ADT Lab 12 (10pts)

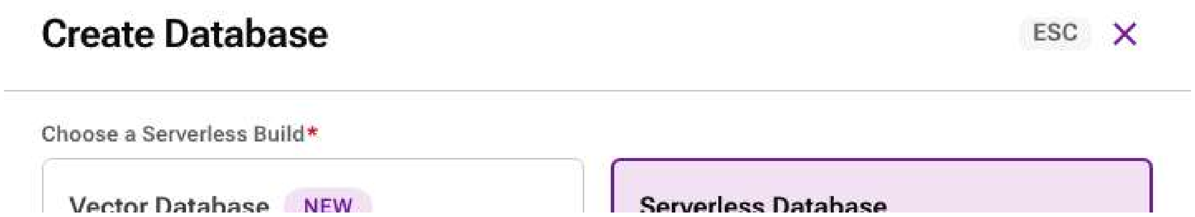
Cassandra

Submitted by : Atharv J

Sem - Fall 23

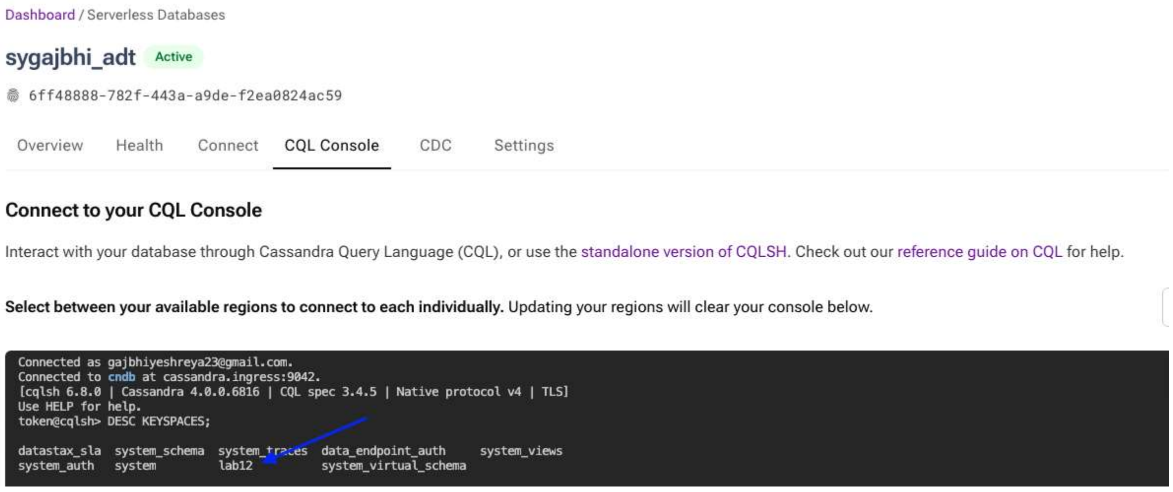
Setup Instructions-

- Go to the Astra DB website (https://astra.datastax.com/ (https://astra.datastax.com/)) and sign up for a free account.
- Click on database and hit Create Database
- Give database name and Keyspace Name



- Go to the CQL Console and see if keyspace was created properly by running the command

```
DESC KEYSPACES;
```


Dashboard / Serverless Databases
sygajbhi_adt Active
6ff48888-782f-443a-a9de-f2ea0824ac59
Overview
Health
Connect
CQL Console
CDC
Settings

Connect to your CQL Console

Interact with your database through Cassandra Query Language (CQL), or use the standalone version of CQLSH. Check out our reference guide on CQL for help.

Select between your available regions to connect to each individually. Updating your regions will clear your console below.

```
Connected as gajbhiyeshreya23@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS]
Use HELP for help.
token@cqlsh> DESC KEYSPACES;

datastax_sla  system_schema  system_traces  data_endpoint_auth     system_views
system_auth   system         lab12          system_virtual_schema
```

You should see the keyspace that you created earlier. Keyspace is similar to a schema in relational dbms

Astra DB

Astra DB is a cloud-native database-as-a-service (DBaaS) offering provided by DataStax, which is designed to provide a fully managed and scalable NoSQL database service for modern applications. Astra DB is built on top of the Apache Cassandra database, but it includes additional features and capabilities that make it easier to use and more powerful.

