ADVANCED DATABASE MANAGEMENT SYSTEM

ASSIGNMENT NO : 3

SUBMITTED BY,

GROUP III (11-15)

ARUN V DAS

TOPIC: Design database schemas and implement min 10 queries using Cassandra column based databases

Aim
To Design database schemas and implement queries using Cassandra databases

Objective(s)	
1	Study of NOSQL Cassandra.
2	Study the procedure to execute a query using Apache Cassandra.
3	Execute min 10 queries using Cassandra column based database.

1. STUDY OF NOSQL CASSANDRA

Cassandra

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

Features

1) Scalability:

- Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.

2) Fault-tolerant:

- Data is automatically replicated to multiple nodes for fault-tolerance.
- Replication across multiple data centers is supported.
- Failed nodes can be replaced with no downtime.

3) MapReduce support:

- Cassandra has Hadoop integration, with MapReduce support.

4) Query language:

- Cassandra introduces CQL (Cassandra Query Language), a SQL-like alternative to the traditional RPC interface.
  - **Keyspace:** Keyspace is the outermost container for data.It is similar to the schema in a relational database.

Basic attributes of Keyspace are:

- **Replication Factor:** It is the number of machines in the cluster that will receive copies of the same data
- **Replica Placement Strategy:** It is a strategy to place replicas in the ring
- Simple Strategy,
- Old Network Topology Strategy
- Network Topology Strategy

Column Families: Column family is a NoSQL object that contains columns of related data. It is a tuple (pair) that consists of a key-value pair, where the key is mapped to a value that is a set of columns. It is similar to a table in a relational database and each key-value pair being a row.

Each column is a tuple (triplet) consisting of

- Column name
- Value
- Timestamp

1.STUDY THE PROCEDURE TO EXECUTE A QUERY USING APACHE CASSANDRA

Introduction

DataStax Community Edition must be installed on the system before installing Cassandra. Verify the Cassandra installation using the following command:

- **$ Cassandra version**

- If Cassandra is already installed on system, then you will get the following response:

- **Connected to Test Cluster at 127.0.0.1:9042.**
- **[cqlsh 5.0.1 | Cassandra 2.2.4 | CQL spec 3.3.1 | Native protocol v4] Use HELP for help.**
- **WARNING: pyreadline dependency missing. Install to enable tab completion. From source with checksum 79e53ce7994d1628b240f09af91e1af4**

Creating KEYSPACE :

Cassandra is technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it.

Create KEYSPACE Statement

- Create KEYSPACE is a statement used to create a KEYSPACE in Cassandra. A KEYSPACE in Cassandra is aKEYSPACE or a collection of tables. The syntax for this statement is as follows:
  cqlsh>CREATE KEYSPACE ABC userdb replication={ 'class':'SimpleStrategy','replication_factor':'1'};

- Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.

- The following query is executed to create a database named userdb:

cqlsh> userdb;

Or
cqlsh> CREATE SCHEMA userdb;

The following query is used to verify a databases list:

cqlsh>:userdb> show tables; Improper show command. default
userdb

For creating Table:

Create Table

```
CREATE TABLE test_table (
id int, address text, name text,
PRIMARY KEY ((id))
);
```

CURD using cql Updating Table:

Update Table
insert into test_table (id, name, address) values (4, 'somnath', 'Sus');

CURD using cql Delete Table

Deleting rows from Table

delete from test_table where id =1;

The following queries are used to drop a database. Let us assume that the database name is **userdb.**
cqlsh:userdb> delete from Tablename where condition;

For describing tables

cqlsh:userdb> describe tables; show all table names
cqlsh:userdb>

For Help of any Topic

Cqshl> Help;

Display topics

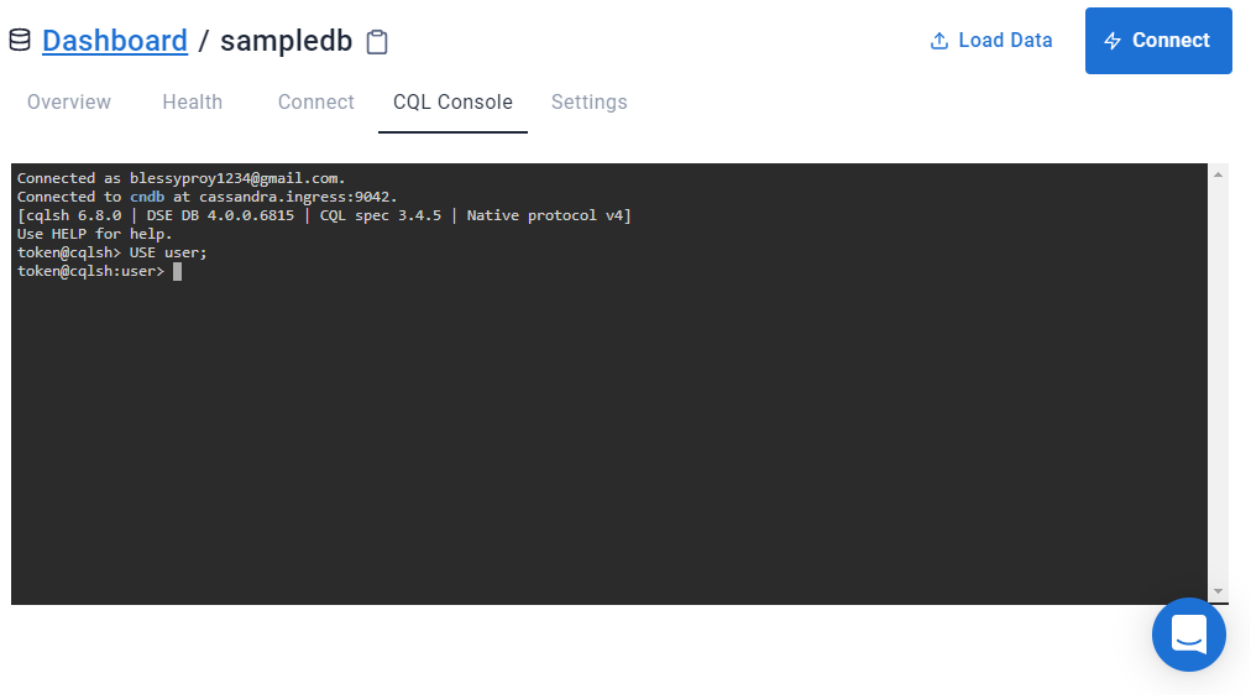
Cqshl> Help topic name;

Help open in Browser.

