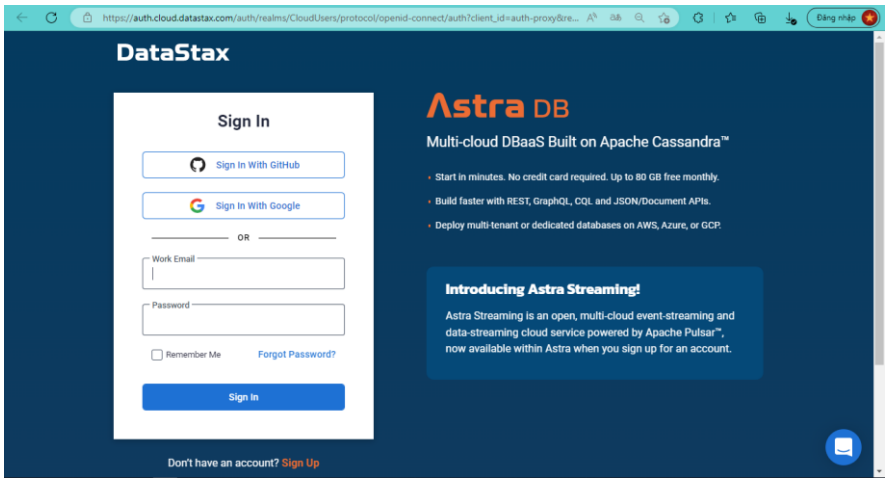
Instructions for Connecting ESP8266 with Datastax Astra DB

(In this tutorial use module ESP8266 NodeMCU)

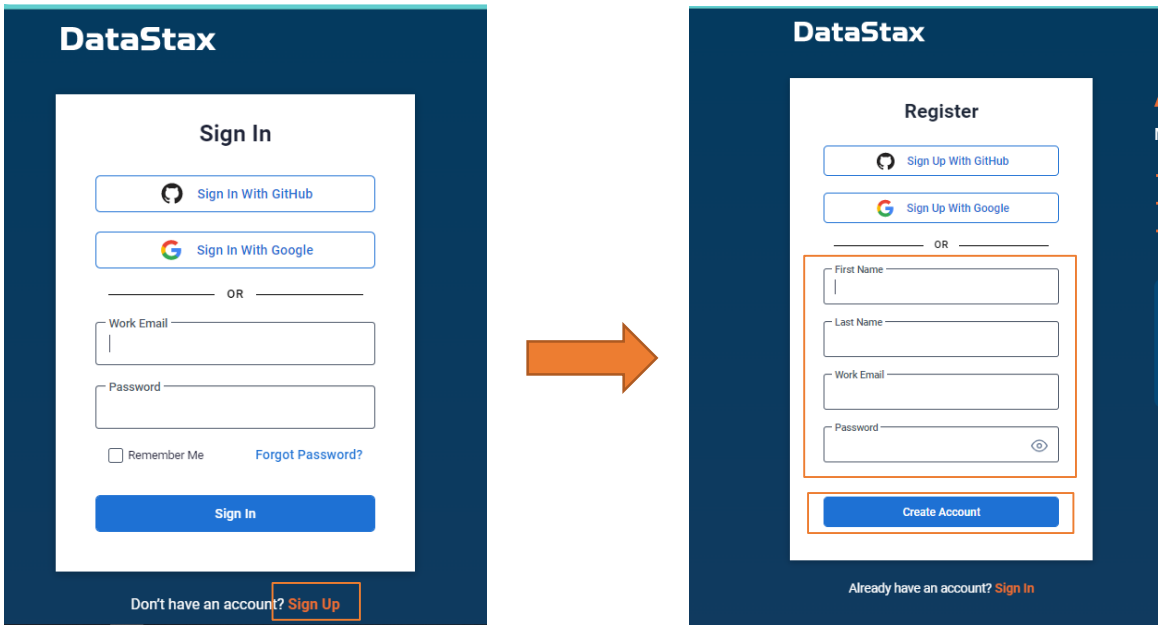
<u>Note:</u> This guide was created on November 11, 2022. After this time, the Datastax website may change its appearance and function, so this document is for reference only.

Step 1: Register or login account DataStax AstraDB.

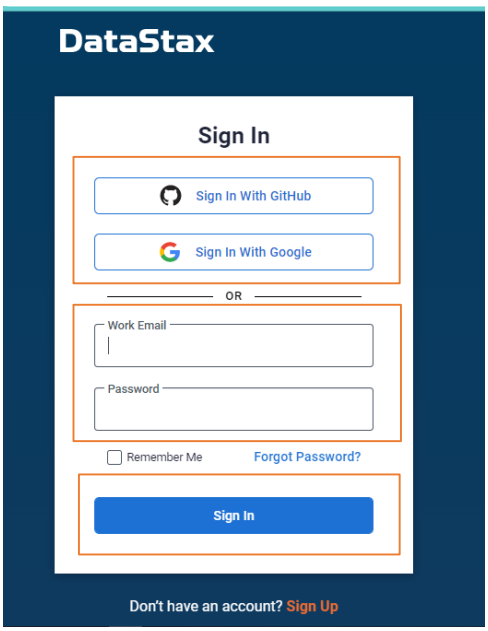
- Access the following path: https://auth.cloud.datastax.com/



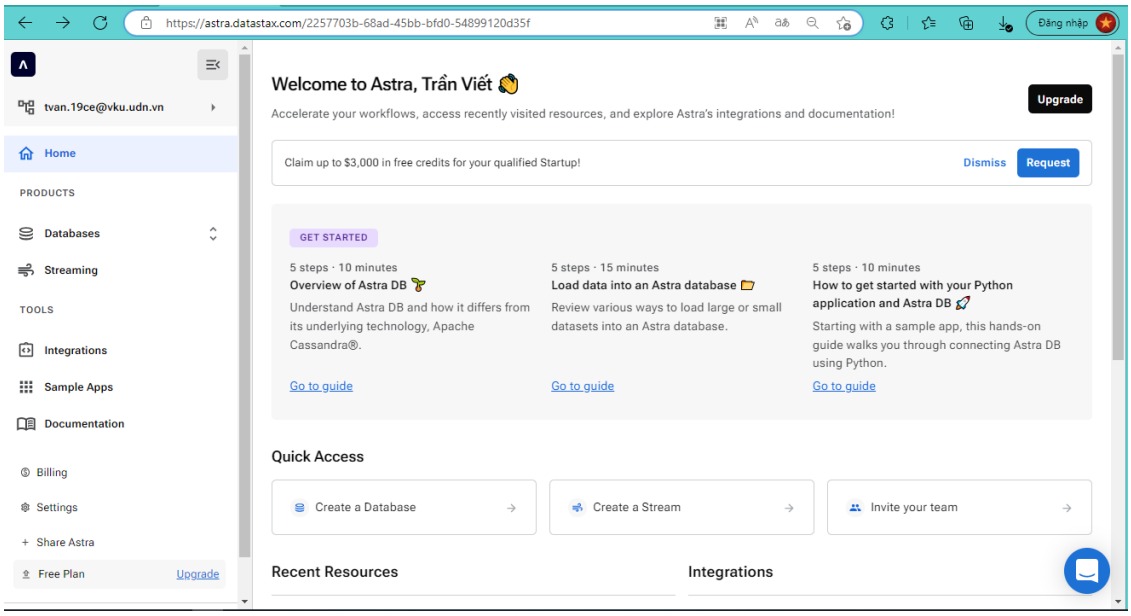
- If you do not have an account, click **Sign Up** and fill in the information to create an account:



