Jetty, Lalitha Kumari – Class ID 5

Vinay Chandra Vasamsetti – Class ID 25

Team 11

BigData Programming

Lab1 Assignment

1. **Hadoop MapReduce Algorithm**

Implement MapReduce algorithm for finding Facebook common friends problem and run the MapReduce job on Apache Hadoop. Show your implementation through map-reduce diagram as shown in Lesson Plan 2 : (<https://umkc.box.com/s/epp04s3m6g8jw6ulnnb4meqgnbwbz5uc>)

Write a report including your algorithm and result screenshots.

**Finding Facebook common friends**: Facebook has a list of friends (note that friends are a bi-directional thing on Facebook. If I'm your friend, you're mine). They also have lots of disk space and they serve hundreds of millions of requests everyday. They've decided to pre-compute calculations when they can to reduce the processing time of requests. One common processing request is the "You and Joe have 230 friends in common" feature. When you visit someone's profile, you see a list of friends that you have in common. We're going to use MapReduce so that we can calculate everyone's common friends once a day and store those results. Later on it's just a quick lookup. We've got lots of disk, it's cheap.

**Example (What is the Key/Value Pair?)**

Assume the friends are stored as Person->[List of Friends], our friends list is then:

A -> B C D

B -> A C D E

C -> A B D E

D -> A B C E

E -> B C D

The result after reduction is:

(A B) -> (C D)

(A C) -> (B D)

(A D) -> (B C)

(B C) -> (A D E)

(B D) -> (A C E)

(B E) -> (C D)

(C D) -> (A B E)

(C E) -> (B D)

(D E) -> (B C)

When D visits B's profile, we can quickly look up (B D) and see that they have three friends in common, (A C E).



Input

A **:** **B** **C** **D**

B **:** **A** **C** **D** **E**

C **:** **A** **B** **D** **E**

D **:** **A** **B** **C** **E**

E **:** **B** **C** **D**

**Mapper**

Generates key value pairs for each user to its friends.

A **:** **B** **C** **D**

map **=>**

**(**A**,** B**)** **=>** B C D

**(**A**,** C**)** **=>** B C D

**(**A**,** D**)** **=>** B C D

**Reducer**

Finds the common friends to the each user from the keyvalue pairs.

(A, B) => [B C D, A C D E]

