| DattaMeghe College of Engineering<br>Airoli, Navi Mumbai | **DEPARTMENT OF COMPUTER ENGINEERING**<br>**ACADEMIC YEAR : 2025 -26 (TERM – I)**<br>**List of Experiments**<br>**Course Name : Subject: Big Data Analytics**<br>**Course Code:CSC702** |
|---|---|

| Sr. No. | Name of the Experiment | Cos Covered | Page No. | Date of Performance | Date of Submission | Marks & Signature |
|---|---|---|---|---|---|---|
| **1** | Study of Hadoop Ecosystem and Hands on Hadoop Commands | CSC702.1 | | | | |
| **2** | a) Installation of Hadoop.<br>b) Implementation of WordCount using MapReduce. | CSC702.2 | | | | |
| **3** | Installation of MongoDB & Execution of queries using MongoDB | CSC702.3 | | | | |
| **4** | a) Installation of Hive & Data aggregation using Hive.<br>b) Installation of Pig & Implementation of Pig Script for displaying contents of file stored in local system. | CSC702.3 | | | | |
| **5** | Implementation of Matrix Multiplication using MapReduce | CSC702.2 | | | | |
| **6** | Implement Bloom Filter and DGIM filter using python | CSC702.4 | | | | |
| **7** | a) Implementation of recommendation engine (CF) using python. | CSC702.5 | | | | |
| **8** | Data visualization using R. | CSC702.6 | | | | |
| **9** | Data import and export in between Mysql and Hive using sqoop | CSC702.3 | | | | |
| **10** | Implementation of Girvan-Newman Algorithm | CSC702.5 | | | | |
| **11** | Mini Project | CSC702.2<br>….<br>CSC702.6 | | | | |
| **12** | Assignment No. 1 | CSC702.1, 2,3 | | | | |
| **13** | Assignment No. 2 | CSC702.4<br>CSC702.5<br>CSC702.6 | | | | |

This is to certify that Mr. / Miss_____ of **SEM VII Class**: _____ **Roll No**. ___has performed the Experiments/ Assignment mentioned above in the premises of the institution.

_____
Practical In charge

# DATTA MEGHE COLLEGE OF ENGINEERING,

# AIROLI, NAVI MUMBAI

## DEPARTMENT OF COMPUTER ENGINEERING

**Institute Vision** : **To create value - based technocrats to fit in the world of work and research**

**Institute Mission** :

- **To adopt the best engineering practices**
- **To empower students to work in the world of technology and research**
- **To create competent human beings**

**Department Vision** : **To provide an intellectually stimulating environment for education, technological excellence in computer engineering field and professional training along with human values.**

**Department Mission:**

**M1: To promote an educational environment that combines academics with intellectual curiosity.**

**M2: To develop human resource with sound knowledge of theory and practical in the discipline of Computer Engineering and the ability to apply the knowledge to the benefit of society at large.**

**M3: To assimilate creative research and new technologies in order to facilitate students to be a lifelong learner who will contribute positively to the economic well-being of the nation.**

**Program Educational Objectives (PEO)**

**PEO1 - To explicate optimal solutions through application of innovative computer science techniques that aid towards betterment of society.**

**PEO2 - To adapt recent emerging technologies for enhancing their career opportunity prospects.**

**PEO3 - To effectively communicate and collaborate as a member or leader in a team to manage multidisciplinary projects**

**PEO4 - To prepare graduates to involve in research, higher studies or to become entrepreneurs in long run.**

**Program Specific Outcomes (PSO)**

**PSO1- To apply basic and advanced computational and logical skills to provide solutions to computer engineering problems**

**PSO2 - Ability to apply standard practices and strategies in design and development of software and hardware based systems and adapt to evolutionary changes in computing to meet the challenges of the future.**

**PSO3 - To develop an approach for lifelong learning and utilize multi-disciplinary knowledge** required for satisfying industry or global requirements.