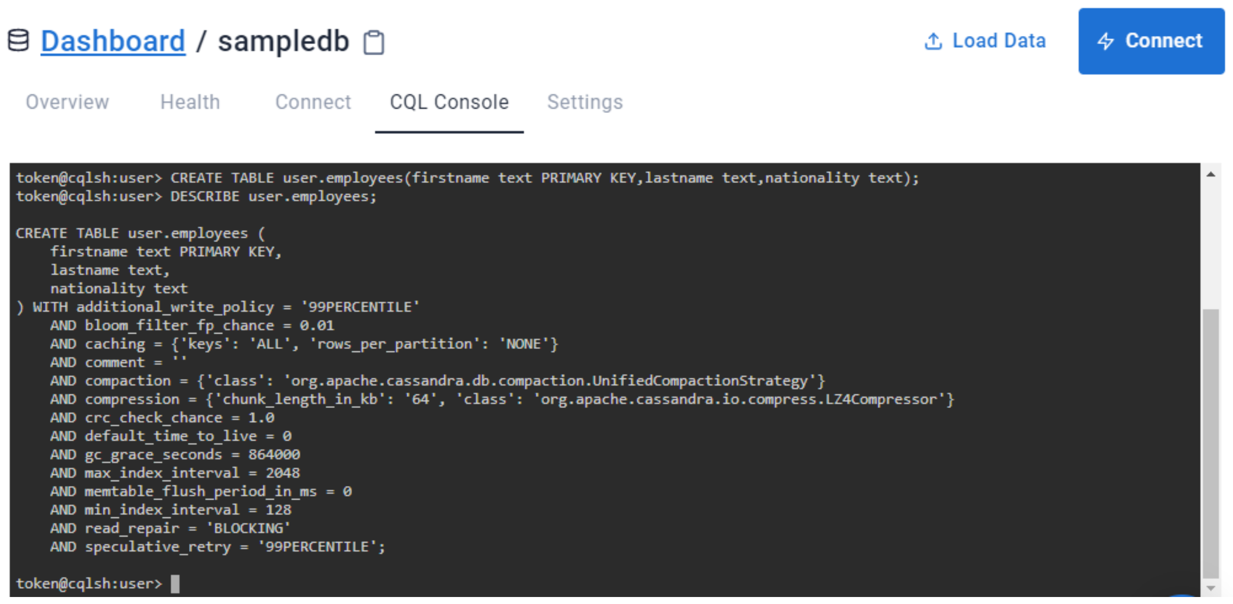
3.Execute minimum 10 queries using Cassandra column-based database

INITIAL PROCEDURE:

Step 1: Here we consider the database "sampledb"
Step 2: Creating and updating a keyspace ,here keyspace is "user"
Step 3: Creating a table "employees"
Step 4: Describe table "employees"



```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user>
```



```
token@cqlsh:user> CREATE TABLE user.employees(firstname text PRIMARY KEY,lastname text,nationality text);
token@cqlsh:user> DESCRIBE user.employees;

CREATE TABLE user.employees (
    firstname text PRIMARY KEY,
    lastname text,
    nationality text
) WITH additional_write_policy = '99PERCENTILE'
    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.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    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 = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';

token@cqlsh:user>
```

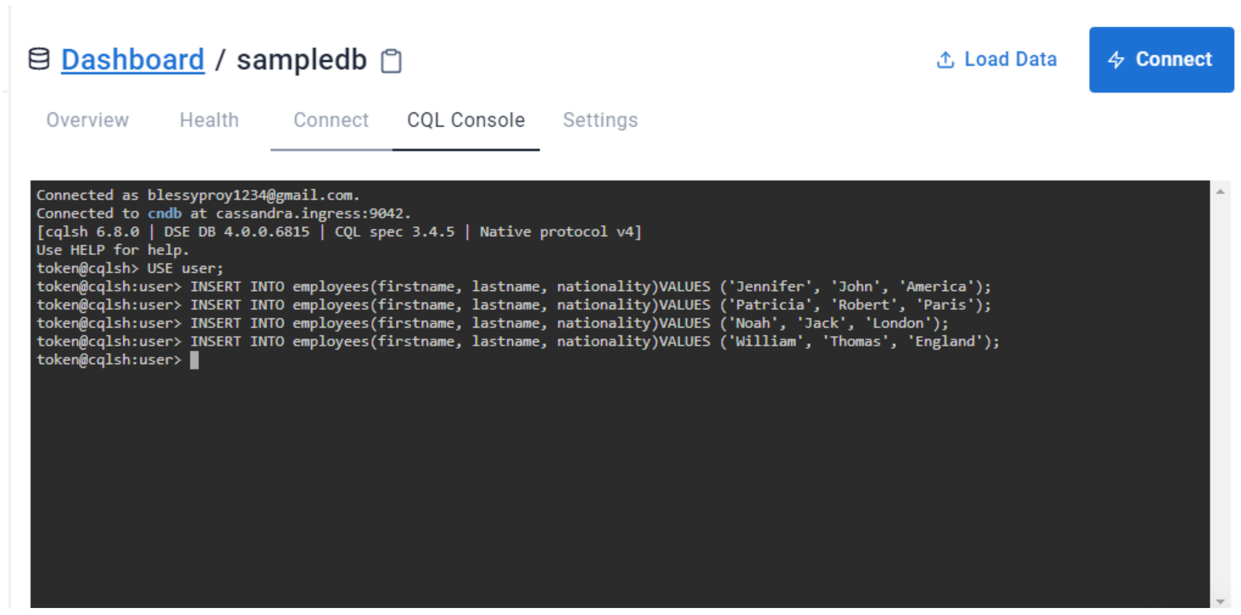
QUERIES

1. INSERTING DATA INTO A TABLE

Procedure:

