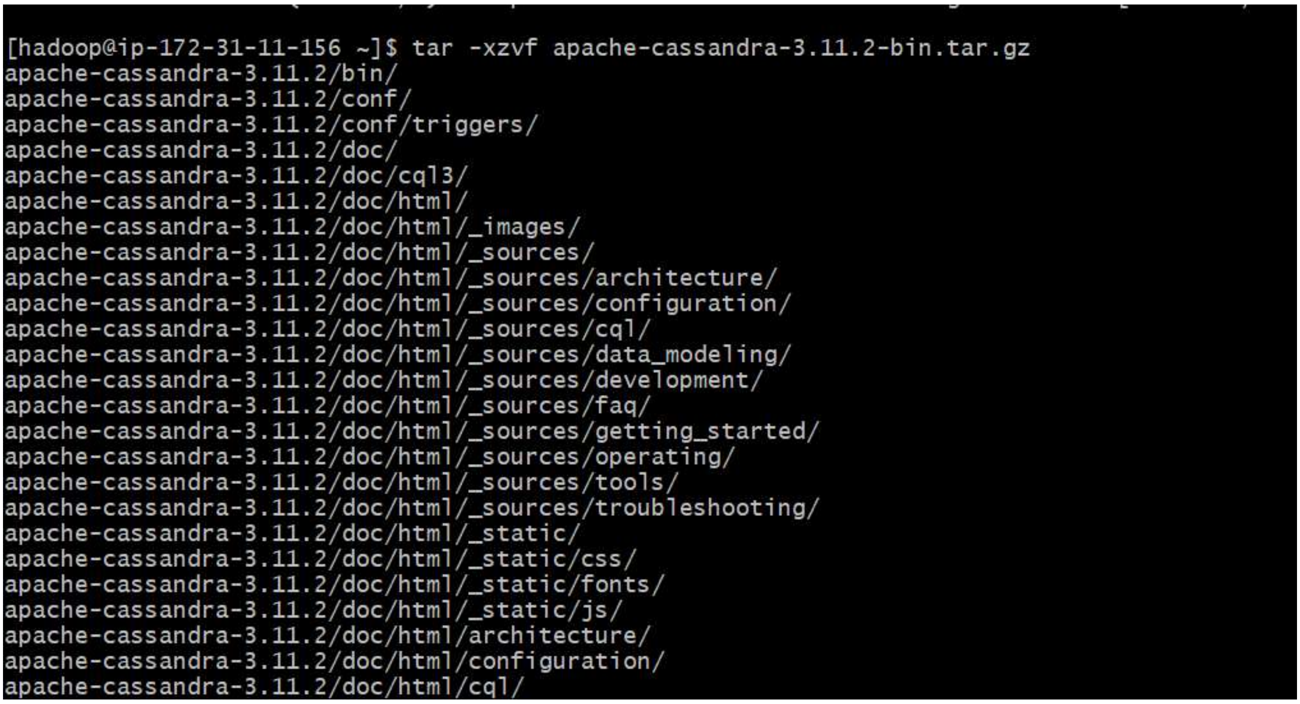
- **ssh -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer hadoop@ec2-3-231-147-225.compute-1.amazonaws.com**



```
[1]+  Done                    apache-cassandra-3.11.2/bin/cassandra
[hadoop@ip-172-31-11-156 ~]$ wget https://archive.apache.org/dist/cassandra/3.11.2/apache-cassandra-3.11.2-bin.tar.gz
--2023-04-29 22:33:19--  https://archive.apache.org/dist/cassandra/3.11.2/apache-cassandra-3.11.2-bin.tar.gz
Resolving archive.apache.org (archive.apache.org)... 138.201.131.134, 2a01:4f8:172:2ec5::2
Connecting to archive.apache.org (archive.apache.org)|138.201.131.134|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38436262 (37M) [application/x-gzip]
Saving to: ‘apache-cassandra-3.11.2-bin.tar.gz.1’

100%[=================================================================================================================================>] 38,436,262  20.1MB/s   in 1.8s

2023-04-29 22:33:22 (20.1 MB/s) - ‘apache-cassandra-3.11.2-bin.tar.gz.1’ saved [38436262/38436262]

[hadoop@ip-172-31-11-156 ~]$ |
```

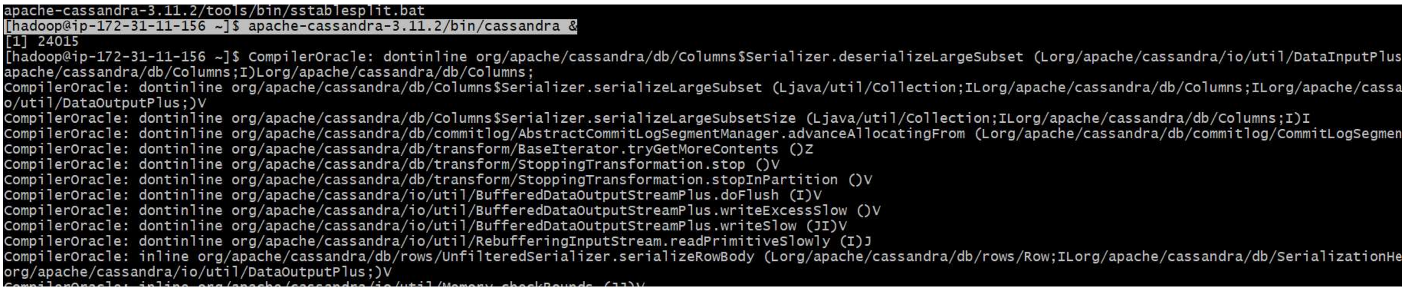
Step B – Install the Cassandra database software and start it

- ***wget https://archive.apache.org/dist/cassandra/3.11.2/apache-cassandra-3.11.2-bin.tar.gz***
- ***tar -xzvf apache-cassandra-3.11.2-bin.tar.gz***



```
[hadoop@ip-172-31-11-156 ~]$ tar -xzvf apache-cassandra-3.11.2-bin.tar.gz
apache-cassandra-3.11.2/bin/
apache-cassandra-3.11.2/conf/
apache-cassandra-3.11.2/conf/triggers/
apache-cassandra-3.11.2/doc/
apache-cassandra-3.11.2/doc/cql3/
apache-cassandra-3.11.2/doc/html/
apache-cassandra-3.11.2/doc/html/_images/
apache-cassandra-3.11.2/doc/html/_sources/
apache-cassandra-3.11.2/doc/html/_sources/architecture/
apache-cassandra-3.11.2/doc/html/_sources/configuration/
apache-cassandra-3.11.2/doc/html/_sources/cql/
apache-cassandra-3.11.2/doc/html/_sources/data_modeling/
apache-cassandra-3.11.2/doc/html/_sources/development/
apache-cassandra-3.11.2/doc/html/_sources/faq/
apache-cassandra-3.11.2/doc/html/_sources/getting_started/
apache-cassandra-3.11.2/doc/html/_sources/operating/
apache-cassandra-3.11.2/doc/html/_sources/tools/
apache-cassandra-3.11.2/doc/html/_sources/troubleshooting/
apache-cassandra-3.11.2/doc/html/_static/
apache-cassandra-3.11.2/doc/html/_static/css/
apache-cassandra-3.11.2/doc/html/_static/fonts/
apache-cassandra-3.11.2/doc/html/_static/js/
apache-cassandra-3.11.2/doc/html/architecture/
apache-cassandra-3.11.2/doc/html/configuration/
apache-cassandra-3.11.2/doc/html/cql/
```

