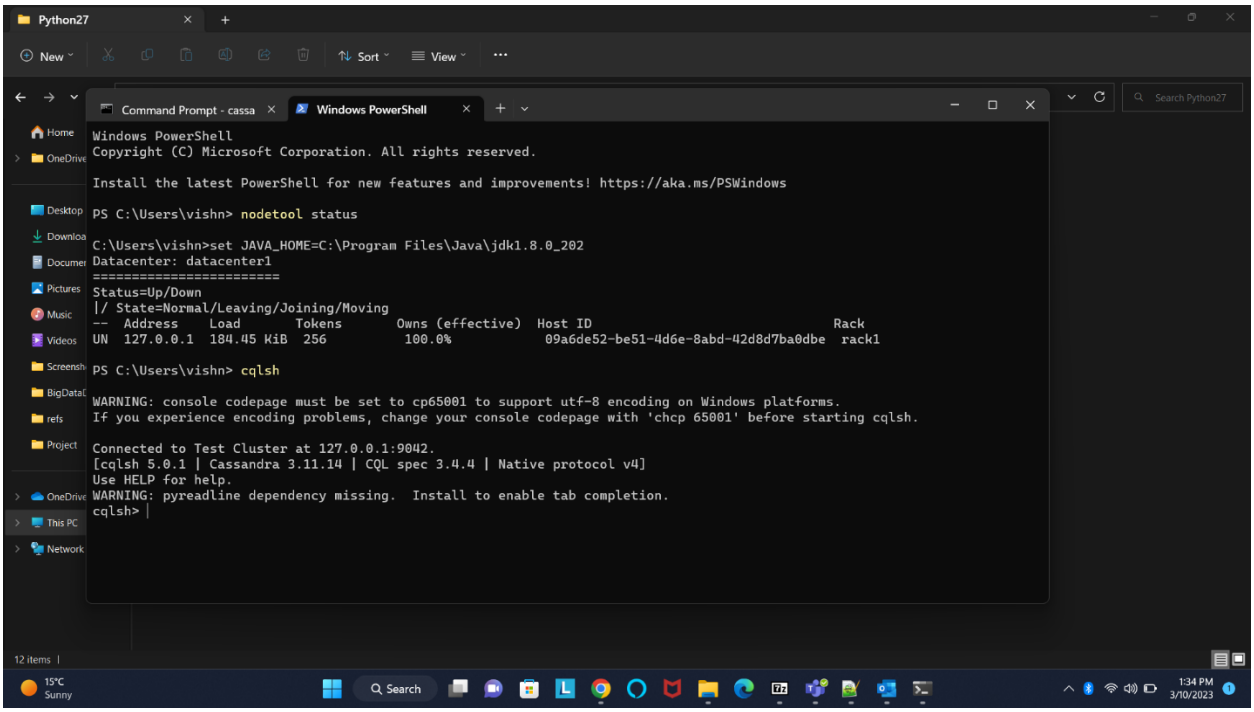
CSCE 5300 - Introduction to Big Data and Data Science