- INSERT INTO employees(firstname, lastname, nationality)VALUES ('Jennifer', 'John', 'America');
- INSERT INTO employees(firstname, lastname, nationality)VALUES ('Patricia', 'Robert', 'Paris');
- INSERT INTO employees(firstname, lastname, nationality)VALUES ('Noah', 'Jack', 'London');
- INSERT INTO employees(firstname, lastname, nationality)VALUES ('William', 'Thomas', 'England');

OUTPUT



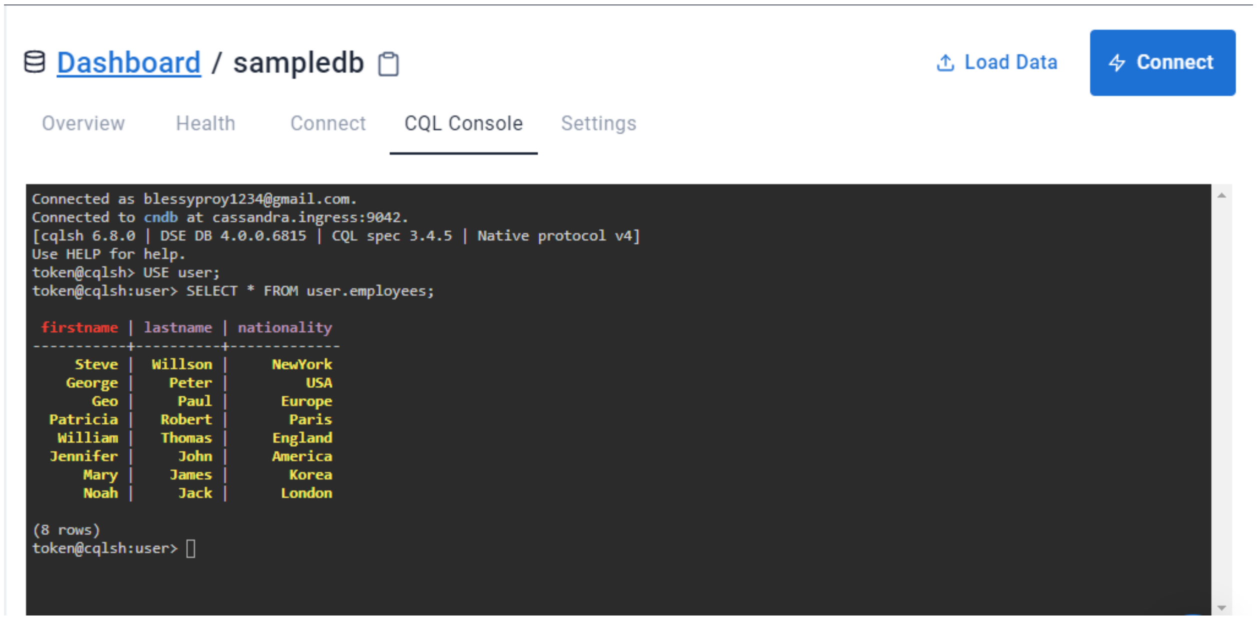
```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user> INSERT INTO employees(firstname, lastname, nationality)VALUES ('Jennifer', 'John', 'America');
token@cqlsh:user> INSERT INTO employees(firstname, lastname, nationality)VALUES ('Patricia', 'Robert', 'Paris');
token@cqlsh:user> INSERT INTO employees(firstname, lastname, nationality)VALUES ('Noah', 'Jack', 'London');
token@cqlsh:user> INSERT INTO employees(firstname, lastname, nationality)VALUES ('William', 'Thomas', 'England');
token@cqlsh:user>
```

2.RETRIEVAL OF ALL DATA

Procedure:

- SELECT * FROM user.employees;