# Program Outcomes as defined by NBA (PO)

**Engineering Graduates will be able to:**

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

# DATTA MEGHE COLLEGE OF ENGINEERING

## Department of Computer Engineering

**Course Name: Big Data Analytics Lab (R-19)**

**Course Code: CSC702**

**Year of Study: B.E., Semester: VII**

# Course Outcomes

| | |
|---|---|
| **CSC702.1** | Student will be understood the building blocks of Big Data Analytics |
| **CSC702.2** | Student will be applying fundamental enabling techniques like Hadoop and MapReduce in solving real world problems. |
| **CSC702.3** | Student will be understood different NoSQL systems and how it handles big data. |
| **CSC702.4** | Student will be applying advanced techniques for emerging applications like stream analytics |
| **CSC702.5** | Student will be achieved adequate perspectives of big data analytics in various applications like recommender systems, social media applications, etc. |
| **CSC702.6** | Student will be applying statistical computing techniques and graphics for analyzing big data |

# DATTA MEGHE COLLEGE OF ENGINEERING

**DEPARTMENT OF COMPUTER ENGINEERING**

**ACADEMIC YEAR 2025-26 (TERM I)**

**SUBJECT: BIG DATA ANALYTICS**

**SEM: VII**

**RUBRICS FOR GRADING EXPERIMENTS**

| Rubric Number | Rubric Title | Criteria | Marks (out of 15) |
|---|---|---|---|
| R1 | Time Line | On-time | 4 |
| | | Delayed by not more than a Week | 3 |
| | | Delayed more than a Week | 2 |
| R2 | Clarity of Concept | Clear understanding | 4 |
| | | Partially understood | 3 |
| | | Weak understanding | 2 |
| R3 | Identification of problem and objectives | Correct implementation with Results | 4 |
| | | Implementation with some errors | 3 |
| | | Partial implementation | 2 |
| R4 | Documentation | Correct Documentation | 3 |
| | | Not documented properly | 2 |

# DATTA MEGHE COLLEGE OF ENGINEERING

**DEPARTMENT OF COMPUTER ENGINEERING**

**ACADEMIC YEAR 2025-26 (TERM I)**

**SUBJECT: BIG DATA ANALYTICS**

**SEM: VII**

**RUBRICS FOR GRADING MINI PROJECT**

| Rubric Number | Rubric Title | Criteria | Marks (out of 15) |
|---|---|---|---|
| R1 | Time Line | On-time | 4 |
| | | Delayed by not more than a Week | 3 |
| | | Delayed more than a Week | 2 |
| R2 | Clarity of Concept | Clear understanding | 4 |
| | | Partially understood | 3 |
| | | Weak understanding | 2 |
| R3 | Identification of problem and objectives | Correct implementation with Results | 4 |
| | | Implementation with some errors | 3 |
| | | Partial implementation | 2 |
| R4 | Documentation | Correct Documentation | 3 |
| | | Not documented properly | 2 |

# DATTA MEGHE COLLEGE OF ENGINEERING

**DEPARTMENT OF COMPUTER ENGINEERING**

**ACADEMIC YEAR 2025-26 (TERM I)**

**SUBJECT: BIG DATA ANALYTICS**

**SEM: VII**

## RUBRICS FOR GRADING ASSIGNMENT

| Rubric Number | Rubric Title | Criteria | Marks (out of 05) |
|---|---|---|---|
| R1 | Timeline | On-time | 2 |
| | | Delayed | 1 |
| R2 | Knowledge | Able to answer all questions | 2 |
| | | Able to answer some of the questions | 1 |
| R3 | Neatness | Well Written | 1 |
| | | Fairly Written | 0 |

## EXPERIMENT NUMBER: 1

**Date of Performance :**

**Date of Submission :**

**Aim:** Study of Hadoop Ecosystem and Hands on Hadoop Commands.

### Objectives:

- To introduce the tools required to manage and analyze big data
- To be familiar with the open source framework like Hadoop and features and tools of it.