reduce =>

(A, B) => C D

**Output**

**A B: C D**

**A C: B D**

**A D: B C**

**B C: A D E**

**B D: A C E**

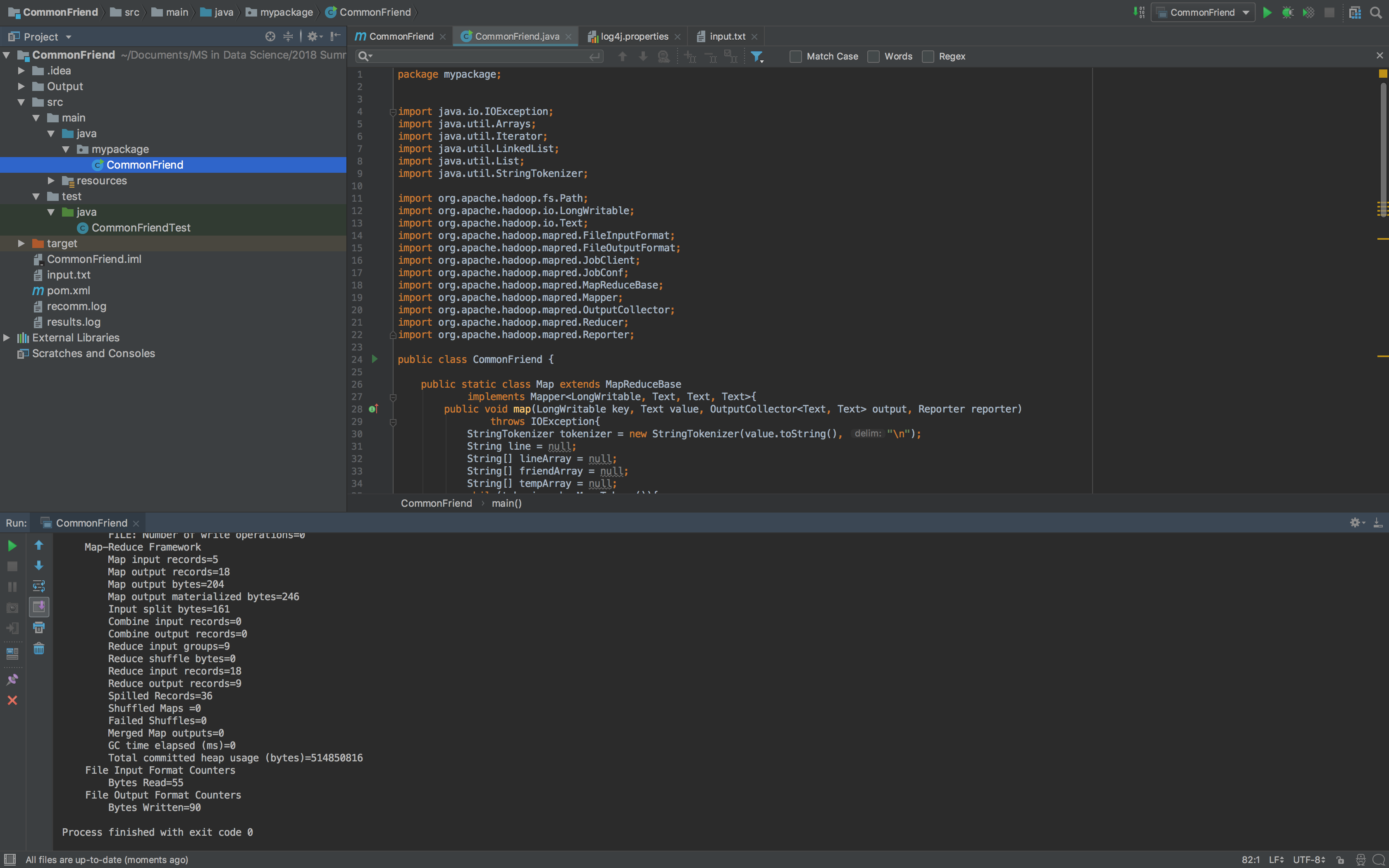
**B E: C D**

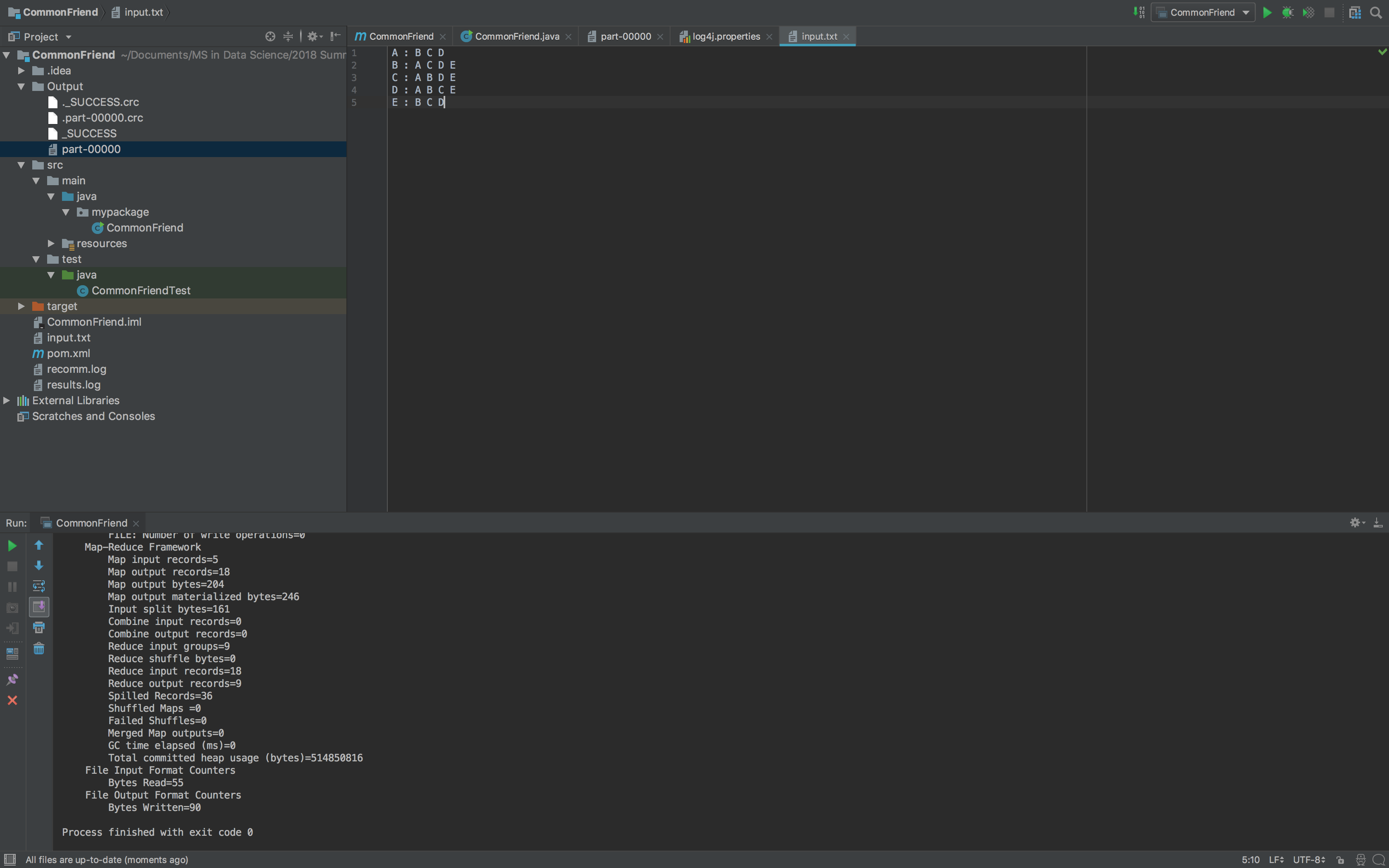
**C D: A B E**

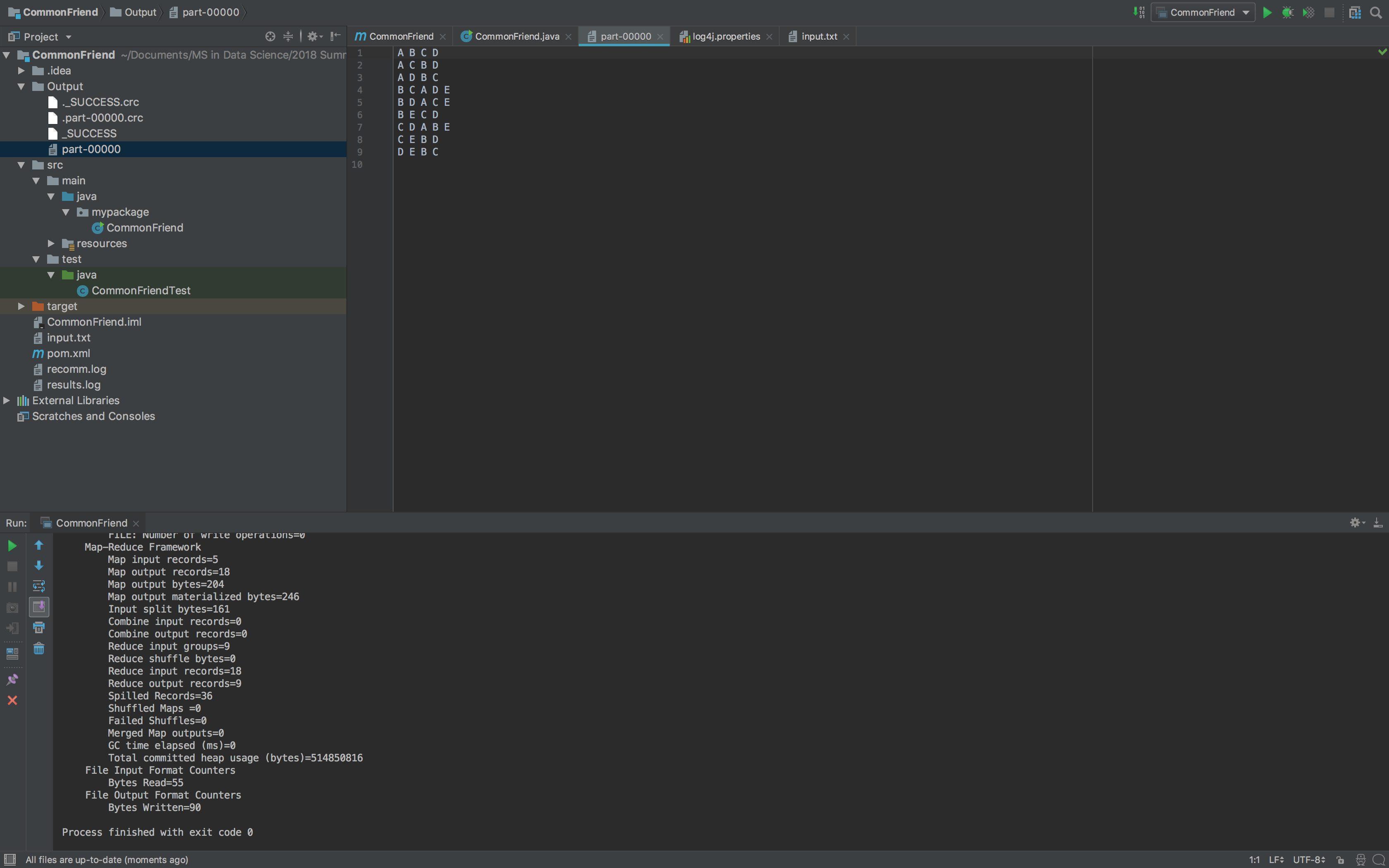
**C E: B D**

**D E: B C**

Code Screenshots







Reference - http://ernie55ernie.github.io/mapreduce/2016/06/30/map-reduce-common-friend.html

1. **Use Case Based No SQL Comparison**

Consider one of the use cases from the below link: <https://umkc.box.com/s/q64fvjm6yd454w5v3ky0he4854g6m1fq>

These use cases were discussed in Lecture 1: Cassandra.

1. Consider one of the use case and use a simple dataset. Describe the use case considered based on your assumptions, report the dataset, its fields, datatype etc.
