HBase

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.

Answer-

Table creation:

```
hbase:019:0> create 'csp554Tbl', {NAME => 'cf1'}, {NAME => 'cf2'}
Created table csp554Tbl
Took 8.6924 seconds
=> Hbase::Table - csp554Tbl
```

describe 'csp554Tbl'

```
hbase:021:0> describe 'csp554Tbl'
Table csp554Tbl is ENABLED
csp554Tbl
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACH
E => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf2', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACH
E => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

2 row(s)
Quota is disabled
Took 0.7572 seconds
hbase:022:0>
```

Exercise 2)
Put the following data into the table created in exercise 1

Row Key	Column Family	Column (Qualifier)	Value
Row1	cf1	name	Sam
Row2	cf1	name	Ahmed
Row1	cf2	job	Pilot
Row2	cf2	job	Doctor
Row1	cf2	level	LZ3
Row2	cf2	level	AR7

Execute the SCAN command on this table returning all rows, column families and columns. Provide the command and its result as the output of this exercise.

Putting data into table:

```
hbase:022:0> put 'csp554Tbl', 'Row1', 'cf1:name', 'Sam'
Took 0.4257 seconds
hbase:023:0> put 'csp554Tbl', 'Row2', 'cf1:name', 'Ahmed'
Took 0.0185 seconds
hbase:024:0> put 'csp554Tbl', 'Row1', 'cf2:job', 'Pilot'
Took 0.0092 seconds
hbase:025:0> put 'csp554Tbl', 'Row2', 'cf2:job', 'Doctor'
Took 0.0066 seconds
hbase:026:0> put 'csp554Tbl', 'Row1', 'cf2:level', 'LZ3'
Took 0.0113 seconds
hbase:027:0> put 'csp554Tbl', 'Row2', 'cf2:level', 'AR7'
Took 0.0066 seconds
```

Scan command output:

```
hbase:029:0> scan 'csp554Tbl'
ROW                                                    COLUMN+CELL
 Row1                                                  column=cf1:name, timestamp=2023-11-17T02:07:36.480, value=Sam
 Row1                                                  column=cf2:job, timestamp=2023-11-17T02:08:16.868, value=Pilot
 Row1                                                  column=cf2:level, timestamp=2023-11-17T02:08:45.446, value=LZ3
 Row2                                                  column=cf1:name, timestamp=2023-11-17T02:07:59.596, value=Ahmed
 Row2                                                  column=cf2:job, timestamp=2023-11-17T02:08:31.737, value=Doctor
 Row2                                                  column=cf2:level, timestamp=2023-11-17T02:08:57.819, value=AR7
2 row(s)
Took 0.0497 seconds
hbase:030:0>
```

Exercise 3)
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.

Command and result are as follows:

```
hbase:030:0> get 'csp554Tbl', 'Row1', {COLUMN => ['cf2', 'cf2:level']}
COLUMN                                                 CELL
 cf2:level                                             timestamp=2023-11-17T02:08:45.446, value=LZ3
1 row(s)
Took 0.0359 seconds
```

Exercise 4)
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.

Command and result are as follows:

```
hbase:031:0> get 'csp554Tbl', 'Row2', {COLUMN => ['cf1', 'cf1:name']}
COLUMN                                                 CELL
 cf1:name                                              timestamp=2023-11-17T02:07:59.596, value=Ahmed
1 row(s)
Took 0.0138 seconds
```

Exercise 5)
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.

Command and result are as follows:

```
hbase:032:0> scan 'csp554Tbl', {LIMIT => 2}
ROW                                                    COLUMN+CELL
 Row1                                                  column=cf1:name, timestamp=2023-11-17T02:07:36.480, value=Sam
 Row1                                                  column=cf2:job, timestamp=2023-11-17T02:08:16.868, value=Pilot
 Row1                                                  column=cf2:level, timestamp=2023-11-17T02:08:45.446, value=LZ3
 Row2                                                  column=cf1:name, timestamp=2023-11-17T02:07:59.596, value=Ahmed
 Row2                                                  column=cf2:job, timestamp=2023-11-17T02:08:31.737, value=Doctor
 Row2                                                  column=cf2:level, timestamp=2023-11-17T02:08:57.819, value=AR7
2 row(s)
Took 0.0461 seconds
```

Cassandra

Exercise 1)
Read the article "A Big Data Modeling Methodology for Apache Cassandra" and provide a ½ page summary including your comments and impressions.