**Software Used:** Hadoop

### Theory:

**Hadoop** is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

**Hadoop Architecture:**

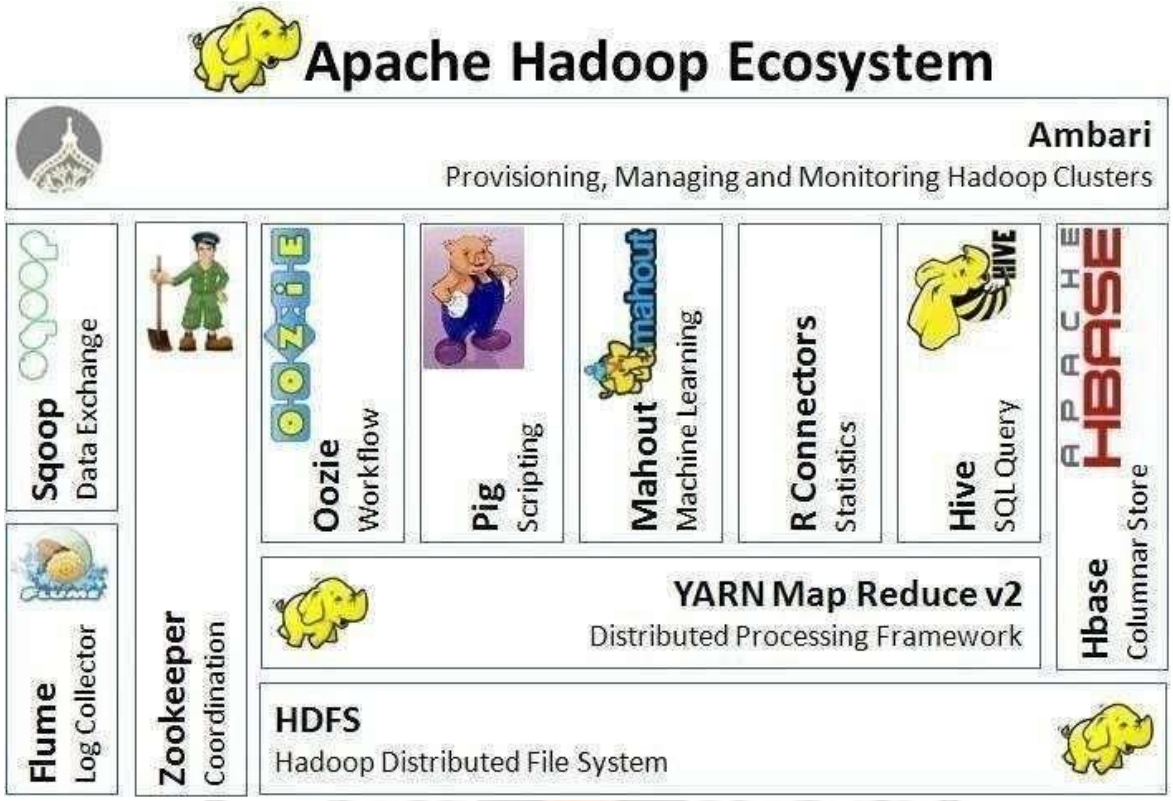
The Apache Hadoop framework includes following four modules:

**Hadoop Common**: Contains Java libraries and utilities needed by other Hadoop modules. These libraries give file system and OS level abstraction and comprise of the essential Java files and scripts that are required to start Hadoop.

**Hadoop Distributed File System (HDFS):** A distributed file-system that provides high-throughput access to application data on the community machines thus providing very high aggregate bandwidth across the cluster.

**Hadoop YARN:** A resource-management framework responsible for job scheduling and cluster resource management.

**Hadoop MapReduce:** This is a YARN- based programming model for parallel processing of large data sets.




Hadoop has gained its popularity due to its ability of storing, analyzing and accessing large amount of data, quickly and cost effectively through clusters of commodity hardware. It won't be wrong if we say that Apache Hadoop is actually a collection of several components and not just a single product.