apache-cassandra-3.11.2/bin/cassandra &

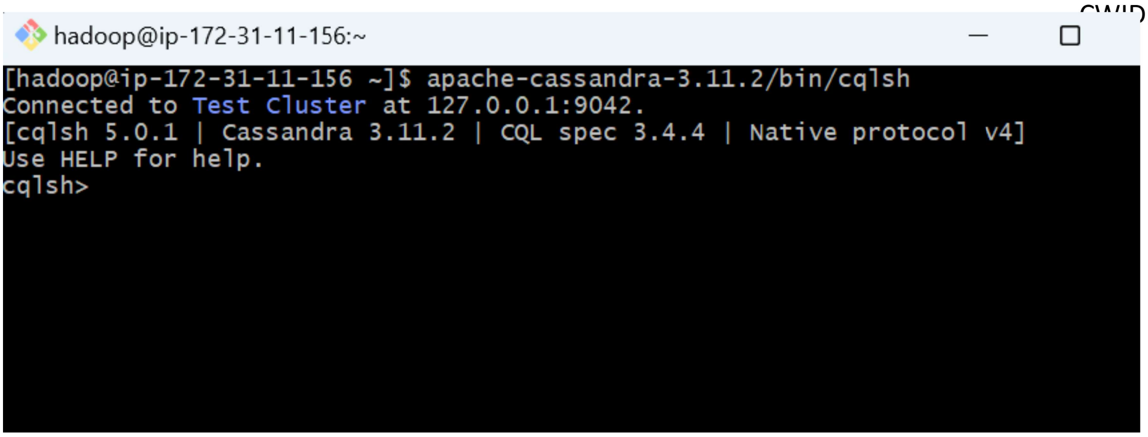


```
apache-cassandra-3.11.2/tools/bin/sstablesplit.bat
[hadoop@ip-172-31-11-156 ~]$ apache-cassandra-3.11.2/bin/cassandra &
[1] 24015
[hadoop@ip-172-31-11-156 ~]$ CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.deserializeLargeSubset (Lorg/apache/cassandra/io/util/DataInputPlus
apache/cassandra/db/Columns;I)Lorg/apache/cassandra/db/Columns;
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.serializeLargeSubset (Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;ILorg/apache/cassa
o/util/DataOutputPlus;)V
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.serializeLargeSubsetSize (Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;I)I
CompilerOracle: dontinline org/apache/cassandra/db/commitlog/AbstractCommitLogSegmentManager.advanceAllocatingFrom (Lorg/apache/cassandra/db/commitlog/CommitLogSegmen
CompilerOracle: dontinline org/apache/cassandra/db/transform/BaseIterator.tryGetMoreContents ()Z
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTransformation.stop ()V
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTransformation.stopInPartition ()V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.doFlush (I)V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.writeExcessSlow ()V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.writeSlow (JI)V
CompilerOracle: dontinline org/apache/cassandra/io/util/RebufferingInputStream.readPrimitiveSlowly (I)J
CompilerOracle: inline org/apache/cassandra/db/rows/UnfilteredSerializer.serializeRowBody (Lorg/apache/cassandra/db/rows/Row;ILorg/apache/cassandra/db/SerializationHe
org/apache/cassandra/io/util/DataOutputPlus;)V
CompilerOracle: inline org/apache/cassandra/io/util/Memory.checkBounds (JJ)V
```

Step C – Run the Cassandra interactive command line interface

- **ssh -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer hadoop@ec2-3-231-147-225.compute-1.amazonaws.com**

- ***apache-cassandra-3.11.2/bin/cqlsh***

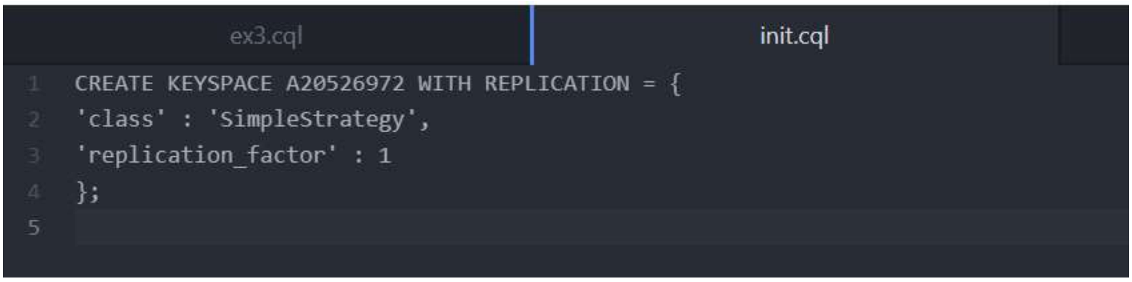


```
[hadoop@ip-172-31-11-156 ~]$ apache-cassandra-3.11.2/bin/cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

Step D – Prepare to edit your Cassandra code

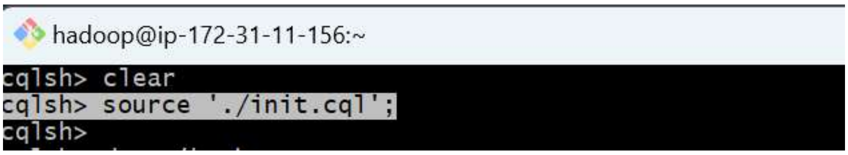
a) Create a file in your working (home) directory called init.cql using your Edit-term (or using your PC/MAC and then scp it to the EMR master node) and enter the following command. Use your IIT id as the name of your keyspace… For example, if your id is A1234567, then replace below with that value:

- ***vi init.cql (** and enter the following command in the file).*
- ***CREATE KEYSPACE A20526972 WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };***



```
CREATE KEYSPACE A20526972 WITH REPLICATION = {
'class' : 'SimpleStrategy',
'replication_factor' : 1
};
```

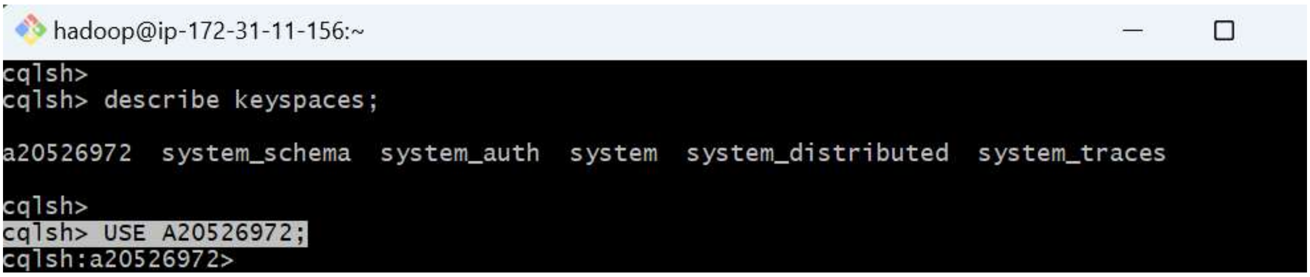
b) ***source './init.cql'***



```
cqlsh> clear
cqlsh> source './init.cql';
cqlsh>
```

c) To check if your script file has created a keyspace execute the following in the CQL shell:

- ***describe keyspaces;***

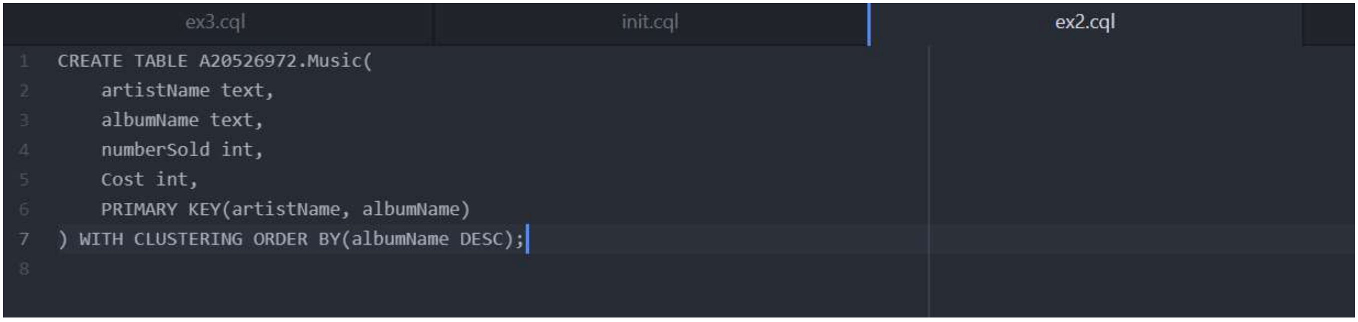