- Now let's use the keyspace to execute further queries, similar to what we do in RDBMS. First we select the schema to be used

```
USE lab11;
```

- Now lets create a people table using this query-

CREATE TABLE lab11.people (

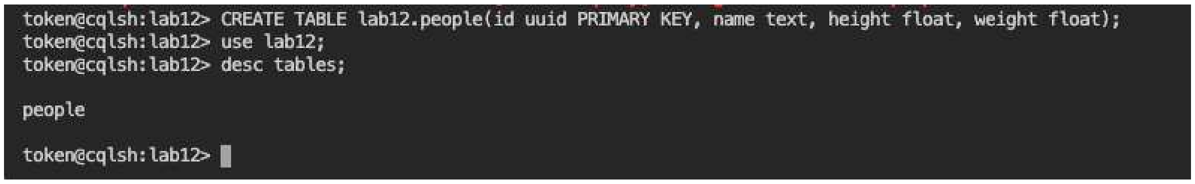
```
id uuid PRIMARY KEY,
name text,
height float,
weight float
```

);

- Verify if the table is created by running

```
DESC TABLES;
```

Your output should be like-



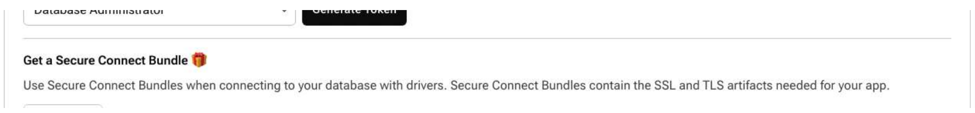
```
token@cqlsh:lab12> CREATE TABLE lab12.people(id uuid PRIMARY KEY, name text, height float, weight float);
token@cqlsh:lab12> use lab12;
token@cqlsh:lab12> desc tables;

people

token@cqlsh:lab12>
```

In [1]:
```
!pip install cassandra-driver
```

```
Requirement already satisfied: cassandra-driver in c:\users\athar\anaconda3\l
ib\site-packages (3.26.0)
Requirement already satisfied: six>=1.9 in c:\users\athar\appdata\roaming\pyt
hon\python39\site-packages (from cassandra-driver) (1.16.0)
Requirement already satisfied: geomet<0.3,>=0.1 in c:\users\athar\anaconda3\l
ib\site-packages (from cassandra-driver) (0.2.1.post1)
Requirement already satisfied: click in c:\users\athar\anaconda3\lib\site-pac
kages (from geomet<0.3,>=0.1->cassandra-driver) (8.0.4)
Requirement already satisfied: colorama in c:\users\athar\anaconda3\lib\site-
packages (from click->geomet<0.3,>=0.1->cassandra-driver) (0.4.4)
```

In [2]:
```
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
from cassandra.query import BatchStatement
from uuid import uuid4
import csv
import os
```

- Go to connect tab and download the bundle as shown below

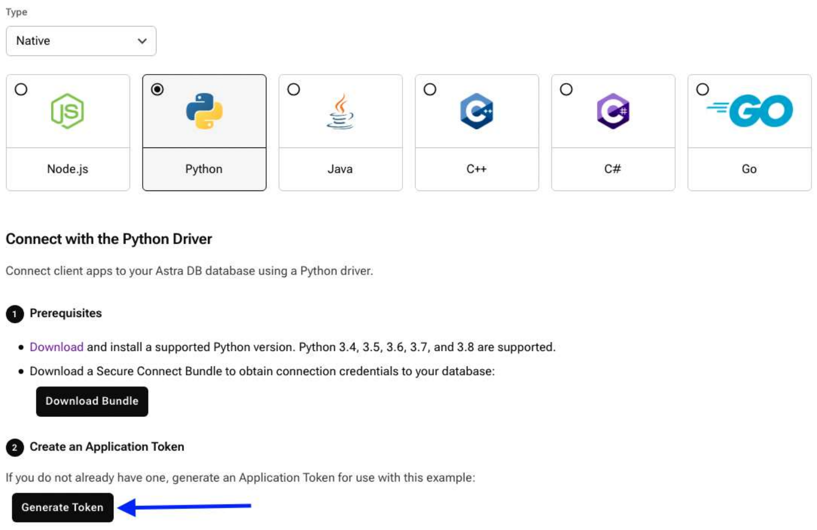

Get a Secure Connect Bundle

Use Secure Connect Bundles when connecting to your database with drivers. Secure Connect Bundles contain the SSL and TLS artifacts needed for your app.

