Lab sheet 5

Name: J Viswaksena **RollNo:** AM.EN.U4AIE21035

Do the following statements in Casandra and draft a report with screenshots and your own explanation wherever necessary.

To get a docker container with cassandra:

```
docker pull cassandra
docker run --name medtech-cassandra -d cassandra:latest

docker exec -it medtech-cassandra cqlsh
```

create a keyspace:

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

```
(base) viswaksenajayam@Viswaksenas-MacBook-Pro ~ % docker pull cassandra
Using default tag: latest
(base) viswaksenajayam@Viswaksenas-MacBook-Pro ~ % docker run --name medtech-cassandra -d cassandra:latest
3f90d576fe1a38b6291f156324b9b5af4258a26aaa54b6c63036654819b536e1
(base) viswaksenajayam@Viswaksenas-MacBook-Pro ~ % docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' medtech-cassandra

172.17.0.2
(base) viswaksenajayam@Viswaksenas-MacBook-Pro ~ % docker exec -it medtech-cassandra cqlsh 172.17.0.2

Connected to Test Cluster at 172.17.0.2:9042
[cqlsh 6.1.0 | Cassandra 4.1.3 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
```

Pulled the latest Cassandra Docker image using docker pull cassandra, ran a container named medtech-cassandra. Retrieved the container's IP with docker inspect. Connected to Cassandra inside the container using docker exec -it.

```
cqlsh> create KEYSPACE mykeyspace WITH REPLICATION = {'class':'SimpleStrategy','replication_factor':1};
```

It establishes a new Cassandra keyspace named 'mykeyspace' using a simple replication strategy with a single replication factor for fault tolerance in the database.

```
cqlsh> describe keyspaces

mykeyspace  system_auth         system_schema  system_views
system      system_distributed  system_traces  system_virtual_schema
```

The CQL command DESCRIBE KEYSPACES in Cassandra's cqlsh shell provides a list of available keyspaces, presenting metadata about the existing keyspaces in the database, aiding in schema exploration and management.

```
cqlsh> use mykeyspace;
```

The command DESCRIBE KEYSPACES in Cassandra's cqlsh shell retrieves a list of existing keyspaces, displaying essential information about their structure. This aids in understanding the database schema and managing keyspaces within the Cassandra database.

```
cqlsh:mykeyspace> create table users(user_id int primary key, fname text, lname text);
```

The CQL command CREATE TABLE users (user_id int PRIMARY KEY, fname text, lname text); creates a table named 'users' in the 'mykeyspace' keyspace, defining columns for user_id, first name (fname), and last name (lname) with appropriate data types and constraints.

```
cqlsh:mykeyspace> describe tables;

users
```

The command DESCRIBE TABLES; in Cassandra's cqlsh shell provides a list of tables in the 'mykeyspace' keyspace. In this case, it returns a single table named 'users,' revealing the available tables within the specified

keyspace.

```
cqlsh:mykeyspace> INSERT INTO users (user_id,fname,Iname) VALUES (1745,'john','smith');
cqlsh:mykeyspace> select * from users;

 user_id | fname | iname
---------+-------+-------
    1745 |  john | smith

(1 rows)
```

The CQL command INSERT INTO users (user_id, fname, Iname) VALUES (1745, 'john', 'smith'); adds a new row to the 'users' table in the 'mykeyspace' keyspace, assigning values to columns: user_id, first name (fname), and last name (Iname).

```
cqlsh:mykeyspace> INSERT INTO users (user_id, fname, Iname) VALUES (1801, 'Alice', 'Johnson');
cqlsh:mykeyspace> INSERT INTO users (user_id, fname, Iname) VALUES (1950, 'Bob', 'Williams');
cqlsh:mykeyspace> INSERT INTO users (user_id, fname, Iname) VALUES (2005, 'Emma', 'Davis');
cqlsh:mykeyspace> INSERT INTO users (user_id, fname, Iname) VALUES (2123, 'Chris', 'Miller');
```

Likewise we inserted some entries into the user table

```
cqlsh:mykeyspace> SELECT * FROM users WHERE user_id < 2000 ALLOW FILTERING;

 user_id | fname | iname
---------+-------+----------
    1745 |  john |    smith
    1950 |   Bob | Williams
    1801 | Alice |  Johnson

(3 rows)
```

The first query retrieves all rows from the 'users' table in the 'mykeyspace' keyspace where the 'user_id' is less than 2000, allowing filtering due to the inequality condition. The second query fetches a specific row where 'user_id' is 1745. The third query retrieves rows where 'Iname' is 'smith,' also allowing filtering. The ALLOW FILTERING clause is used because these queries involve non-indexed columns, potentially impacting performance.

```
cqlsh:mykeyspace> CREATE INDEX ON users (Iname);
```