```
cqlsh>
cqlsh> describe keyspaces;

a20526972  system_schema  system_auth  system  system_distributed  system_traces

cqlsh>
cqlsh> USE A20526972;
cqlsh:a20526972>
```

d) At this point you have created a keyspace unique to you. So, make that keyspace the default by entering the following into the CQL shell:

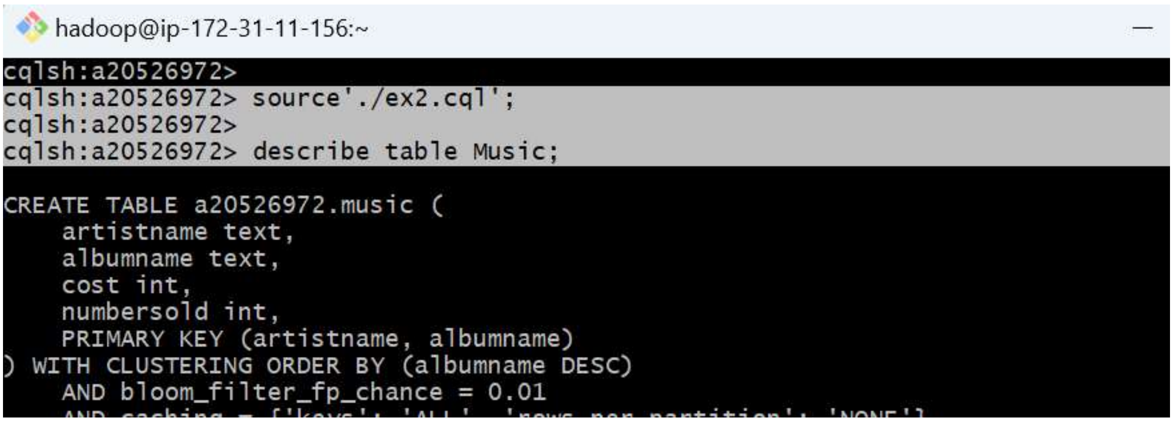
- ***USE A20526972;***
- ***nano ex2.cql***
- ***CREATE TABLE A20526972.Music(***
  artistName text,
  albumName text,
  numberSold int,

Cost int,
PRIMARY KEY(artistName, albumName)) WITH CLUSTERING ORDER BY(albumName DESC);



```
CREATE TABLE A20526972.Music(
    artistName text,
    albumName text,
    numberSold int,
    Cost int,
    PRIMARY KEY(artistName, albumName)
) WITH CLUSTERING ORDER BY(albumName DESC);
```

- ***source './ex2.cql';***
- ***DESCRIBE TABLE Music;***



```
cqlsh:a20526972>
cqlsh:a20526972> source'./ex2.cql';
cqlsh:a20526972>
cqlsh:a20526972> describe table Music;

CREATE TABLE a20526972.music (
    artistname text,
    albumname text,
    cost int,
    numbersold int,
    PRIMARY KEY (artistname, albumname)
) WITH CLUSTERING ORDER BY (albumname DESC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
```

Exercise 3) (1 point)

Now create a file in your working directory called ex3.cql using the Edit-Term. In this file write the commands to insert the following records into table ‘Music

a) Execute ex3.cql. Provide the content of this file as the result of this exercise.

- ***vi ex3.cql***

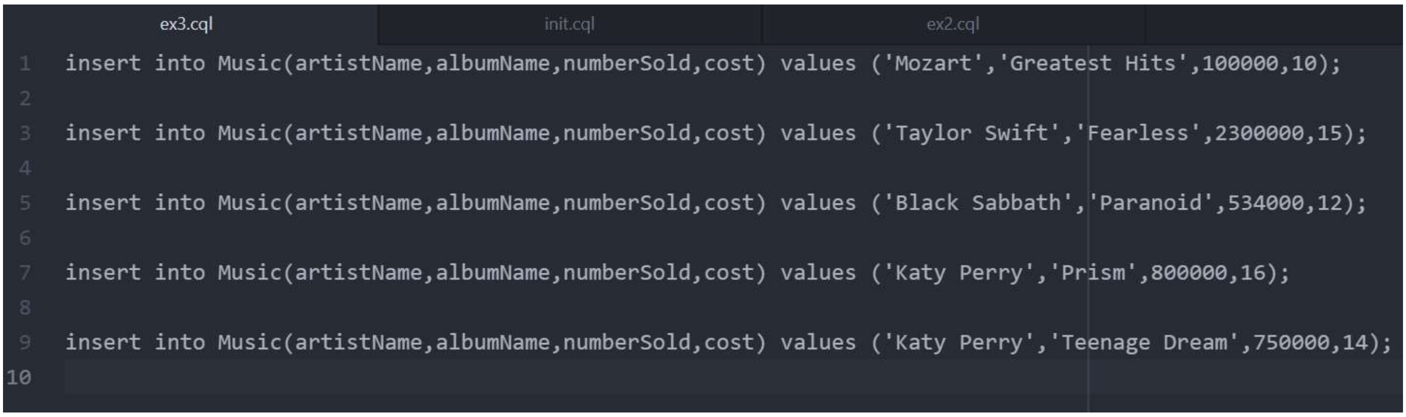
insert into Music(artistName,albumName,numberSold,cost) values ('Mozart','Greatest Hits',100000,10);

insert into Music(artistName,albumName,numberSold,cost) values ('Taylor Swift','Fearless',2300000,15);

insert into Music(artistName,albumName,numberSold,cost) values ('Black Sabbath','Paranoid',534000,12);

insert into Music(artistName,albumName,numberSold,cost) values ('Katy Perry','Prism',800000,16);

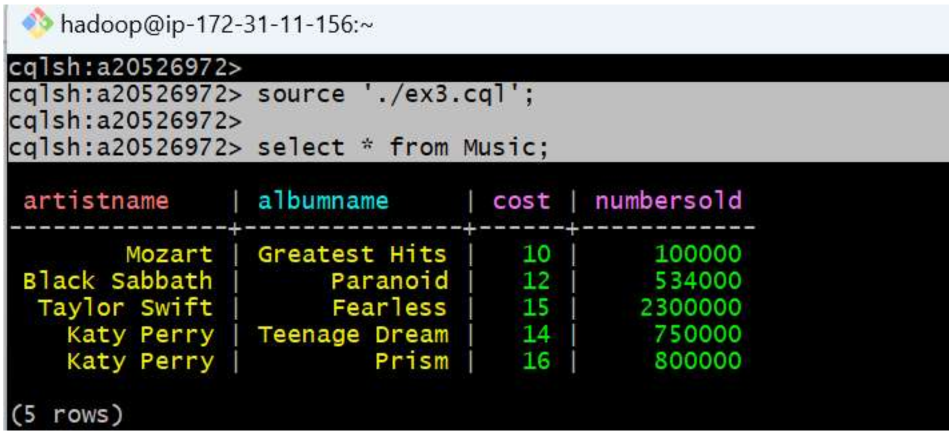
insert into Music(artistName,albumName,numberSold,cost) values ('Katy Perry','Teenage Dream',750000,14);