2. Use HBase to implement a Solution for the use case. Report at least 3 queries, their input and output. The query’s relevance towards solving the use case is important.
3. Use Cassandra to implement a Solution for the use case. Report at least 3 queries, their input and output. The query’s relevance towards solving the use case is important.
4. Compare Cassandra and HBase for your use case. Present a table with comparison of your use case being implemented in both NO SQL Systems.

**Use Case – Coursera**

**Description** – Coursera is an Education Platform which partners with top universities and organizations worldwide, to offer courses online for anyone to take, for free.

**Challenges** –

1. My SQL was insufficient
2. Unstable performance
3. Unexpected downtime
4. Limitation in introducing new features

**Solution** – After evaluating emerging database technologies, it chooses Cassandra (data stax)

**Reason** – 100% application uptime needed and scalability (enabling storage of growing user data)

**Creating Table in Hbase – Courses by Learner with column families as LearnerDetails, CourseDetails.**

Commands Used:

create 'Coursera', 'LearnerDetails','CourseDetails'

put 'Coursera','1','LearnerDetails:LearnerID','111'

put 'Coursera', '1', 'LearnerDetails:LearnerName’, ’Lalitha’

put 'Coursera', ‘2’, 'LearnerDetails:LearnerID’, ’222’

put 'Coursera', '2, 'LearnerDetails:LearnerName’, ’Vinay’

put 'Coursera', '1', ‘CourseDetails:CourseID’ ,’ML001’

put 'Coursera', '1', ‘CourseDetails:CourseName’ ,’Machine Learning’

put 'Coursera', ‘2, ‘CourseDetails:CourseID’ ,’DL001’

