CS422 - Database Management System

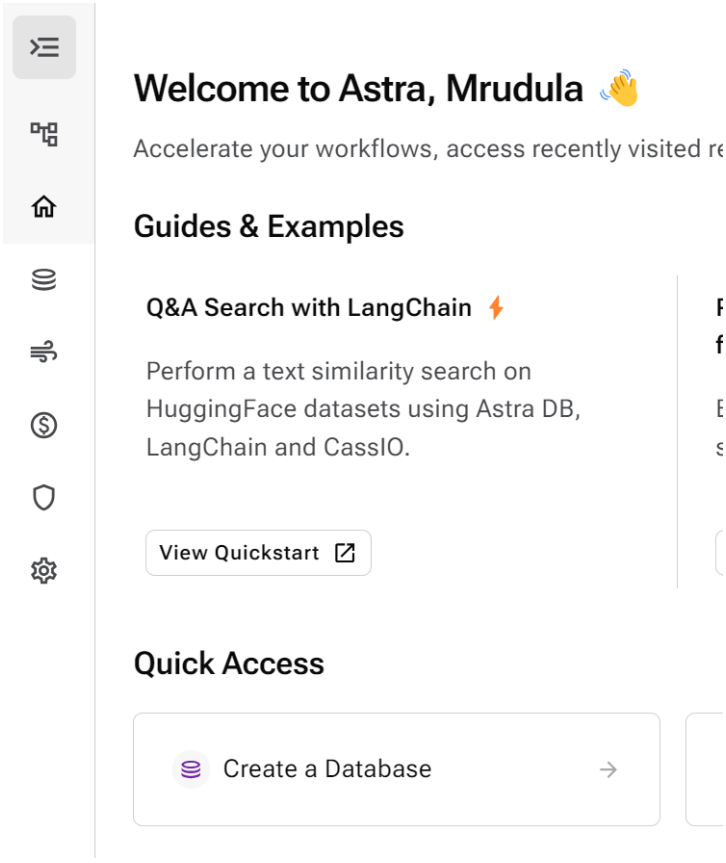
Lab 4 – NoSQL DB - Cassandra

- Submit your *own work* on time. No credit will be given if the lab is submitted after the due date.
- Note that the completed lab should be submitted in .docx, .pdf or .zip format only.

DataStax Astra DB is a cloud-native, scalable Database-as-a-Service built on Apache Cassandra. Apache Cassandra is the open-source NoSQL database behind the largest applications in the world, including Netflix and Instagram.