```
insert into Music(artistName,albumName,numberSold,cost) values ('Mozart','Greatest Hits',100000,10);

insert into Music(artistName,albumName,numberSold,cost) values ('Taylor Swift','Fearless',2300000,15);

insert into Music(artistName,albumName,numberSold,cost) values ('Black Sabbath','Paranoid',534000,12);

insert into Music(artistName,albumName,numberSold,cost) values ('Katy Perry','Prism',800000,16);

insert into Music(artistName,albumName,numberSold,cost) values ('Katy Perry','Teenage Dream',750000,14);
```

b) Execute the command 'SELECT * FROM Music;' and provide the output of this command as another result of the exercise.

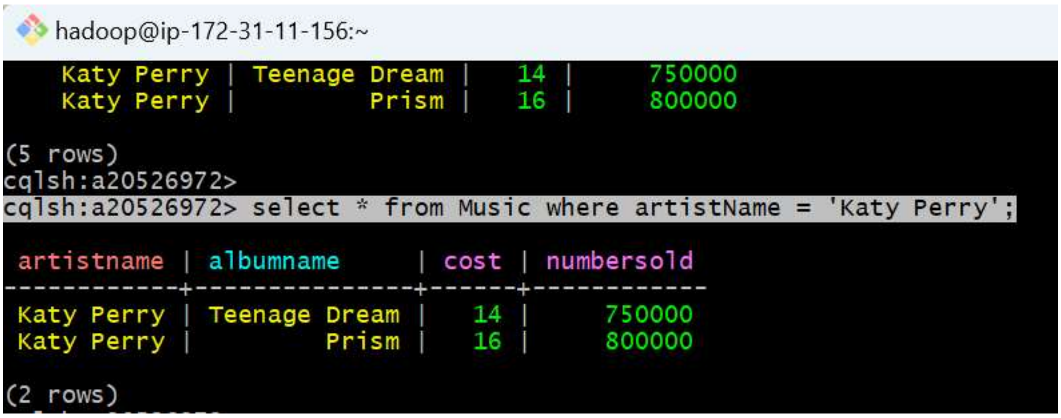
- ***source './ex3.cql'***
- ***SELECT * FROM Music;***



```
cqlsh:a20526972>
cqlsh:a20526972> source './ex3.cql';
cqlsh:a20526972>
cqlsh:a20526972> select * from Music;

 artistname    | albumname     | cost | numbersold
---------------+---------------+------+------------
        Mozart | Greatest Hits |   10 |     100000
 Black Sabbath |      Paranoid |   12 |     534000
  Taylor Swift |      Fearless |   15 |    2300000
    Katy Perry | Teenage Dream |   14 |     750000
    Katy Perry |         Prism |   16 |     800000

(5 rows)
```

Exercise 4) (1 point)

- **Now enter this command directly into the cql shell**
  - SELECT * FROM Music where artistName = 'Katy Perry';

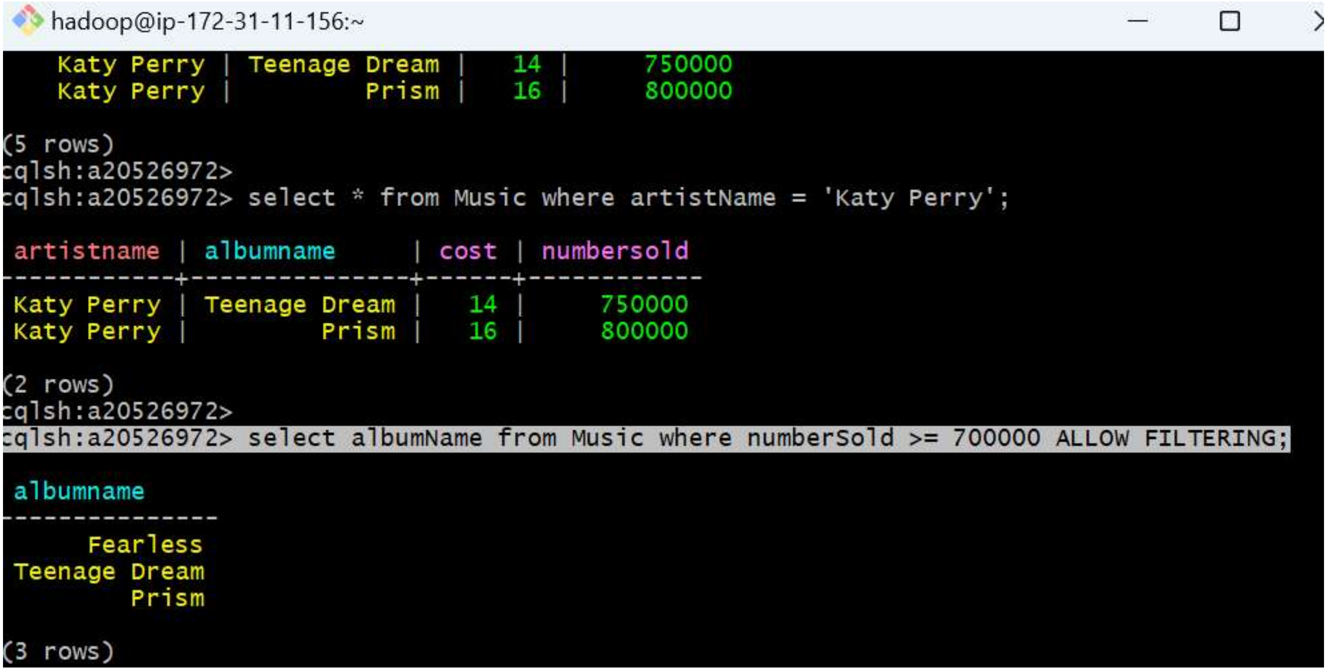


```
    Katy Perry | Teenage Dream |   14 |     750000
    Katy Perry |         Prism |   16 |     800000

(5 rows)
cqlsh:a20526972>
cqlsh:a20526972> select * from Music where artistName = 'Katy Perry';

 artistname | albumname     | cost | numbersold
------------+---------------+------+------------
 Katy Perry | Teenage Dream |   14 |     750000
 Katy Perry |         Prism |   16 |     800000

(2 rows)
```

Exercise 5) (1 point)

Now enter this command in the cassandra shell.

- SELECT FROM Music where numberSold >= 700000 ALLOW FILTERING;



```
    Katy Perry | Teenage Dream |   14 |     750000
    Katy Perry |         Prism |   16 |     800000

(5 rows)
cqlsh:a20526972>
cqlsh:a20526972> select * from Music where artistName = 'Katy Perry';

 artistname | albumname     | cost | numbersold
------------+---------------+------+------------
 Katy Perry | Teenage Dream |   14 |     750000
 Katy Perry |         Prism |   16 |     800000

