Exercise: 7

Working with Collections in Cassandra using DataStax Astra

Aim:

The aim is to demonstrate how to manage and manipulate various types of collections (sets, lists, maps) in Cassandra through creating, updating, and querying data within a "users" table.

Procedure:

Initialize Cassandra Table:

Create a table named users in Cassandra, incorporating diverse data types for columns: an int for user_id (Primary Key), text for fname and lname, and collections such as a set<text> for emails, a list<text> for top_places, and a map<timestamp, text> for todo.

Populate Table with Data:

Sets: Insert a record for a user including a set of email addresses.

Lists: After altering the table to include a list<text> column, update the record to include a list of top places.

Maps: Alter the table to include a map<timestamp, text> column and update the user's record to map tasks to specific timestamps.

Modify Collection Data:

Sets: Demonstrate adding to and removing from a set of emails, including clearing the entire set.

Lists: Show how to append to a list, modify a specific element within it, remove an element, and subtract occurrences of specific elements.

Maps: Illustrate adding to and removing from a map, including updating specific key-value pairs and clearing a key-value pair.

Cleanup:

Optionally, demonstrate the deletion of specific columns or the entire row for a user to showcase how to manage and clean up data within Cassandra, ensuring the database remains organized and efficient.

Query and Retrieve Data:

Execute queries to retrieve updated collections (set, list, map) from the users table for specific user_id values to demonstrate the changes made during the modification steps.

Queries :

Collections – SETS:

A set is an unordered set of values. It is possible to solve the problem of multi-valued fields, like emails.

1. create a table named users in Cassandra with a column named emails as a set of text Values.

CREATE TABLE users (

user_id int PRIMARY KEY,

fname text,

lname text,

emails set<text>

);

```
token@cqlsh> describe keyspaces;

datastax_sla   default_keyspace    system_traces
system_auth    data_endpoint_auth  system_views
system_schema  system              system_virtual_schema

token@cqlsh> use default_keyspace;
token@cqlsh:default_keyspace> CREATE TABLE users (
                    ... user_id int PRIMARY KEY,
                    ... fname text,
                    ... lname text,
                    ... emails set<text>
                    ... );
```

2. insert data into the emails set for a user with user_id 1234.

INSERT INTO users (user_id, fname, lname, emails)

VALUES(1234, 'Frodo', 'Baggins', {'f@baggins.com', 'baggins@gmail.com'});

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

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

(1 rows)
```

3. Retrieve the user_id and emails for the user with user_id 1234.

SELECT user_id, emails FROM users WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> SELECT user_id, emails FROM users WHERE user_id = 1234;

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

4. Add the email address 'fb@friendsofmordor.org' to the set of emails for the user with user_id 1234

UPDATE users SET emails = emails + {'fb@friendsofmordor.org'} WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> UPDATE users SET emails = emails + {'fb@friendsofmordor.org'} WHERE use
r_id = 1234;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails                                                                | fname | lname
---------+-----------------------------------------------------------------------+-------+---------
    1234 | {'baggins@gmail.com', 'f@baggins.com', 'fb@friendsofmordor.org'} | Frodo | Baggins
```


5. Retrieve the user_id and updated emails for the user with user_id 1234 after adding the new email.

SELECT user_id, emails FROM users WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> SELECT user_id, emails FROM users WHERE user_id = 1234;

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

6.Remove the email address 'fb@friendsofmordor.org' from the set of emails for the user with user_id 1234

UPDATE users SET emails = emails - {'fb@friendsofmordor.org'} WHERE user_id = 1234;

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

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

7.Remove all email addresses from the set for the user with user_id 1234.

UPDATE users SET emails = {} WHERE user_id = 1234;

```
(1 rows)
token@cqlsh:default_keyspace> UPDATE users SET emails = {} WHERE user_id = 1234;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname
---------+--------+-------+---------
    1234 |   null | Frodo | Baggins

(1 rows)
```

Collections – LISTS:

A list is a sorted collection of non-unique values where elements are ordered by their positions in the list.