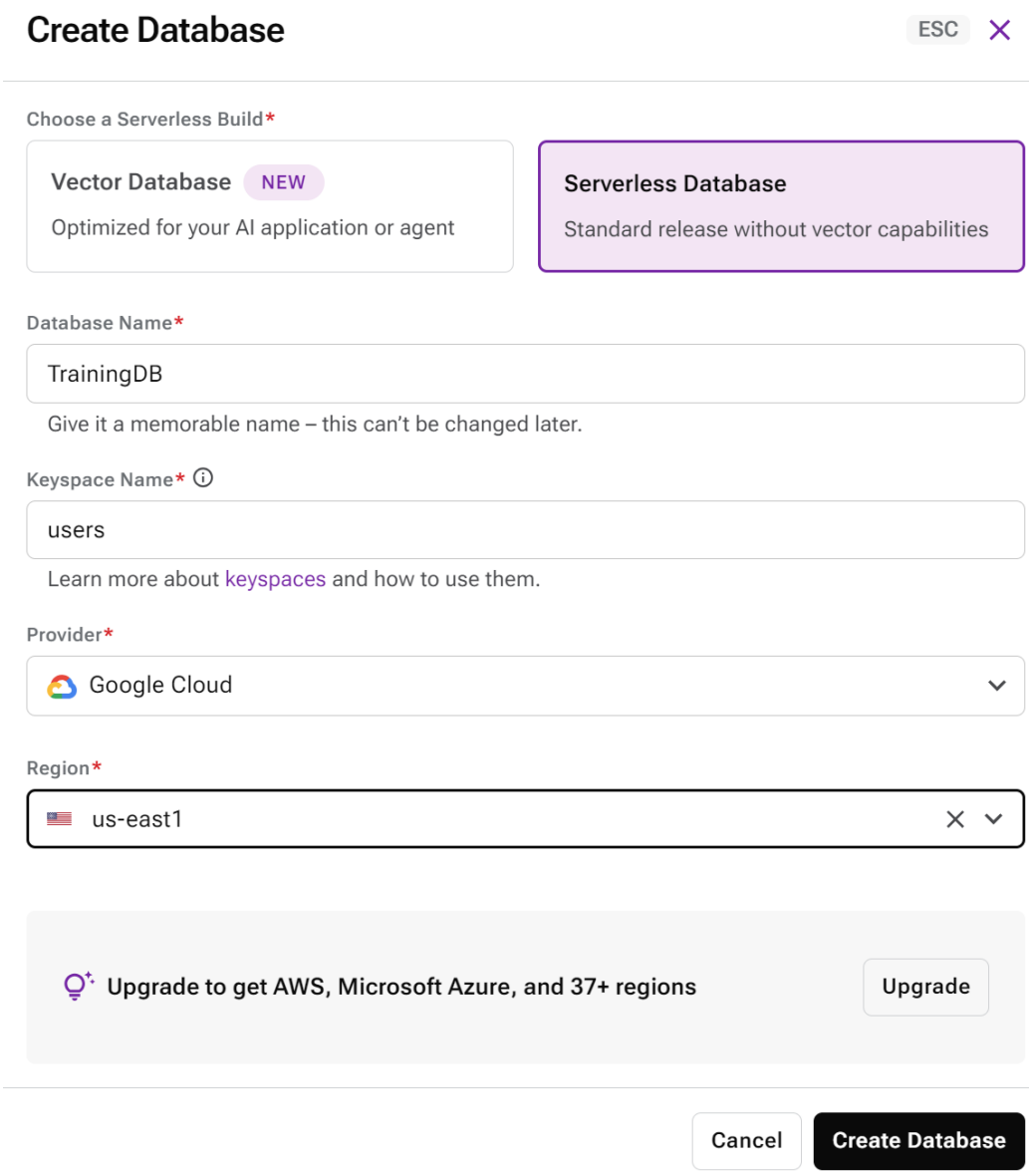
Part 1

Sign up for an account at astra.datastax.com

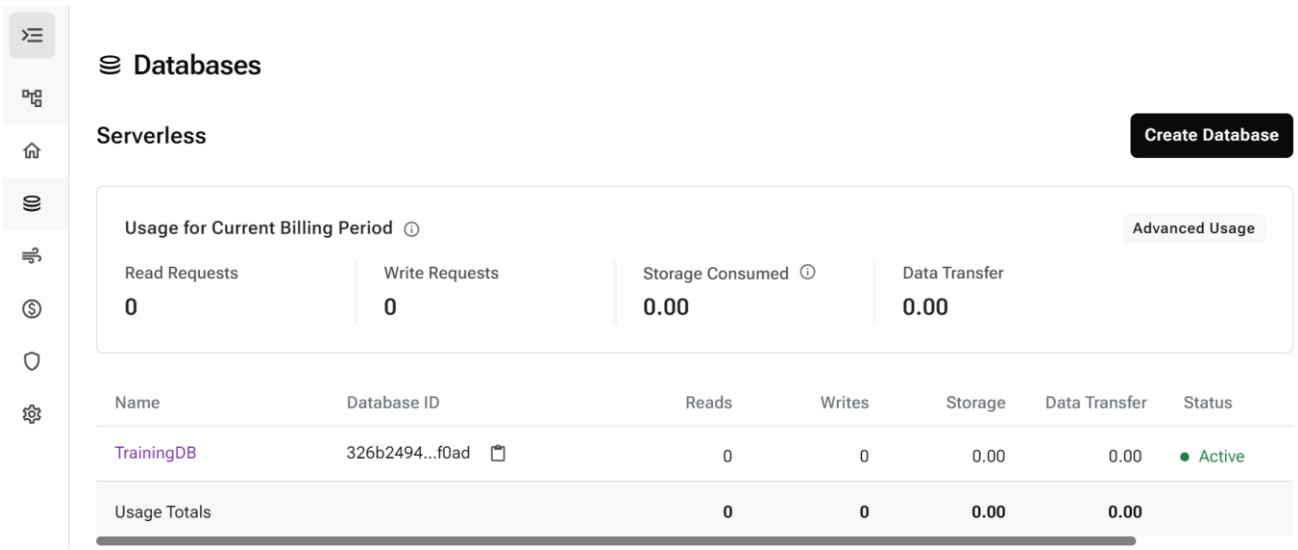