(2 rows)
cqlsh:a20526972>
cqlsh:a20526972> select albumName from Music where numberSold >= 700000 ALLOW FILTERING;

 albumname
---------------
      Fearless
 Teenage Dream
         Prism

(3 rows)
```

MongoDB

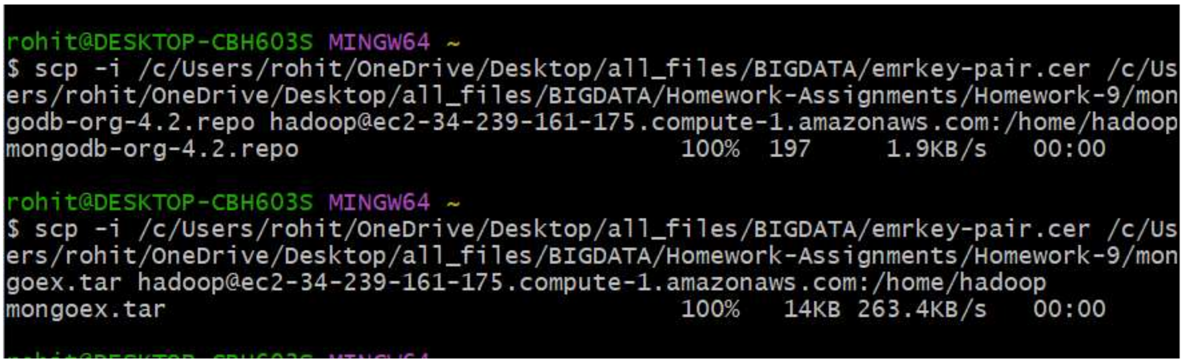
Step A – Start an EMR cluster

- ssh -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer hadoop@ec2-34-200-220-199.compute-1.amazonaws.com

Step B – Download the assignment software (mongoex.tar, mongodb-org-4.2.repo) to master node

- scp -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/Homework-Assignments/Homework-9/mongodb-org-4.2.repo hadoop@ec2-34-239-161-175.compute-1.amazonaws.com:/home/hadoop

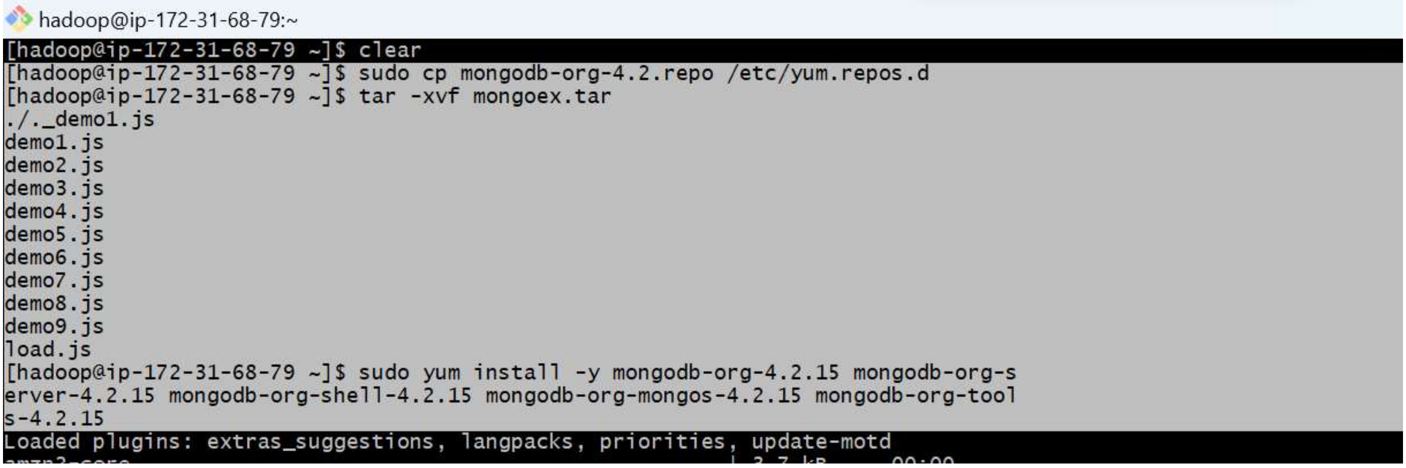
- scp -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/Homework-Assignments/Homework-9/mongoex.tar hadoop@ec2-34-239-161-175.compute-1.amazonaws.com:/home/hadoop



```
rohit@DESKTOP-CBH603S MINGW64 ~
$ scp -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer /c/Us
ers/rohit/OneDrive/Desktop/all_files/BIGDATA/Homework-Assignments/Homework-9/mon
godb-org-4.2.repo hadoop@ec2-34-239-161-175.compute-1.amazonaws.com:/home/hadoop
mongodb-org-4.2.repo                          100%  197      1.9KB/s   00:00

rohit@DESKTOP-CBH603S MINGW64 ~
$ scp -i /c/Users/rohit/OneDrive/Desktop/all_files/BIGDATA/emrkey-pair.cer /c/Us
ers/rohit/OneDrive/Desktop/all_files/BIGDATA/Homework-Assignments/Homework-9/mon
goex.tar hadoop@ec2-34-239-161-175.compute-1.amazonaws.com:/home/hadoop
mongoex.tar                                   100%   14KB  263.4KB/s   00:00
```

Step C – Install assignment software (mongoex.zip, mongodb-org-4.2.repo)

- sudo cp mongodb-org-4.2.repo /etc/yum.repos.d
- tar -xvf mongoex.tar



```
[hadoop@ip-172-31-68-79 ~]$ clear
[hadoop@ip-172-31-68-79 ~]$ sudo cp mongodb-org-4.2.repo /etc/yum.repos.d
[hadoop@ip-172-31-68-79 ~]$ tar -xvf mongoex.tar
./._demo1.js
demo1.js
demo2.js
demo3.js
demo4.js
demo5.js
demo6.js
demo7.js
demo8.js
demo9.js
load.js
[hadoop@ip-172-31-68-79 ~]$ sudo yum install -y mongodb-org-4.2.15 mongodb-org-s
erver-4.2.15 mongodb-org-shell-4.2.15 mongodb-org-mongos-4.2.15 mongodb-org-tool
s-4.2.15
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
```

Step D – Install and start MongoDB

- **sudo yum install -y mongodb-org-4.2.15 mongodb-org-server-4.2.15 mongodb-org-shell-4.2.15 mongodb-org-mongos-4.2.15 mongodb-org-tools-4.2.15**
- **sudo systemctl start mongod**

Step E – Start the MongoDB Shell (Command Line Interpreter)

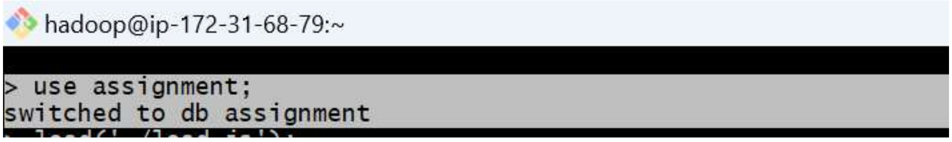
- **mongo**

Step F – Edit mongo query language files

Step G – Setting up the assignment database

Now, in the MongoDB shell, using the CLI-Term, create a database called "assignment" by entering the following into the MongoDB shell:

- **use assignment;**

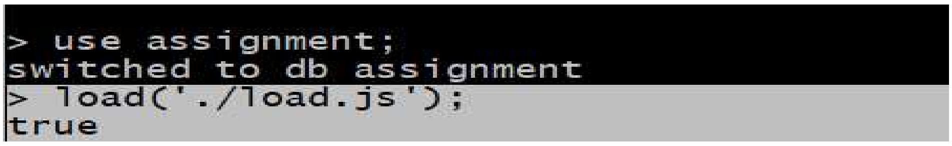