In [4]:
```
# Make sure your current working directory contains the downloaded bundle (sec
os.listdir()
```

Out[4]:
```
['.ipynb_checkpoints',
 'ADT_Lab_12-2-1.ipynb',
 'hw_200-1.csv',
 'secure-connect-ajangam-adt.zip']
```

- Now again in the connect tab generate and download the token as shown below



Connect with the Python Driver

Connect client apps to your Astra DB database using a Python driver.

1 Prerequisites

- Download and install a supported Python version. Python 3.4, 3.5, 3.6, 3.7, and 3.8 are supported.
- Download a Secure Connect Bundle to obtain connection credentials to your database:

2 Create an Application Token

If you do not already have one, generate an Application Token for use with this example:

- Open the token file and put your clientID and secret below

In [5]:
```
cloud_config= {
  'secure_connect_bundle': 'secure-connect-ajangam-adt.zip'
}
auth_provider = PlainTextAuthProvider('OzumBsEsWABveOMprBziPZBs', 'T.DUjLCdil+
cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
session = cluster.connect()
```

In [7]:
```
# Read the data from the CSV file
with open('hw_200-1.csv', 'r') as f:
    reader = csv.reader(f)
    next(reader)
    data = [(uuid4(), row[0], float(row[1]), float(row[2])) for row in reader]
```

In [8]:
```
data[:10]
```

Out[8]:
```
[(UUID('1a7d7f01-e231-492b-870c-58abe99030dd'), '1', 65.78, 112.99),
 (UUID('a161fdd8-44d8-4eda-9799-b886fddc57fc'), '2', 71.52, 136.49),
 (UUID('ee0b0e05-2526-4654-bc06-75392cd97e45'), '3', 69.4, 153.03),
 (UUID('25b29f79-35e9-440f-b43a-cf4264d386dd'), '4', 68.22, 142.34),
 (UUID('2df0e02b-5bb6-43ce-9913-c09f5647d103'), '5', 67.79, 144.3),
 (UUID('2a1acca9-9972-48ba-a06f-7bddd42603ee'), '6', 68.7, 123.3),
 (UUID('5b014d0a-70a3-470f-8a11-0129b51cabca'), '7', 69.8, 141.49),
 (UUID('25762772-2733-4e9f-b3cd-8ef3baa399e3'), '8', 70.01, 136.46),
 (UUID('d723c0b0-ae89-4e15-b647-533b01af2c36'), '9', 67.9, 112.37),
 (UUID('d46ad495-baca-42d5-b56c-c477bed5b43c'), '10', 66.78, 120.67)]
```

In [9]:
```
# Set the keyspace
session.set_keyspace('lab12')
```

In [10]:
```
# Insert the data into the database
query = "INSERT INTO people (id, name, height, weight) VALUES (?, ?, ?, ?)"
prepared = session.prepare(query)
batch = BatchStatement()
for row in data:
    batch.add(prepared, row)
session.execute(batch)
```

Out[10]:
```
<cassandra.cluster.ResultSet at 0x23b10abc430>
```

- Now lets go to CQL console and see if values were inserted properly
- Go to CQL console; use lab12 keyspace and run the command-

```
SELECT * FROM people;
```

- You should see the following

- Now you can run some queries on the people's table

1. Retrieve 10 records from the "people" table

In [11]:
```
rows = session.execute("SELECT * FROM people limit 10")
for row in rows:
    print(row.id, row.name, row.height, row.weight)
```

```
67105059-ccc7-452f-931d-3126fbf81a34 28 67.48999786376953 131.5500030517578
4baa1e96-db10-4eee-954a-8ec3ee9a7329 84 66.27999877929688 128.94000244140625
74860221-a226-47d6-b6ab-07803fafc1c8 77 68.36000061035156 138.60000610351562
586e499e-4e79-4490-b04f-f0c91409ebf4 111 67.72000122070312 122.05999755859375
649a21f6-1b0c-4923-a9ea-690c95e79200 132 71.2300033569336 130.6999969482422
96b11863-84dc-46d8-8830-74b2e8e7bb29 57 70.41000366210938 155.89999389648438
b8dd869a-89e8-40b8-a18d-9953f2c49d00 160 65.30999755859375 115.91000366210938
7df2cba3-26f9-42a3-97ac-eb892d40a173 103 68.3499984741211 134.17999267578125
6378c6b2-9c66-4b33-a050-efbaad2ec5d6 162 64.38999938964844 109.87999725341797
1a7d7f01-e231-492b-870c-58abe99030dd 1 65.77999877929688 112.98999786376953
```