The CQL command CREATE INDEX ON users (Iname); creates an index on the 'Iname' column in the 'users' table within the 'mykeyspace' keyspace. This index enhances query performance when searching for rows based on the 'Iname' column.

```
cqlsh:mykeyspace> CREATE TABLE tab2 ( id1 int, id2 int, first_name varchar, last_name varchar, PRIMARY KEY(id1,id2));
cqlsh:mykeyspace> ALTER TABLE users ADD telephone text;
cqlsh:mykeyspace> UPDATE users SET telephone = '21212121' where user_id = 1745;
cqlsh:mykeyspace> select * from users
              ... ;

 user_id | fname | iname    | telephone
---------+-------+----------+-----------
    1745 |  john |    smith |  21212121
    2005 |  Emma |    Davis |      null
    1950 |   Bob | Williams |      null
    1801 | Alice |  Johnson |      null
    2123 | Chris |   Miller |      null

(5 rows)
```

The first command creates a table named 'tab2' in the 'mykeyspace' keyspace with columns 'id1', 'id2', 'first_name', and 'last_name', defining a composite primary key on 'id1' and 'id2'. The second command alters the 'users' table, adding a new column 'telephone' with a text data type. The third command updates the 'telephone' column for the user with 'user_id' 1745 to '21212121'. These actions collectively involve table creation, altering an existing table, and updating data, demonstrating schema modification and data manipulation capabilities in Cassandra's CQL (Cassandra Query Language).

```
cqlsh:mykeyspace> TRUNCATE users;
cqlsh:mykeyspace> DROP table users;
```

The `TRUNCATE users;` command removes all data from the 'users' table in the 'mykeyspace' keyspace. The subsequent `DROP TABLE users;` command deletes the 'users' table entirely, including its schema. These operations clear data and remove the table from the Cassandra database.

```
cqlsh:mykeyspace> CREATE TABLE users(user_id int PRIMARY KEY,fname text,Iname text,emails set<text>);
```

The CQL command `CREATE TABLE users (user_id int PRIMARY KEY, fname text, Iname text, emails set<text>);` defines a 'users' table in the Cassandra keyspace with columns for user ID, first name, last name, and a set of email addresses. The user_id column serves as the primary key.

```
cqlsh:mykeyspace> INSERT INTO users (user_id,fname,Iname,emails) VALUES(1234,'Frodo','Baggins',{'f@baggins.com','baggins@gmail.com'});
cqlsh:mykeyspace> select * from users
             ... ;

 user_id | emails                                  | fname | iname
---------+-----------------------------------------+-------+---------
    1234 | {'baggins@gmail.com', 'f@baggins.com'} | Frodo | Baggins

(1 rows)
```

Adds a new row to the 'users' table, providing values for user ID, first name, last name, and a set of emails.

```
cqlsh:mykeyspace> UPDATE users SET emails = emails + {'fb@friendsofmorder.org'} WHERE user_id = 1234;
cqlsh:mykeyspace> SELECT user_id,emails FROM users WHERE user_id =1234;

 user_id | emails
---------+-------------------------------------------------------------------
    1234 | {'baggins@gmail.com', 'f@baggins.com', 'fb@friendsofmorder.org'}

(1 rows)
```

modifies the 'emails' column for the user with ID 1234 by adding the email 'fb@friendsofmorder.org' to the existing set of emails.

```
cqlsh:mykeyspace> UPDATE users SET emails = emails-{'fb@friendsofmorder.org'} WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails                                  | fname | iname
---------+-----------------------------------------+-------+---------
    1234 | {'baggins@gmail.com', 'f@baggins.com'} | Frodo | Baggins

(1 rows)
```

modifies the 'emails' column for the user with ID 1234 by removing the email 'fb@friendsofmorder.org' from the existing set of emails.

```
cqlsh:mykeyspace> DELETE emails FROM users WHERE user_id = 1234;
cqlsh:mykeyspace> ALTER TABLE users ADD top_places list<text>;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | top_places
---------+--------+-------+---------+------------
    1234 |   null | Frodo | Baggins |       null

(1 rows)
```

The first command removes the entire 'emails' column data for the user with ID 1234 in the 'users' table. The second command alters the table, adding a new column 'top_places' with a list of text. The final SELECT statement displays the updated row, reflecting the changes in the table structure and data.

```
cqlsh:mykeyspace> UPDATE users SET top_places = ['riverdell','rohan'] WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | top_places
---------+--------+-------+---------+------------------------
    1234 |   null | Frodo | Baggins | ['riverdell', 'rohan']

(1 rows)
```

modifies the 'top_places' column for the user with ID 1234, assigning the list of text values ['riverdell', 'rohan'] to this column.

```
cqlsh:mykeyspace> UPDATE users SET top_places = top_places + ['morder'] WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | top_places
---------+--------+-------+---------+-----------------------------------
    1234 |   null | Frodo | Baggins | ['riverdell', 'rohan', 'morder']

(1 rows)
```

```
cqlsh:mykeyspace> UPDATE users SET top_places[1] = 'riddermark' WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | top_places
---------+--------+-------+---------+-----------------------------------------
    1234 |   null | Frodo | Baggins | ['riverdell', 'riddermark', 'morder']

(1 rows)
```

modifies the 'top_places' list column of the 'users' table. It sets the second element (index 1) of the list to 'riddermark' for the user with 'user_id' 1234.

```
cqlsh:mykeyspace> DELETE top_places[2] FROM users WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | top_places
---------+--------+-------+---------+-----------------------------
    1234 |   null | Frodo | Baggins | ['riverdell', 'riddermark']

(1 rows)
```

