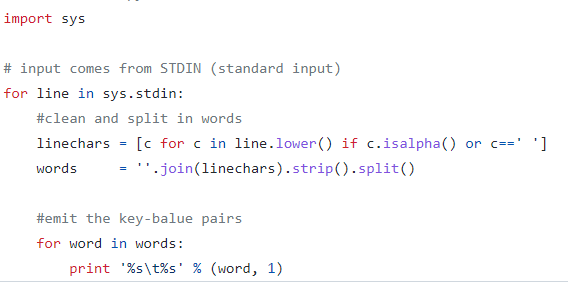
**WEEK -**

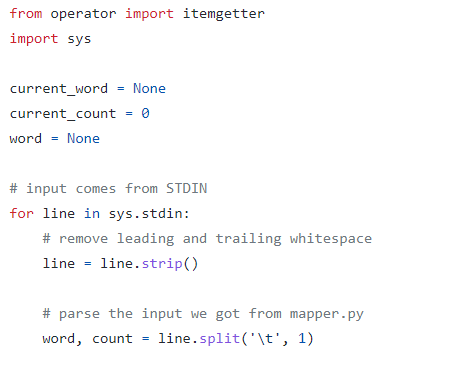
**Aim**: To Implement MapReduce in Python.

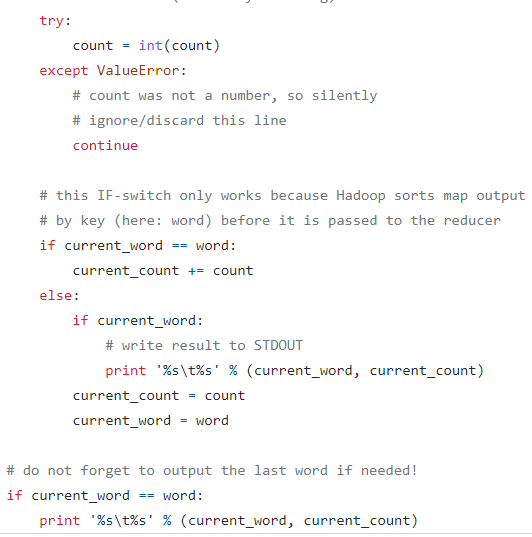
**Program**:

**Mapper.py**



**Reducer.py**



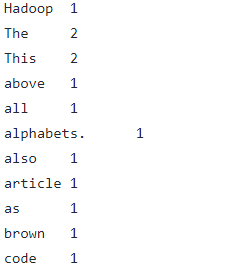


**Command:**

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/share/hadoop/tools/lib/hadoop-streaming-2.3.0.jar \ -mapper ./mapper.py -reducer ./reducer.py -input wordcount-input -output wordcount-mapreduce-streaming-python-output

$HADOOP\_HOME/bin/hadoop fs cat wordcount-mapreduce-streaming-python-output/\*

**Output:**



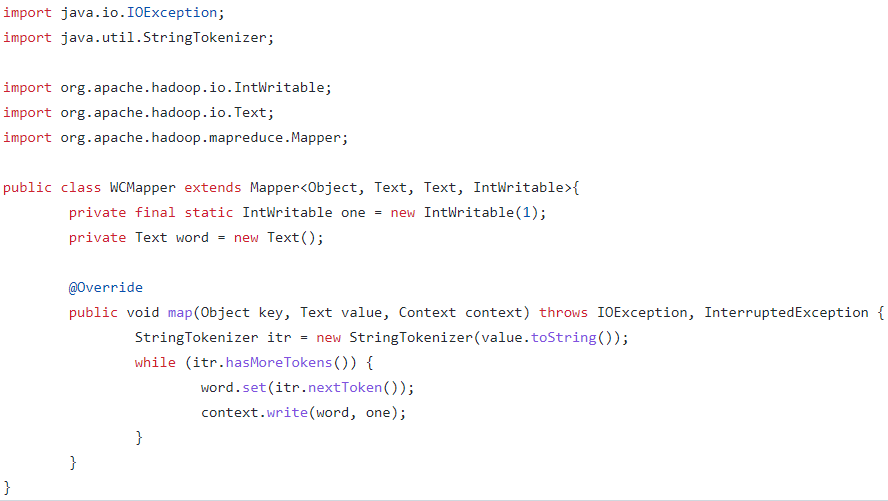
**Result:** Successfully Implemented Word Count using MapReduce in Python.

**WEEK -**

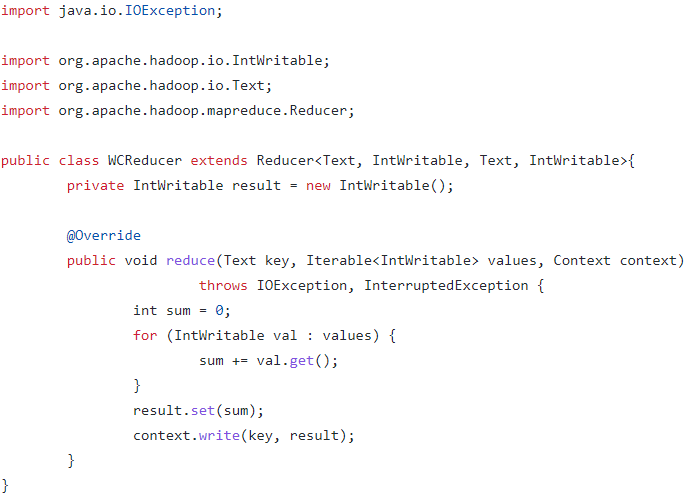
**Aim**: To Implement MapReduce in java.

**Program**:

**Mapper.java**



**Reducer.java**



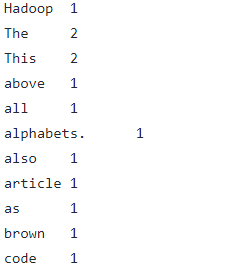
**Driver.java**



**Command:**

hadoop jar wordcount.jar Driver /user/cloudera/words.txt /user/cloudera/output/

**Output:**



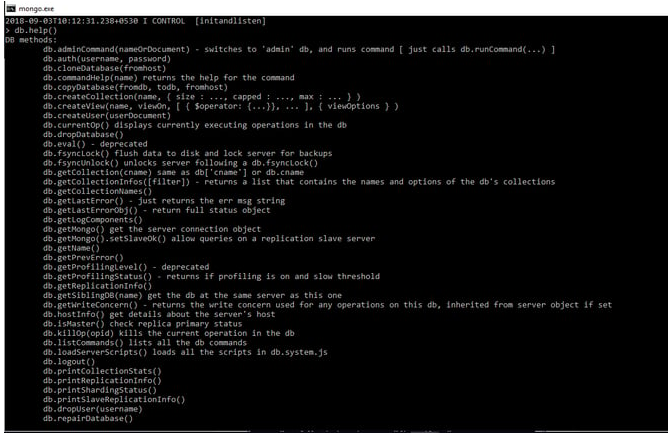
**Result:** Successfully Implemented Word Count using MapReduce in java.

**WEEK -**

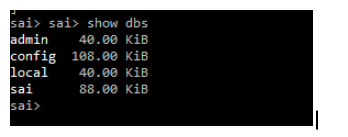
**Aim**: To Implement Mongo Db.

**Commands Below:**

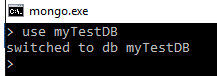
To get a list of commands, type db.help() in MongoDB client. This will give you a list of commands as shown in the following screenshot.  
db.help



**Show All Databases**  
  
Use below command to get list of all databases.  
show dbs



**Create new database**To create a new database execute the following command.  
use DATABASE\_NAME

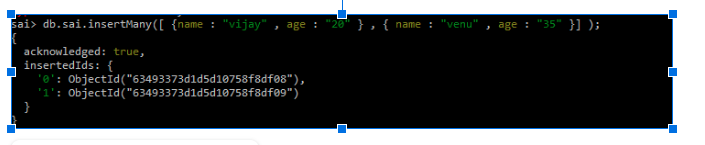


To know your current working/selected database execute the following command  
db

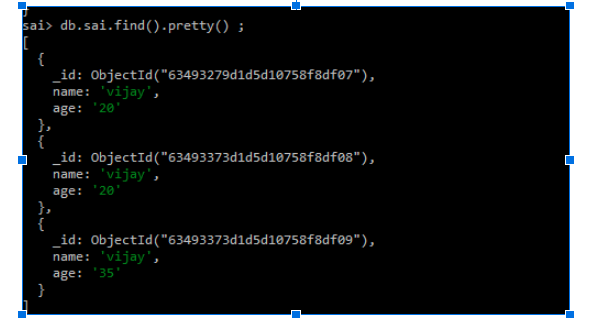


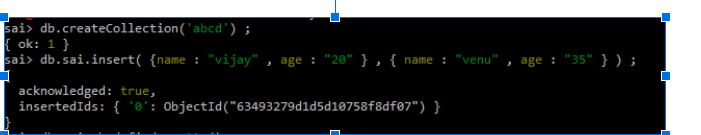
**Drop database**To drop the database execute following command, this will drop the selected database  
db.dropDatabase()  
  


**Insert document in collection**>db.COLLECTION\_NAME.insert(document)

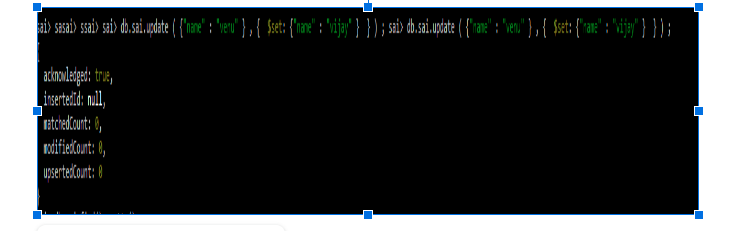


DISPLAY THE RECORDS :





UPDATE THE RECORS :





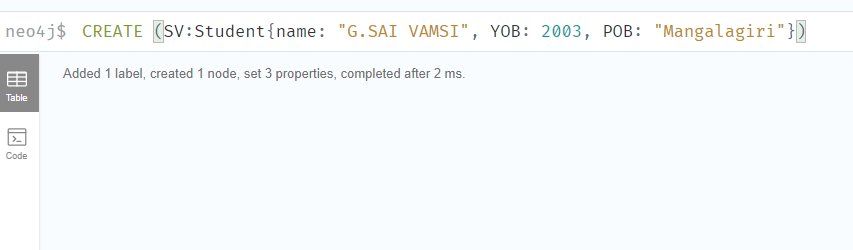
**Result**: Successfully Implemented MongoDB commands.

**WEEK –**

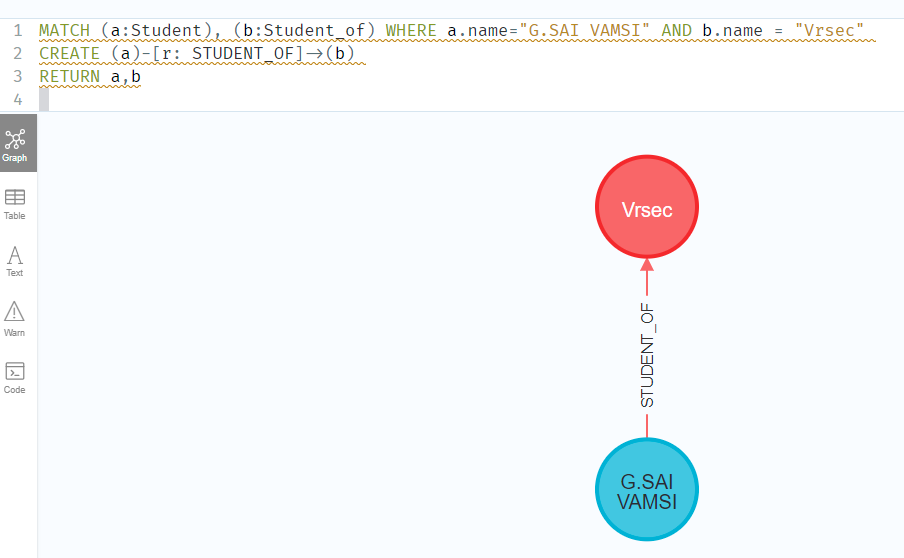
**Aim**: To Implement Neo4J Graph Database.

**Screenshots of execution:**

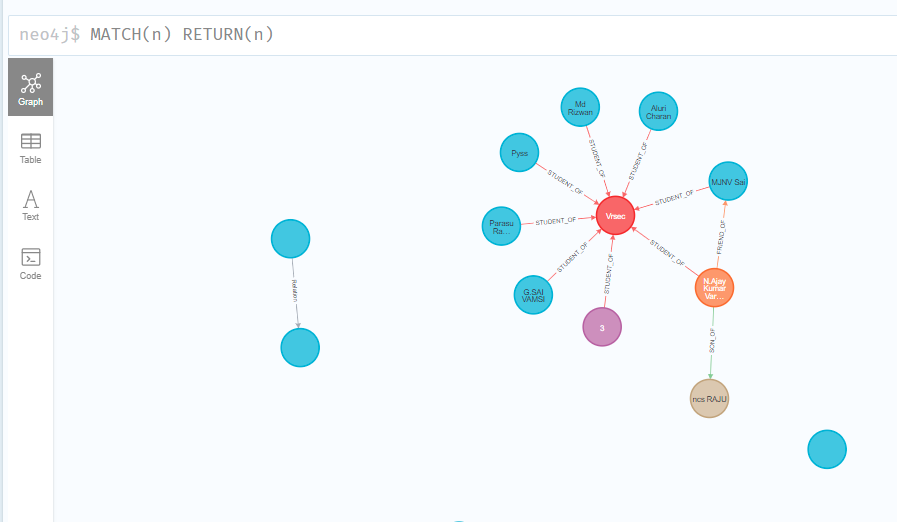
1) Create command



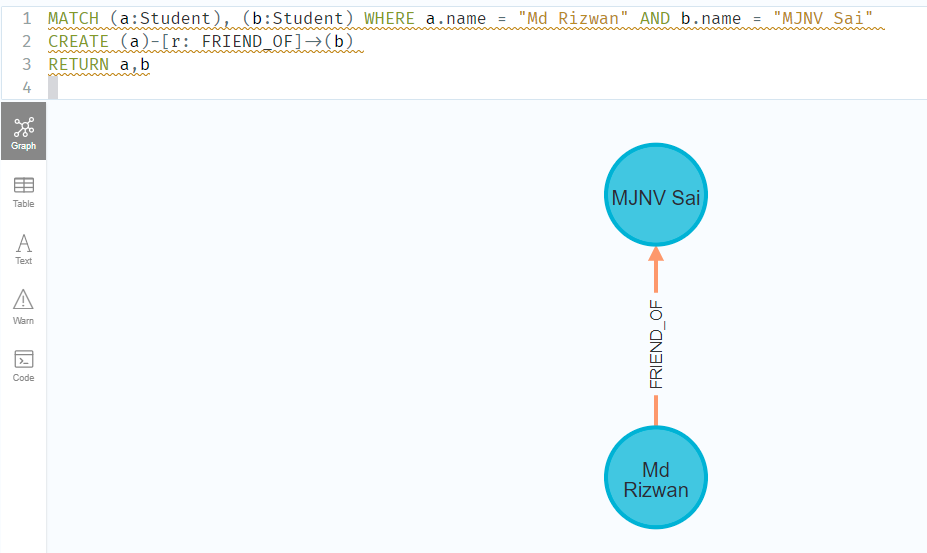
2) Search for some node using match and then creating relations between two nodes (edges)

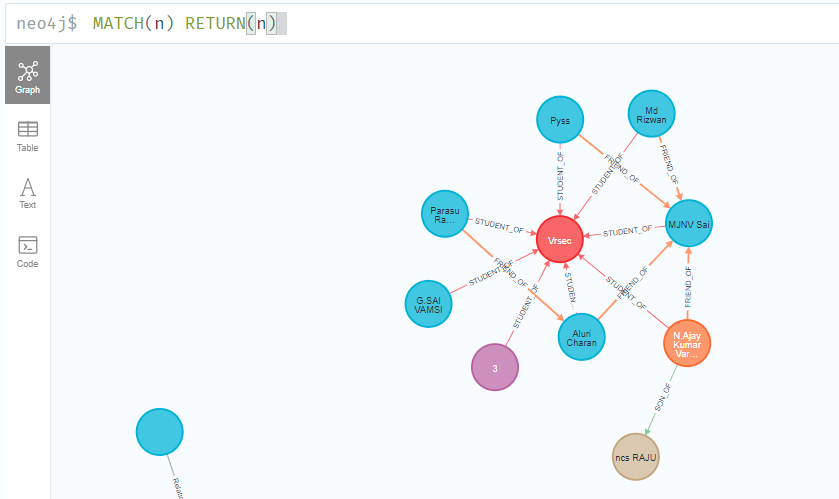


3) MATCH(n) RETURN(n) -> Return to display output graph

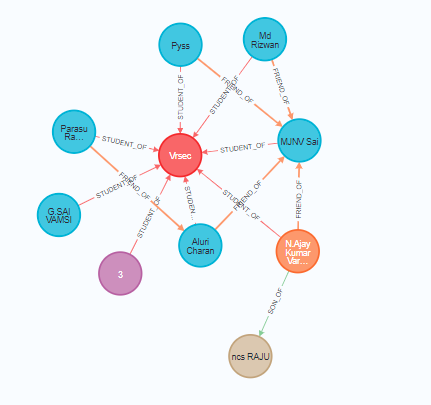


🡪Creating more edges and establishing Relations between them.





**Final Graph Obtained:**



**WEEK -**

**Aim**: To Implement Pig Latin.

**Procedure**:

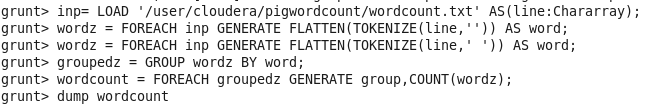
The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.

Pig Latin Simple DataTypes are: int, long, float, double, chararray, bytearray, datetime, Boolean, biginteger, bigdecimal

Pig Latin Complex Data Types: Tuple, Bag, Map

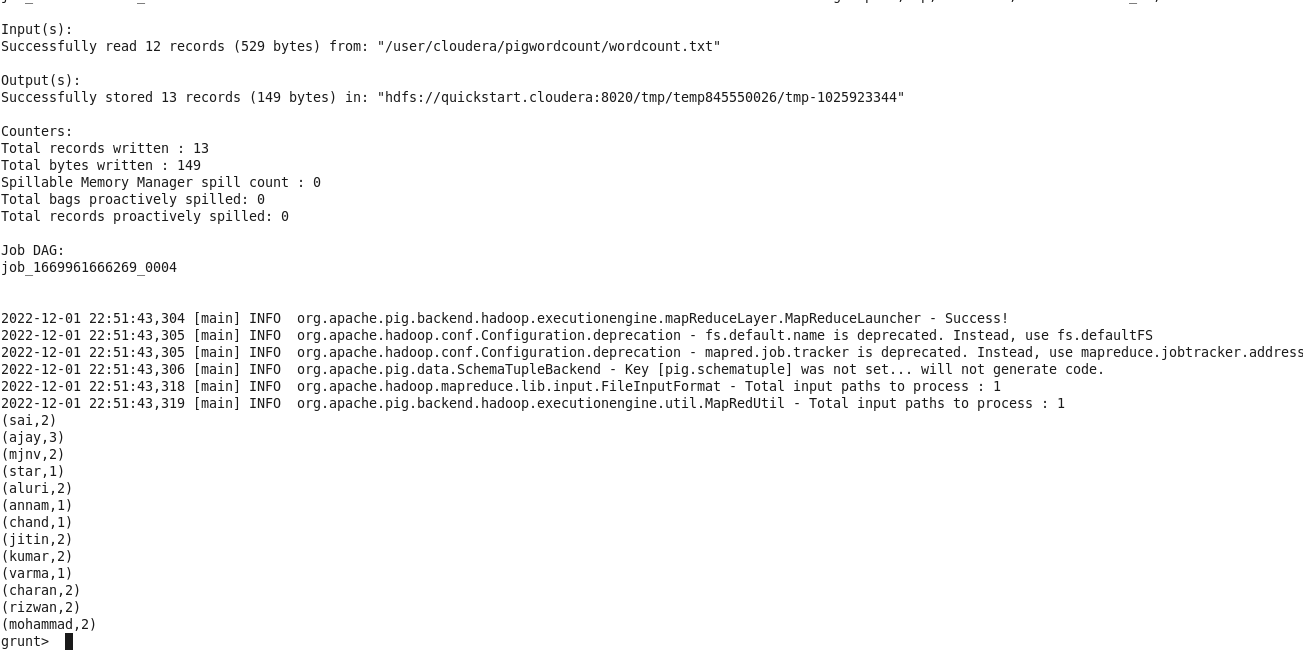
**Word count with pig:**

The Pig Queries and Results are in below Screenshot



**Output:**





Result: Successfully Implemented Word Count using Pig Latin.

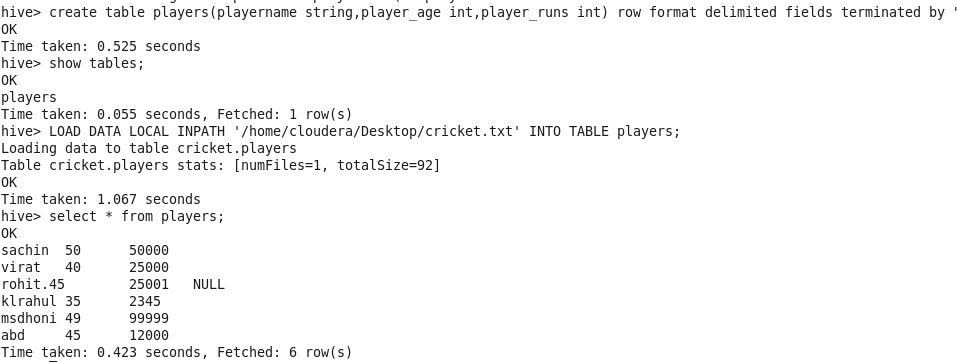
**WEEK -**

**Aim:** To Implement Hive.

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 analyzing easy.

Below are Screenshots of Queries and Execution.

1. Create a new database and then a new table using Hive DDL commands.
2. Loading data with same datatypes from a text or a csv file using below queries.



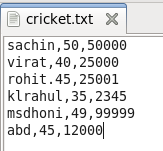
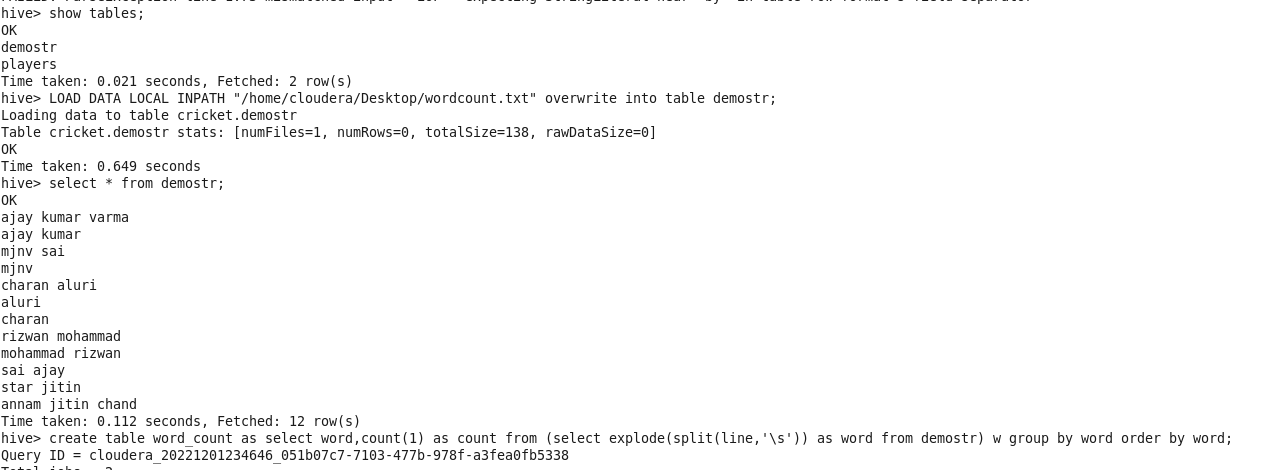


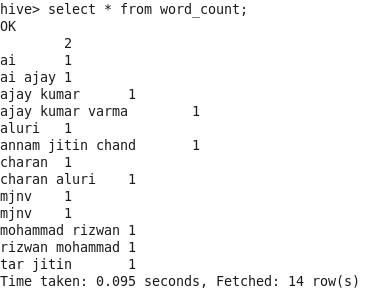
Fig: Text Present inside Text File

**Implementing Word count using Hive**:

Load data to hive and then write query present in below screenshot.



**Output**:



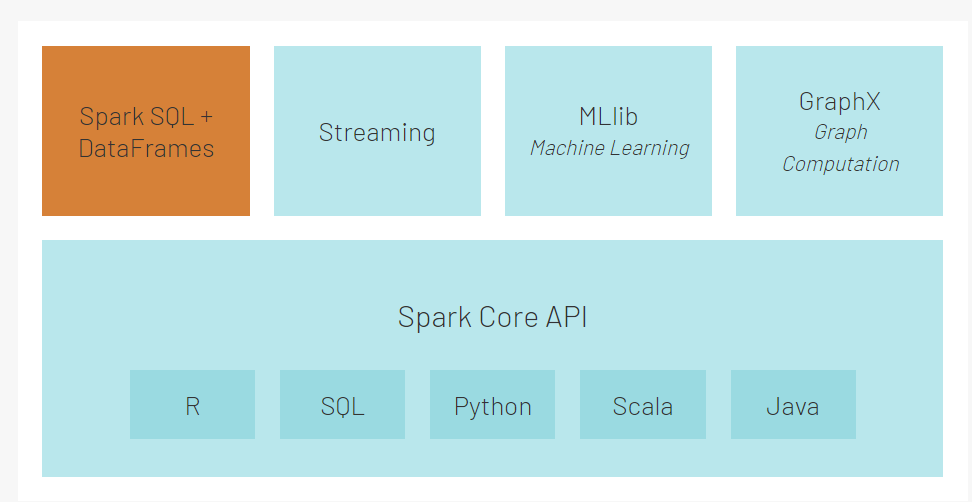
**Result:** Successfully Implemented Hive.

**WEEK -**

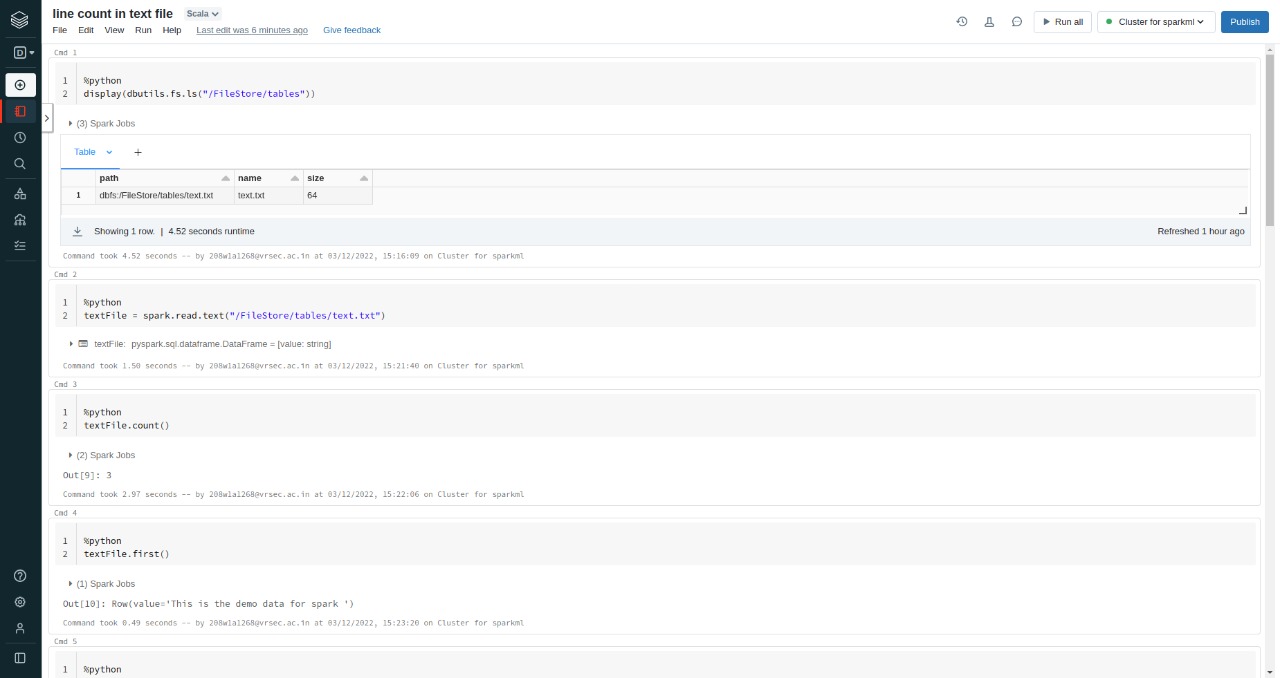
**Aim:** To Implement Apache Spark in Data Bricks.

Apache Spark is a lightning-fast unified analytics engine for big data and machine learning.

**Apache Spark Eco System**



Word Count using Spark in Data Bricks



**Output**:

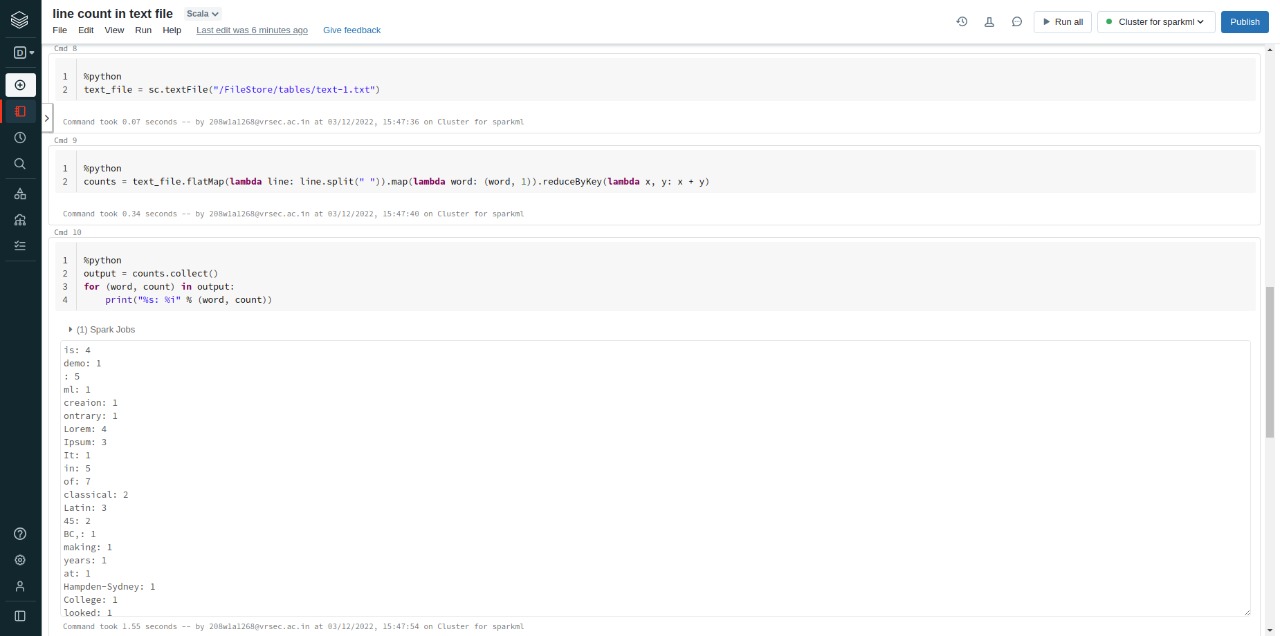
****

Fig: Results of Word count using Spark

**Result:** Successfully Implemented Spark in DataBricks.