```
> use assignment;
switched to db assignment
```

This will set the shell variable 'db' to this new database.

Load a collection called 'unicorns' with sample data by executing the script load.js in the MongoDB shell as follows (don't cut and paste this, type it in manually):

- **load('./load.js');**

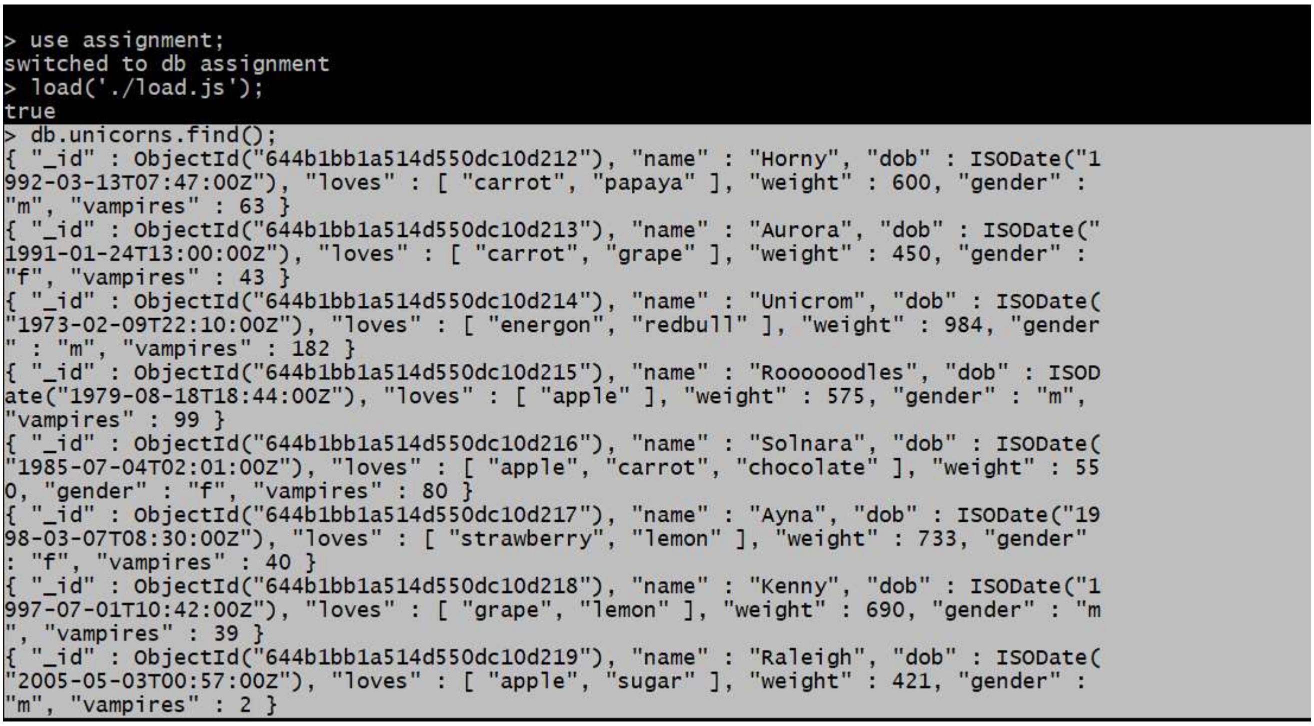


```
> use assignment;
switched to db assignment
> load('./load.js');
true
```

Note, look at the content of the script file (via the other terminal window you have opened to the EC2 instance) to see how each unicorn is described.

Confirm this has all worked by executing the following command in the MongoDB shell:

- **db.unicorns.find();**



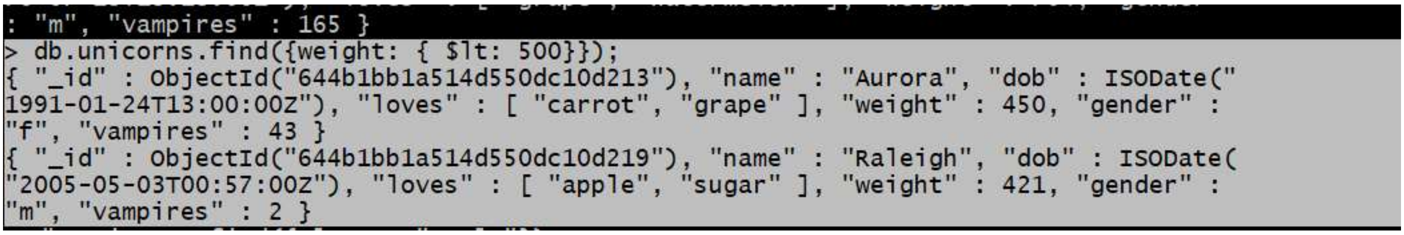
```
> use assignment;
switched to db assignment
> load('./load.js');
true
> db.unicorns.find();
{ "_id" : ObjectId("644b1bb1a514d550dc10d212"), "name" : "Horny", "dob" : ISODate("1
992-03-13T07:47:00Z"), "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" :
"m", "vampires" : 63 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d213"), "name" : "Aurora", "dob" : ISODate("
1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" :
"f", "vampires" : 43 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d214"), "name" : "Unicrom", "dob" : ISODate(
"1973-02-09T22:10:00Z"), "loves" : [ "energon", "redbull" ], "weight" : 984, "gender
" : "m", "vampires" : 182 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d215"), "name" : "Roooooodles", "dob" : ISOD
ate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender" : "m",
"vampires" : 99 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d216"), "name" : "Solnara", "dob" : ISODate(
"1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 55
0, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d217"), "name" : "Ayna", "dob" : ISODate("19
98-03-07T08:30:00Z"), "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender"
: "f", "vampires" : 40 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d218"), "name" : "Kenny", "dob" : ISODate("1
997-07-01T10:42:00Z"), "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m
", "vampires" : 39 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d219"), "name" : "Raleigh", "dob" : ISODate(
"2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" :
"m", "vampires" : 2 }
```

Note, the files named "demo*.js" (also included in the mongoex.tar file) provide examples of how to operate in the unicorn collection. These are a VERY good idea to review and understand and will present you with information helpful in completing the assignment. Also, try them out by typing something like

Exercise 1) (1 point)

Write a command that finds all unicorns having weight less than 500 pounds. Include the code you executed and some sample output as the result of this exercise. Recall you can place the command, if you choose, into a file, say 'ex1.js' and execute it with the load command as above and similarly for the following exercises.

- **db.unicorns.find({weight : { $lt : 500 }});**

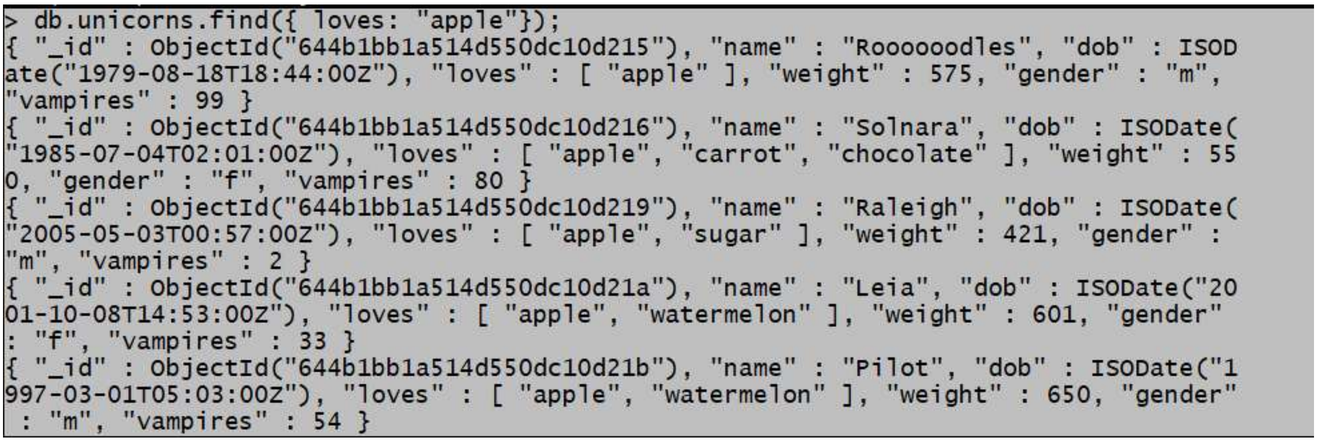