Assignment 6

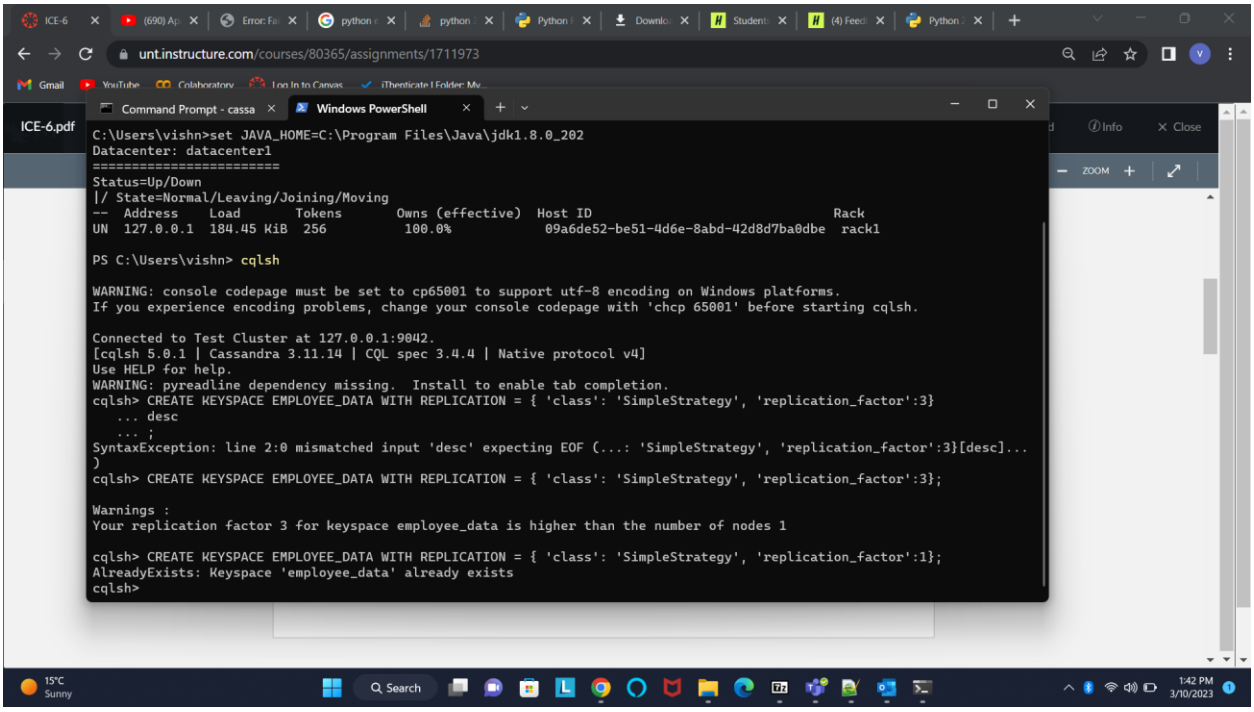
Initial setup

Installed Python version 2.7 and java version 1.8.0_202 and added the path to environment variables. Then downloaded Cassandra and added path to environment variables. Opened command prompt and checked for versions for java, python and Cassandra. Ran '**Cassandra -f**' in command prompt to start Cassandra. Open a new command prompt without closing the previous one and run 'cqlsh' in new command prompt to start Cassandra in shell.

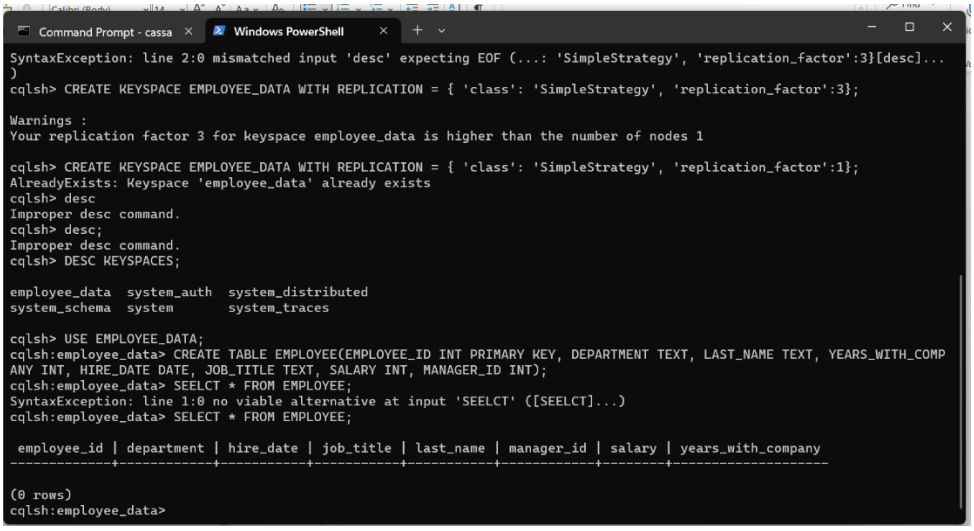


Creating Keyspace:

Created a keyspace named ‘employee_data’ with simple strategy and replication factor 1. Simple strategy is as we have only simple data without any images in it and number of nodes is also less. Then created used the command to ‘use employee_data’ to work on the employee_data keyspace.



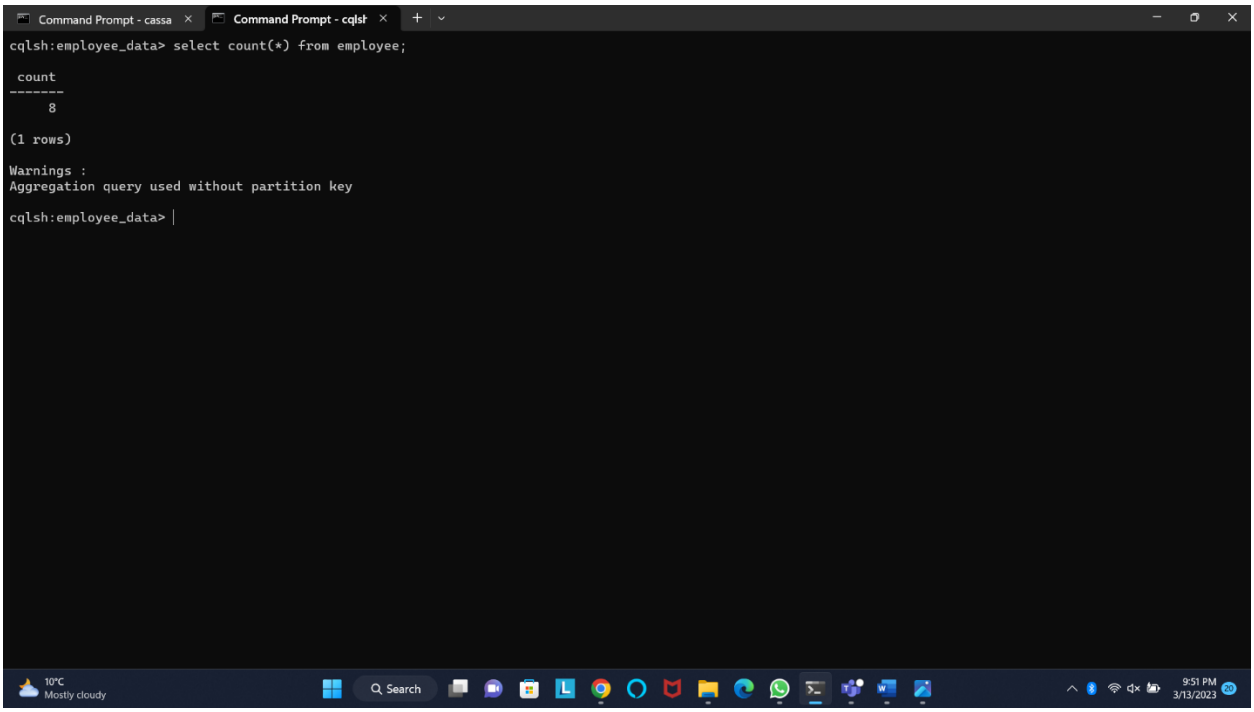
Created a table employee in employee data keyspace with their respective datatypes that are being observed from the dataset provided in the cavas along with the assignment.



Q1. To get the total count of employees in the dataset.

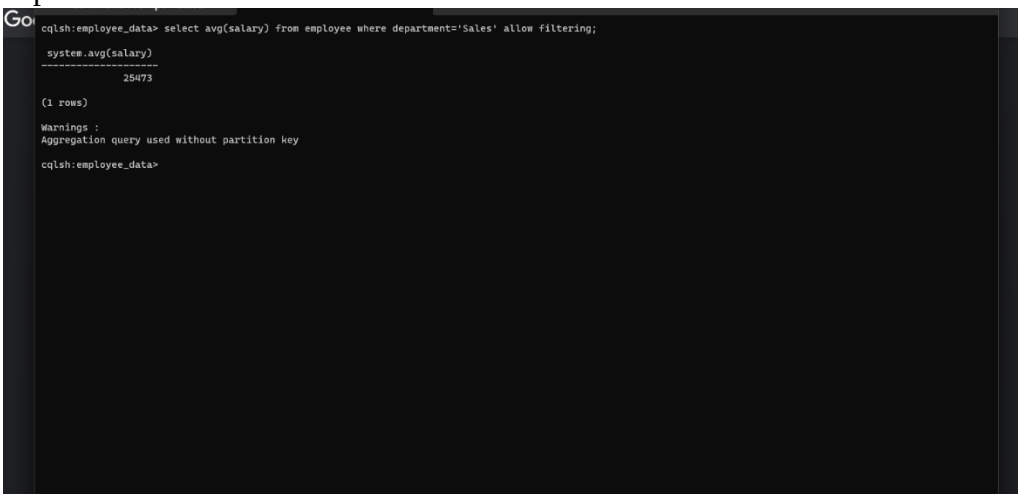
By using the count aggregate function in the select query over the whole table the total number of rows in the table is 8. The syntax used is

‘Select count(*) from <table name>’;



Q2. To write a query that fetches the average salary of the employees in the department of sales.

The average salary of all the employes who work in the sales department is 25473. This is achieved by using ‘average’ aggregate function.
The command used to get the result is ‘Select avg(salary) from employee where department=’Sales’.



Q3. To get the employee who is drawing the highest salary from the marketing department.

Smith is the employee with employe id 3 in the marketing department who is drawing the highest salary which is 80000. The query used to fetch the details of the employee who is drawing the highest salary from the given dataset is as provided in the below picture. Max is the aggregate function used to fetch the maximum number from the salary column.

The syntax used is as follows:

Select <column name1>, <column name2> … max(<salary column>) from <table name> where condition;

```
cqlsh:employee_data> select employee_id, last_name, department, max(salary) from employee where department = 'Marketing' allow filtering;

 employee_id | last_name | department | system.max(salary)
-------------+-----------+------------+--------------------
           3 |     smith |  Marketing |              80000

(1 rows)
```

Q4. Fetch the average number of years that an employee has spent in the company under the engineering department.