- You can sign in with your **Google** or **Github** account or with the one you registered in the step above

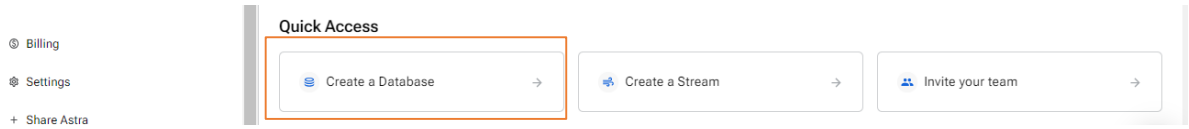


- After successful login, you will be redirected to your work page.

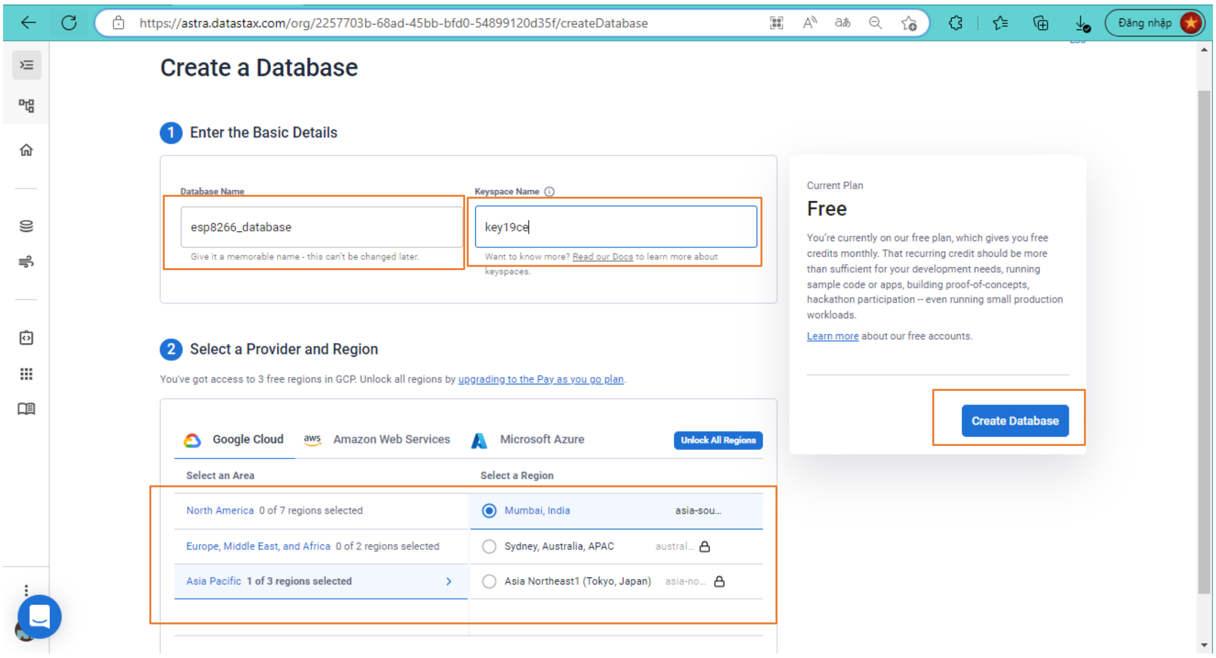


- Complete step 1.

Step 2: Create Database.



- At: **Quick Access** -> select **Create a Database**
- Then fill in the **Database name**, **Keyspace name** and **Select the provider and region**. (Here is using a trial account, so it will be limited to some provider and regions, if you want a better experience, please register for **Unlock all regions**). Then click **Create Database** and wait for a moment to initialize the database.



Create a Database

1 Enter the Basic Details

Database Name: esp8266_database

Give it a memorable name - this can't be changed later.

Keyspace Name: key19ce

Want to know more? Read our Docs to learn more about keyspaces.

2 Select a Provider and Region

You've got access to 3 free regions in GCP. Unlock all regions by upgrading to the Pay as you go plan.

Select an Area	Select a Region
North America 0 of 7 regions selected	● Mumbai, India asia-sou...
Europe, Middle East, and Africa 0 of 2 regions selected	○ Sydney, Australia, APAC austral...
Asia Pacific 1 of 3 regions selected	○ Asia Northeast1 (Tokyo, Japan) asia-no...

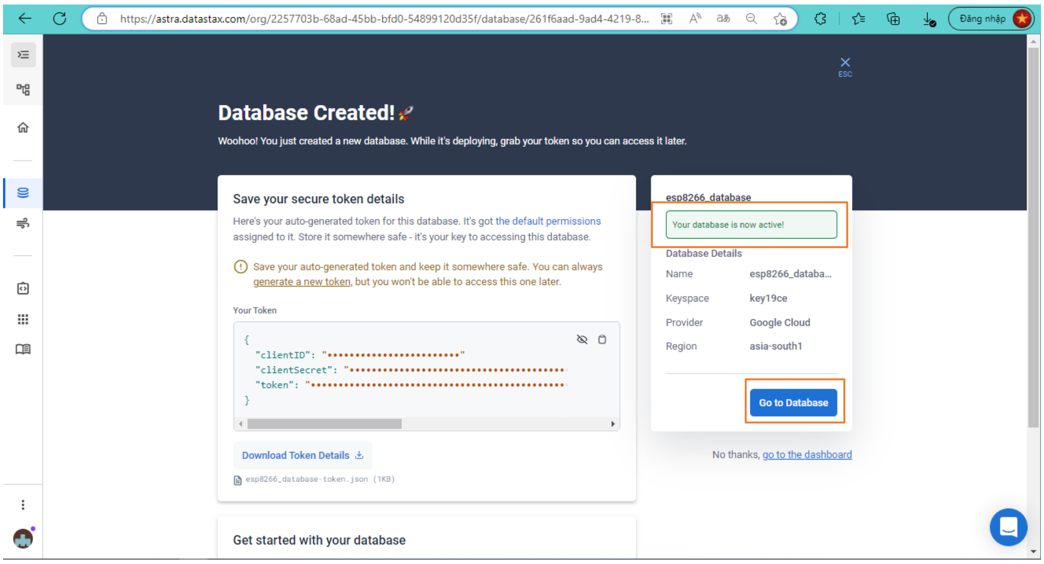
Current Plan

Free

You're currently on our free plan, which gives you free credits monthly. That recurring credit should be more than sufficient for your development needs, running sample code or apps, building proof-of-concepts, hackathon participation -- even running small production workloads.