```
: "m", "vampires" : 165 }
> db.unicorns.find({weight: { $lt: 500}});
{ "_id" : ObjectId("644b1bb1a514d550dc10d213"), "name" : "Aurora", "dob" : ISODate("
1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" :
"f", "vampires" : 43 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d219"), "name" : "Raleigh", "dob" : ISODate(
"2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" :
"m", "vampires" : 2 }
```

Exercise 2) (1 point)

Write a command that finds all unicorns who love apples. Hint, search for "apple". Include the code you executed and some sample output as the result of this exercise.

- **db.unicorns.find({loves: 'apple'});**

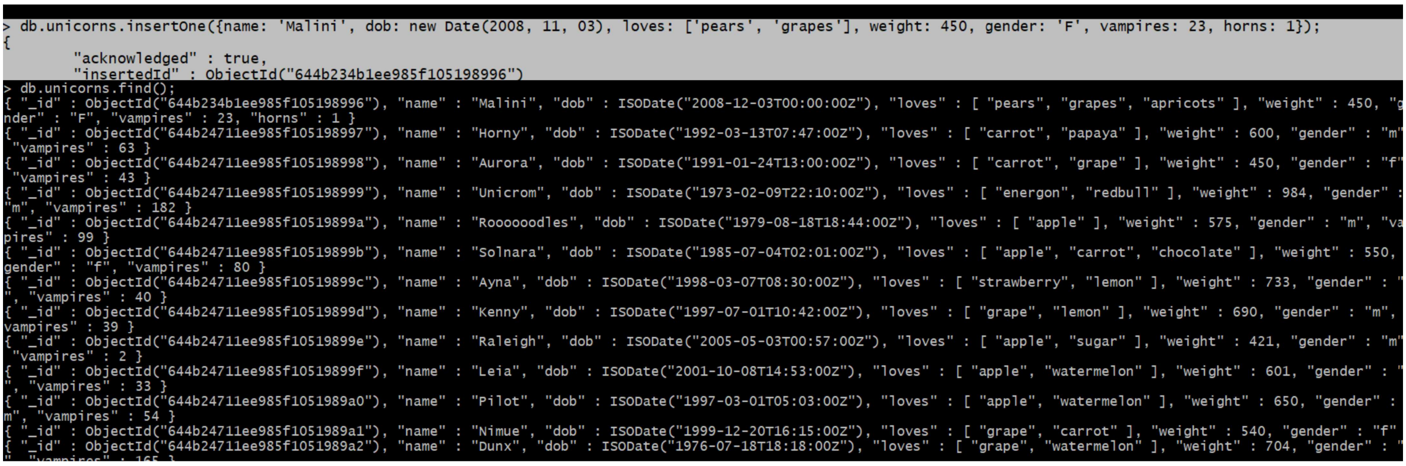


```
> db.unicorns.find({ loves: "apple"});
{ "_id" : ObjectId("644b1bb1a514d550dc10d215"), "name" : "Roooooodles", "dob" : ISOD
ate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender" : "m",
"vampires" : 99 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d216"), "name" : "Solnara", "dob" : ISODate(
"1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 55
0, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d219"), "name" : "Raleigh", "dob" : ISODate(
"2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" :
"m", "vampires" : 2 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d21a"), "name" : "Leia", "dob" : ISODate("20
01-10-08T14:53:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender"
: "f", "vampires" : 33 }
{ "_id" : ObjectId("644b1bb1a514d550dc10d21b"), "name" : "Pilot", "dob" : ISODate("1
997-03-01T05:03:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender"
 : "m", "vampires" : 54 }
```

Exercise 3) (1 point)

Write a command that adds a unicorn with the following attributes to the collection. Note dob means “Date of Birth.”

- **db.unicorns.insert({name: 'Malini', dob: new Date(2008, 11, 03), loves:['pears', 'grapes'], weight: 450, gender: 'F', vampires: 23, horns : 1});**
- **db.unicorns.find();**

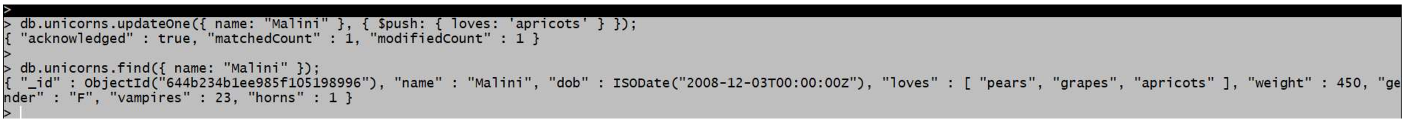