With Hadoop Ecosystem there are several commercial along with an open source products which are broadly used to make Hadoop laymen accessible and more usable.

**MapReduce**

Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner. In terms of programming, there are **two functions** which are most common in MapReduce.

- **The Map Task:** Master computer or node takes input and convert it into divide it into smaller parts and distribute it on other worker nodes. All worker nodes solve their own small problem and give answer to the master node.
- **The Reduce Task:** Master node combines all answers coming from worker node and forms it in some form of output which is answer of our big distributed problem.

Generally both the input and the output are reserved in a file-system. The framework is responsible for scheduling tasks, monitoring them and even re-executes the failed tasks.

**Hadoop Distributed File System (HDFS)**

HDFS is a distributed file-system that provides high throughput access to data. When data is pushed to HDFS, it automatically splits up into multiple blocks and stores/replicates the data thus ensuring high availability and fault tolerance.

***Note: A file consists of many blocks (large blocks of 64MB and above).***

Here are the **main components of HDFS:**

- **Name Node:** It acts as the master of the system. It maintains the name system i.e., directories and files and manages the blocks which are present on the Data Nodes.
- **Data Nodes:** They are the slaves which are deployed on each machine and provide the actual storage. They are responsible for serving read and write requests for the clients.
- **Secondary Name Node:** It is responsible for performing periodic checkpoints. In the event of Name Node failure, you can restart the Name Node using the checkpoint.

**Hive**

Hive is part of the Hadoop ecosystem and provides an SQL like interface to Hadoop. It is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems.

It provides a mechanism to project structure onto this data and query the data using a SQL-like language called Hive QL. Hive also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

The **main building blocks of Hive are –**

1. **Metastore** – To store metadata about columns, partition and system catalogue.
2. **Driver** – To manage the lifecycle of a HiveQL statement
3. **Query Compiler** – To compiles HiveQL into a directed acyclic graph.
4. **Execution Engine** – To execute the tasks in proper order which are produced by the compiler.
5. **HiveServer** – To provide a Thrift interface and a JDBC / ODBC server.

**HBase (Hadoop DataBase)**

HBase is a distributed, column oriented database and uses HDFS for the underlying storage. As said earlier, HDFS works on write once and read many times pattern, but this isn't a case always. We may require real time read/write random access for huge dataset; this is where HBase comes into the picture. HBase is built on top of HDFS and distributed on column-oriented database.

Here are the **main components of HBase:**

- **HBase Master:** It is responsible for negotiating load balancing across all Region Servers and maintains the state of the cluster. It is not part of the actual data storage or retrieval path.
- **Region Server:** It is deployed on each machine and hosts data and processes I/O requests.

**Zookeeper**

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization and providing group services which are very useful for a variety of distributed systems. HBase is not operational without ZooKeeper.

**Mahout**

Mahout is a scalable machine learning library that implements various different approaches machine learning. At present Mahout contains four **main groups of algorithms:**

- Recommendations, also known as collective filtering
- Classifications, also known as categorization
- Clustering
- Frequent item set mining, also known as parallel frequent pattern mining

Algorithms in the Mahout library belong to the subset that can be executed in a distributed fashion and have been written to be executable in MapReduce. Mahout is scalable along three dimensions: It scales to reasonably large data sets by leveraging algorithm properties or implementing versions based on Apache Hadoop.

**Sqoop (SQL-to-Hadoop)**

Sqoop is a tool designed for efficiently transferring structured data from SQL Server and SQL Azure to HDFS and then uses it in MapReduce and Hive jobs. One can even use Sqoop to move data from HDFS to SQL Server.

**Apache Spark:**

Apache Spark is a general compute engine that offers fast data analysis on a large scale. Spark is built on HDFS but bypasses MapReduce and instead uses its own data processing framework. Common uses cases for Apache Spark include real-time queries, event stream processing, iterative algorithms, complex operations and machine learning.

**Pig**

Pig is a platform for analyzing and querying huge data sets that consist of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. Pig's built-in operations can make sense of semi-structured data, such as log files, and the language is extensible using Java to add support for custom data types and transformations.

The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets. At the present

time, Pig's infrastructure layer consists of a compiler that produces sequences of MapReduce programs.

**Oozie**

Apache Oozie is a workflow/coordination system to manage Hadoop jobs.

**Flume**

Flume is a framework for harvesting, aggregating and moving huge amounts of log data or text files in and out of Hadoop. Agents are populated throughout ones IT infrastructure inside web servers, application servers and mobile devices. Flume itself has a query processing engine, so it's easy to transform each new batch of data before it is shuttled to the intended sink.

**Ambari**

Ambari was created to help manage Hadoop. It offers support for many of the tools in the Hadoop ecosystem including Hive, HBase, Pig, Sqoop and Zookeeper. The tool features a management dashboard that keeps track of cluster health and can help diagnose performance issues.

## Sample HDFS commands:

hadoop version

hdfs dfs -mkdir /test

hdfs dfs -put ~/hadoop/README.txt /test

hdfs dfs -ls /test

hdfs dfs -cat /test/README.txt

hdfs dfs -copyToLocal /test/README.txt ~/Desktop

hdfs dfs -rm /test/README.txt

hdfs dfs -rmdir /test

hdfs dfsadmin -report

jps

**Output:**

```
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /home/jinit/hadoop/share/hadoop/common/hadoop-common-3.3.6.jar
Found 1 items
-rw-r--r--   1 jinit supergroup        175 2025-07-27 14:38 /test/README.txt
For the latest information about Hadoop, please visit our website at:

   http://hadoop.apache.org/

and our wiki, at:

   https://cwiki.apache.org/confluence/display/HADOOP/
copyToLocal: `/home/jinit/Desktop': File exists
Deleted /test/README.txt
Configured Capacity: 1081101176832 (1006.85 GB)
Present Capacity: 1021699207354 (951.53 GB)
DFS Remaining: 1021699182592 (951.53 GB)
DFS Used: 24762 (24.18 KB)
DFS Used%: 0.00%
Replicated Blocks:
        Under replicated blocks: 0
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0
Erasure Coded Block Groups:
        Low redundancy block groups: 0
        Block groups with corrupt internal blocks: 0
        Missing block groups: 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0
```

```
Live datanodes (1):

Name: 127.0.0.1:9866 (localhost)
Hostname: Pratham-Patade
Decommission Status : Normal
Configured Capacity: 1081101176832 (1006.85 GB)
DFS Used: 36864 (36 KB)
Non DFS Used: 4411285504 (4.11 GB)
DFS Remaining: 1021697499136 (951.53 GB)
DFS Used%: 0.00%
DFS Remaining%: 94.51%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 0
Last contact: Sun Jul 27 14:51:51 UTC 2025
Last Block Report: Sun Jul 27 14:50:57 UTC 2025
Num of Blocks: 0


8624 ResourceManager
8752 NodeManager
9280 Jps
8354 SecondaryNameNode
7989 NameNode
8126 DataNode
```

**Conclusion:**

**Thus, we have studied Hadoop Ecosystem and Hands on Hadoop Commands.**

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 2

**Date of Performance :**

**Date of Submission :**

**Aim:** Write a Program To Implement Word Count Program Using MapReduce.

**Objective**:

- To learn the key issues in big data management and its tools and techniques, specifically programming module of Hadoop.
- To understand the working of map-reduce programming.
- To understand the use of map-reduce for big data analytics.

**Software Used:** Virtual Machine, Cloudera, eclipse

**Theory:**

### a) MapReduce

**MapReduce** is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map andReduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller setof tuples. As the sequence of the nameMapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change.

MapReduce program executes in three stages, namely map stage, shuffle stage, and reducestage.

1. **Map stage**: The map or mapper's job is to process the input data. Generally the inputdata is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

2. **Reduce stage**: This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriateservers in the cluster. The framework manages all the details of data- passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes. Most of the computing takes place on nodes with data on local disks that reduces the network traffic. After completion of the given tasks, the cluster collects and reduces the data to form an appropriate

result, and sends it back to the Hadoop server.

**MAP-REDUCE Working**

The MapReduce framework operates on <key, value> pairs, that is, the framework views theinput to the job as a set of <key, value> pairs and produces a set of <key, value> pairs asthe output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classeshave to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output typesof a MapReduce job: (Input) <k1, v1> -> map -
> <k2, v2>-> reduce -> <k3, v3>(Output).

## Algorithm

map(key, value):
for each word w in value: emit(w, 1)

reduce(key,values):

result = 0

for each count v invalues:result+=values emit(key, result)

## Program:

*******WordCount.java**********

```
import java.io.IOException; import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job; import org.apache.hadoop.mapreduce.Mapper;import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{
private final static IntWritable one = new IntWritable(1);private Text word = new Text();
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException { StringTokenizer itr = new
StringTokenizer(value.toString());while (itr.hasMoreTokens()) {
word.set(itr.nextToken());context.write(word, one);}}}
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {private IntWritable result = new
IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,Context context
) throws IOException, InterruptedException {int sum = 0;
for (IntWritable val : values) {sum += val.get();}
result.set(sum); context.write(key, result);}}
public static void main(String[] args) throws Exception {Configuration conf = new
```

Configuration();
Job job = new Job (conf, "word count")
job.setJarByClass(WordCount.class); job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class); job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0])); FileOutputFormat.setOutputPath(job, new Path(args[1]));System.exit(job.waitForCompletion(true) ? 0 : 1);}}
hadoop fs -rm -r /input hadoop fs -rm -r /output hadoop fs -mkdir /input hadoop fs -put input.txt /input
javac -Xlint -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-3.21.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar WordCount.java
jar cvf WordCount.jar *.class
hadoop jar WordCount.jar WordCount /input /outputhadoop fs -cat /output/part-r-00000

**On virtual machine and Cloudera**

Install Virtual machine
1. Open Cloudera on virtual machine
2. Open eclipse
3. Create new java project and create new WordCount class
4. Paste wordcount program in file
6. Right click on project-> build path->libraries-> open hadoop-> add all jar files
7. Check error free code
8. Right click on class and export jar file on desktop of virtual machine
9. on terminal, click below command to upload input text file to hadoop directories and run jar file to see output in output file.

hadoop fs -rm -r training/input
hadoop fs -rm -r training/output
hadoop fs -mkdir training/input
hadoop fs -put input.txt
training/input
hadoop jar home/training/Desktop/WordCount.jar WordCount user/training/input
/outputhadoop fs -cat /output/part-r-00000

## Output:

```
[training@localhost ~]$ hadoop fs -ls MRDir1

Found 3 items

-rw-r--r--   1 training supergroup          0 2023-08-08 03:36 /user/training/MRDir1/_SUCCESS
drwxr-xr-x   - training supergroup          0 2023-08-08 03:36 /user/training/MRDir1/_logs
-rw-r--r--   1 training supergroup         20 2023-08-08 03:36 /user/training/MRDir1/part-r-00000
```

```
[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000
BUS     7
CAR     4
TRAIN   6
```

**Conclusion: Thus installed Hadoop and ran Wordcount program using MapReduce.**

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 3

**Date of Performance :**

**Date of Submission :**

**Aim:** Install and configure MongoDB and Execute NoSQL Commands

**Objectives:**

To learn installation of MongoDB to execute NoSQL command

**Software Used:** MongoDB

**Theory:**

**Follow these steps to install MongoDB Enterprise Edition using the apt package manager.**

1. Import the public key used by the package management system.
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add

2. Create a /etc/apt/sources.list.d/mongodb-enterprise.list file for MongoDB.

echo "deb [ arch=amd64,arm64,s390x ] http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list