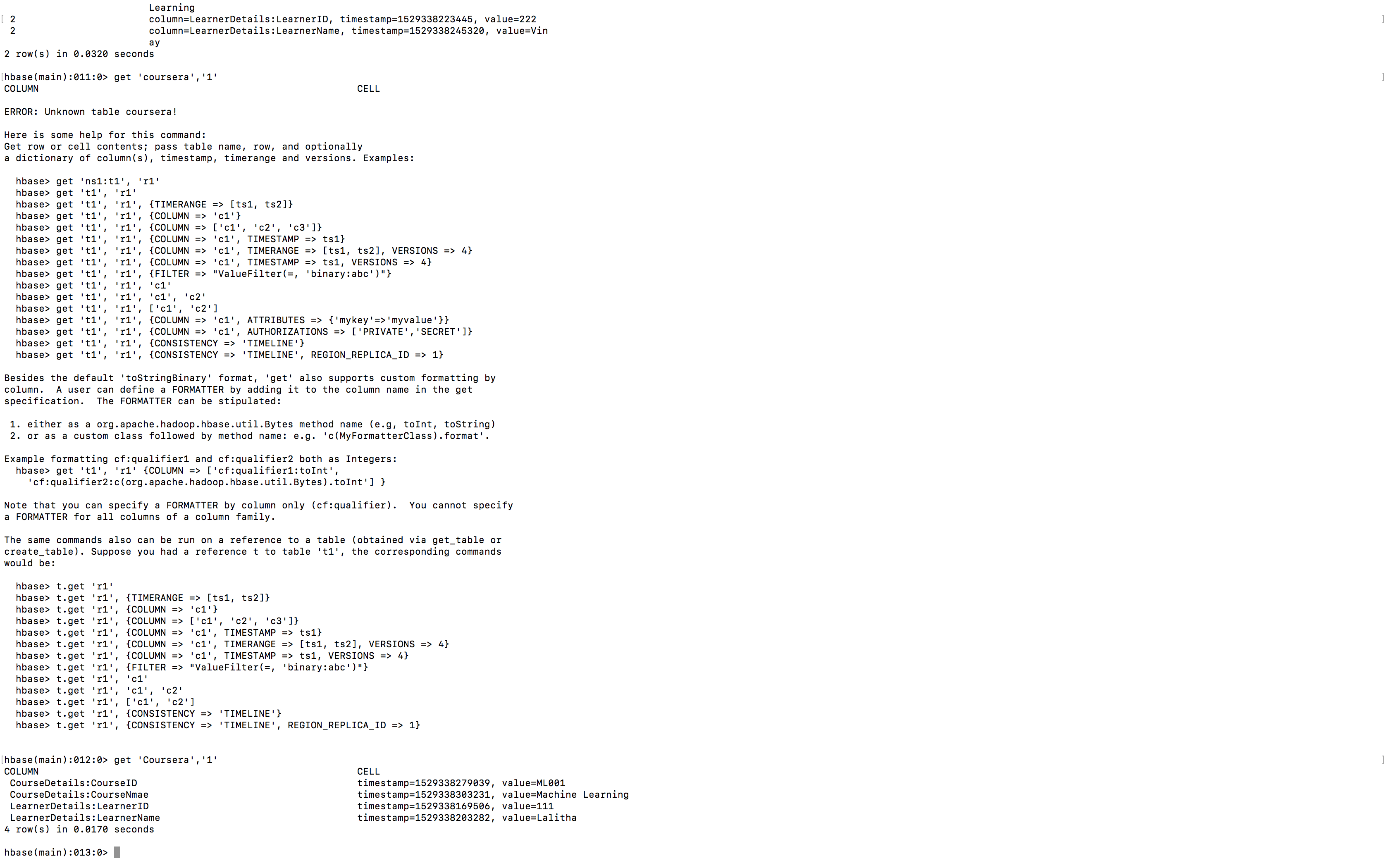
put 'Coursera', ‘2’, ‘CourseDetails:CourseName’ ,’Deep Learning’

scan ‘Coursera’

List

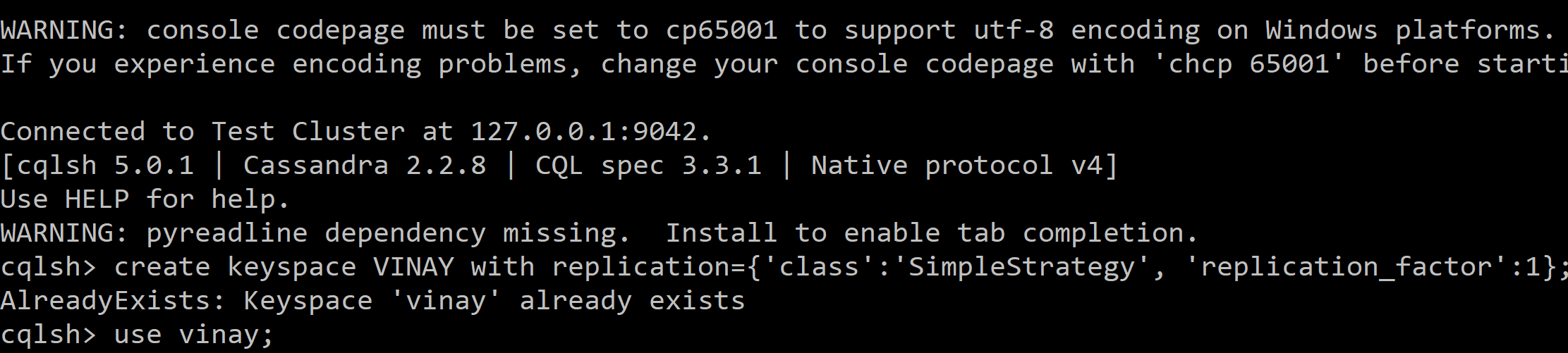
get ‘Coursera’, ’1’