Answer-

Popular distributed database Apache Cassandra is renowned for its fault tolerance, scalability, and capacity for handling big datasets. Because of its flexible data model—which developed from BigTable—and its intuitive query language, CQL, it is a top option for big data applications, such as messaging, fraud detection, product catalogues, and sensor data. Applications needing high transaction rates can also benefit greatly from Cassandra's write and read access methods.

The new data modelling paradigm for Apache Cassandra presented in this article places a strong emphasis on a query-driven methodology. This technique incorporates patterns and guidelines for translating conceptual data models to Cassandra-specific logical data models, which sets it apart from the relational data modelling approach. In addition to introducing Diagrams—a tool for visualising intricate logical and physical data models—the study emphasises the importance of physical data modelling. The article also covers KDM, a data modelling tool that streamlines challenging, labour-intensive, and error-prone operations through automation.

Compared to conventional relational data modelling, Cassandra's query-driven data modelling paradigm has a number of benefits. It allows for effective data duplication and layering, enhancing scalability and query performance. Additionally, it gives schema designers greater freedom, which makes it simpler to adjust to shifting business requirements. The document offers mapping rules and patterns that make the process of transforming conceptual data models into logical data models tailored to Cassandra easier.

The study concludes with a query-driven data modelling paradigm for Apache Cassandra, which has a number of benefits over conventional relational data modelling. With its focus on efficiency and flexibility, this innovative method is a great option for distributed and large-scale databases. In order to streamline and boost productivity, the paper also presents new tools, such as Diagrams and KDM. Future big data applications are expected to continue using Apache Cassandra because of its scalable architecture, flexible data model, and easy-to-use query language.

Exercise 2)

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 <IIT id> below with that value:

   CREATE KEYSPACE <IIT id> WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

```
[hadoop@ip-172-31-5-182 ~]$ touch inti.cql
[hadoop@ip-172-31-5-182 ~]$ vim inti.cql
[hadoop@ip-172-31-5-182 ~]$ cat inti.cql
CREATE KEYSPACE A20551908 WITH REPLICATION = { 'class' : 'SimpleStrategy', 'repl
ication_factor' : 1 };
```

b) Then execute this file in the CQL shell using the Cqlsh-Term as follows:

source './init.cql';

```
cqlsh> source './inti.cql';
```

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

describe keyspaces;

```
cqlsh> describe keyspaces;

a20551908  system_schema  system_auth  system  system_distributed  system_traces

cqlsh>
```

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 <IIT id>;

```
cqlsh> USE A20551908;
```

Now create a file in your working directory called ex2.cql using the Edit-Term (or PC/MAC and scp). In this file write the command to create a table named ‘Music’ with the following characteristics:

Attribute Name	Attribute Type	Primary Key / Cluster Key
artistName	text	Primary Key
albumName	text	Cluster Key
numberSold	int	Non Key Column
Cost	int	Non Key Column

Execute ex2.cql in the CQL shell. Then execute the shell command ‘DESCRIBE TABLE Music;’ and include the output as the result of this exercise.

```
cqlsh:a20551908> source './ex2.cql';
cqlsh:a20551908> DESCRIBE TABLE Music

CREATE TABLE a20551908.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'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCom
pactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandr
a.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

Exercise 3)

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'.

artistName	albumName	numberSold	cost
Mozart	Greatest Hits	100000	10
Taylor Swift	Fearless	2300000	15
Black Sabbath	Paranoid	534000	12
Katy Perry	Prism	800000	16
Katy Perry	Teenage Dream	750000	14

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