1. Alter the users table to include a column named top_places as a list of text values

ALTER TABLE users ADD top_places list<text>;

```
(1 rows)
token@cqlsh:default_keyspace> ALTER TABLE users ADD top_places list<text>;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname   | top_places
---------+--------+-------+---------+------------
    1234 |   null | Frodo | Baggins |       null
```

2. Insert a list of the best places visited into the top_places column for the user with user_id 1234.

UPDATE users SET top_places = ['rivendell', 'rohan'] WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> UPDATE users SET top_places = [ 'rivendell', 'rohan' ] WHERE user_id = 1234;
token@cqlsh:default_keyspace> select*from users;
 user_id | emails | fname | lname   | top_places
---------+--------+-------+---------+------------------------
    1234 |   null | Frodo | Baggins | ['rivendell', 'rohan']

(1 rows)
```

3.Add the place 'mordor' to the end of the list of top_places for the user with user_id 1234

UPDATE users SET top_places = top_places + ['mordor'] WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> UPDATE users SET top_places = top_places + [ 'mordor' ] WHERE user_id = 1234;
token@cqlsh:default_keyspace> select*from users;
 user_id | emails | fname | lname   | top_places
---------+--------+-------+---------+---------------------------------
    1234 |   null | Frodo | Baggins | ['rivendell', 'rohan', 'mordor']

(1 rows)
```

4. Update the second place in the list top_places to 'riddermark' for the user with user_id

1234.

UPDATE users SET top_places[1] = 'riddermark' WHERE user_id = 1234;

```
token@cqlsh:default_keyspace>
token@cqlsh:default_keyspace> ALTER TABLE users ADD top_places_map map<int, text>;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname   | top_places                      | top_places_map
---------+--------+-------+---------+---------------------------------+----------------
    1234 |   null | Frodo | Baggins | ['rivendell', 'rohan', 'mordor'] |           null

(1 rows)
token@cqlsh:default_keyspace> UPDATE users SET top_places_map[1] = 'riddermark' WHERE user_id = 1234;token@cqlsh:default_keyspace> select*from users;
 user_id | emails | fname | lname   | top_places                      | top_places_map
---------+--------+-------+---------+---------------------------------+------------------
    1234 |   null | Frodo | Baggins | ['rivendell', 'rohan', 'mordor'] | {1: 'riddermark'}

(1 rows)
token@cqlsh:default_keyspace>
```

5.Remove the first place from the top_places list for the user with user_id 1234

DELETE top_places[1] FROM users WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> DELETE top_places_map[1] FROM users WHERE user_id = 1234;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname   | top_places                        | top_places_map
---------+--------+-------+---------+-----------------------------------+---------------
    1234 |   null | Frodo | Baggins | ['rivendell', 'rohan', 'mordor'] |          null

(1 rows)
```

6.Retrieve the user_id and top_places list for the user with user_id 1234.

SELECT user_id, top_places FROM users WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> SELECT user_id, top_places FROM users WHERE user_id = 1234;

 user_id | top_places
---------+-----------------------------------
    1234 | ['rivendell', 'rohan', 'mordor']

(1 rows)
```


Collections – MAP:

A map allows us to associate two elements, in the form of a key / value. It can be used, for example, to save the schedules of different events in a user profile. Using Cassandra, each element in a Map is stored as a column that we can modify, replace and query.

1.Alter the users table to include a column named todo as a map with timestamp keys and text values

ALTER TABLE users ADD todo map<timestamp, text>;

```
token@cqlsh:default_keyspace> ALTER TABLE users ADD todo map<timestamp, text>;
token@cqlsh:default_keyspace> select*from users;
 user_id | emails | fname | lname   | todo | top_places                        | top_places_map
---------+--------+-------+---------+------+-----------------------------------+---------------
    1234 |   null | Frodo | Baggins | null | ['rivendell', 'rohan', 'mordor'] |          null

(1 rows)
```

2. Insert tasks into the todo map for the user with user_id 1234.

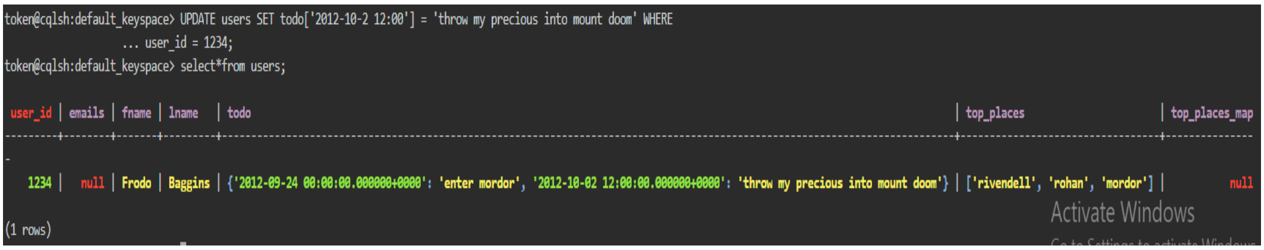
UPDATE users SET todo = { '2012-9-24' : 'enter mordor', '2012-10-2 12:00' : 'throw ring into mount doom' } WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> UPDATE users SET todo = { '2012-09-24' : 'enter mordor', '2012-10-02T12:00:00.000Z' : 'throw ring into mount doom' } WHERE user_id = 1234;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname   | todo                                                                                                            | top_places                     | top_places_map
---------+--------+-------+---------+-----------------------------------------------------------------------------------------------------------------+--------------------------------+----------------
    1234 |   null | Frodo | Baggins | {'2012-09-24 00:00:00.000000+0000': 'enter mordor', '2012-10-02 12:00:00.000000+0000': 'throw ring into mount doom'} | ['rivendell', 'rohan', 'mordor'] |           null