Learn more about our free accounts.

- Next, click Go To Database to use the database on Datastax AstraDB.



Database Created!

Woohoo! You just created a new database. While it's deploying, grab your token so you can access it later.

Save your secure token details

Here's your auto-generated token for this database. It's got the default permissions assigned to it. Store it somewhere safe - it's your key to accessing this database.

Save your auto-generated token and keep it somewhere safe. You can always generate a new token, but you won't be able to access this one later.

Your Token

```
{
  "clientID": "••••••••••••••••••••••••",
  "clientSecret": "••••••••••••••••••••••••••••••••••••••••
  "token": "••••••••••••••••••••••••••••••••••••••••••••
}
```

Download Token Details

esp8266_database-token.json (1KB)

esp8266_database

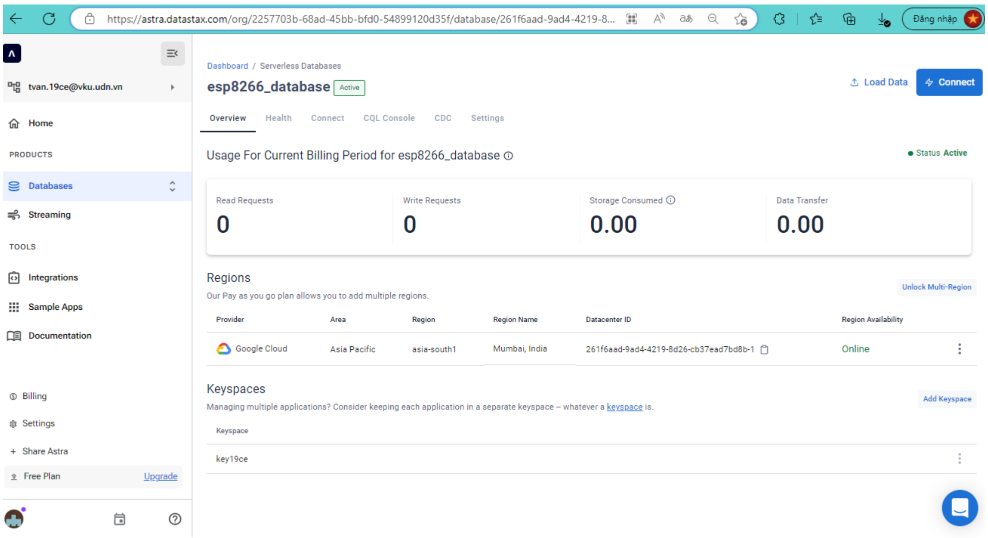
Your database is now active!

Database Details

Name	esp8266_databa...
Keyspace	key19ce
Provider	Google Cloud
Region	asia-south1

No thanks, go to the dashboard

Get started with your database



esp8266_database Active

Usage For Current Billing Period for esp8266_database

Status Active

Read Requests	Write Requests	Storage Consumed	Data Transfer
0	0	0.00	0.00

Regions

Our Pay as you go plan allows you to add multiple regions.

Provider	Area	Region	Region Name	Datacenter ID	Region Availability
Google Cloud	Asia Pacific	asia-south1	Mumbai, India	261f6aad-9ad4-4219-8d26-cb37ead7bd8b-1	Online

Keyspaces

Managing multiple applications? Consider keeping each application in a separate keyspace – whatever a keyspace is.

Keyspace
key19ce

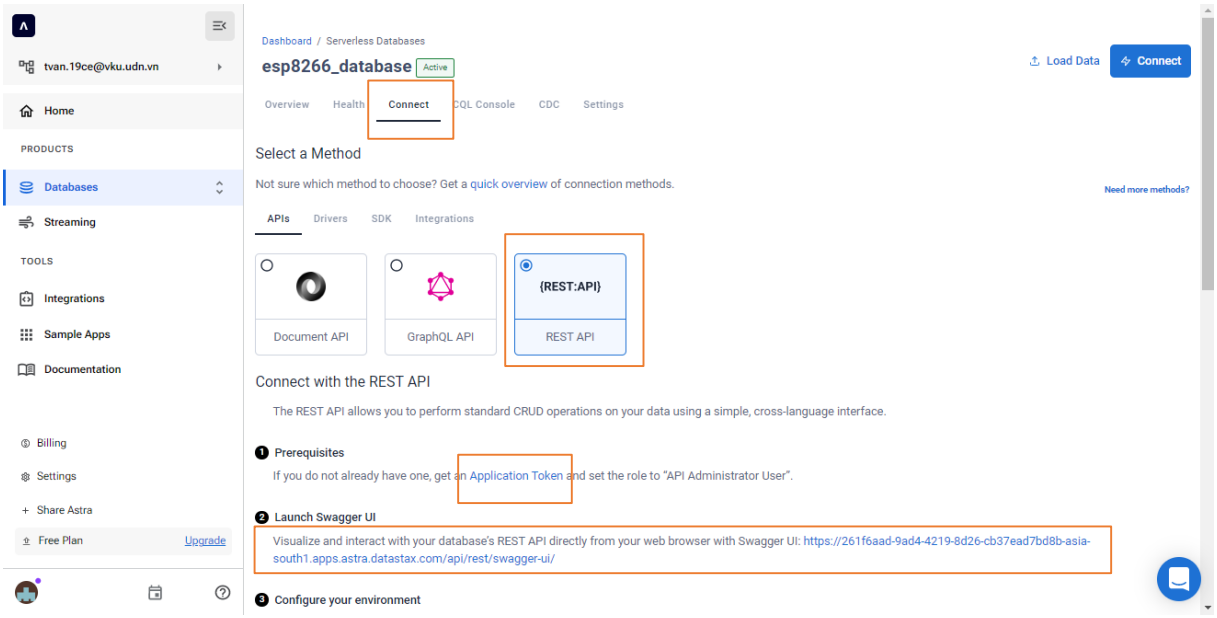
- Complete step 2

Step 3: Create database tables and data fields

- Below is the sample database used for this tutorial, you can change the database table for your own.

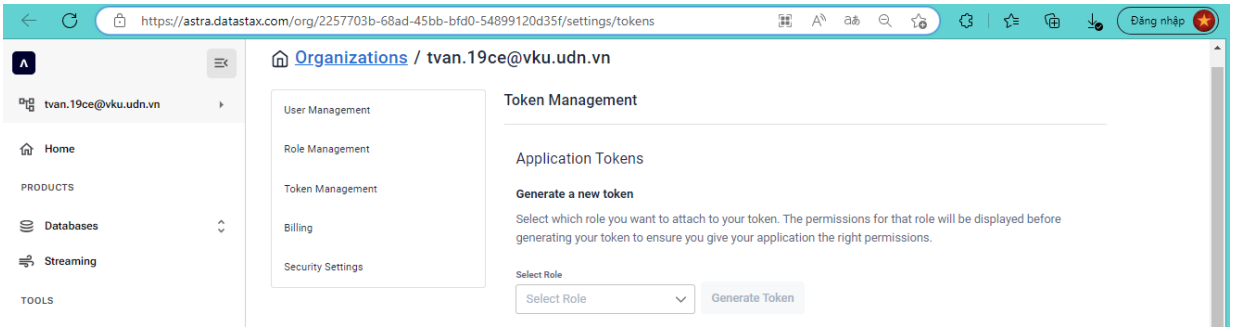
Table name: collection					
data field	id	position	temprature	humility	amount_of_rain
datatypes	text	text	text	text	text
key	primaryKey				

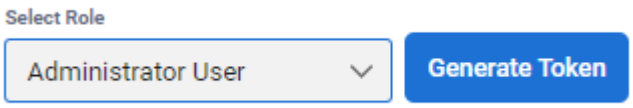
- In this tutorial, to connect and send data from ESP8266 to DataStax AstraDB will be through REST API, go to **Connect** section and select **{REST:API}**.

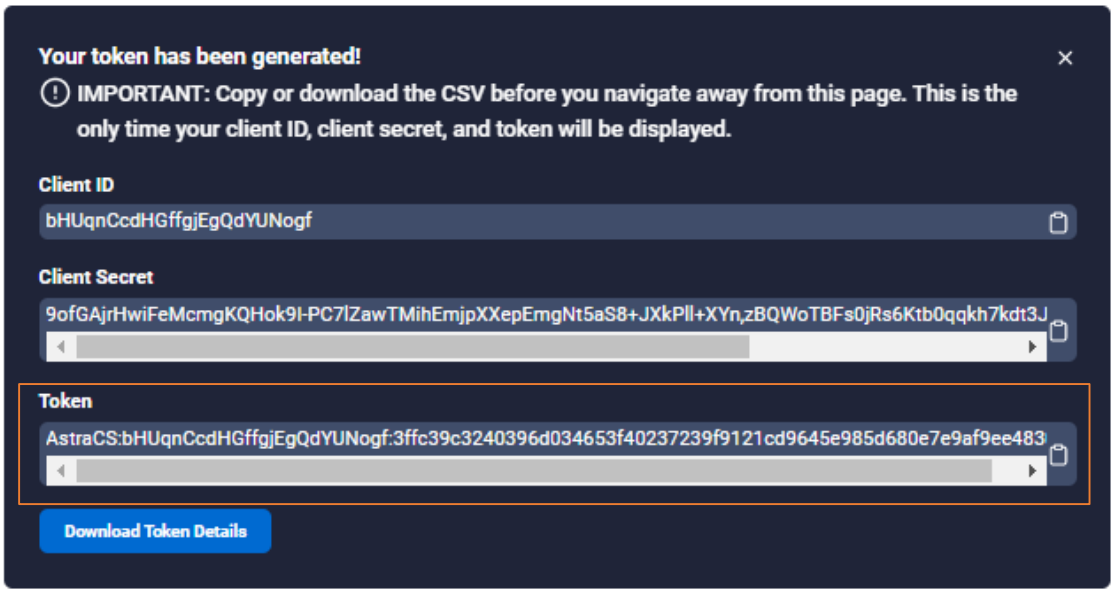


- Next we will generate a token to serve in the connection between ESP8266 and AstraDB by clicking **Application Token**, and create database table in Swagger UI (click link in section **2.Lauchn Swagger UI**)

  + **Create Token:** After clicking on **Application Token** will go to token management page. In select role select **Administrator User** and then select **Generate Token**.