```
[hadoop@ip-172-31-5-182 ~]$ cat 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);
```

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

```
cqlsh:a20551908> source './ex3.cql';
cqlsh:a20551908> 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)

Now create a file in your working directory called ex4.cql using the Edit-Term. In this file write the commands to query and output only Katy Perry songs. Execute ex4.cql. Provide the content of this file and output of executing this file as the result of this exercise.

```
[hadoop@ip-172-31-5-182 ~]$ touch ex4.cql
[hadoop@ip-172-31-5-182 ~]$ vim ex4.cql
[hadoop@ip-172-31-5-182 ~]$ cat ex4.cql
SELECT * FROM Music WHERE artistName = 'Katy Perry';
```

```
cqlsh:a20551908> source './ex4.cql';

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

(2 rows)
```

Exercise 5)

Now create a file in your working directory called ex5.cql using the Edit-Term. In this file write the commands to query only albums that have sold 700000 copies or more. Execute ex5.cql. Provide the content of this file and the output of executing this file as the result of this exercise.

```
[hadoop@ip-172-31-5-182 ~]$ touch ex5.cql
[hadoop@ip-172-31-5-182 ~]$ vim ex5.cql
[hadoop@ip-172-31-5-182 ~]$ cat ex5.cql
SELECT * FROM Music WHERE numberSold >= 700000 ALLOW FILTERING;
```

```
cqlsh:a20551908> source './ex5.cql';

 artistname   | albumname     | cost | numbersold
--------------+---------------+------+------------
 Taylor Swift |      Fearless |   15 |    2300000
   Katy Perry | Teenage Dream |   14 |     750000
   Katy Perry |         Prism |   16 |     800000

(3 rows)
cqlsh:a20551908>
```

MongoDB

Exercise 1)

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}})
{ "_id" : ObjectId("6556a447629d5aabbd11b6bf"), "name" : "Aurora", "dob" : ISODa
te("1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "ge
nder" : "f", "vampires" : 43 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c5"), "name" : "Raleigh", "dob" : ISOD
ate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "ge
nder" : "m", "vampires" : 2 }
```

Exercise 2)

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'})
{ "_id" : ObjectId("6556a447629d5aabbd11b6c1"), "name" : "Roooooodles", "dob" :
ISODate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender"
 : "m", "vampires" : 99 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c2"), "name" : "Solnara", "dob" : ISOD
ate("1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weig
ht" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c5"), "name" : "Raleigh", "dob" : ISOD
ate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "ge
nder" : "m", "vampires" : 2 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c6"), "name" : "Leia", "dob" : ISODate
("2001-10-08T14:53:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 601, "
gender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c7"), "name" : "Pilot", "dob" : ISODat
e("1997-03-01T05:03:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 650,
"gender" : "m", "vampires" : 54 }
```

Exercise 3)

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

Attribute	Value(s)
name	Malini
dob	11/03/2008
loves	pears, grapes
weight	450
gender	F
vampires	23
horns	1