OUTPUT



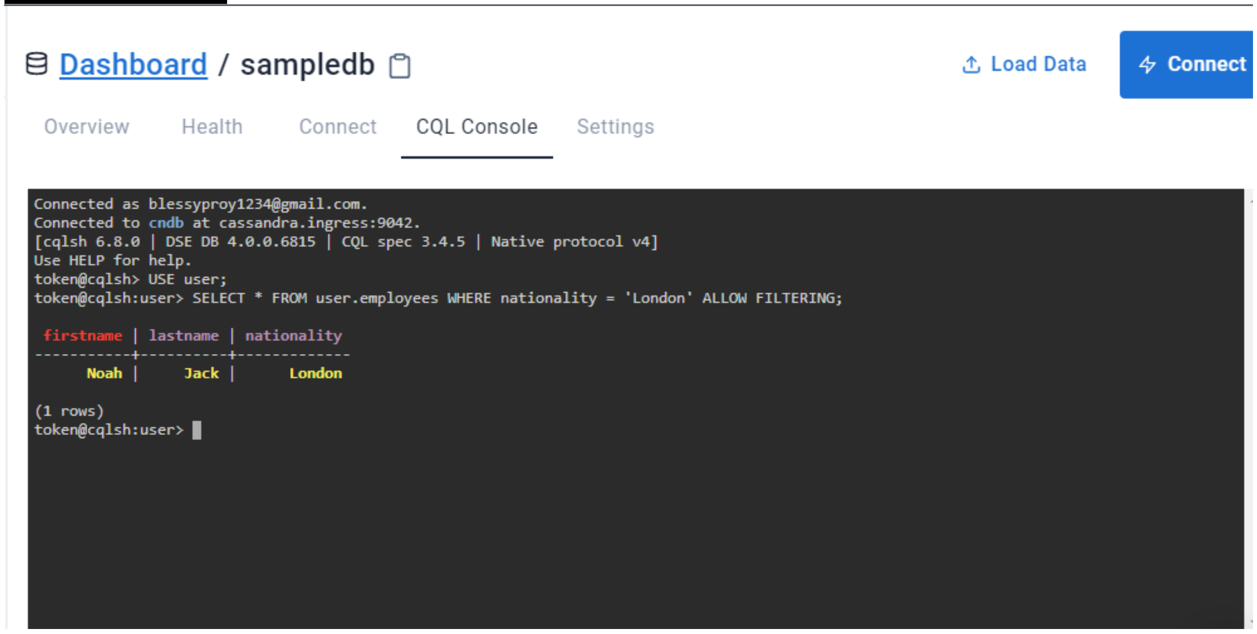
```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user> SELECT * FROM user.employees;

 firstname | lastname | nationality
-----------+----------+-------------
     Steve |  Willson |     NewYork
    George |    Peter |         USA
       Geo |     Paul |      Europe
  Patricia |   Robert |       Paris
   William |   Thomas |     England
  Jennifer |     John |     America
      Mary |    James |       Korea
      Noah |     Jack |      London

(8 rows)
token@cqlsh:user>
```

3.RETRIEVE THE DETAILS OF STAFFS USING CONDITIONS

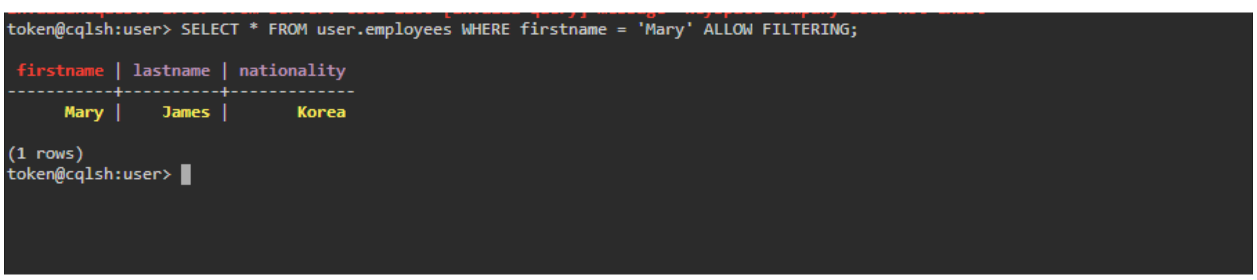
Procedure:

- SELECT * FROM user.employees WHERE nationality = 'London' ALLOW FILTERING;
- SELECT * FROM user.employees WHERE firstname = 'Mary' ALLOW FILTERING;

OUTPUT



```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user> SELECT * FROM user.employees WHERE nationality = 'London' ALLOW FILTERING;

 firstname | lastname | nationality
-----------+----------+-------------
      Noah |     Jack |      London

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



```
token@cqlsh:user> SELECT * FROM user.employees WHERE firstname = 'Mary' ALLOW FILTERING;

 firstname | lastname | nationality
-----------+----------+-------------
      Mary |    James |       Korea

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

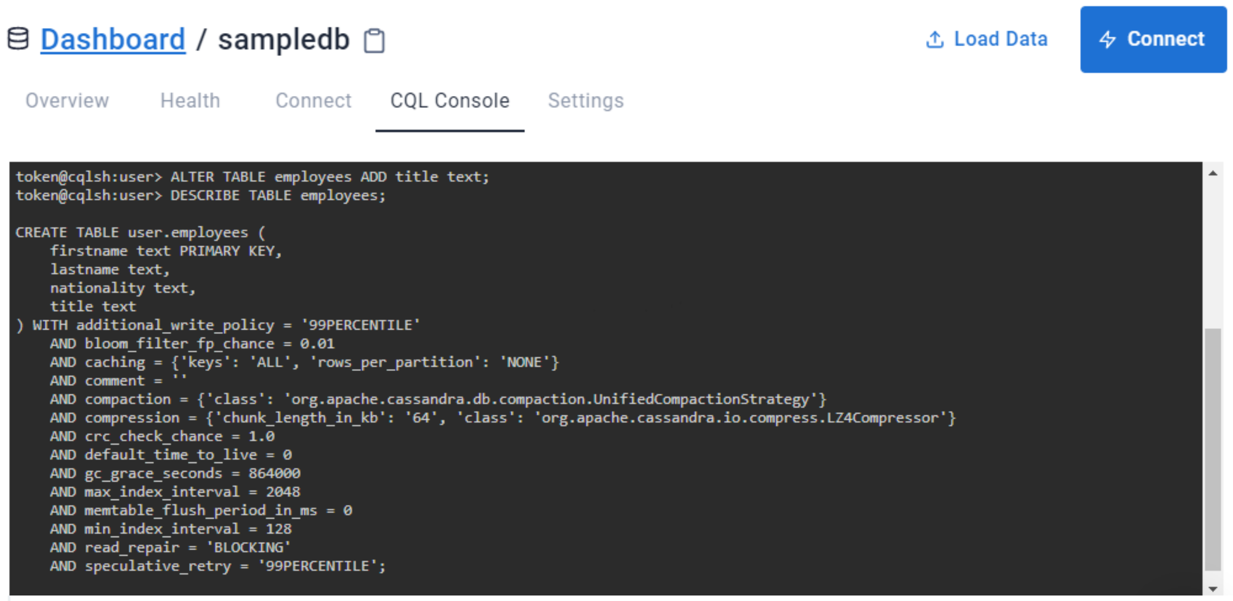
1. ALTER TABLE command

- Adding new column 'title' into the table 'employees'

