BIG DATA ANALYTICS

ASSIGNMENT 7: Hands on over Apache HIVE.

(Amitabh Saini, TY-E-4, 141541)

* **Introduction**

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analysing easy. It is a platform used to develop SQL type scripts to do MapReduce operations.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not

* A relational database
* A design for OnLine Transaction Processing (OLTP)
* A language for real-time queries and row-level updates

Features of Hive

* It stores schema in a database and processed data into HDFS.
* It is designed for OLAP.
* It provides SQL type language for querying called HiveQL or HQL.
* It is familiar, fast, scalable, and extensible.
* **Architecture**

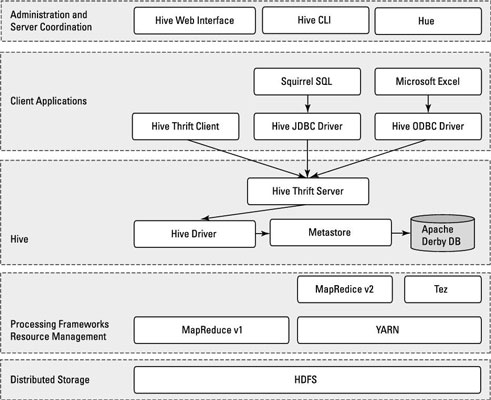
In the case of MapReduce, the figure shows both the Hadoop 1 and Hadoop 2 components. With Hadoop 1, Hive queries are converted to MapReduce code and executed using the MapReduce v1 (MRv1) infrastructure, like the JobTracker and Task Tracker.

With Hadoop 2, YARN has decoupled resource management and scheduling from the MapReduce framework. Hive queries can still be converted to MapReduce code and executed, now with MapReduce v2 (MRv2) and the YARN infrastructure.

There is a new framework under development called Apache Tez, which is designed to improve Hive performance for batch-style queries and support smaller interactive (also known as real-time) queries. At the time of writing, the Apache Tez project is still in incubation, and doesn’t yet have a production-ready release.

If it helps you visualize how all the pieces fit together, think of the HDFS and MapReduce systems as being parts of the Apache Hadoop operating system, with Hive — as well as other components, such as HBase — as higher-level functions or applications.

Moving up the diagram, you find the Hive Driver, which compiles, optimizes, and executes the HiveQL. The Hive Driver may choose to execute HiveQL statements and commands locally or spawn a MapReduce job, depending on the task at hand. The Hive Driver stores table metadata in the metastore and its database.



By default, Hive includes the Apache Derby RDBMS configured with the metastore in what’s called embedded mode. Embedded mode means that the Hive Driver, the metastore, and Apache Derby are all running in one Java Virtual Machine (JVM).

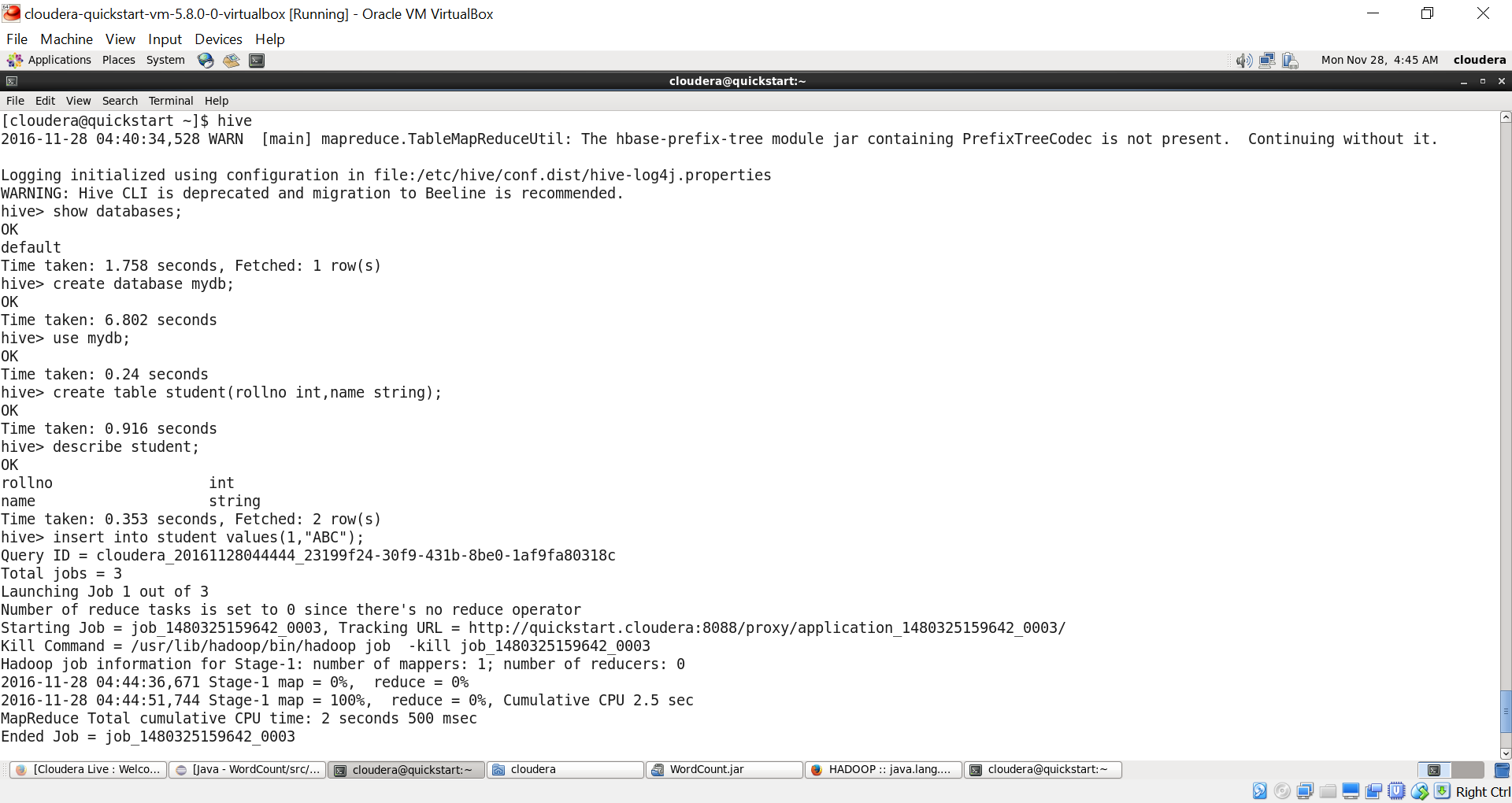
This configuration is fine for learning purposes, but embedded mode can support only a single Hive session, so it normally isn’t used in multi-user production environments. Two other modes exist — local and remote — which can better support multiple Hive sessions in production environments. Also, you can configure any RDBMS that’s compliant with the Java Database Connectivity (JDBC) Application Programming Interface (API) suite. (Examples here include MySQL and DB2.)

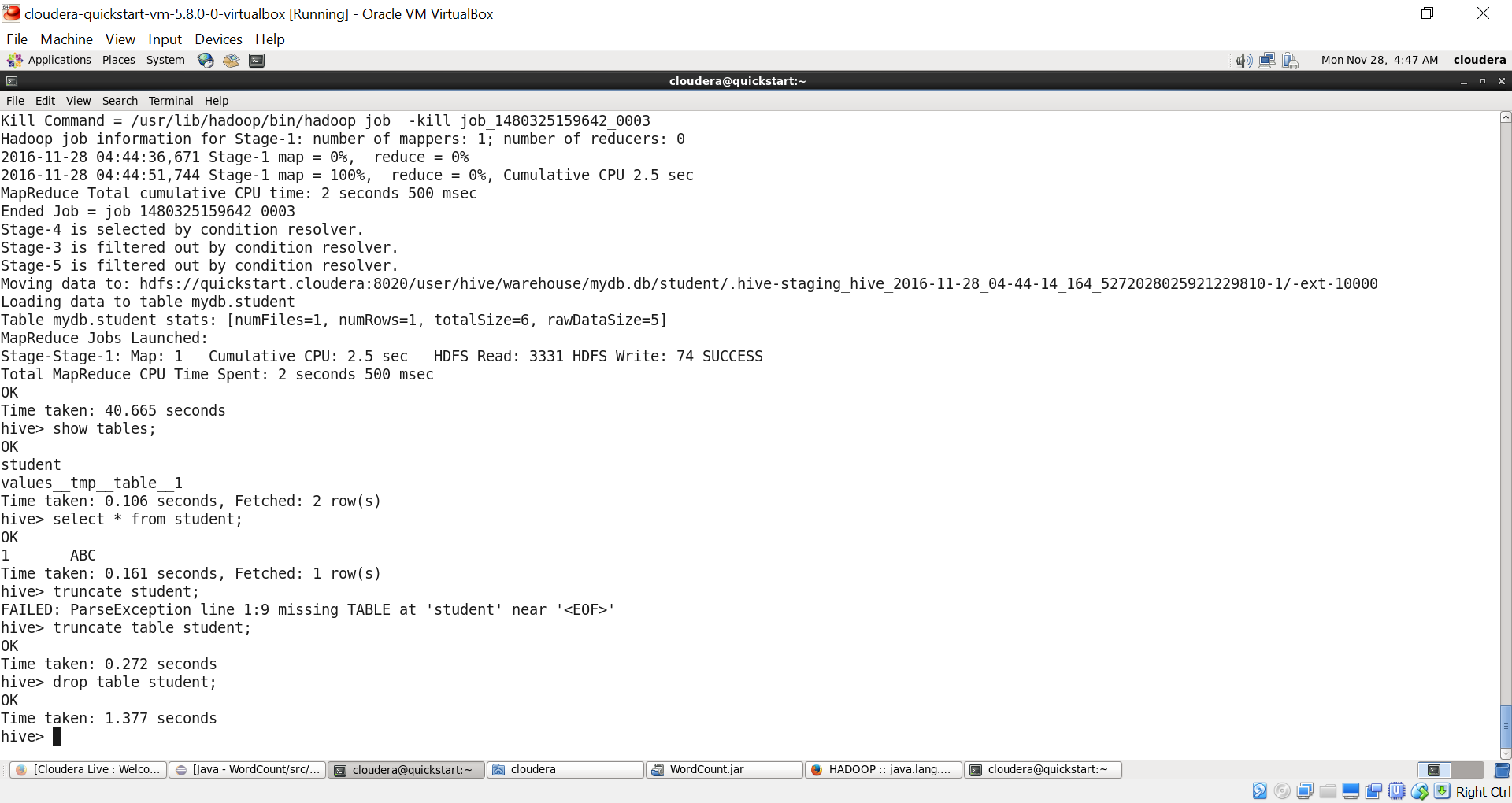
The key to application support is the **Hive Thrift Server**, which enables a rich set of clients to access the Hive subsystem. The open source SQuirreL SQL client is included as an example. The main point is that any JDBC-compliant application can access Hive via the bundled JDBC driver.

The same statement applies to clients compliant with Open Database Connectivity (ODBC) — for example, unixODBC and the isql utility, which are typically bundled with Linux, enable access to Hive from remote Linux clients.

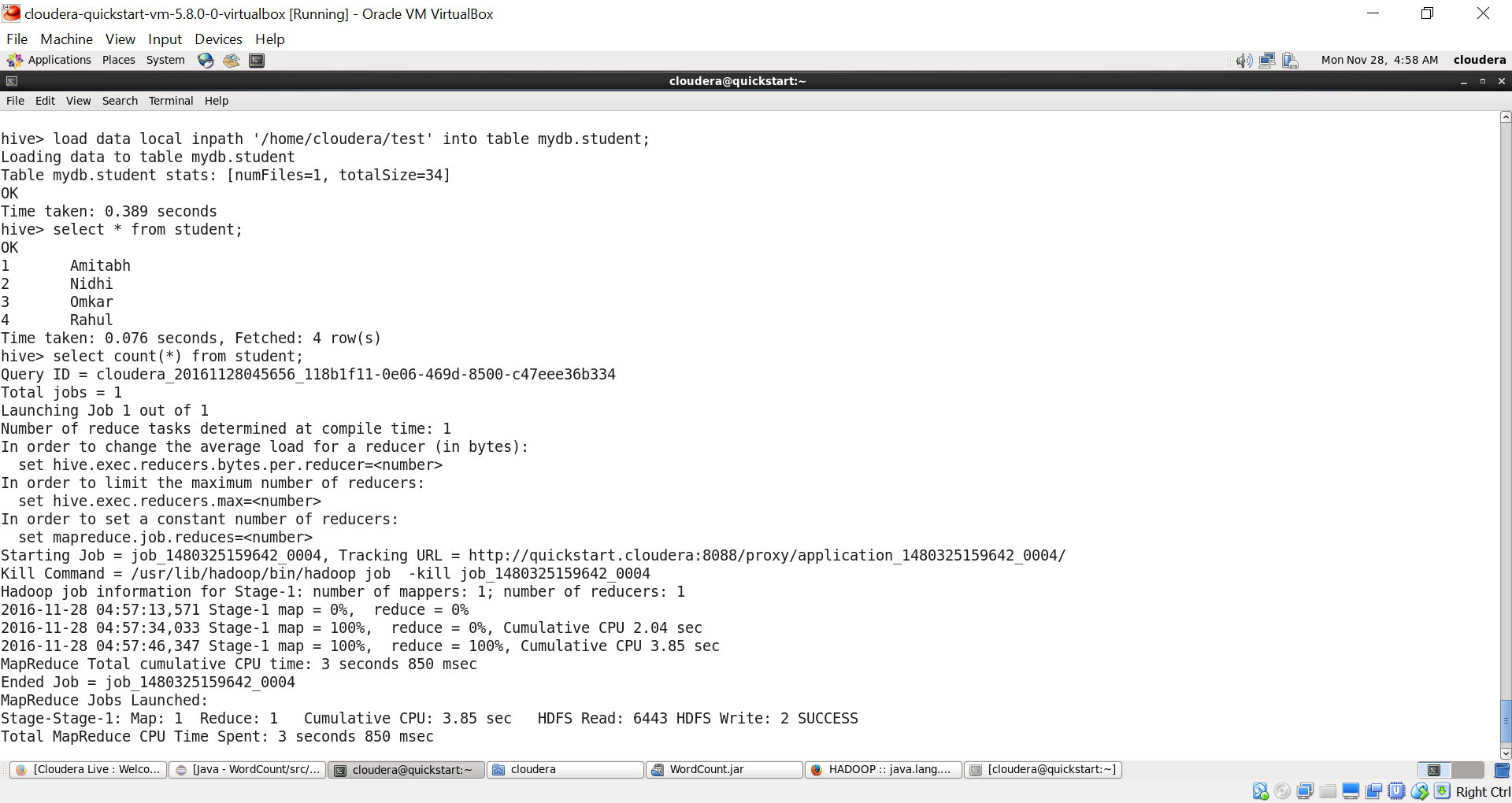
Additionally, if you use Microsoft Excel, you’ll be pleased to know that you can access Hive after you install the Microsoft ODBC driver on your client system. Finally, if you need to access Hive from programming languages other than Java (PHP or Python, for example), Apache Thrift is the answer. Apache Thrift clients connect to Hive via the Hive Thrift Server, just as the JDBC and ODBC clients do.

* **Hands on over HIVE**

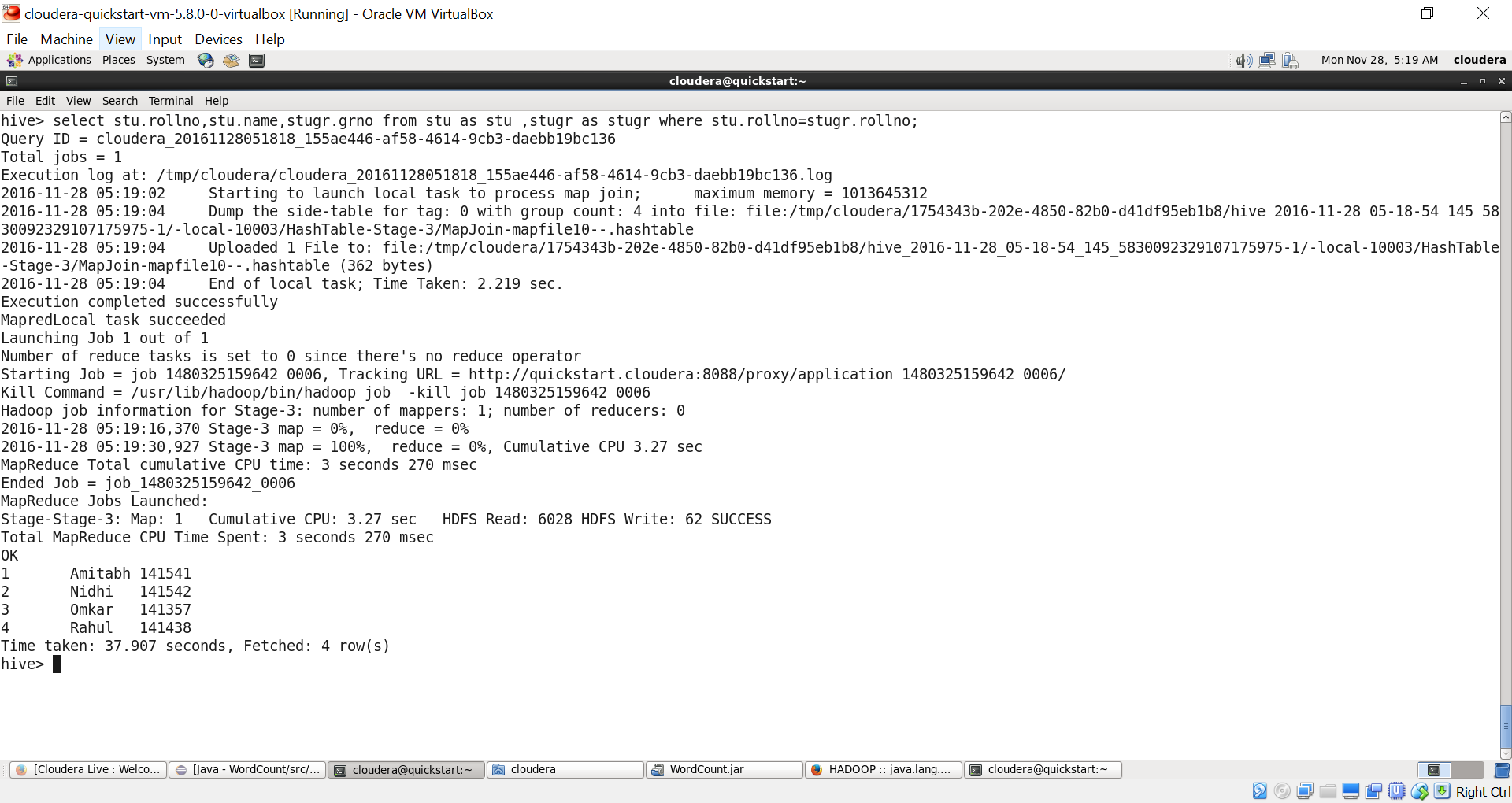


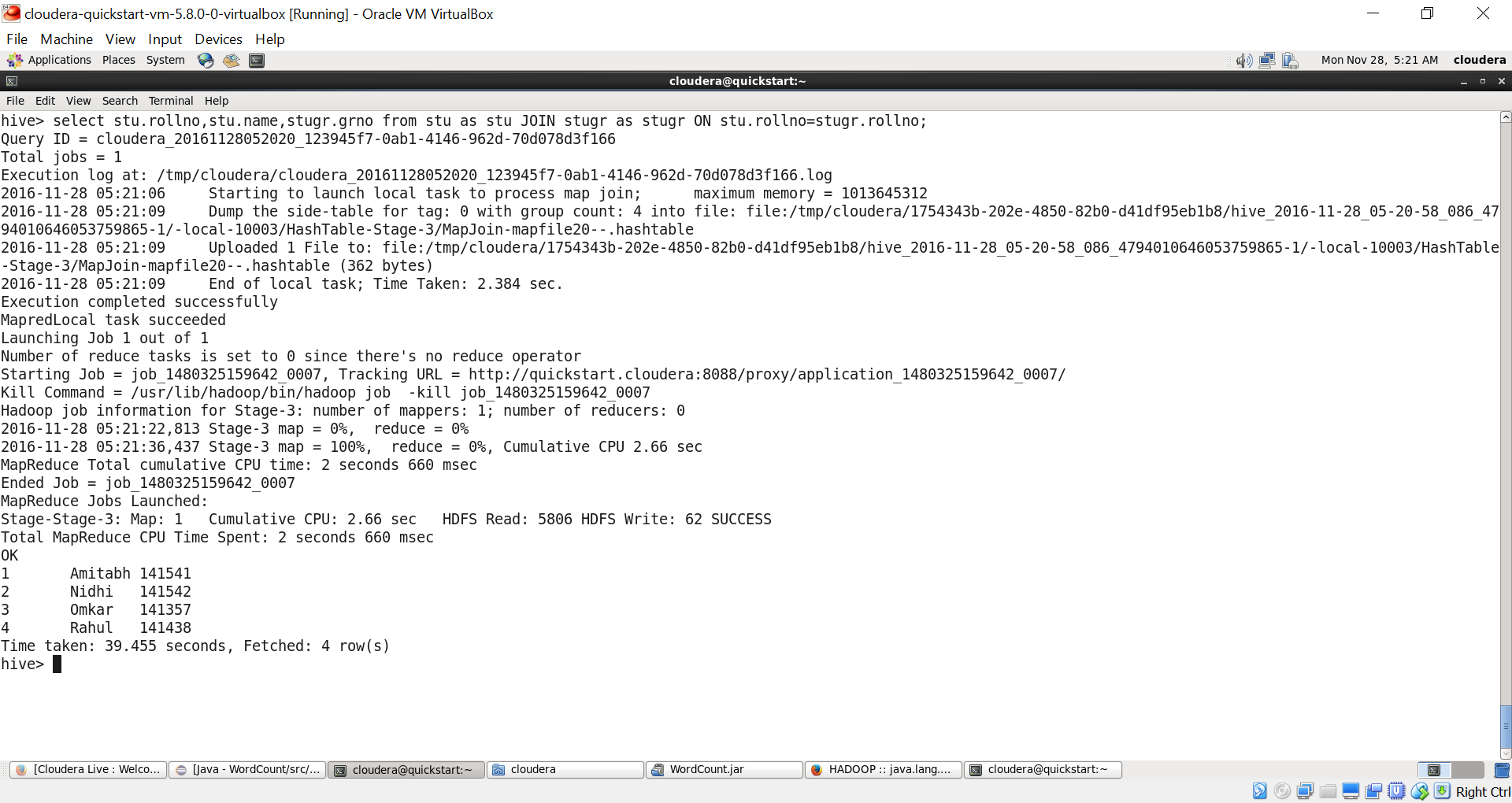


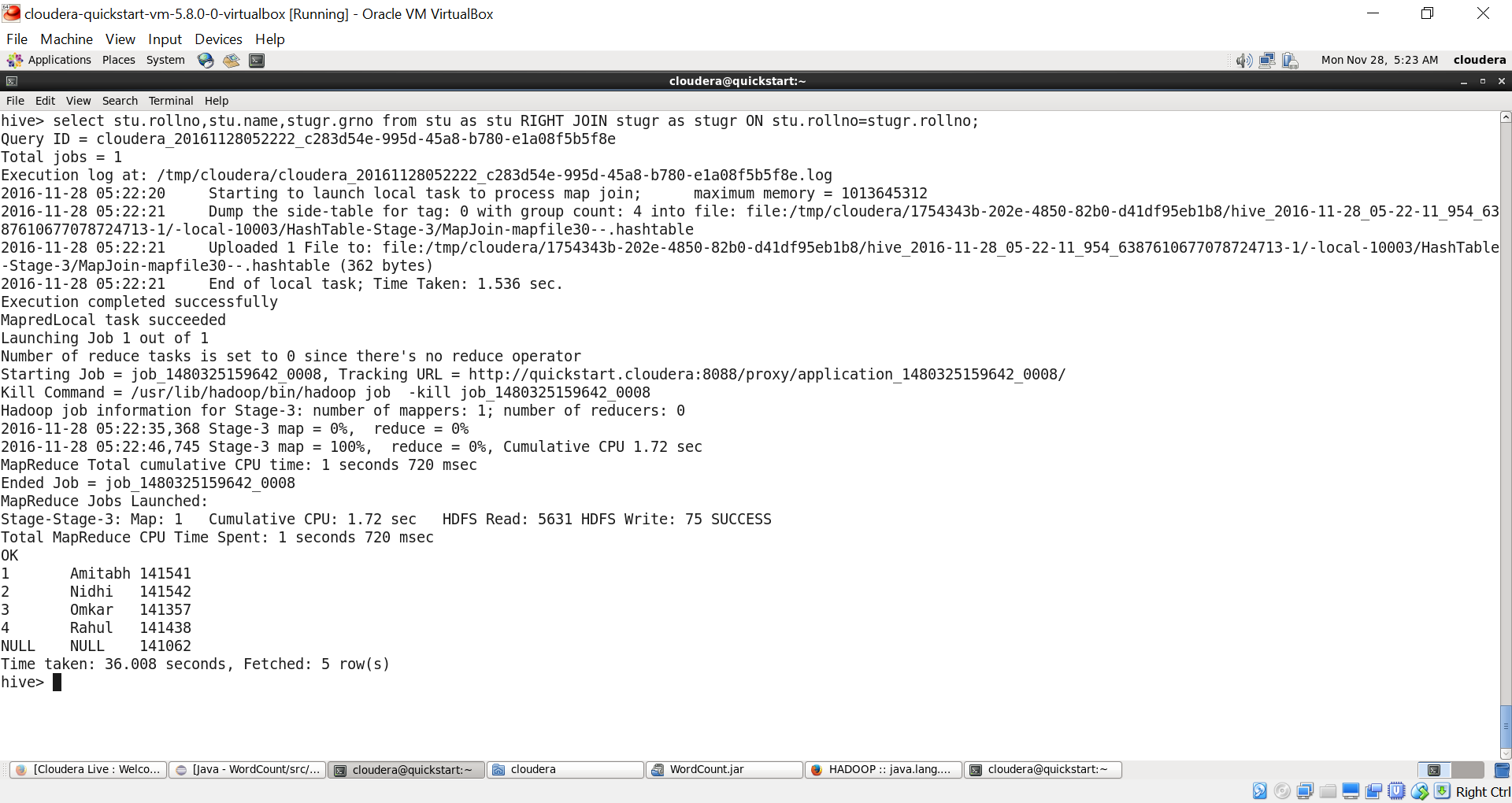
🡪create table student(rollno int,name string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';











* You can write a .hql file and directly run as:

hive –e file.hql