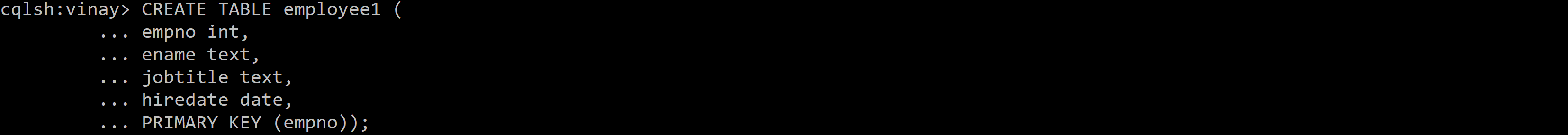


**Creating Table in Cassandra – employee with ename, job title and hiredate**

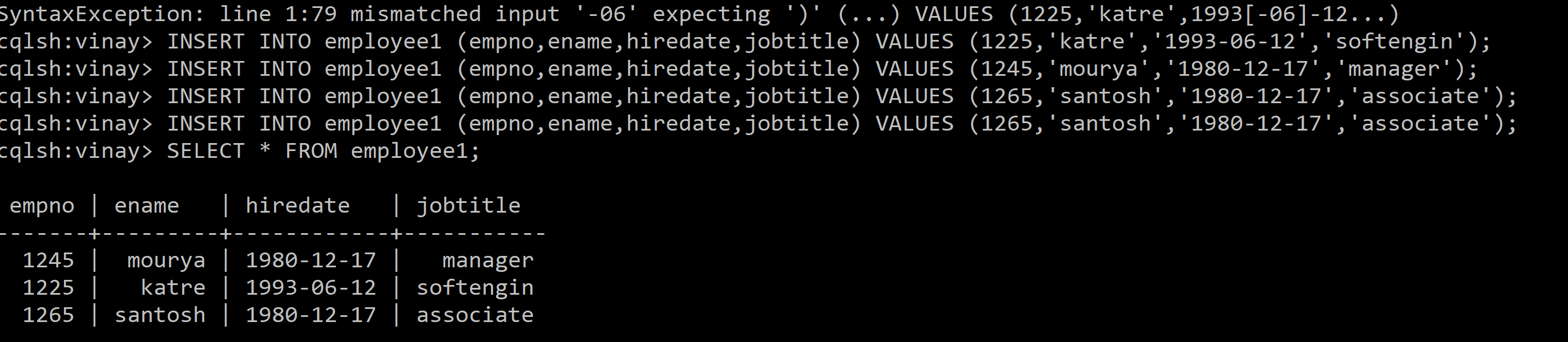
Creating Keyspace name Vinay



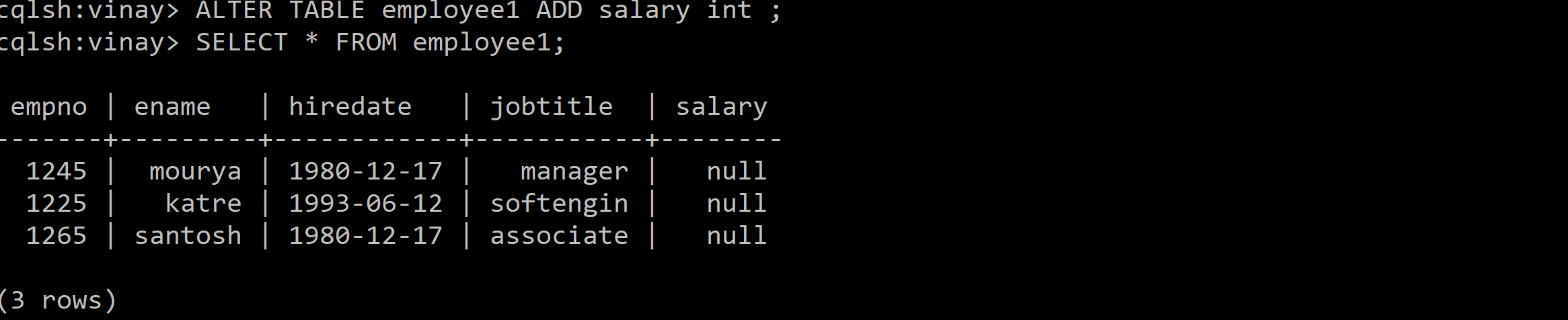
Creating Table employee1



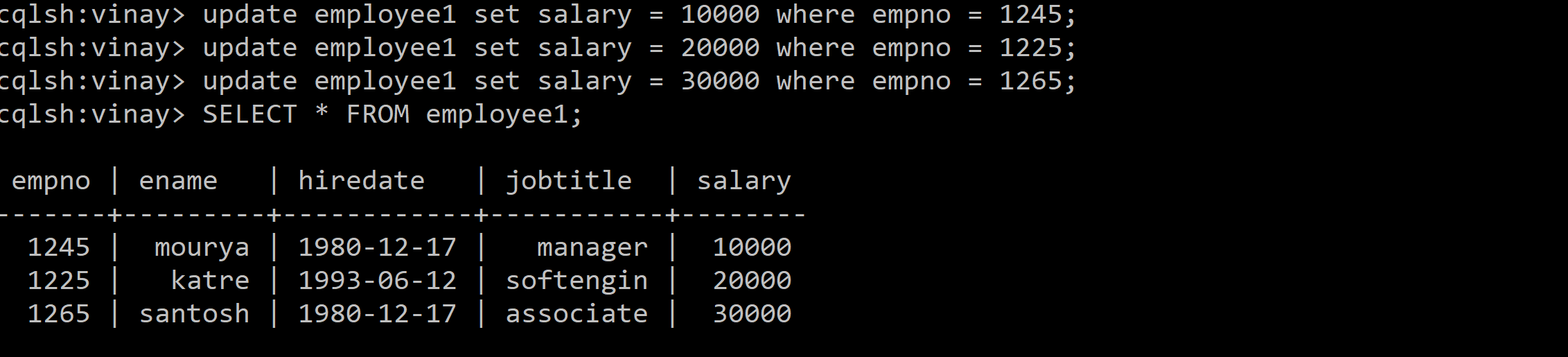
Inserting values in to table



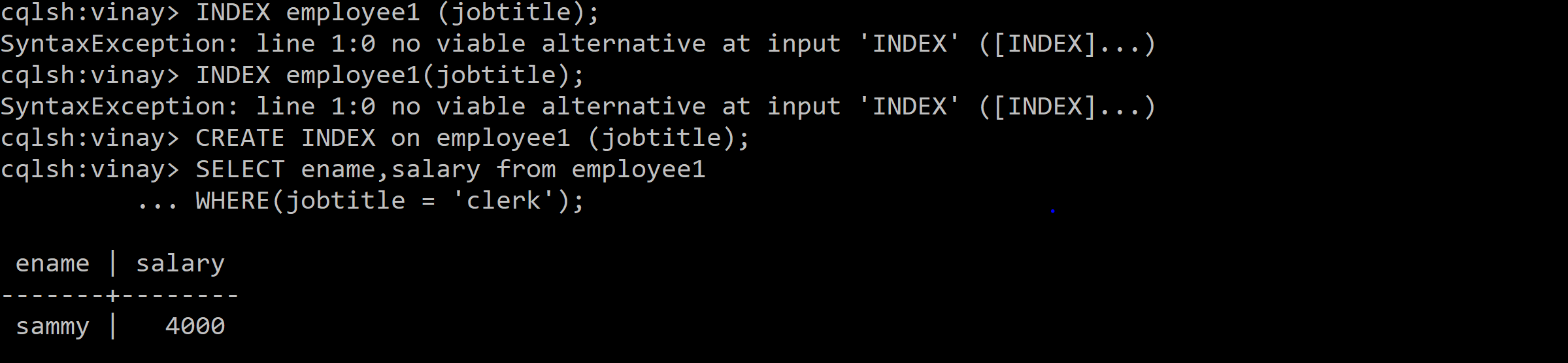
Adding Column using Alter command



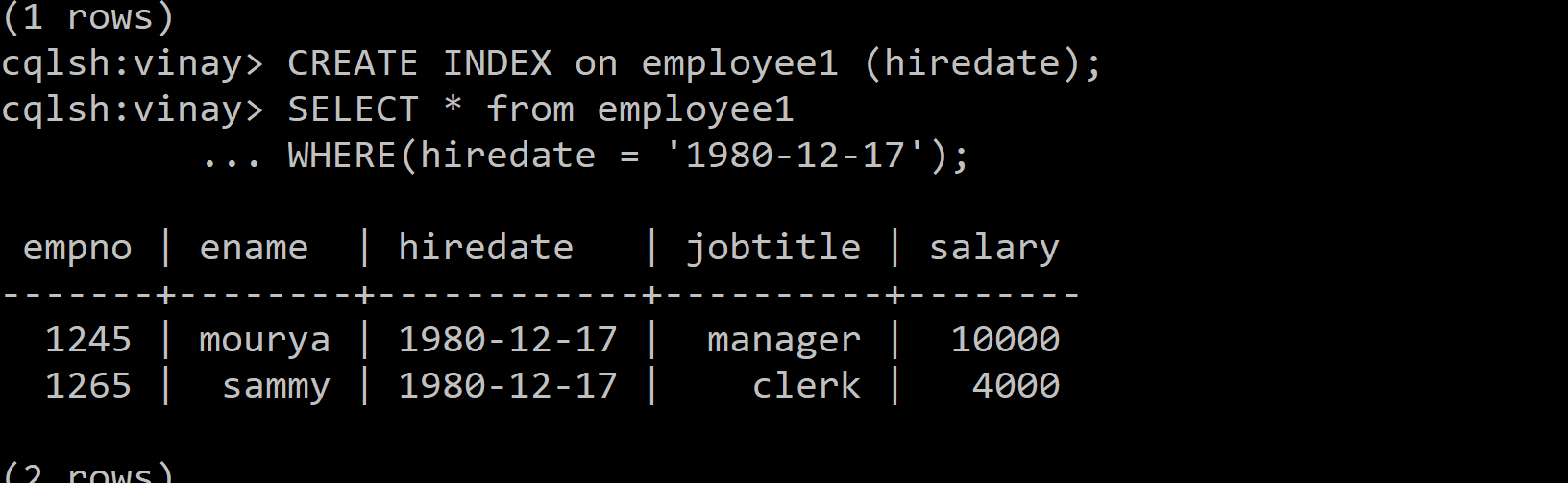
Updating salaries if the employees



List name salary of the employee who are clerks



List the name,job,salary for every employee joined on dec 17 1980



**Similarities and differences between Cassandra and Hbase:**

|  |  |  |
| --- | --- | --- |
|  | **Cassandra** | **Hbase** |
| **Database** | open-source NoSQL Database, They both are distributed database that can manage extremely large data sets and handle non-relational data. | open-source NoSQL Database, They both are distributed database that can manage extremely large data sets and handle non-relational data. |
| **Scalability** | high linear scalability. That means to handle more data, the user should simply increase the number of nodes in the cluster. Because of this feature, they both are an excellent choice for handling a large amount of data. | high linear scalability. That means to handle more data, the user should simply increase the number of nodes in the cluster. Because of this feature, they both are an excellent choice for handling a large amount of data. |
| **Replication** | But in both Cassandra and HBase, there is a safeguard that prevents data loss even after failure | But in both Cassandra and HBase, there is a safeguard that prevents data loss even after failure |
| **Programming/Coding** | Both are column-oriented databases that implement similar write paths | Both are column-oriented databases that implement similar write paths |
| **Infrastructure** | Cassandra, on the other hand, has different infrastructure and operation than Hadoop | This HBase-Hadoop infrastructure consists of several moving parts like Zookeeper, HBase master, Data nodes and Name Node. |
| **Support** | Cassandra, on the other hand, supports ordered partitioning. Cassandra is also limited in supporting range based row scans. | HBase, do not supports ordered partitioning. HBase offers a coprocessor capability |

Reference - https://data-flair.training/blogs/hbase-vs-cassandra/