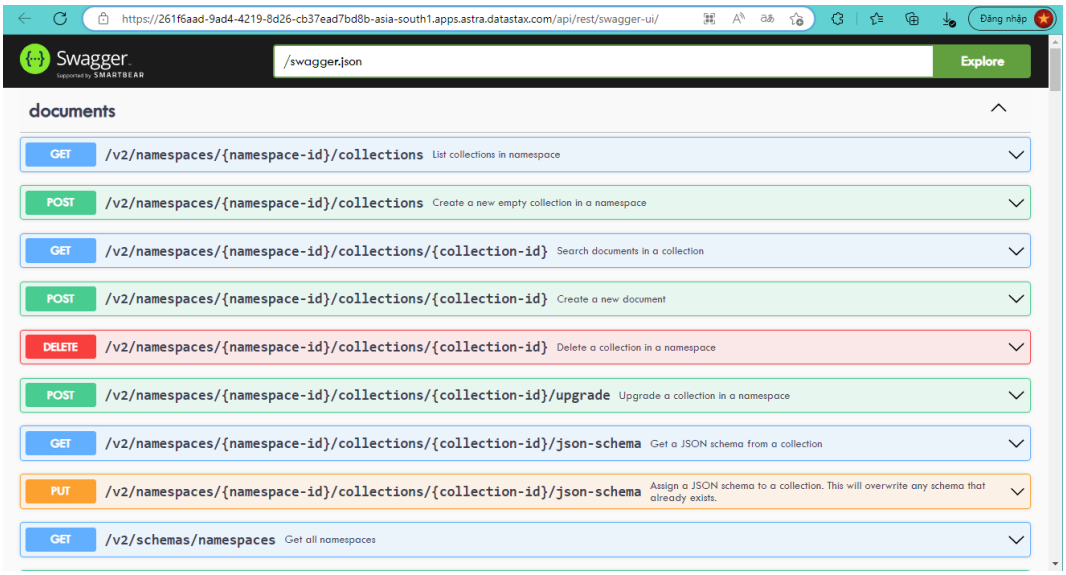






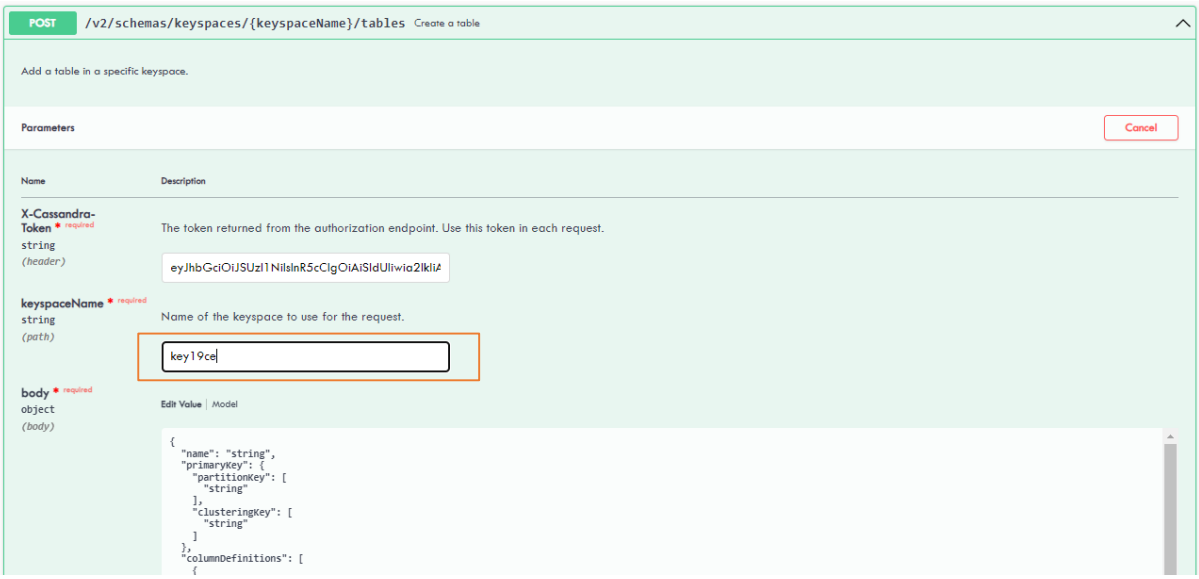
<u>Note:</u> After the token is generated, copy or save it. (1)

+ **Create database table in Swagger UI:** After clicking on the link in the **Lauchn Swagger UI**, you will be redirected to the working interface of Swagger UI.



+ In the Schemas section select the line: POST /v2/schemas/keyspace/…..

+ Click select **Try it out.**
+ Enter the keyspace used by the database in the **keyspaceName** field



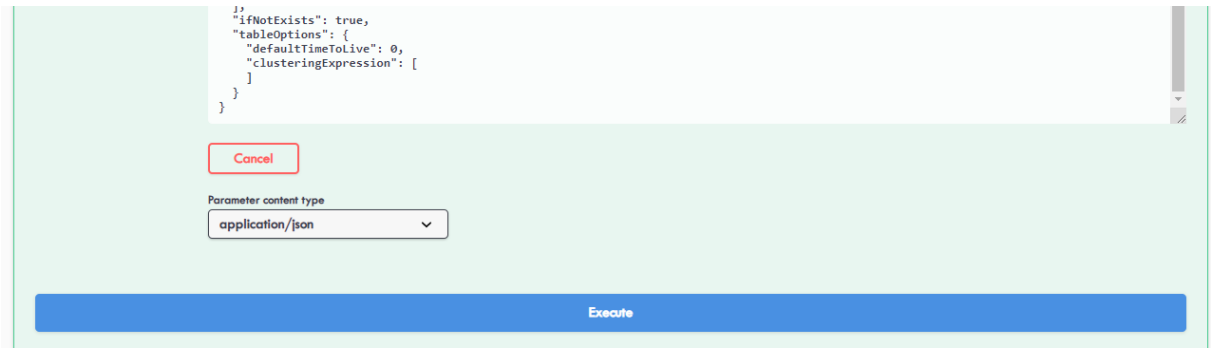
+ In the body part, we will create the database table in the form of json string. create a database like the following table:

Table name: collection					
data field	id	position	temprature	humility	amount_of_rain
datatypes	text	text	text	text	text
key	primaryKey				

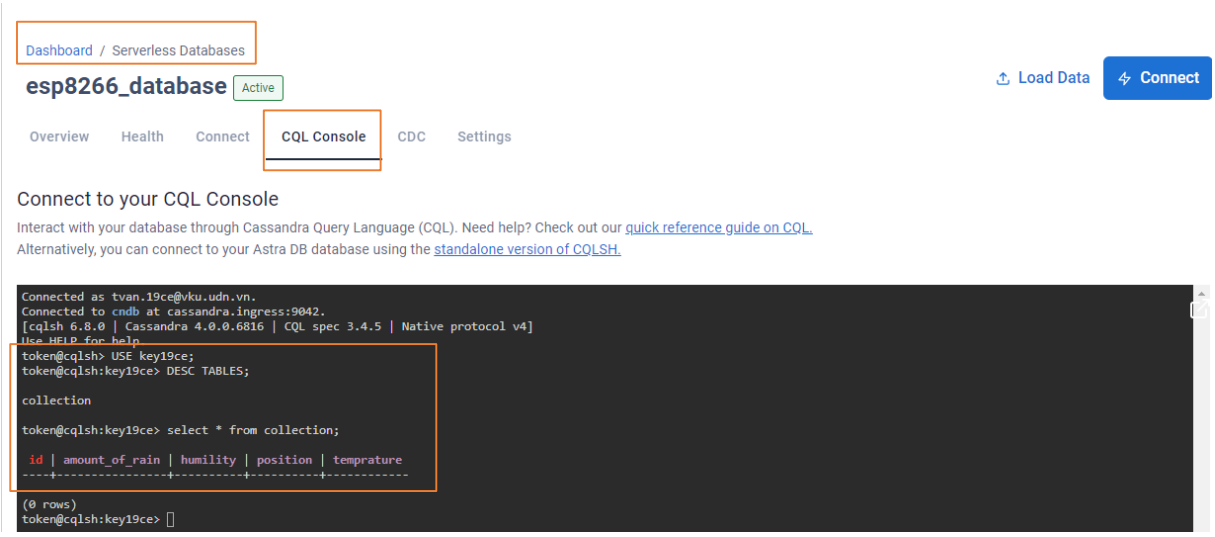
```
{
 "name": "collection",
 "primaryKey": {
  "partitionKey": [
   "id"
  ],
  "clusteringKey": [
  ]
 },
 "columnDefinitions": [
  {
```

```
      "name": "id",
      "typeDefinition": "text",
      "static": false
    },
    {
      "name": "position",
      "typeDefinition": "text",
      "static": false
    },
    {
      "name": "temprature",
      "typeDefinition": "text",
      "static": false
    },
    {
      "name": "humility",
      "typeDefinition": "text",
      "static": false
    },
    {
      "name": "amount_of_rain",
      "typeDefinition": "text",
      "static": false
    }
  ],
  "ifNotExists": true,
  "tableOptions": {
    "defaultTimeToLive": 0,
    "clusteringExpression": [
    ]
  }
}
```