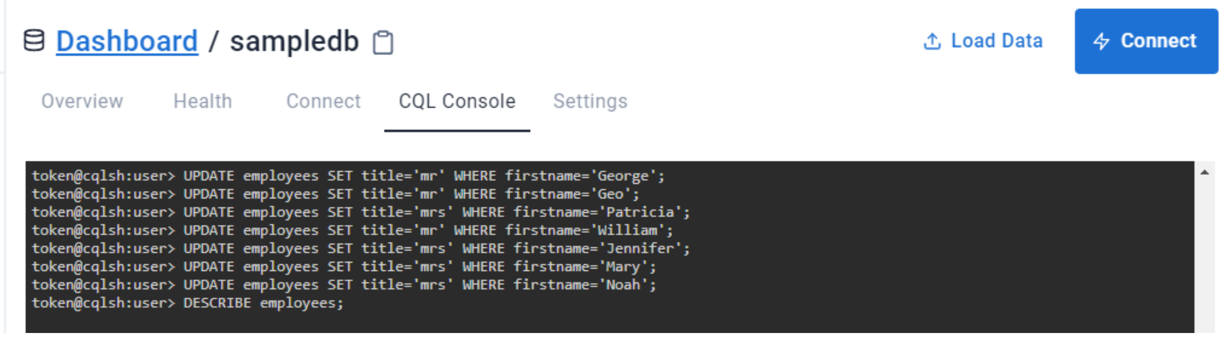
Procedure:

- ALTER TABLE employees ADD title text;
- DESCRIBE TABLE employees;
- Add data into that column using update command
- Then displayed the details using select command

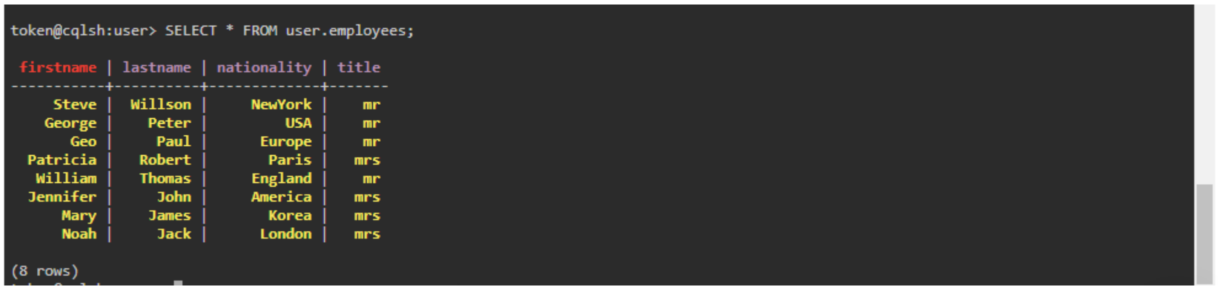
OUTPUT



```
token@cqlsh:user> ALTER TABLE employees ADD title text;
token@cqlsh:user> DESCRIBE TABLE employees;

CREATE TABLE user.employees (
    firstname text PRIMARY KEY,
    lastname text,
    nationality text,
    title text
) WITH additional_write_policy = '99PERCENTILE'
    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.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    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 = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';
```



```
token@cqlsh:user> UPDATE employees SET title='mr' WHERE firstname='George';
token@cqlsh:user> UPDATE employees SET title='mr' WHERE firstname='Geo';
token@cqlsh:user> UPDATE employees SET title='mrs' WHERE firstname='Patricia';
token@cqlsh:user> UPDATE employees SET title='mr' WHERE firstname='William';
token@cqlsh:user> UPDATE employees SET title='mrs' WHERE firstname='Jennifer';
token@cqlsh:user> UPDATE employees SET title='mrs' WHERE firstname='Mary';
token@cqlsh:user> UPDATE employees SET title='mrs' WHERE firstname='Noah';
token@cqlsh:user> DESCRIBE employees;
```



```
token@cqlsh:user> SELECT * FROM user.employees;

 firstname | lastname | nationality | title
-----------+----------+-------------+-------
     Steve |  Willson |     NewYork |    mr
    George |    Peter |         USA |    mr
       Geo |     Paul |      Europe |    mr
  Patricia |   Robert |       Paris |   mrs
   William |   Thomas |     England |    mr
  Jennifer |     John |     America |   mrs
      Mary |    James |       Korea |   mrs
      Noah |     Jack |      London |   mrs

(8 rows)
```

4.RETRIEVING TIMESTAMPS

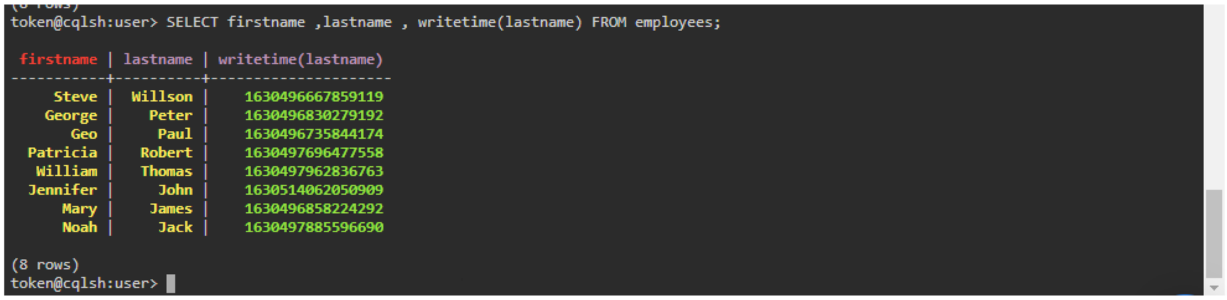
Timestamps

Each time you write data into Cassandra, a timestamp is generated for each column value that is updated. Internally, Cassandra uses these timestamps for resolving any conflicting changes that are made to the same value.