Removes the third element (index 2) from the 'top_places' list column of the 'users' table. This action is applied to the user with 'user_id' 1234.

```
cqlsh:mykeyspace> UPDATE users SET top_places = top_places - ['riddermark'] WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | top_places
---------+--------+-------+---------+---------------
    1234 |   null | Frodo | Baggins | ['riverdell']

(1 rows)
```

Modifies the 'top_places' list column of the 'users' table for the user with 'user_id' 1234

```
cqlsh:mykeyspace> ALTER TABLE users ADD todo map<timestamp,text>;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | todo | top_places
---------+--------+-------+---------+------+---------------
    1234 |   null | Frodo | Baggins | null | ['riverdell']

(1 rows)
```

The CQL ALTER TABLE statement enhances the 'users' table by adding a new column 'todo' of type map, where keys are timestamps and values are corresponding text entries. This allows associating timestamped text data with each user in the 'users' table.

```
cqlsh:mykeyspace> UPDATE users
              ... SET todo = {'2012-09-24':'enter morder', '2012-10-02 12:00':'throw ring into mount doom', toTimestamp(now()):'die!'}
              ... WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | todo
                                    | top_places
---------+--------+-------+---------+------------------------------------------------------------------------------------------------------------------------------
-----------------------+--------------
    1234 |   null | Frodo | Baggins | {'2012-09-24 00:00:00.000000+0000': 'enter morder', '2012-10-02 12:00:00.000000+0000': 'throw ring into mount doom', '2024-01-15 13:43:25
.954000+0000': 'die!'} | ['riverdell']

(1 rows)
```

Modifies the 'users' table, specifically the 'todo' column, for the user with ID 1234. It adds or updates entries in the map 'todo' with timestamped text tasks, including the current timestamp for a new task.

```
cqlsh:mykeyspace> UPDATE users SET todo['2012-10-2 12:00'] = 'throw my precious into mount doom' WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | todo
                                           | top_places
---------+--------+-------+---------+------------------------------------------------------------------------------------------------------------------------------
------------------------------+--------------
    1234 |   null | Frodo | Baggins | {'2012-09-24 00:00:00.000000+0000': 'enter morder', '2012-10-02 12:00:00.000000+0000': 'throw my precious into mount doom', '2024-01-15 1
3:43:25.954000+0000': 'die!'} | ['riverdell']
```

Modifies the 'users' table, updating the 'todo' map for the user with ID 1234. It adds or updates the entry with the timestamp '2012-10-2 12:00' to contain the task 'throw my precious into mount doom'.

```
cqlsh:mykeyspace> INSERT INTO users(user_id,todo) VALUES(1234,{'2013-9-22 12:01':'birthday wishes to Bilbo','2013-10-1 18:00':'Check into Inn of Prancing Pony'});
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | todo                                                                                                                  | t
op_places
---------+--------+-------+---------+-----------------------------------------------------------------------------------------------------------------------+--
-------------
    1234 |   null | Frodo | Baggins | {'2013-09-22 12:01:00.000000+0000': 'birthday wishes to Bilbo', '2013-10-01 18:00:00.000000+0000': 'Check into Inn of Prancing Pony'} | [
'riverdell']

(1 rows)
```

Adds a new row to the 'users' table, assigning user_id 1234 and a 'todo' map containing timestamped tasks, such as 'birthday wishes to Bilbo' on '2013-9-22 12:01' and 'Check into Inn of Prancing Pony' on '2013-10-1 18:00'.

```
cqlsh:mykeyspace> DELETE todo['2012-09-24'] FROM users WHERE user_id = 1234;
cqlsh:mykeyspace> select * from users;

 user_id | emails | fname | iname   | todo                                                                                                                  | t
op_places
---------+--------+-------+---------+-----------------------------------------------------------------------------------------------------------------------+--
-------------
    1234 |   null | Frodo | Baggins | {'2013-09-22 12:01:00.000000+0000': 'birthday wishes to Bilbo', '2013-10-01 18:00:00.000000+0000': 'Check into Inn of Prancing Pony'} | [
'riverdell']

(1 rows)
```

Removes the entry with the timestamp '2012-09-24' from the 'todo' map within the 'users' table where user_id