(1 rows)
```

3. Add a task with the timestamp '2012-10-2 12:00' to the todo map for the user with user_id 1234

UPDATE users SET todo['2012-10-2 12:00'] = 'throw my precious into mount doom' WHERE

user_id = 1234;



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

 user_id | emails | fname | lname   | todo                                                                                                                   | top_places                       | top_places_map
---------+--------+-------+---------+------------------------------------------------------------------------------------------------------------------------+----------------------------------+----------------
-
    1234 |   null | Frodo | Baggins | {'2012-09-24 00:00:00.000000+0000': 'enter mordor', '2012-10-02 12:00:00.000000+0000': 'throw my precious into mount doom'} | ['rivendell', 'rohan', 'mordor'] |           null

(1 rows)
```

4. Insert tasks into the todo map for the user with user_id 1234 using the INSERT statement.

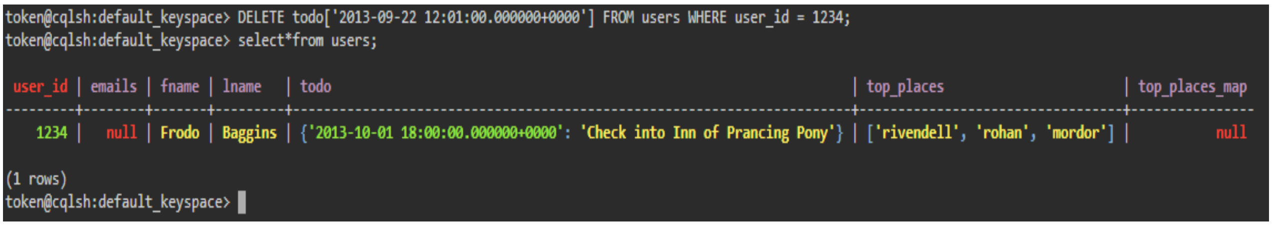
INSERT INTO users (user_id, todo) VALUES (1234, { '2013-9-22 12:01' : 'birthday wishes to

Bilbo', '2013-10-1 18:00' : 'Check into Inn of Prancing Pony' });

```
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname   | todo                                                                                                                         | top_places                       | top_places_map
---------+--------+-------+---------+------------------------------------------------------------------------------------------------------------------------------+----------------------------------+----------------
    1234 |   null | Frodo | Baggins | {'2013-09-22 12:01:00.000000+0000': 'birthday wishes to\nBilbo', '2013-10-01 18:00:00.000000+0000': 'Check into Inn of Prancing Pony'} | ['rivendell', 'rohan', 'mordor'] |           null

(1 rows)
token@cqlsh:default_keyspace>
```

5. Remove the task with the timestamp '2012-9-24' from the todo map for the user with user_id 1234

DELETE todo['2012-9-24'] FROM users WHERE user_id = 1234;



```
token@cqlsh:default_keyspace> DELETE todo['2013-09-22 12:01:00.000000+0000'] FROM users WHERE user_id = 1234;
token@cqlsh:default_keyspace> select*from users;

 user_id | emails | fname | lname   | todo                                                               | top_places                       | top_places_map
---------+--------+-------+---------+--------------------------------------------------------------------+----------------------------------+----------------
    1234 |   null | Frodo | Baggins | {'2013-10-01 18:00:00.000000+0000': 'Check into Inn of Prancing Pony'} | ['rivendell', 'rohan', 'mordor'] |           null

(1 rows)
token@cqlsh:default_keyspace>
```

6.Retrieve the user_id and todo map for the user with user_id 1234.

SELECT user_id, todo FROM users WHERE user_id = 1234;

```
token@cqlsh:default_keyspace> SELECT user_id, todo FROM users WHERE user_id = 1234;

 user_id | todo
---------+-----------------------------------------------------------------------
    1234 | {'2013-10-01 18:00:00.000000+0000': 'Check into Inn of Prancing Pony'}

(1 rows)
token@cqlsh:default_keyspace>
```

RESULT:

The algorithm demonstrates the management of set, list, and map collections in Cassandra, from initialization and data population to modification and retrieval, concluding with optional data cleanup.