Data Science Salaries in 2023: An Exploration with Cassandra

CS673 Assignment #4

Aayushi Verma

M.S. Data Science

Pace University

INTRODUCTION

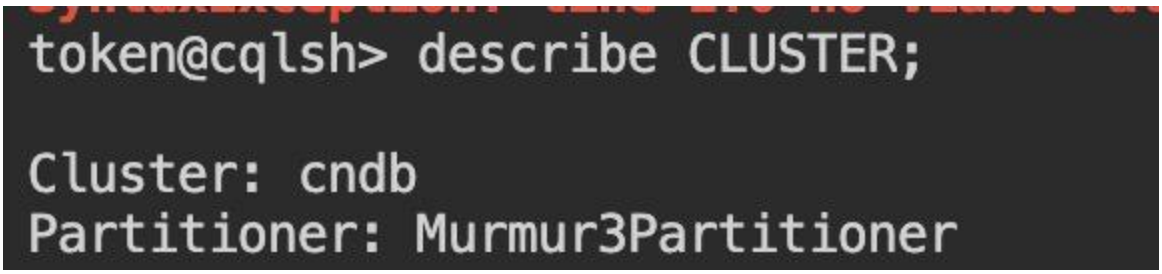
In Assignment #1, I did an analysis on the Data Science Salaries 2023 dataset to glean insights about data science salaries using SQL (specifically PostgreSQL). In this assignment, the task is to use either HBASE or Cassandra to analyze the same dataset. I decided to use Cassandra in this assignment to demonstrate my understanding of Cassandra Query Language (CQL), which is Cassandra's variant of SQL.

ANALYSIS

I used Astra DataStax to stand up an online database and use the inbuilt CQL console to perform the queries. Here, I list my queries and the results.

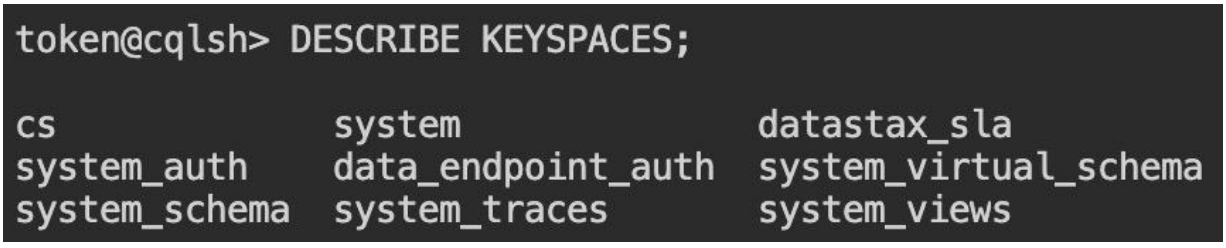
First, we start by describing the cluster that has been assigned through DataStax. We see that our cluster is called 'cndb' and has also been assigned a partitioner.

```
DESCRIBE CLUSTER;
```



We then describe the keyspaces that come with the cluster. When I was creating the cluster in DataStax, I was asked to create a keyspace, which I called 'cs'.

```
DESCRIBE KEYSPACES;
```



Before performing any queries, we need to choose a keyspace. I found that I did not have access to any other keyspaces except for the one I created, 'cs'. So I selected that keyspace, and then

created a simplified version of the Data Science Salaries data table, and then described it. We see that the underlying structure of the table I just created is now described in CQL.

```
USE cs;
CREATE TABLE ds_salaries (emp_id TEXT, job_title TEXT, salary INT,
PRIMARY KEY (emp_id));
DESCRIBE TABLE ds_salaries;
```

```
token@cqlsh> USE cs;
token@cqlsh:cs> CREATE TABLE ds_salaries (emp_id TEXT, job_title TEXT, salary INT, PRIMARY KEY (emp_id));
token@cqlsh:cs> DESCRIBE TABLE ds_salaries;

CREATE TABLE cs.ds_salaries (
    emp_id text PRIMARY KEY,
    job_title text,
    salary int
) 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';
```