Procedure:

- SELECT firstname ,lastname , writetime(lastname) FROM employees;

OUTPUT



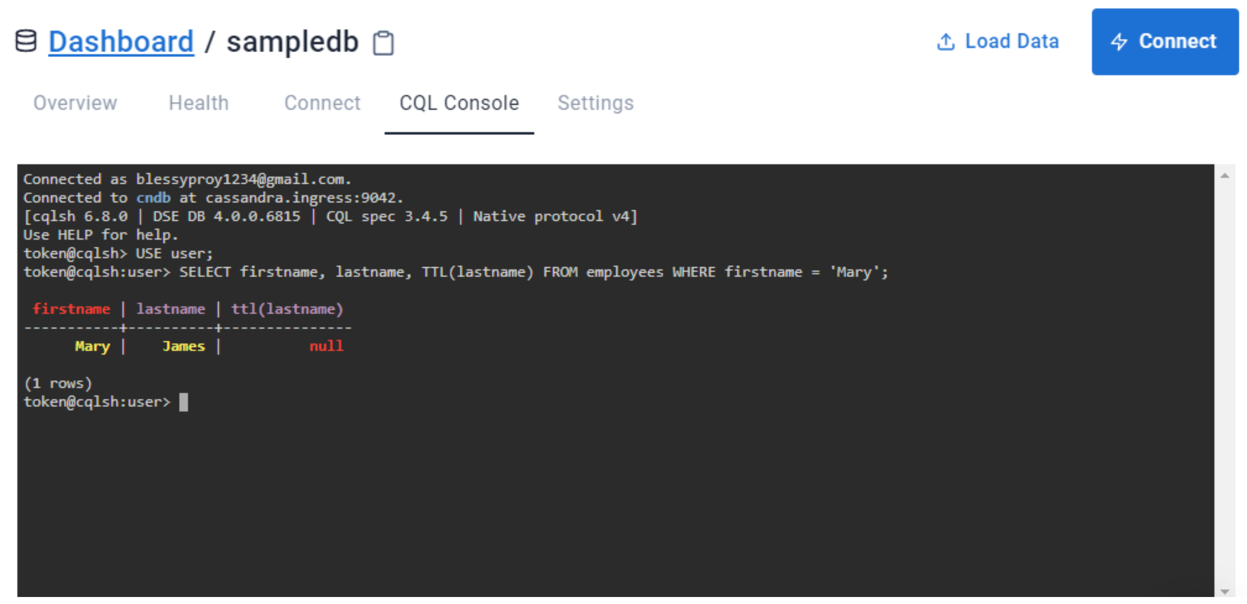
```
token@cqlsh:user> SELECT firstname ,lastname , writetime(lastname) FROM employees;

 firstname | lastname | writetime(lastname)
-----------+----------+---------------------
     Steve |  Willson |    1630496667859119
    George |    Peter |    1630496830279192
       Geo |     Paul |    1630496735844174
  Patricia |   Robert |    1630497696477558
   William |   Thomas |    1630497962836763
  Jennifer |     John |    1630514062050909
      Mary |    James |    1630496858224292
      Noah |     Jack |    1630497885596690

(8 rows)
token@cqlsh:user>
```

5.RETRIEVE THE TTL VALUE FOR ‘ Mary’s lastname’

Procedure:

- SELECT firstname, lastname, TTL(lastname) FROM employees WHERE firstname = 'Mary';

OUTPUT



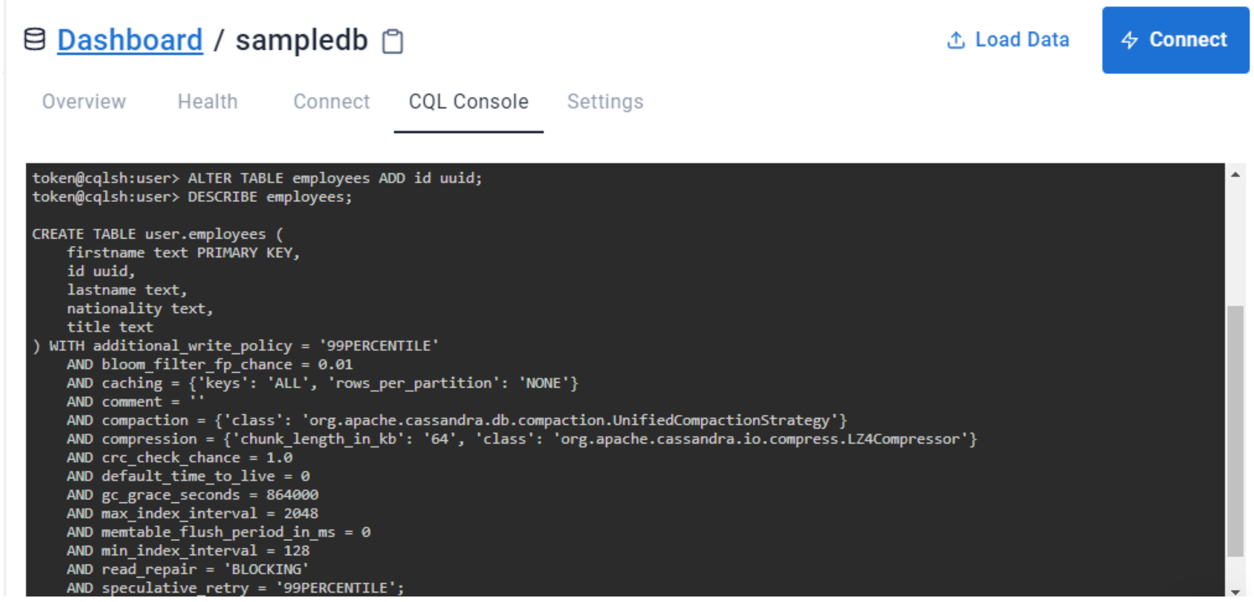
```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user> SELECT firstname, lastname, TTL(lastname) FROM employees WHERE firstname = 'Mary';

 firstname | lastname | ttl(lastname)
-----------+----------+---------------
      Mary |    James |          null

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

6. QUERY TO ADD THE IDENTIFIER USING A UUID

Procedure:

- ALTER TABLE employees ADD id uuid;
- DESCRIBE employees;

OUTPUT