Include the code you executed to insert this unicorn into the collection along with the output of a find command showing it is in the collection.

```
> db.unicorns.insert({name : 'Malini', dob : new Date(2008,10,3,0,0), loves : ['
pears', 'grapes'], weight : 450, gender : 'f', vampires : 23, horns : 1})
WriteResult({ "nInserted" : 1 })
```

```
> db.unicorns.find();
{ "_id" : ObjectId("6556a447629d5aabbd11b6be"), "name" : "Horny", "dob" : ISODat
e("1992-03-13T07:47:00Z"), "loves" : [ "carrot", "papaya" ], "weight" : 600, "ge
nder" : "m", "vampires" : 63 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6bf"), "name" : "Aurora", "dob" : ISODa
te("1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "ge
nder" : "f", "vampires" : 43 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c0"), "name" : "Unicrom", "dob" : ISOD
ate("1973-02-09T22:10:00Z"), "loves" : [ "energon", "redbull" ], "weight" : 984,
 "gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c1"), "name" : "Roooooodles", "dob" :
ISODate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender"
 : "m", "vampires" : 99 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c2"), "name" : "Solnara", "dob" : ISOD
ate("1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weig
ht" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c3"), "name" : "Ayna", "dob" : ISODate
("1998-03-07T08:30:00Z"), "loves" : [ "strawberry", "lemon" ], "weight" : 733, "
gender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c4"), "name" : "Kenny", "dob" : ISODat
e("1997-07-01T10:42:00Z"), "loves" : [ "grape", "lemon" ], "weight" : 690, "gend
er" : "m", "vampires" : 39 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c5"), "name" : "Raleigh", "dob" : ISOD
ate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "ge
nder" : "m", "vampires" : 2 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c6"), "name" : "Leia", "dob" : ISODate
("2001-10-08T14:53:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 601, "
gender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c7"), "name" : "Pilot", "dob" : ISODat
e("1997-03-01T05:03:00Z"), "loves" : [ "apple", "watermelon" ], "weight" : 650,
"gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c8"), "name" : "Nimue", "dob" : ISODat
e("1999-12-20T16:15:00Z"), "loves" : [ "grape", "carrot" ], "weight" : 540, "gen
der" : "f" }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c9"), "name" : "Dunx", "dob" : ISODate
("1976-07-18T18:18:00Z"), "loves" : [ "grape", "watermelon" ], "weight" : 704, "
gender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("6556a5b1629d5aabbd11b6ca"), "name" : "Malini", "dob" : ISODa
te("2008-11-03T00:00:00Z"), "loves" : [ "pears", "grapes" ], "weight" : 450, "ge
nder" : "f", "vampires" : 23, "horns" : 1 }
```

Exercise 4)

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'}, {$addToSet:{loves: 'apricots'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.unicorns.find({name: 'Malini'})
{ "_id" : ObjectId("6556a5b1629d5aabbd11b6ca"), "name" : "Malini", "dob" : ISODa
te("2008-11-03T00:00:00Z"), "loves" : [ "pears", "grapes", "apricots" ], "weight
" : 450, "gender" : "f", "vampires" : 23, "horns" : 1 }
```

Exercise 5)

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.deleteMany({weight:{$gt:600}})
{ "acknowledged" : true, "deletedCount" : 6 }
```

```
> db.unicorns.find();
{ "_id" : ObjectId("6556a447629d5aabbd11b6be"), "name" : "Horny", "dob" : ISODat
e("1992-03-13T07:47:00Z"), "loves" : [ "carrot", "papaya" ], "weight" : 600, "ge
nder" : "m", "vampires" : 63 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6bf"), "name" : "Aurora", "dob" : ISODa
te("1991-01-24T13:00:00Z"), "loves" : [ "carrot", "grape" ], "weight" : 450, "ge
nder" : "f", "vampires" : 43 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c1"), "name" : "Roooooodles", "dob" :
ISODate("1979-08-18T18:44:00Z"), "loves" : [ "apple" ], "weight" : 575, "gender"
 : "m", "vampires" : 99 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c2"), "name" : "Solnara", "dob" : ISOD
ate("1985-07-04T02:01:00Z"), "loves" : [ "apple", "carrot", "chocolate" ], "weig
ht" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c5"), "name" : "Raleigh", "dob" : ISOD
ate("2005-05-03T00:57:00Z"), "loves" : [ "apple", "sugar" ], "weight" : 421, "ge
nder" : "m", "vampires" : 2 }
{ "_id" : ObjectId("6556a447629d5aabbd11b6c8"), "name" : "Nimue", "dob" : ISODat
e("1999-12-20T16:15:00Z"), "loves" : [ "grape", "carrot" ], "weight" : 540, "gen
der" : "f" }
{ "_id" : ObjectId("6556a5b1629d5aabbd11b6ca"), "name" : "Malini", "dob" : ISODa
te("2008-11-03T00:00:00Z"), "loves" : [ "pears", "grapes", "apricots" ], "weight
" : 450, "gender" : "f", "vampires" : 23, "horns" : 1 }
```