+ You can change the above json string to suit the database you want to create such as table name, primary key, data field names, input data type.
+ Click Execute to execute the creation.



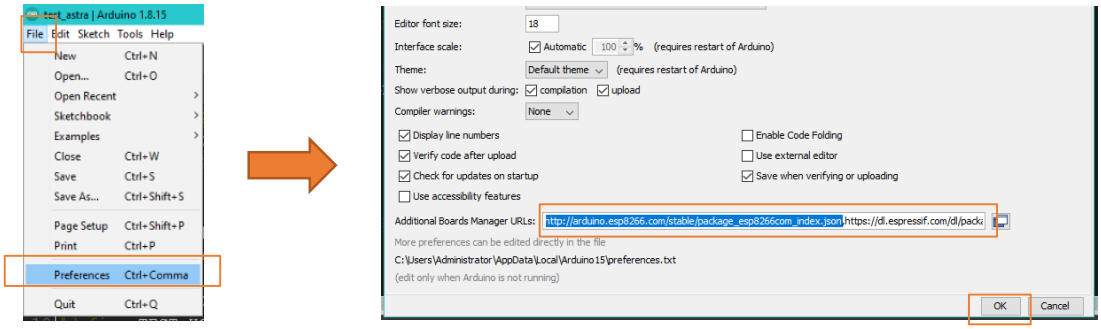
+ To check if the database table has been created successfully, do the following steps: Back to Dashboard -> CQL Console -> run command **USE key19ce;** (key19ce is keyspacename of database); run command **DESC TABLES;** (used to display all created database tables). You see that the collection table has been created successfully. Run command **select * from collection;** to view the database fields of the collection table.



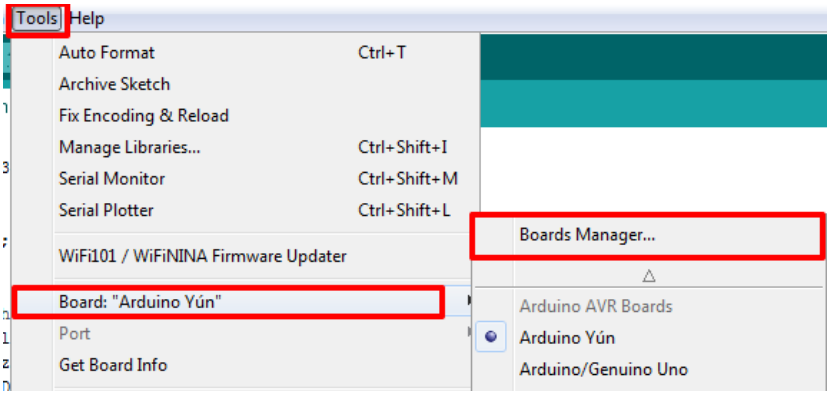
Step 4: Code ESP8266 connect WiFi and DataStax Astra DB

- Link code: https://github.com/TranVietAn1606/AstraDBConnectESP8266
- Arduino IDE working mode setting with ESP8266:

+ Start the Arduino IDE, click **File** on the toolbar and select **Preferences (Ctrl+Comma)**. Next, insert a link so that the Arduino IDE can receive the ESP8266 Board and click **OK.**
+ Link: http://arduino.esp8266.com/stable/package_esp8266com_index.json

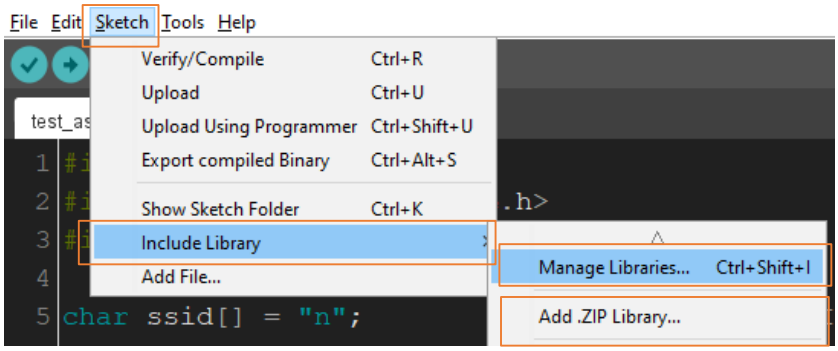


+ Next, go to **Tools > Board > Boards Manager.**



+ The window that opens is search **Esp8266** to download the list of Boards. Click **Install** to proceed with the installation.

+ Next install the following libraries if not available: ESP8266WiFi, WiFiClientSecure, ESP8266HTTPClient.



- Code explanation:

+ Declare the library to use.

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ESP8266HTTPClient.h>
```

+ Enter the WiFi name and password that you use.

```
char ssid[] = "NotNew";          // your network SSID (name)
char password[] = "abc123123"; // your network key
```

+ Initializes available data arrays for **temperature, humidity, positions, amount_of_rain**

```
const char *temprature[25] = {"27.05", "27.90", "28.00", "28.24", "28.47", "28.85", "29.14", "29.33", "29.78", "30.10", "30.23", "30.44", "30.67", "31.00", "31.54", "32.78", "33.06", "33.
const char *amount_of_rain[25] = {"16", "20", "29", "40", "47", "53", "59", "62", "67", "70", "73", "77", "81", "86", "92", "95", "98", "100", "108", "111", "115", "120", "131", "149", "1
const char *humility[25] = {"49.00", "50.59", "51.65", "", "52.34", "55.04", "58.43", "59.76", "60.28", "63.01", "63.59", "64.66", "65.21", "68.23", "68.81", "71.00", "73.37", "77.77", "8
const char *positions[25] = {"16,17", "16,25", "18,87", "20,130", "21,88", "22,111", "23,97", "24,109", "25,54", "25,124", "26,01,", "26,97", "32,10", "32,32", "33,39", "33,44", "33,99",
```

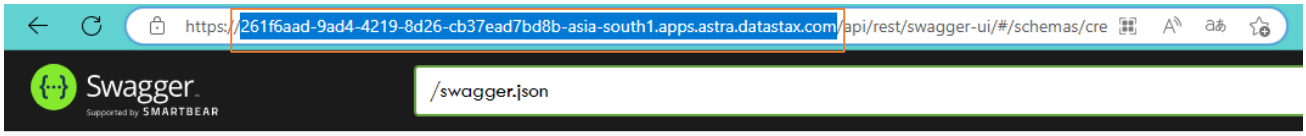
+ Declare variable **client** using the library WiFiClientSecure.

```
WiFiClientSecure client;
```

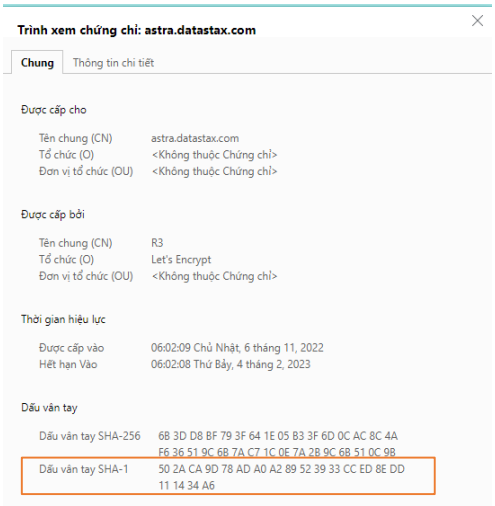
+ Declare TEST_HOST and TEST_HOST_FINGERPRINT to connect to DataStax AstraDB.

```
#define TEST_HOST "261f6aad-9ad4-4219-8d26-cb37ead7bd8b-asia-south1.apps.astra.datastax.com"
#define TEST_HOST_FINGERPRINT "50 2A CA 9D 78 AD A0 A2 89 52 39 33 CC ED 8E DD 11 14 34 A6"
```

- TEST_HOST:



- TEST_HOST_FINGERPRINT:



- Funcition setup():

+ Establish serial connection.
+ Establish WiFi connection
+ Check the fingerprint.

```
void setup() {

  Serial.begin(115200);

  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  IPAddress ip = WiFi.localIP();
  Serial.println(ip);

  client.setFingerprint(TEST_HOST_FINGERPRINT);
}
```

- Funcition makeHTTPRequest(String id, String temp, String hum, String amount, String pos):

  + Input values are id, temp, hum, amount, pos (This is data added to the database table, you can change it accordingly).
  + Check connection to DataStax Astra with port 443 (HTTPS)
  + Establish connection with created data table with **serverName** and **Token**

    - serverName: https://**261f6aad-9ad4-4219-8d26-cb37ead7bd8b-asia-south1.apps.astra.datastax.com**/api/rest/v2/keyspaces/**key19ce/collection**
    - Token: the code generated (1) in **step 3** (you can recreate it) AstraCS:bHUqnCcdHGffgjEgQdYUNogf:3ffc39c3240396d034653f40237239f9121cd9645e985d680e7e9af9ee4830ef

  + Define the content you want to send to Astra DB
  + Send data to AstraDB.
  + Check if the data has been successfully sent, if it is successful, it will retrieve the data and display the Serial, if not, it will report an error.

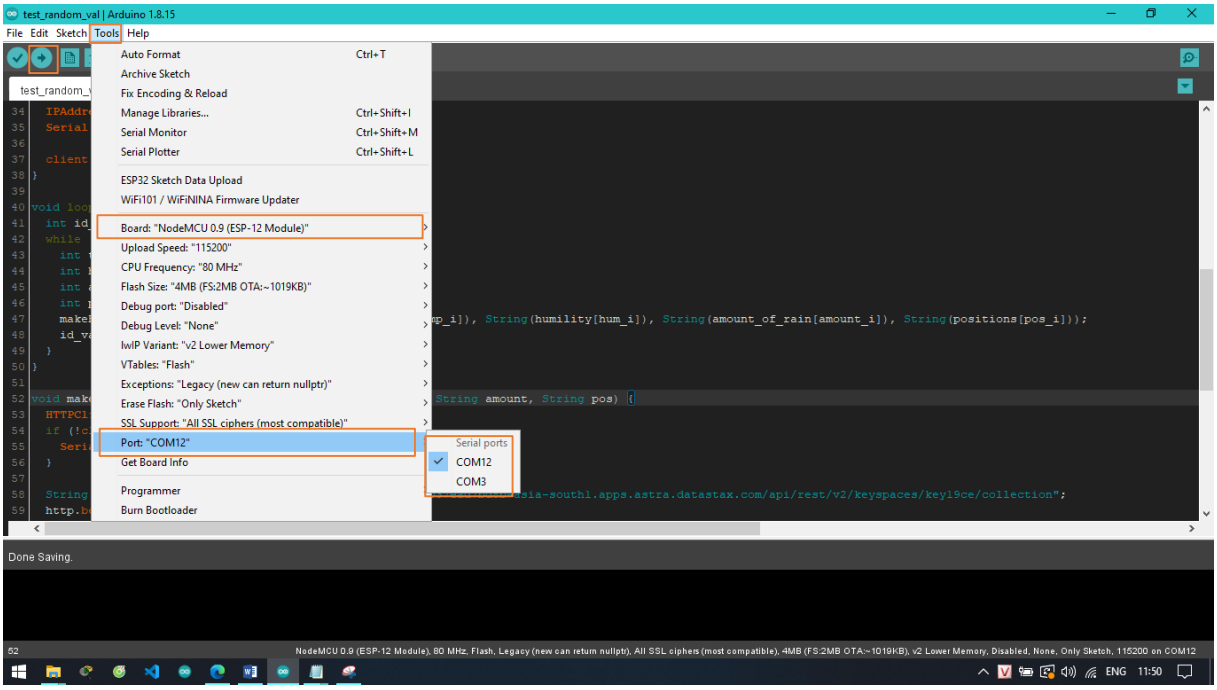
```
void makeHTTPRequest(String id, String temp, String hum, String amount, String pos) {
  HTTPClient http;
  if (!client.connect(TEST_HOST, 443)) {
    Serial.println(F("Connection failed"));
  }

  String serverName = "https://261f6aad-9ad4-4219-8d26-cb37ead7bd8b-asia-south1.apps.astra.datastax.com/api/rest/v2/keyspaces/key19ce/collection";
  http.begin(client, serverName.c_str());
  http.addHeader("Content-Type", "application/json");
  http.addHeader("X-Cassandra-Token", "AstraCS:bHUqnCcdHGffgjEgQdYUNogf:3ffc39c3240396d034653f40237239f9121cd9645e985d680e7e9af9ee4830ef");

  String bodystr = "{\"id\": \"";
  bodystr.concat(id);
  bodystr.concat("\",\"position\":\"");
  bodystr.concat(pos);
  bodystr.concat("\",\"temprature\":\"");
  bodystr.concat(temp);
  bodystr.concat("\",\"humility\":\"");
  bodystr.concat(hum);
  bodystr.concat("\",\"amount_of_rain\":\"");
  bodystr.concat(amount);
  bodystr.concat("\"}");
  int httpResponseCode = http.POST(bodystr.c_str());
  if (httpResponseCode > 0) {
    String payload = http.getString();
    Serial.println("payload: " + payload);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
  delay(5000);
}
```

- Funcition loop():

  + **id** will be 0 and will increase by 1 after each while loops
  + The values of **temperature**, **humidity**, **position**, **amount_of_rain** will be randomly taken in the respective data arrays.
  + Implement funcition makeHTTPRequest(…) to upload data to DataStax AstraDB

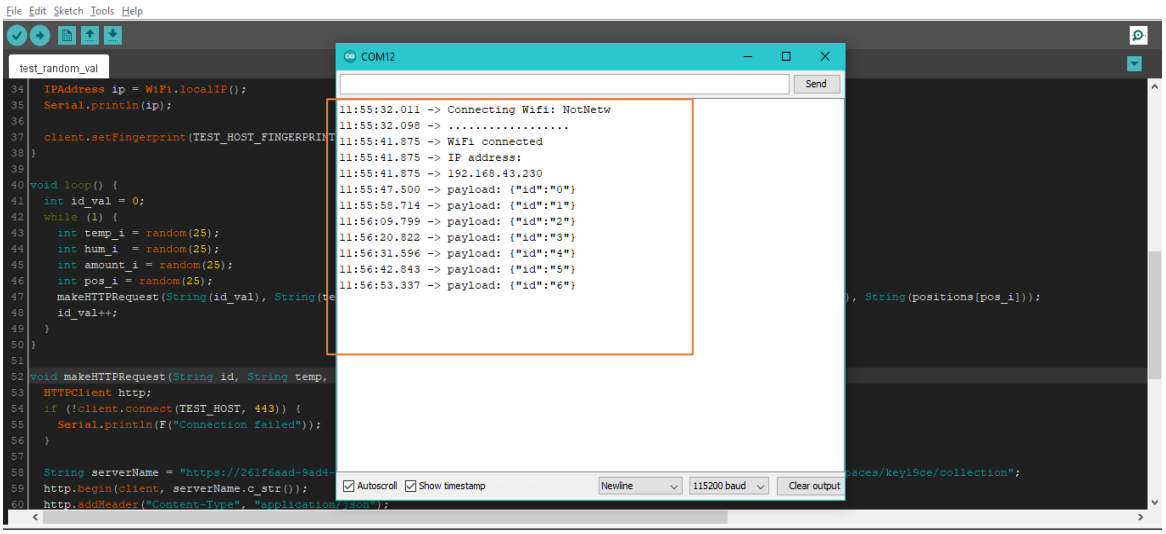
```
void loop() {
  int id_val = 0;
  while (1) {
    int temp_i = random(25);
    int hum_i  = random(25);
    int amount_i = random(25);
    int pos_i = random(25);
    makeHTTPRequest(String(id_val), String(temprature[temp_i]), String(humility[hum_i]), String(amount_of_rain[amount_i]), String(positions[pos_i]));
    id_val++;
  }
}
```

- Load code for module ESP8266: select the corresponding **board** and **port**

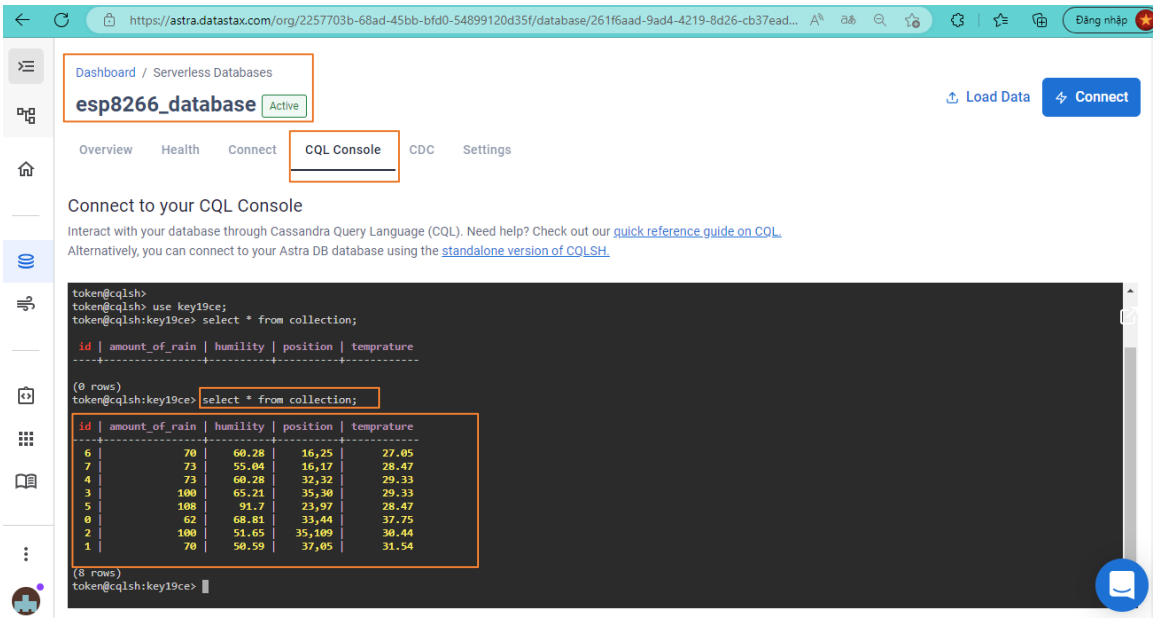


- Result:

  + Connect to WiFi and upload data to Datastax Astra successfully.



  + Go to **CQL Console** and run **select * from collection;** to see the database table.



- Through this database and the support of DataStax on many platforms, you can use it for many different purposes such as building monitoring systems, data collection, data processing.

This part of the tutorial is temporarily finished, wish you success.