```
token@cqlsh:user> ALTER TABLE employees ADD id uuid;
token@cqlsh:user> DESCRIBE employees;

CREATE TABLE user.employees (
    firstname text PRIMARY KEY,
    id uuid,
    lastname text,
    nationality text,
    title text
) WITH additional_write_policy = '99PERCENTILE'
    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.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    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 = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';
```

7.QUERY TO INSERT AN ID FOR MARY USING UUID() FUNCTION AND THEN VIEW THE RESULTS

uuid
A universally unique identifier (UUID) is a 128-bit value in which the bits conform to one of several types, of which the most commonly used are known as Type 1 and Type 4. The CQL uuid type is a Type 4 UUID, which is based entirely on random numbers. UUIDs are typically represented as dash-separated sequences of hex digits. For example:

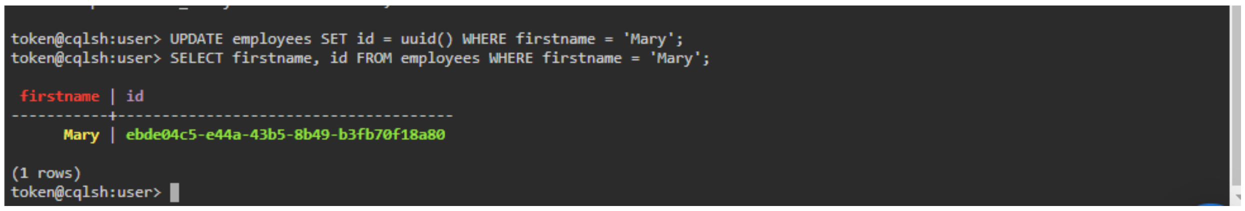
1a6300ca-0572-4736-a393-c0b7229e193e

The uuid type is often used as a surrogate key, either by itself or in combination with other values.

Procedure:

- UPDATE employees SET id = uuid() WHERE firstname = 'Mary';
- SELECT firstname, id FROM employees WHERE firstname = 'Mary';

OUTPUT



```
token@cqlsh:user> UPDATE employees SET id = uuid() WHERE firstname = 'Mary';
token@cqlsh:user> SELECT firstname, id FROM employees WHERE firstname = 'Mary';

 firstname | id
-----------+--------------------------------------
      Mary | ebde04c5-e44a-43b5-8b49-b3fb70f18a80

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

8.QUERY TO MODIFY OUR STAFFS TABLE TO ADD A SET OF EMAIL ADDRESSES

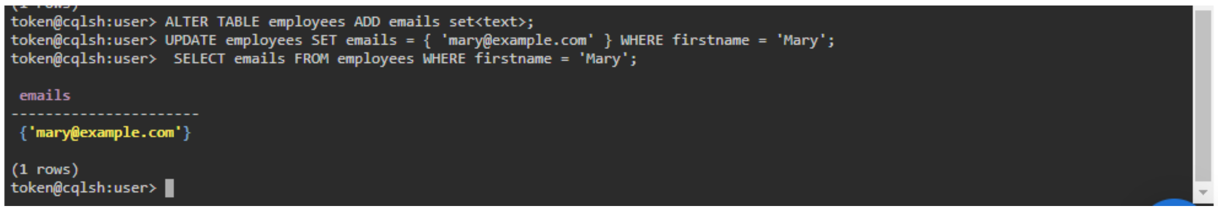
Set

The set data type stores a collection of elements. The elements are unordered, but cqlsh returns the elements in sorted order. For example, text values are returned in alphabetical order. One advantage of using set is the ability to insert additional items without having to read the contents first.

Procedure:

- ALTER TABLE employees ADD emails set<text>;
- UPDATE employees SET emails = { 'mary@example.com' } WHERE firstname = 'Mary';
- SELECT emails FROM employees WHERE firstname = 'Mary';