I then inserted values row-by-row into the table and selected all records to make sure the data was recorded to the table ok.

```
INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0001',
'Principal Data Scientist', 85847);

SELECT * FROM ds_salaries;

INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0002',
'ML Engineer', 30000);

INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0003',
'ML Engineer', 25500);

INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0004',
'Data Scientist', 175000);

INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0005',
'Data Scientist', 120000);

SELECT * FROM ds_salaries;
```

```
token@cqlsh:cs> INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0001', 'Principal Data Scientist', 85847);
token@cqlsh:cs> SELECT * FROM ds_salaries;

 emp_id | job_title               | salary
--------+-------------------------+--------
   0001 | Principal Data Scientist |  85847

(1 rows)
token@cqlsh:cs> INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0002', 'ML Engineer', 30000);
token@cqlsh:cs> INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0003', 'ML Engineer', 25500);
token@cqlsh:cs> INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0004', 'Data Scientist', 175000);
token@cqlsh:cs> INSERT INTO ds_salaries (emp_id, job_title, salary) VALUES ('0005', 'Data Scientist', 120000);
token@cqlsh:cs> SELECT * FROM ds_salaries;
 emp_id | job_title               | salary
--------+-------------------------+--------
   0005 |          Data Scientist | 120000
   0003 |             ML Engineer |  25500
   0004 |          Data Scientist | 175000
   0002 |             ML Engineer |  30000
   0001 | Principal Data Scientist |  85847

(5 rows)
```

I then deleted one value from a column and row, and then checked to see what the table looked like. The value I deleted was replaced with null! Then I deleted the row and checked the table to be sure.

```
DELETE salary FROM ds_salaries WHERE emp_id='0005';

SELECT * FROM ds_salaries;

DELETE FROM ds_salaries WHERE emp_id='0005';

SELECT * FROM ds_salaries;
```

```
token@cqlsh:cs> DELETE salary FROM ds_salaries WHERE emp_id='0005';
token@cqlsh:cs> SELECT * FROM ds_salaries;

 emp_id | job_title               | salary
--------+-------------------------+--------
   0005 |          Data Scientist |   null
   0003 |             ML Engineer |  25500
   0004 |          Data Scientist | 175000
   0002 |             ML Engineer |  30000
   0001 | Principal Data Scientist |  85847

(5 rows)
token@cqlsh:cs> DELETE FROM ds_salaries WHERE emp_id='0005';
token@cqlsh:cs> SELECT * FROM ds_salaries;

 emp_id | job_title               | salary
--------+-------------------------+--------
   0003 |             ML Engineer |  25500
   0004 |          Data Scientist | 175000
   0002 |             ML Engineer |  30000
   0001 | Principal Data Scientist |  85847

(4 rows)
token@cqlsh:cs> ALTER TABLE ds_salaries ADD experience_level set<text>;
token@cqlsh:cs> SELECT * FROM ds_salaries;

 emp_id | experience_level | job_title               | salary
--------+------------------+-------------------------+--------
   0003 |             null |             ML Engineer |  25500
   0004 |             null |          Data Scientist | 175000
   0002 |             null |             ML Engineer |  30000
   0001 |             null | Principal Data Scientist |  85847

(4 rows)
```

I then updated the column structure of the table and added a new table called ‘experience_level’ and updated a row to add the relevant value.

```
UPDATE ds_salaries SET experience_level = {'SE'} WHERE emp_id =
'0001';
```

```
token@cqlsh:cs> UPDATE ds_salaries SET experience_level = {'SE'} WHERE emp_id = '0001';token@cqlsh:cs> SELECT * FROM ds_salaries;
 emp_id | experience_level | job_title               | salary
--------+------------------+-------------------------+--------
   0003 |             null |              ML Engineer |  25500
   0004 |             null |           Data Scientist | 175000
   0002 |             null |              ML Engineer |  30000
   0001 |          {'SE'} | Principal Data Scientist |  85847

(4 rows)
```

Finally I obtained the average salary of the table.

```
SELECT job_title, AVG(salary) AS avg_salary FROM ds_salaries;
```

```
token@cqlsh:cs> SELECT job_title, AVG(salary) AS avg_salary FROM ds_salaries;
 job_title   | avg_salary
-------------+------------
 ML Engineer |      79086

(1 rows)
```

CONCLUSION

In this project, I used Cassandra to create tables and rows, delete values and rows, insert new columns and data, and aggregate the data.