

# CSEE5590 Big Data Programming

In Class Programming –7 Report  
(Jongkook Son)

## Project Overview:

Independent Column based No SQL Tool – Cassandra

## Requirements/Task(s):

Create KEYSPACE and a table.

Import the employees' data from the given dataset and apply the following quires

1. List the empID,ename,jobtitle,and hiredate of employee from the employee table.
2. List the name,salary of the employees who are clerks.
3. List the name,job,salary of every employee joined on 'february 18,2000',
4. List name and annual salary of all the employees.
5. Display employees' names, salary and manager values of those employees whose salary is 45000 from EMP table using SELECT statement.

## What I learned in ICP:

I could have learned how to download Cassandra on my virtual maschine and installing the dependencies like java and python. Cassandra is an open-source NoSQL database management system. Cassandra uses tables and indexes to store data. Its strength are scalability and availably for the end user. Cassandra can operate strongly on multiple data centres which are distributed over the World, along with replication which gives the Cassandra the ability to operate on low latency servers. Many companies like netfilx and reddit use Cassandra for those good points.

## INSTALLATION OF CASSANDRA AND START

(I installed it on the ubuntu virtual machine. )

```
jk@jk-VirtualBox:~$ echo "deb https://downloads.apache.org/cassandra/debian 39x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
deb https://downloads.apache.org/cassandra/debian 39x main
jk@jk-VirtualBox:~$ curl https://downloads.apache.org/cassandra/KEYS | sudo apt-key add -
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 255k  100 255k    0     0  236k      0  0:00:01  0:00:01 --:--:-- 236k
OK
jk@jk-VirtualBox:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [24.3 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [58.2 kB]
Get:4 https://dl.bintray.com/apache/cassandra 39x InRelease [3,168 B]
Get:7 https://dl.bintray.com/apache/cassandra 39x/main amd64 Packages [682 B]
Get:8 https://dl.bintray.com/apache/cassandra 39x/main i386 Packages [682 B]
```

```
jk@jk-VirtualBox:~$ sudo apt-get install cassandra
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  cassandra-tools
The following NEW packages will be installed:
  cassandra
0 upgraded, 1 newly installed, 0 to remove and 53 not upgraded.
```

```
jk@jk-VirtualBox:~$ sudo service cassandra start
jk@jk-VirtualBox:~$ nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
// State=Normal/Leaving/Joining/Moving
-- Address      Load        Tokens       Owns (effective)  Host ID                               Ra
ck
UN  127.0.0.1    140.12 KiB   256          100.0%            97d00294-d90a-4c56-a718-113160e919e0  r
ack1

jk@jk-VirtualBox:~$ python --version
Python 2.7.18
jk@jk-VirtualBox:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.9 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
cqlsh>
```

```
$ echo "deb http://www.apache.org/dist/cassandra/debian 39x main" | sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list
```

```
$ curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install Cassandra
```

```
$ sudo service cassandra start
```

## CREATE KEY SPACE

```
cqlsh> create keyspace data with replication={'class':'SimpleStrategy', 'replication_factor':3};
cqlsh> desc keyspaces;

'keyspaces' not found in keyspaces
cqlsh> desc keyspaces;

system_schema  system_auth  system  system_distributed  system_traces  data

cqlsh> use data;
cqlsh:data>
```

\$ create keyspace data with replication={'class':'SimpleStrategy', 'replication\_factor':3};

## CREATE TABLE AND IMPORT

```
cqlsh:data> copy employees (employee_id, department, lastname, years_with_company, hiredate, jobtitle, salary, manager
id) from '/home/jk/Downloads/employee_entries.csv' with DELIMITER='|' AND HEADER=TRUE;
Using 1 child processes

Starting copy of data.employees with columns [employee_id, department, lastname, years_with_company, hiredate, jobtitl
e, salary, managerid].
Failed to import 1 rows: ParseError - Invalid row length 5 should be 8, given up without retries
Failed to process 1 rows; failed rows written to import_data_employees.err
Processed: 9 rows; Rate:      14 rows/s; Avg. rate:      21 rows/s
9 rows imported from 1 files in 0.426 seconds (0 skipped).
cqlsh:data> select * from employees;
```

employee_id	department	hiredate	jobtitle	lastname	managerid	salary	years_with_company
5	Engineering	2011-09-23	testengineer	Gonzales	7	20000	2
1	Engineering	2000-02-18	manager	stevens	2	50000	1
8	Sales	2008-01-07	teamlead	Charles	1	19220	8
2	Engineering	1999-06-11	manager	jones	0	70000	2
4	Sales	2003-09-21	softwareengineer	Howard	6	45000	1
7	Sales	2010-01-07	teamlead	Devin	3	12200	2
6	Engineering	2009-08-09	engineer	Griffin	8	80000	2
3	Marketing	1996-03-21	teamlead	smith	5	80000	3

(8 rows)  
cqlsh:data>

\$ use data;

\$ create table data.employees (employee\_id int PRIMARY KEY, department text, lastname text, year\_with\_company int, hiredate text, jobtitle text, salary int, managerid int);

\$ copy employees (employee\_id, department, lastname, year\_with\_company, hiredate, jobtitle, salary, managerid) from '/home/jk/Downloads/employee\_entries.csv' with DELIMITER='|' AND HEADER=TRUE;

\$ select \* from employees;

**QUERY1** (List the employee\_id, lastname, jobtitle, and hiredate of employee from the employee table)

```
cqlsh:data> select employee_id, lastname, jobtitle, hiredate from employees;
```

employee_id	lastname	jobtitle	hiredate
5	Gonzales	testengineer	2011-09-23
1	stevens	manager	2000-02-18
8	Charles	teamlead	2008-01-07
2	jones	manager	1999-06-11
4	Howard	softwareengineer	2003-09-21
7	Devin	teamlead	2010-01-07
6	Griffin	engineer	2009-08-09
3	smith	teamlead	1996-03-21

(8 rows)

\$ select employee\_id, lastname, jobtitle, hiredate from employees;

**QUERY2** (List the lastname, salary, of the employees who are clerks.)

```
(0 rows)
cqlsh:data> SELECT lastname, salary from employees where jobtitle = 'clerk' ALLOW FILTERING;
```

lastname	salary
----------	--------

(0 rows)

```
cqlsh:data> SELECT lastname, salary from employees where jobtitle = 'teamlead' ALLOW FILTERING;
```

lastname	salary
Charles	19220
Devin	12200
smith	80000

(3 rows)

\$ select lastname, salary from employees where jobtitle = 'clerk' ALLOW FILTERING;

**QUERY3 (List the lastname, job, salary, of every employee joined on 'February 18,2000'.)**

```
cqlsh:data> SELECT lastname, jobtitle, salary from employees where hiredate='2000-02-18' ALLOW FILTERING;
```

lastname	jobtitle	salary
stevens	manager	50000

\$ select lastname, jobtitle, salary from employees where hiredate='2000-02-18' ALLOW FILTERING;

**QUERY4 (List lastname, and annual salary of all the employees.)**

```
cqlsh:data> SELECT lastname, salary from employees;
```

lastname	salary
Gonzales	20000
stevens	50000
Charles	19220
jones	70000
Howard	45000
Devin	12200
Griffin	80000
smith	80000

\$ select lastname, salary from employees;

**QUERY5 (Display employees' lastname, salary and manager values of those employees whose salary is 45000 from employees table using select statement.)**

```
cqlsh:data> SELECT lastname, salary, managerid from employees where salary=45000 ALLOW FILTERING;
```

lastname	salary	managerid
Howard	45000	6

(1 rows)

\$ select lastname, salary, managerid from employees where salary=45000 ALLOW FILTERING;

## BONUS

=>I used catfood\_entries dataset for import it and apply some commands like above

```
cqlsh:data> copy catfood (item_id, brand, flavor, can_price) from '/home/jk/Downloads/catfood_entries.csv' with DELIMITER='|' AND HEADER=TRUE;
Using 1 child processes

Starting copy of data.catfood with columns [item_id, brand, flavor, can_price].
Processed: 10 rows; Rate:      20 rows/s; Avg. rate:      28 rows/s
10 rows imported from 1 files in 0.357 seconds (0 skipped).
cqlsh:data> SELECT * FROM catfood;
```

item_id	brand	can_price	flavor
5	Iams	1.49	chicken
10	Evangers	1.85	chicken
1	Friskies	1.25	chicken
8	Wellness	1.34	chicken
2	Friskies	1.45	beef
4	TikiCat	1.58	fish
7	Iams	1.05	beef
6	Iams	1.25	pork
9	Wellness	1.34	turkey
3	TikiCat	1.79	chicken

```
(10 rows)
cqlsh:data> SELECT brand, can_price from catfood;
```

brand	can_price
Iams	1.49
Evangers	1.85
Friskies	1.25
Wellness	1.34
Friskies	1.45
TikiCat	1.58
Iams	1.05
Iams	1.25
Wellness	1.34

\$ create table catfood (item\_id int PRIMARY KEY, brand text, flavor text, can\_price float);

\$ copy catfood (item\_id, brand, flavor, can\_price) from  
'/home/jk/Downloads/catfood\_entries.csv' with DELIMITER='|' AND HEADER=TRUE;

\$SELECT \* FROM catfood;

\$SELECT brand, can\_price from catfood;

```
cqlsh:data> select brand, can_price from catfood where flavor = 'chicken' ALLOW FILTERING;
```

brand	can_price
Iams	1.49
Evangers	1.85
Friskies	1.25
Wellness	1.34
TikiCat	1.79

```
(5 rows)
```

\$ select brand, can\_price from catfood where flavor = 'chicken' ALLOW FILTERING;

=>Display the brand, can\_price where the flavor's value is chicken

```
cqlsh:data> select item_id, brand, flavor from catfood where can_price<1.5 ALLOW FILTERING;
```

item_id	brand	flavor
5	Iams	chicken
1	Friskies	chicken
8	Wellness	chicken
2	Friskies	beef
7	Iams	beef
6	Iams	pork
9	Wellness	turkey

(7 rows)

\$ select item\_id, brand, flavor from catfood where can\_price<1.5 ALLOW FILTERING;

=>Display item id and brand where can price is below 1.5

```
cqlsh:data> SELECT brand, flavor, can_price from catfood where can_price>1.5 AND flavor='chicken' ALLOW FILTERING;
```

brand	flavor	can_price
Evangers	chicken	1.85
TikiCat	chicken	1.79

\$ select brand, flavor, can\_price from catfood where can\_price>1.5 AND flavor='chicken'  
ALLOW FILTERING;

=>Display flavor and can price where can price is over 1.5 and flavor value is chicken

```
cqlsh:data> SELECT brand, flavor from catfood where brand='Iams' ALLOW FILTERING;
```

brand	flavor
Iams	chicken
Iams	beef
Iams	pork

(3 rows)

\$ select brand, flavor from catfood where brand='Iams' ALLOW FILTERING;

=>Display brand and flavor where brand value is Iams