```
> db.unicorns.insertOne({name: 'Malini', dob: new Date(2008, 11, 03), loves: ['pears', 'grapes'], weight: 450, gender: 'F', vampires: 23, horns: 1});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("644b234b1ee985f105198996")
> db.unicorns.find();
{ "_id" : ObjectId("644b234b1ee985f105198996"), "name" : "Malini", "dob" : ISODate("2008-12-03T00:00:00Z"), "loves" : [ "pears", "grapes", "apricots" ], "weight" : 450, "g
nder" : "F", "vampires" : 23, "horns" : 1 }
{ "_id" : ObjectId("644b24711ee985f105198997"), "name" : "Horny", "dob" : ISODate("1992-03-13T07:47:00Z"), "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m"
 "vampires" : 63 }
{ "_id" : ObjectId("644b24711ee985f105198998"), "name" : "Aurora", "dob" : ISODate("1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f"
 "vampires" : 43 }
{ "_id" : ObjectId("644b24711ee985f105198999"), "name" : "Unicrom", "dob" : ISODate("1973-02-09T22:10:00Z"), "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" :
"m", "vampires" : 182 }
{ "_id" : ObjectId("644b24711ee985f10519899a"), "name" : "Roooooodles", "dob" : ISODate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "va
pires" : 99 }
{ "_id" : ObjectId("644b24711ee985f10519899b"), "name" : "Solnara", "dob" : ISODate("1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550,
gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("644b24711ee985f10519899c"), "name" : "Ayna", "dob" : ISODate("1998-03-07T08:30:00Z"), "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "
", "vampires" : 40 }
{ "_id" : ObjectId("644b24711ee985f10519899d"), "name" : "Kenny", "dob" : ISODate("1997-07-01T10:42:00Z"), "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m",
vampires" : 39 }
{ "_id" : ObjectId("644b24711ee985f10519899e"), "name" : "Raleigh", "dob" : ISODate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m"
 "vampires" : 2 }
{ "_id" : ObjectId("644b24711ee985f10519899f"), "name" : "Leia", "dob" : ISODate("2001-10-08T14:53:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "
", "vampires" : 33 }
{ "_id" : ObjectId("644b24711ee985f1051989a0"), "name" : "Pilot", "dob" : ISODate("1997-03-01T05:03:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" :
m", "vampires" : 54 }
{ "_id" : ObjectId("644b24711ee985f1051989a1"), "name" : "Nimue", "dob" : ISODate("1999-12-20T16:15:00Z"), "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f"
{ "_id" : ObjectId("644b24711ee985f1051989a2"), "name" : "Dunx", "dob" : ISODate("1976-07-18T18:18:00Z"), "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "
", "vampires" : 165 }
```

Exercise 4) (1 point)

Write a command that updates the above record to add apricots to the list of things Malini loves. Include the code you executed and some sample output showing the addition.

- **db.unicorns.update({name: 'Malini'}, {$push : {loves: ['apricots'] } });**
- **db.unicorns.find({ name: "Malini" });**

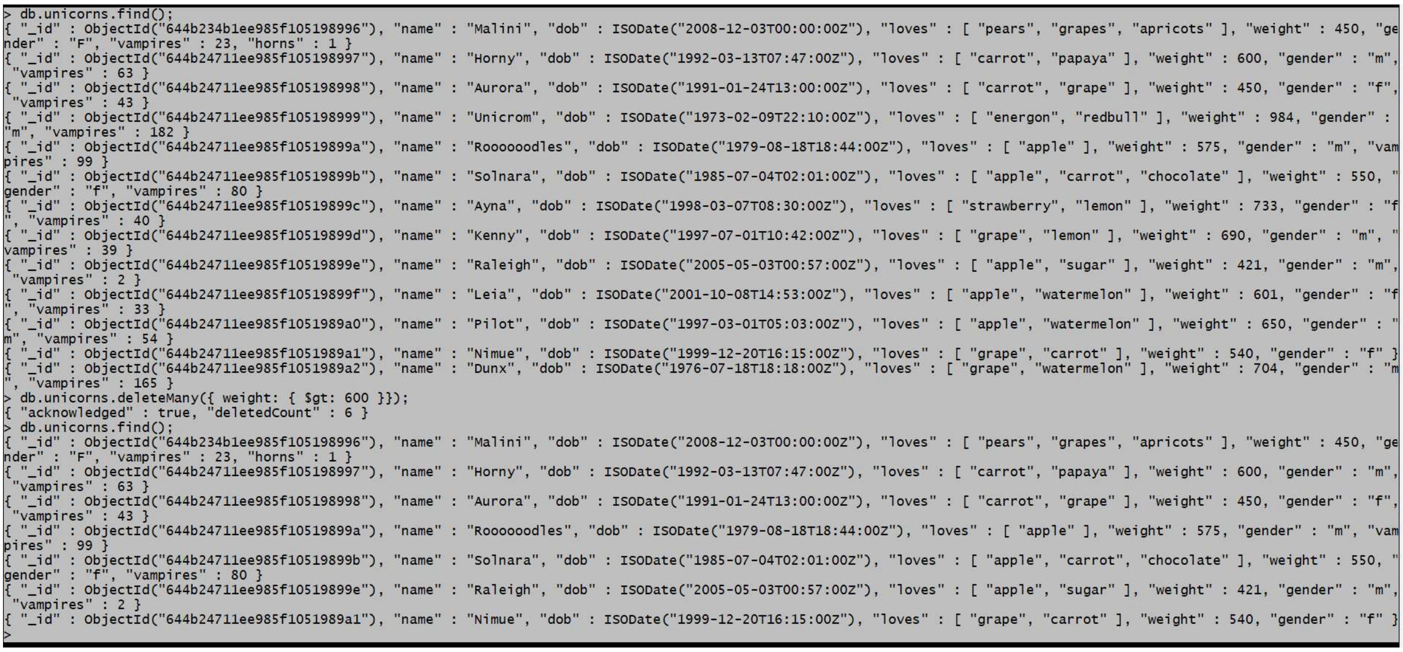


```
>
> db.unicorns.updateOne({ name: "Malini" }, { $push: { loves: 'apricots' } });
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
> db.unicorns.find({ name: "Malini" });
{ "_id" : ObjectId("644b234b1ee985f105198996"), "name" : "Malini", "dob" : ISODate("2008-12-03T00:00:00Z"), "loves" : [ "pears", "grapes", "apricots" ], "weight" : 450, "ge
nder" : "F", "vampires" : 23, "horns" : 1 }
>
```

Exercise 5) (1 point)

Write a command that deletes all unicorns with weight more than 600 pounds. Include the code you executed and some sample output as the result of this exercise

- **db.unicorns.find();**
- **db.unicorns.deleteMany({ weight: { $gt : 600 } });**
- **db.unicorns.find();**



```
> db.unicorns.find();
{ "_id" : ObjectId("644b234b1ee985f105198996"), "name" : "Malini", "dob" : ISODate("2008-12-03T00:00:00Z"), "loves" : [ "pears", "grapes", "apricots" ], "weight" : 450, "ge
nder" : "F", "vampires" : 23, "horns" : 1 }
{ "_id" : ObjectId("644b24711ee985f105198997"), "name" : "Horny", "dob" : ISODate("1992-03-13T07:47:00Z"), "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m",
 "vampires" : 63 }
{ "_id" : ObjectId("644b24711ee985f105198998"), "name" : "Aurora", "dob" : ISODate("1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f",
 "vampires" : 43 }
{ "_id" : ObjectId("644b24711ee985f105198999"), "name" : "Unicrom", "dob" : ISODate("1973-02-09T22:10:00Z"), "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" :
"m", "vampires" : 182 }
{ "_id" : ObjectId("644b24711ee985f10519899a"), "name" : "Roooooodles", "dob" : ISODate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vam
pires" : 99 }
{ "_id" : ObjectId("644b24711ee985f10519899b"), "name" : "Solnara", "dob" : ISODate("1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "
gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("644b24711ee985f10519899c"), "name" : "Ayna", "dob" : ISODate("1998-03-07T08:30:00Z"), "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "f
", "vampires" : 40 }
{ "_id" : ObjectId("644b24711ee985f10519899d"), "name" : "Kenny", "dob" : ISODate("1997-07-01T10:42:00Z"), "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "
vampires" : 39 }
{ "_id" : ObjectId("644b24711ee985f10519899e"), "name" : "Raleigh", "dob" : ISODate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m",
 "vampires" : 2 }
{ "_id" : ObjectId("644b24711ee985f10519899f"), "name" : "Leia", "dob" : ISODate("2001-10-08T14:53:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f
", "vampires" : 33 }
{ "_id" : ObjectId("644b24711ee985f1051989a0"), "name" : "Pilot", "dob" : ISODate("1997-03-01T05:03:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "
m", "vampires" : 54 }
{ "_id" : ObjectId("644b24711ee985f1051989a1"), "name" : "Nimue", "dob" : ISODate("1999-12-20T16:15:00Z"), "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
{ "_id" : ObjectId("644b24711ee985f1051989a2"), "name" : "Dunx", "dob" : ISODate("1976-07-18T18:18:00Z"), "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m
", "vampires" : 165 }
> db.unicorns.deleteMany({ weight: { $gt: 600 }});
{ "acknowledged" : true, "deletedCount" : 6 }
> db.unicorns.find();
{ "_id" : ObjectId("644b234b1ee985f105198996"), "name" : "Malini", "dob" : ISODate("2008-12-03T00:00:00Z"), "loves" : [ "pears", "grapes", "apricots" ], "weight" : 450, "ge
nder" : "F", "vampires" : 23, "horns" : 1 }
{ "_id" : ObjectId("644b24711ee985f105198997"), "name" : "Horny", "dob" : ISODate("1992-03-13T07:47:00Z"), "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m",
 "vampires" : 63 }
{ "_id" : ObjectId("644b24711ee985f105198998"), "name" : "Aurora", "dob" : ISODate("1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f",
 "vampires" : 43 }
{ "_id" : ObjectId("644b24711ee985f10519899a"), "name" : "Roooooodles", "dob" : ISODate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vam
pires" : 99 }
{ "_id" : ObjectId("644b24711ee985f10519899b"), "name" : "Solnara", "dob" : ISODate("1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "
gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("644b24711ee985f10519899e"), "name" : "Raleigh", "dob" : ISODate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m",
 "vampires" : 2 }
{ "_id" : ObjectId("644b24711ee985f1051989a1"), "name" : "Nimue", "dob" : ISODate("1999-12-20T16:15:00Z"), "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
>
```