2. Retrieve all data for individuals with a height greater than 70 inches

In [12]:
```
rows = session.execute("SELECT * FROM people WHERE height > 70")
for row in rows:
    print(row.id, row.name, row.height, row.weight)
```

```
---------------------------------------------------------------------------
InvalidRequest                            Traceback (most recent call last)
Input In [12], in <cell line: 1>()
----> 1 rows = session.execute("SELECT * FROM people WHERE height > 70")
      2 for row in rows:
      3     print(row.id, row.name, row.height, row.weight)

File ~\anaconda3\lib\site-packages\cassandra\cluster.py:2618, in Session.exec
ute(self, query, parameters, timeout, trace, custom_payload, execution_profil
e, paging_state, host, execute_as)
   2575 def execute(self, query, parameters=None, timeout=_NOT_SET, trace=Fal
se,
   2576             custom_payload=None, execution_profile=EXEC_PROFILE_DEFAU
LT,
   2577             paging_state=None, host=None, execute_as=None):
   2578     """
   2579     Execute the given query and synchronously wait for the response.
   2580
   (...)
   2615     on a DSE cluster.
   2616     """
-> 2618     return self.execute_async(query, parameters, trace, custom_payloa
d, timeout, execution_profile, paging_state, host, execute_as).result()

File ~\anaconda3\lib\site-packages\cassandra\cluster.py:4901, in ResponseFutu
re.result(self)
   4899     return ResultSet(self, self._final_result)
   4900 else:
-> 4901     raise self._final_exception

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot
execute this query as it might involve data filtering and thus may have unpre
dictable performance. If you want to execute this query despite the performan
ce unpredictability, use ALLOW FILTERING"
```

- You will get an error here. Now run the below cell

In [13]:
```
rows = session.execute("SELECT * FROM people WHERE height > 70 allow filtering
for row in rows:
    print(row.id, row.name, row.height, row.weight)
```

```
649a21f6-1b0c-4923-a9ea-690c95e79200 132 71.2300033569336 130.6999969482422
96b11863-84dc-46d8-8830-74b2e8e7bb29 57 70.41000366210938 155.89999389648438
a2d0ec05-be49-401a-bece-951a9fbc4ac8 93 71.48999786376953 140.61000061035156
7b634e1b-33db-44dd-a9ac-b85fed753982 19 71.2300033569336 137.89999389648438
c8cd0cf6-ac6c-4410-8fcb-859c5950c9fe 115 70.01000213623047 122.04000091552734
a161fdd8-44d8-4eda-9799-b886fddc57fc 2 71.5199966430664 136.49000549316406
1078d33f-1619-4451-b2f0-23bf0fbf50ef 155 72.44000244140625 136.74000549316406
54e83103-27e1-44a9-9ca8-6e0cc1a2fa2a 53 70.2699966430664 125.4800033569336
f786b61f-be11-4c58-9cc8-7c5f1e571ba9 135 71.0999984741211 128.13999938964844
3cb00797-b13e-40be-95dd-1c4bdc5d783d 112 70.05000305175781 127.61000061035156
17ee1f46-6f85-47f3-a152-7776cb074b07 72 70.83999633789062 134.02000427246094
6f211bc3-8546-4d79-96cf-6a61275c1e0f 16 71.08999633789062 140.0
a2d02e2a-0c60-4cc5-b9d4-fab2f0ad31ad 130 70.23999786376953 133.97999572753906
25762772-2733-4e9f-b3cd-8ef3baa399e3 8 70.01000213623047 136.4600067138672
9bcd39ea-43d5-4b8a-8b46-18d678a1fe13 27 70.83999633789062 142.4199981689453
f1c2ac34-a789-4fe5-9407-fbc57cb8a7ed 95 70.05999755859375 133.4600067138672
47ea4663-6543-4b11-a27f-79d7a96b136b 65 70.80999755859375 135.32000732421875
67992c23-4e1c-4607-9f98-3abd0eca606c 92 70.23999786376953 141.49000549316406
c9e79b97-cd52-4091-bb1b-317ab32d1e11 96 70.55999755859375 131.8000030517578
854e9e4e-03ca-4705-9ef7-c1cc0d11369d 56 70.18000030517578 147.88999938964844
1508fe97-1f94-49d3-849a-6999dacf75be 139 73.9000015258789 151.38999938964844
facd2e39-7e53-4eca-aa34-fbd7269e8268 159 72.0199966430664 138.77999877929688
8a59ff80-56bc-4abf-a32d-b0dd6fdafb6b 83 70.05000305175781 155.3800048828125
42ee4c01-d06c-41cc-b41a-bdc08f09ceea 35 71.80000305175781 140.10000610351562
1d825ee7-a806-4559-b69e-c670bfe7f183 175 73.83000183105469 139.3000030517578
8ac38fcf-18cb-49d2-8c6b-2c81cede34ca 157 70.9800033569336 158.9600067138672
796a641b-5dd3-4265-84c7-42d95be3c3f9 34 70.5999984741211 136.22000122070312
67dc515b-7aee-401a-9c1c-ecf565e62b8f 89 70.18000030517578 121.12000274658203
76d73a5e-d57c-4ccb-b672-855e91839919 191 70.19000244140625 144.99000549316406
e80f51d0-7410-46b9-bc57-66ee6b77693a 200 71.38999938964844 127.87999725341797
48585d48-4ccd-4b05-b689-776574932f68 113 70.19000244140625 131.63999938964844
8a86412b-8d81-46e3-b957-146372d92ee7 88 70.08999633789062 131.58999633789062
```