Once you logged in, create a database (Serverless Database).

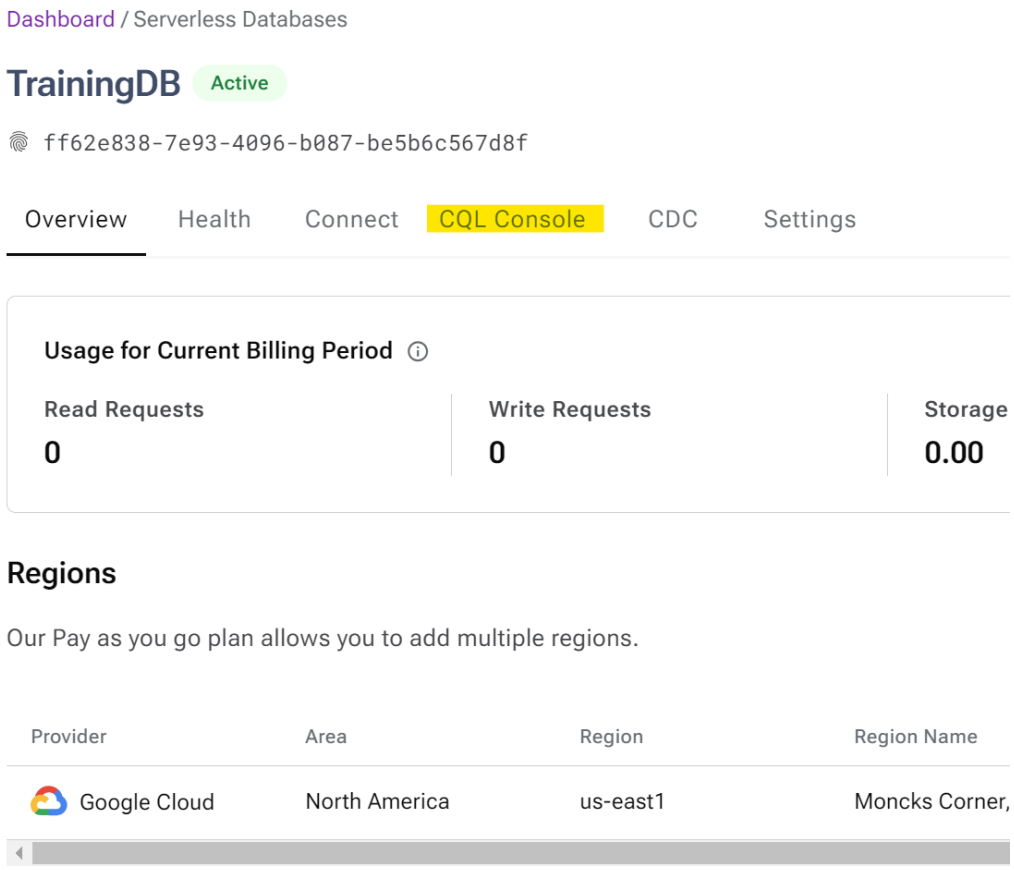
Enter a database name and a keyspace name and click Create Database