OUTPUT



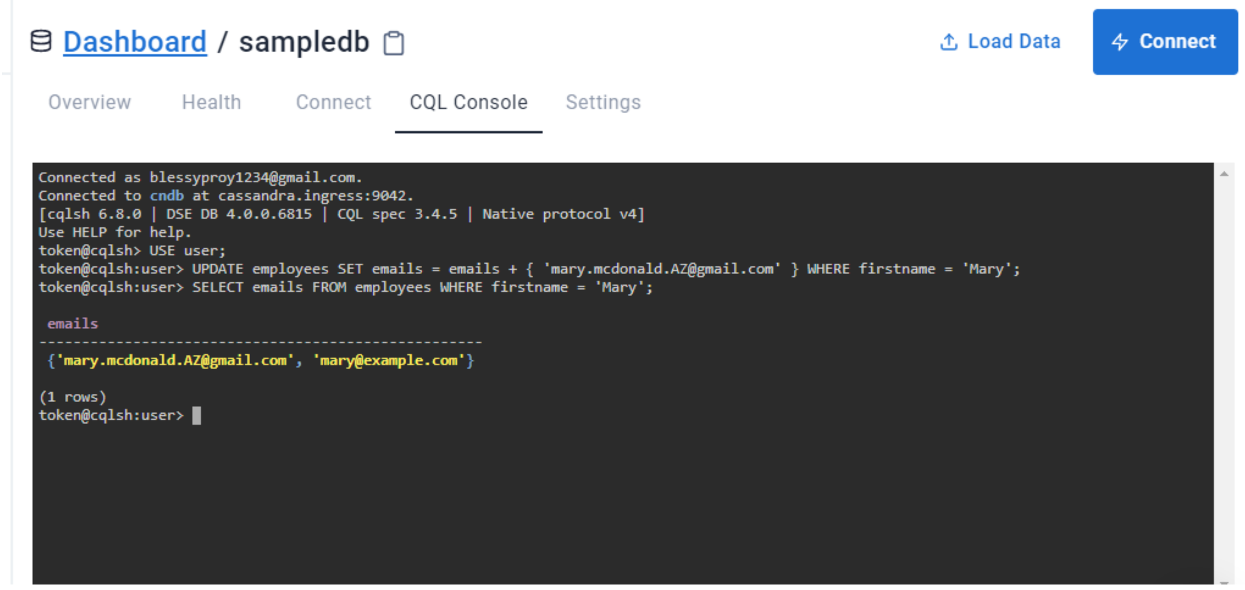
```
token@cqlsh:user> ALTER TABLE employees ADD emails set<text>;
token@cqlsh:user> UPDATE employees SET emails = { 'mary@example.com' } WHERE firstname = 'Mary';
token@cqlsh:user>  SELECT emails FROM employees WHERE firstname = 'Mary';

 emails
----------------------
 {'mary@example.com'}

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

9.QUERY TO ADD ANOTHER EMAIL ADDRESS WITHOUT REPLACING THE WHOLE SET BY USING CONCATENATION

Procedure:

- UPDATE employees SET emails = emails + { 'mary.mcdonald.AZ@gmail.com' } WHERE firstname = 'Mary';
- SELECT emails FROM employees WHERE firstname = 'Mary';

OUTPUT



```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user> UPDATE employees SET emails = emails + { 'mary.mcdonald.AZ@gmail.com' } WHERE firstname = 'Mary';
token@cqlsh:user> SELECT emails FROM employees WHERE firstname = 'Mary';

 emails
----------------------------------------------------
 {'mary.mcdonald.AZ@gmail.com', 'mary@example.com'}

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

<u>10.QUERY TO MODIFY OUR STAFFS TABLE TO ADD A LIST OF PHONE NUMBERS AND ADD A PHONE NUMBER FOR MARY AND CHECK THAT IT WAS ADDED SUCCESSFULLY</u>

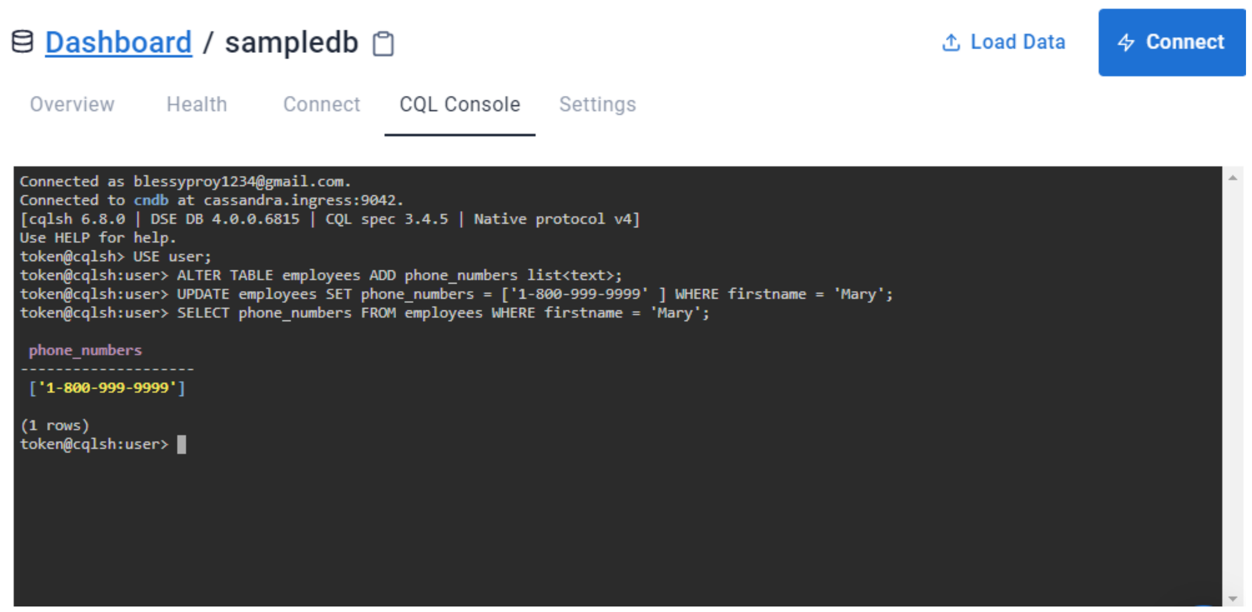
List

The list data type contains an ordered list of elements. By default, the values are stored in order of insertion.

<u>Procedure:</u>

- ALTER TABLE employees ADD phone_numbers list<text>;
- UPDATE employees SET phone_numbers = ['1-800-999-9999'] WHERE firstname = 'Mary';
- SELECT phone_numbers FROM employees WHERE firstname = 'Mary';

OUTPUT



```
Connected as blessyproy1234@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6815 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh> USE user;
token@cqlsh:user> ALTER TABLE employees ADD phone_numbers list<text>;
token@cqlsh:user> UPDATE employees SET phone_numbers = ['1-800-999-9999' ] WHERE firstname = 'Mary';
token@cqlsh:user> SELECT phone_numbers FROM employees WHERE firstname = 'Mary';

 phone_numbers
--------------------
 ['1-800-999-9999']

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