Why is the query running if we add ALLOW FILTERING at the end?

- ALLOW FILTERING is a way to retrieve data from a table that doesn't involve using the primary key, but it can be an expensive operation as it requires scanning the entire table.
- In general, it's a best practice to avoid using ALLOW FILTERING whenever possible and to design your data model so that queries can be satisfied using the primary key. However, there may be cases where using ALLOW FILTERING is necessary or unavoidable.

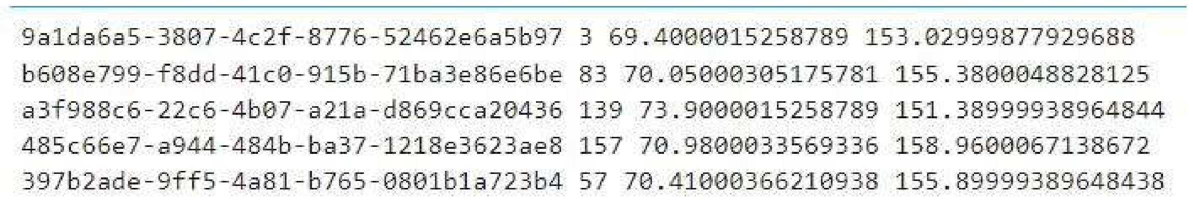
3. Similarly now retrive data for individuals with a weight between 150 and 200 pounds

In [14]:
```
rows = session.execute("SELECT * FROM people WHERE weight >= 150 and weight<=2
for row in rows:
    print(row.id, row.name, row.height, row.weight)
```

```
96b11863-84dc-46d8-8830-74b2e8e7bb29 57 70.41000366210938 155.89999389648438
ee0b0e05-2526-4654-bc06-75392cd97e45 3 69.4000015258789 153.02999877929688
1508fe97-1f94-49d3-849a-6999dacf75be 139 73.9000015258789 151.38999938964844
8a59ff80-56bc-4abf-a32d-b0dd6fdafb6b 83 70.05000305175781 155.3800048828125
8ac38fcf-18cb-49d2-8c6b-2c81cede34ca 157 70.9800033569336 158.9600067138672
```

Hint- Use allow filtering; Syntax will be similiar to normal SQL. We are actually using CQL here which is a subset of SQL

Expected output



```
9a1da6a5-3807-4c2f-8776-52462e6a5b97 3 69.4000015258789 153.02999877929688
b608e799-f8dd-41c0-915b-71ba3e86e6be 83 70.05000305175781 155.3800048828125
a3f988c6-22c6-4b07-a21a-d869cca20436 139 73.9000015258789 151.38999938964844
485c66e7-a944-484b-ba37-1218e3623ae8 157 70.9800033569336 158.9600067138672
397b2ade-9ff5-4a81-b765-0801b1a723b4 57 70.41000366210938 155.89999389648438
```

In [ ]: