Big Data Management and Analytics
Assignment 3

Assignment 3-1

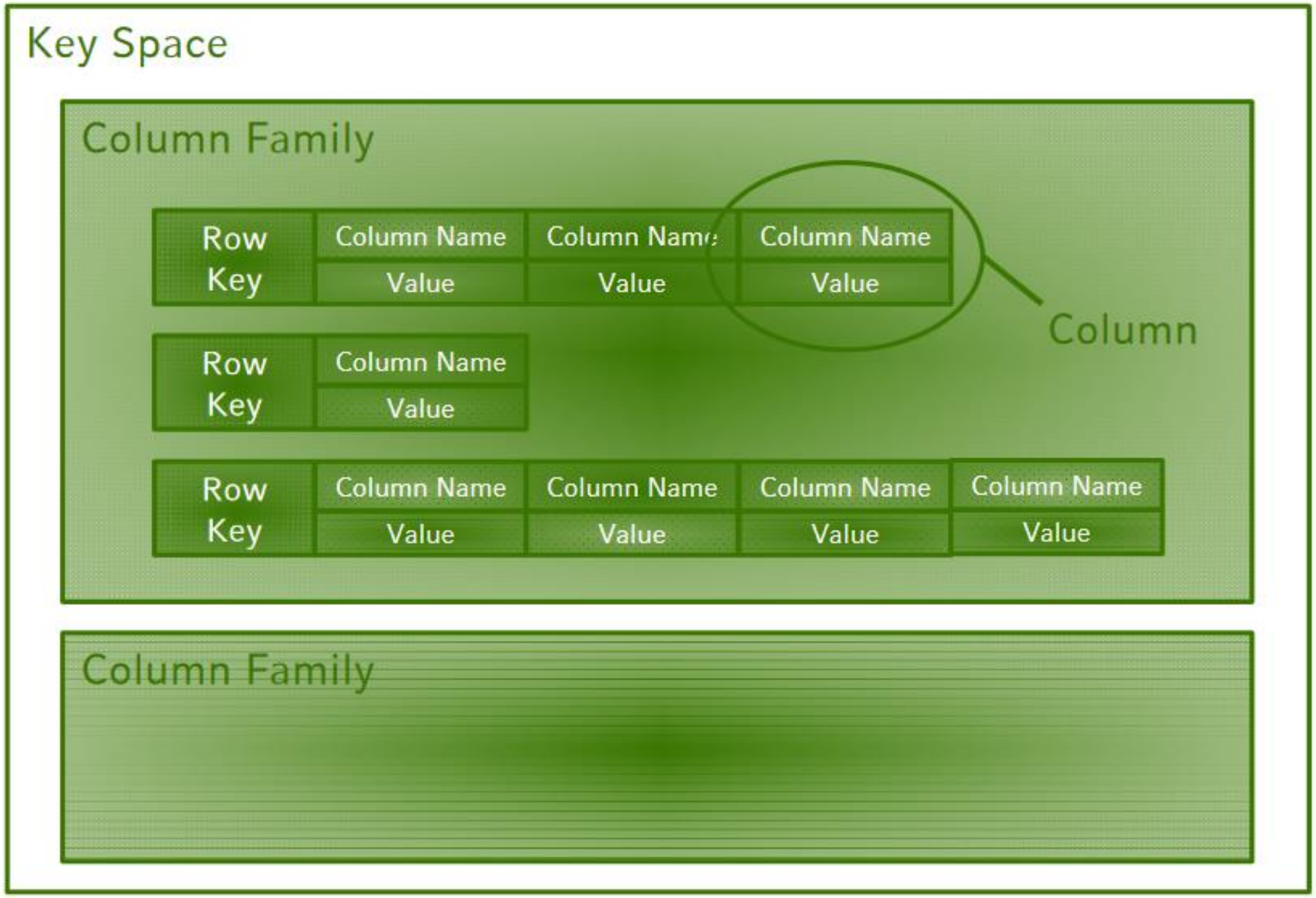
(a) column based vs. RDBMS

RDBMS	Cassandra
Database	Keyspace
Table	Column family
Primary key	Row key
Column name	Column name/key
Column value	Column value

→ Don't use this analogy while designing Cassandra column families

→ Think of a column family as a map of a map!

Assignment 3-1: RECAP



Source: Big Data Management and Analytics, V1: Volume — Chapter 2 Part 1:NoSQL Databases, p.86

Assignment 3-1

(c) Create a keyspace named *mycompany*

Replica Placement Strategy

```
CREATE KEYSPACE mycompany
      WITH replication = {'class':'SimpleStrategy',
                          'replication_factor':3};
```

Number of replicas wanted

Assignment 3-1

(d) Create a column family named employees. It has the following column keys: emp_id, emp_name, emp_city, emp_phone, emp_salary

Alternatively : TABLE

```
USE mycompany;

CREATE COLUMNFAMILY employees(emp_id int PRIMARY KEY,
                              emp_name text,
                              emp_city text,
                              emp_phone varint,
                              emp_sal varint);

Validate your created column family by typing:
SELECT * FROM employees;
```

Assignment 3-1

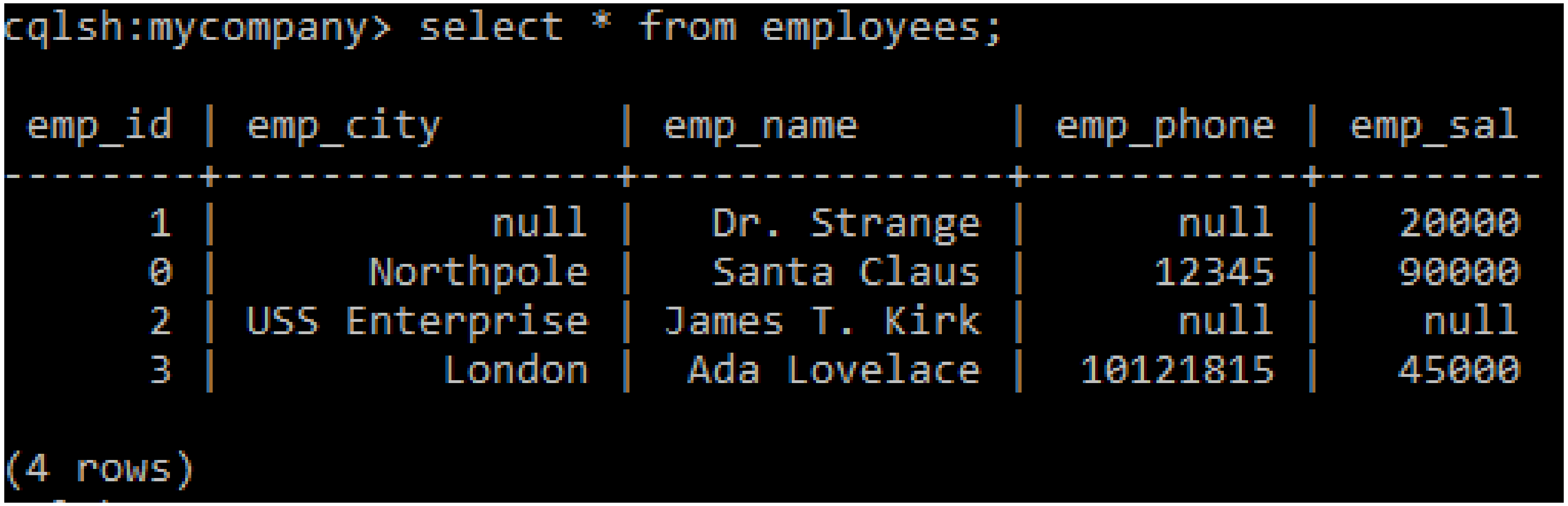
(e) Add the following employees to the employees column family:

emp_id	emp_name	emp_city	emp_phone	emp_sal
0	Santa Claus	Northpole	12345	90000
1	Dr. Strange			20000
2	James T. Kirk	USS Enterprise		
3	Ada Lovelace	London	10121815	45000

```
INSERT INTO employees
      (emp_id, emp_name, emp_city, emp_phone, emp_sal)
      VALUES (0, 'Santa Claus', 'Northpole', 12345, 90000);
```

Assignment 3-1

(e) Add the following employees to the employees column family:



```
cqlsh:mycompany> select * from employees;

 emp_id | emp_city          | emp_name        | emp_phone | emp_sal
--------+-------------------+-----------------+-----------+---------
      1 |              null |    Dr. Strange  |      null |   20000
      0 |         Northpole |    Santa Claus  |     12345 |   90000
      2 |  USS Enterprise   | James T. Kirk   |      null |    null
      3 |            London |   Ada Lovelace  |  10121815 |   45000

(4 rows)
```

Assignment 3-1

(f) Return all employees which have a salary above 30000.

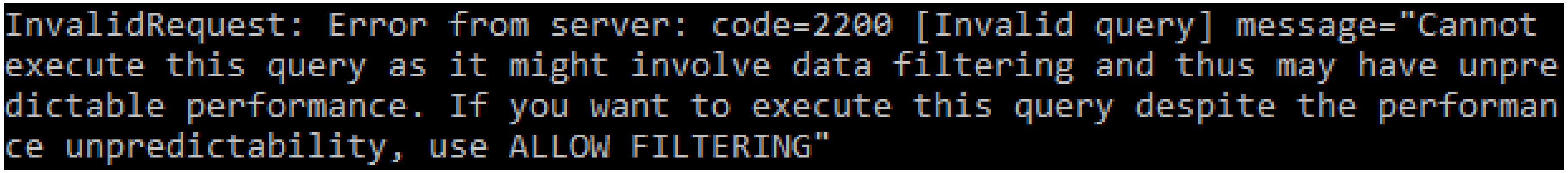
Hint: for this purpose you'll probably use a WHERE clause. Can you use WHERE clauses the way you'd use them in RDBMS?

```
SELECT * FROM employees WHERE emp_sal > 30000
```

Assignment 3-1

(f) Return all employees which have a salary above 30000.

Hint: for this purpose you'll probably use a WHERE clause. Can you use WHERE clauses the way you'd use them in RDBMS?



```
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"
```

Solution:

```
SELECT * FROM employees WHERE emp_sal > 30000 ALLOW FILTERING;
```

DATABASE SYSTEMS GROUP

Assignment 3-2

LMU

(a) What are the main building blocks of the Neo4j data model? What are the corresponding counterparts for a database, a table, a row, a column and join in RDBMS?

RDBMS	Neo4j
Database	Graph
Table	Node label
Row	Node
Column	(node) property
Join	Relationship

Assignment 3-2

(c)In this task you'll become a bit more familiar with the cypher query language (CQL). Based on the table below, create employee nodes with the properties emp_id, emp_name, emp_city, emp_phone and emp_sal. The label of all nodes is *Employee*

node name	emp_id	emp_name	emp_city	emp_phone	emp_sal
santa	0	Santa Claus	Northpole	12345	90000
strange	1	Dr. Strange			20000
kirk	2	James T. Kirk	USS Enterprise		
ada	3	Ada Lovelace	London	10121815	45000

Assignment 3-2

(c)In this task you'll become a bit more familiar with the cypher query language (CQL). Based on the table below, create employee nodes with the properties emp_id, emp_name, emp_city, emp_phone and emp_sal. The label of all nodes is *Employee*

```
CREATE (santa:Employee
             { emp_id:0,
               emp_name:"Santa Claus",
               emp_city:"Northpole",
               emp_phone:12345,
               emp_sal:90000})
```

Assignment 3-2

(d) Write a query (by using a MATCH-RETURN statement) which returns all the employee names.

```
MATCH (emp: Employee)
RETURN emp.emp_name
```

Assignment 3-2

(e) Write a query (by using a MATCH-WHERE-RETURN statement) which returns the names of all employees which have a salary above 40000

```
MATCH (emp: Employee)
WHERE emp.emp_sal > 40000
RETURN emp.emp_name
```

Assignment 3-2

(f) Create the following node with the label *Customer*

node name	cust_name	cust_city	cust_complain
thor	Thor	Asgard	Hammer not working

```
CREATE (thor:Customer
              { cust_name:"Thor",
                cust_city:"Valhalla",
                cust_complain:"Hammer not working"})
```

Assignment 3-2

(g) Santa Claus delivers Thor a new working hammer. Create a relation between Santa and Thor with the type **delivers_to** and the label *DELIVERS* (using the MATCH-CREATE statement). Use the Neo4j graph view in order to see your current graph.

```
MATCH  (santa:Employee
            {emp_name:"Santa Claus"}),
       (thor:Customer
            {cust_name:"Thor"})

CREATE (santa)-[delivers_to:DELIVERS]->(thor)
```