The aggregate function used to fetch the results for the average years an employee has spent in the company is average. The average time that an employee spent in the company is 1.75 years.

The query used to fetch the data is as follows:

Select <column name1>, <column name2> … avg(<years with company column>) from <table name> where condition;

```
cqlsh:employee_data> select lastname,avg(years_with_company) from employees where department='Engineering' allow filtering;

 lastname | system.avg(years_with_company)
----------+--------------------------------
 Gonzales |                           1.75

(1 rows)
```

Q5. To find the count of the employees that are having the job title as teamlead.

The number of employees who have the job role as teamlead is 3. The aggregate function used to fetch the data is count. The command for the query to get the total number of employees they are working as teamlead is given in the below figure. The syntax for the command is as follows.

Select count(*) from <table name> where job_title = <required job title>;

```
cqlsh:employee_data> select count(*) from employee where job_title='teamlead' allow filtering;

 count
-------
     3

(1 rows)
```

Q6. To fetch the manager who has the maximum number of employees working under him.

The below figure shows the list of managers and the number of employees that are working under them. In my observation, there is only one employee under each manager.

The query used is shown in the picture given below.

```
cqlsh:employee_data> select employee_id,lastname, count(*) from employees where managerid>0 group by managerid allow filtering;

 employee_id | lastname | count
-------------+----------+-------
           3 |    smith |     1
           8 |  Charles |     1
           6 |  Griffin |     1
           1 |  stevens |     1
           5 | Gonzales |     1
           4 |   Howard |     1
           7 |    Devin |     1
```

Q7. To get the number of employees with managers having a huge salary of above 50,000.

There are a total of 3 employees who are managers of another employees and drawing a salary of above 50000. The list of the employees and their employee ids are provided in the below figure. The query used to fetch the result is also available in the figure below. This query has a where condition to check the salary condition and a group by clause to group employes based on manager id.

```
cqlsh:employee_data> select employee_id,lastname from employees where salary>50000 group by managerid allow filtering;

 employee_id | lastname
-------------+----------
           3 |    smith
           6 |  Griffin
           2 |    jones

(3 rows)
```

Q8. Query to update the department of an employee to sales who is has the employee id as 3.

Update is the function used to update a value in the table. The desired row is selected based on the employee id. The query to update the department of the employee whose employee id is 3. This query will identify the row that selects the row that has the employee id 3 and changes the value of department to sales irrespective of the previous value present in the table.

```
cqlsh:employee_data> update employee set department='Sales' where employee_id=3;
cqlsh:employee_data> select * from employee where employee_id=3;

 employee_id | department | hire_date  | job_title | last_name | manager_id | salary | years_with_company
-------------+------------+------------+-----------+-----------+------------+--------+--------------------
           3 |      Sales | 1996-03-21 |  teamlead |     smith |          5 |  80000 |                  3
```

Q9. To delete all the employees whose salaries are less than 30000.

The employes who are drawing the salary less than 30000 are removed from the data set by using the delete command. A where command along the delete command is necessary to identify the specific rows to delete. All the data after deleting the records is shown in the figure below. The query to delete selected records is also available in the picture sited below.

```
Command Prompt - cassa    Command Prompt - cqlsh

cqlsh:employee_data> delete from employee where salary <30000 allow filtering;
cqlsh:employee_data>
cqlsh:employee_data> select * from employee;

 employee_id | department  | hire_date  | job_title        | last_name | manager_id | salary | years_with_company
-------------+-------------+------------+------------------+-----------+------------+--------+--------------------
           1 | Engineering | 2000-02-18 |          manager |   stevens |          2 |  50000 |                  1
           2 | Engineering | 1999-06-11 |          manager |     jones |          0 |  70000 |                  2
           4 |       Sales | 2003-09-21 | softwareengineer |    Howard |          6 |  45000 |                  1
           6 | Engineering | 2009-08-09 |         engineer |   Griffin |          8 |  80000 |                  2
           3 |       Sales | 1996-03-21 |         teamlead |     smith |          5 |  80000 |                  3

(5 rows)
cqlsh:employee_data>
```

Q10. To find the count of the employees who worked in marketing department.

In the initial data set the count of the employee who is from marketing department is 1. But after updating the department of the employee with employee id 3, the count of the employees with department marketing is 0. The query to fetch the count is shown in the below figure. To achieve this result the aggregate function used is count.

```
cqlsh:employee_data> select count(*) from employee where department='Marketing' allow filtering;

 count
-------
     0
```