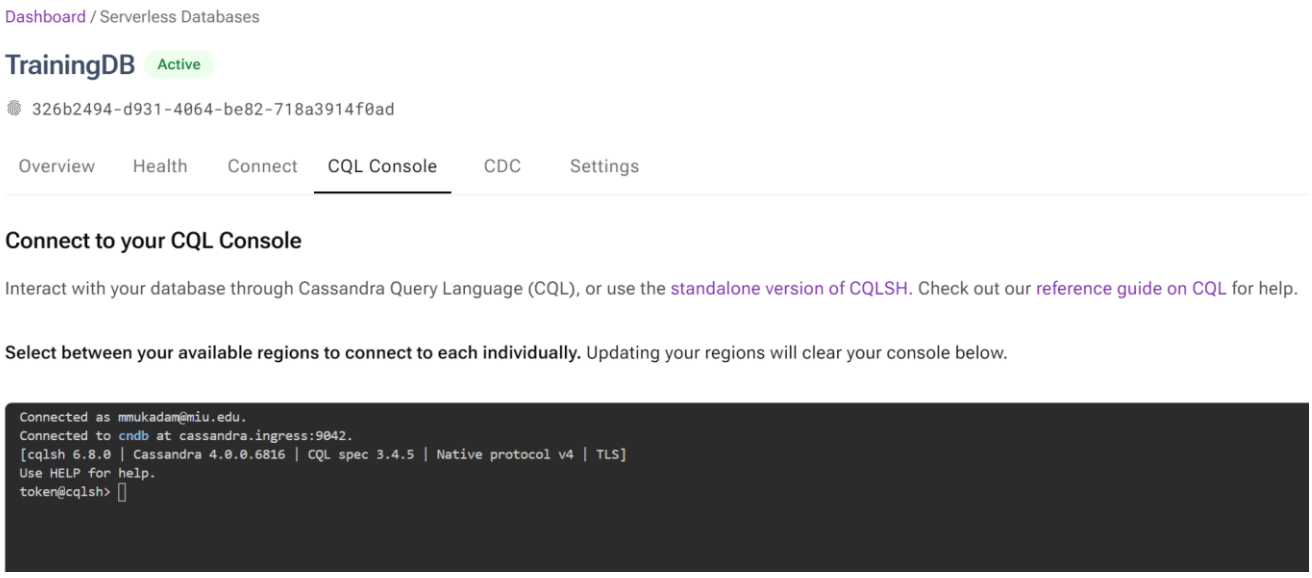
Wait till the database becomes active and then click on your database. It does take some time to initialize the database. Have patience! 😊



Select the CQL Console tab. Wait till the console starts.



CQL Console started:



Now you are in the CQL shell and you can enter CQL commands.

Part 2

token@cqlsh> **DESCRIBE keyspaces;** //lowercase is fine; can use "desc" as well

users system_schema system_traces
datastax_sla data_endpoint_auth system_virtual_schema
system_auth system system_views

token@cqlsh> **USE "users";** //okay to not use quotes

token@cqlsh> **DESCRIBE tables;** //this will list the available tables in this keyspace.

```
token@cqlsh:users> CREATE TABLE ratings_by_user (
      ...   email TEXT,
      ...   title TEXT,describ
      ...   year INT,
      ...   rating INT,
      ...   PRIMARY KEY ((email), title, year)
      ... );
```

token@cqlsh:users> **DESCRIBE ratings_by_user;**

```
CREATE TABLE users.ratings_by_user (
    email text,
    title text,
    year int,
    rating int,
    PRIMARY KEY (email, title, year)
) WITH CLUSTERING ORDER BY (title ASC, year ASC)
    AND 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';
```

Now insert the following data in the table:

```
INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('joe@datastax.com', 'Alice in Wonderland', 2010, 9);

INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('joe@datastax.com', 'Edward Scissorhands', 1990, 10);

INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('jen@datastax.com', 'Alice in Wonderland', 2010, 10);

INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('jen@datastax.com', 'Alice in Wonderland', 1951, 8);
```

Perform the following query:

SELECT * FROM ratings_by_user;

```
 email           | title                | year | rating
-----------------+----------------------+------+--------
 jen@datastax.com | Alice in Wonderland | 1951 |      8
 jen@datastax.com | Alice in Wonderland | 2010 |     10
 joe@datastax.com | Alice in Wonderland | 2010 |      9
 joe@datastax.com | Edward Scissorhands | 1990 |     10

(4 rows)
token@cqlsh:users>
```

Write the following queries:

1) [1] Retrieve all data from the user with email 'joe@datastax.com'.

2) [1] Retrieve all rating data from the user with email 'joe@datastax.com' for the movie 'Alice in Wonderland' from the year 2010

Also submit the screenshot of the CQL shell with the output.

Which columns are allowed in the WHERE clause?

- Try to get the data for all movies with a rating of 10:

  SELECT * FROM ratings_by_user
  WHERE rating= 10;

```
token@cqlsh:users> SELECT * FROM ratings_by_user
         ... WHERE rating= 10;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute
this query as it might involve data filtering and thus may have unpredictable perform
ance. If you want to execute this query despite the performance unpredictability, use
 ALLOW FILTERING"
```

  Notice that this query is not possible because rating is not part of the PK.

- Try to select all ratings for the movie 'Alice in Wonderland'.

  SELECT * FROM ratings_by_user
  WHERE title = 'Alice in Wonderland';

  Notice that this query is not directly possible.

```
token@cqlsh:users> SELECT * FROM ratings_by_user
         ... WHERE title = 'Alice in Wonderland';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute
this query as it might involve data filtering and thus may have unpredictable perform
ance. If you want to execute this query despite the performance unpredictability, use
 ALLOW FILTERING"
```

- But the following query is very well possible.

  SELECT * FROM ratings_by_user
  WHERE email = 'joe@datastax.com' and title = 'Alice in Wonderland';

```
token@cqlsh:users> SELECT * FROM ratings_by_user
        ... WHERE email = 'joe@datastax.com' and title = 'Alice in Wonderland';

 email             | title               | year | rating
-------------------+---------------------+------+--------
 joe@datastax.com | Alice in Wonderland | 2010 |      9
```

- But again, the following query is not possible.

  SELECT * FROM ratings_by_user
  WHERE email = 'joe@datastax.com' and year = 2010;

```
token@cqlsh:users> SELECT * FROM ratings_by_user
         ... WHERE email = 'joe@datastax.com' and year = 2010;
InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY KEY col
umn "year" cannot be restricted as preceding column "title" is not restricted"
```

Part 3

Create the following table:

```
CREATE TABLE ratings_by_movie (
  title TEXT,
  year INT,
  email TEXT,
  rating INT,
  PRIMARY KEY ((title, year), email)
);
```

Notice how the partition key is chosen.

Now populate the table with the following data:

```
INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Alice in Wonderland', 2010, 'jen@datastax.com', 10);

INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Alice in Wonderland', 2010, 'joe@datastax.com', 9);

INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Alice in Wonderland', 1951, 'jen@datastax.com', 8);

INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Edward Scissorhands', 1990, 'joe@datastax.com', 10);
```

Complete the following question:

3) [1] Write the query to select all ratings for 'Alice in Wonderland' from 2010. Submit the CQL code and the output from the CQL shell.

Part 4

Suppose we have the following relational DB tables:

Student

StudentNumber	name	email
S122	James Black	jblack@hotmail.com
S145	Frank Brown	fbrown@gmail.com
S265	James Bond	jb124@yahoo.com

Course

CourseNumber	Name	NrOfCredits
CS522	Databases	4
CS430	Programming 1	4
CS458	Programming 2	3

Student_Course

StudentNumber	CourseNumber
S122	CS522
S122	CS430
S145	CS430
S145	CS522
S145	CS458
S265	CS458

The database that we use cannot handle the amount of data anymore and we want to migrate to Cassandra.

The 2 main use cases for this application are:

a. **Get all courses for a specific student in ascending order of course number.**
b. **Get all students from a specific course.**

Write the following queries:

4) [5] Design the tables that you need in Cassandra.
Create these tables in Cassandra and populate the tables with the given data above.

5) [2] Write the 2 given queries and see if they work.
Submit the CQL code and the output from the CQL shell.