3. Reload local package

database.sudo apt-get update

4. Install the MongoDB Enterprise

packages.sudo apt-get install -y mongodb-enterprise

Use the initialization system appropriate for your platform:

1. start MongoDB

   sudo systemctl start mongod

2. Verify that MongoDB has started successfully

   sudo systemctl enable mongod

3. Stop MongoDB.

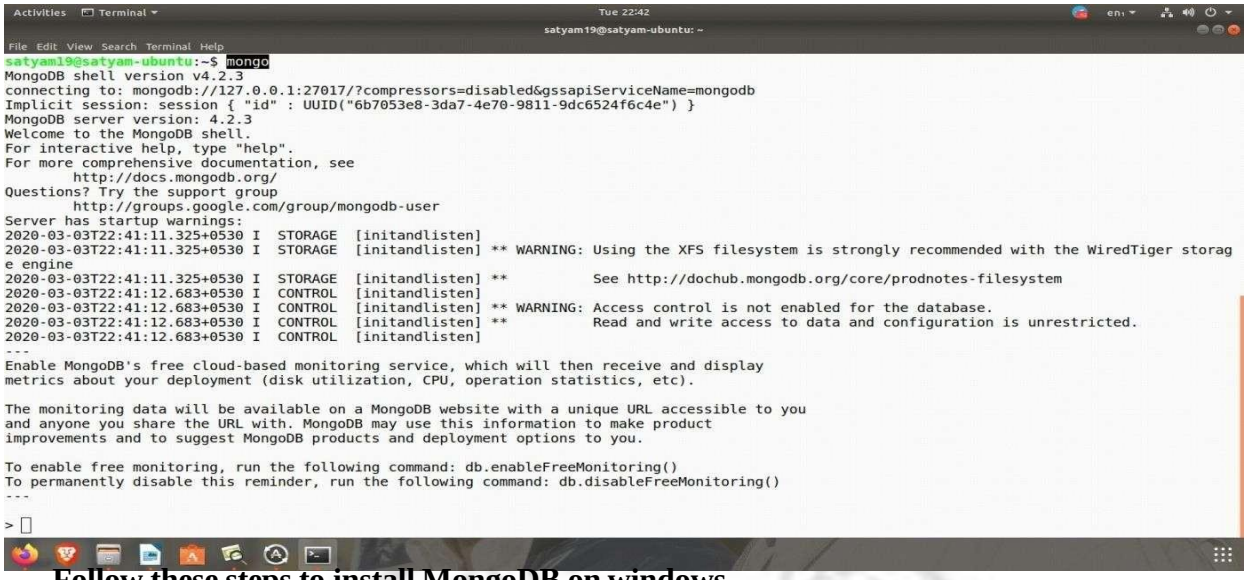
   sudo systemctl stop mongod

4. Restart MongoDB

   sudo systemctl restart mongo

5. Begin using MongoDB

   mongo




**Follow these steps to install MongoDB on windows**

1) Create the data directory where MongoDB stores data. MongoDB's default data directory path is the absolute path \data\db on the drive from which you start MongoDB.
2) Download MongoDB setup file. In zip folder extract mongod and mongo setup file.
3) In command prompt run mongod.exe command
4) Open second command prompt and run mongo.exe command
5) MongoDB is connected on first command window.

**Command:** Implement different data manipulation operation like show database, create collection, insert and delete data from collection, read conditional data using MongoDB NoSQL commands.

**Output**:

```
test> use university
switched to db university
university> db.createCollection("students")
{ ok: 1 }
university> db.students.insertOne({name:"Alice",age:22,course:"CS"})
{
  acknowledged: true,
  insertedId: ObjectId('688e175dfbc3cf4400eec4a9')
}
university> db.students.find()
[
  {
    _id: ObjectId('688e175dfbc3cf4400eec4a9'),
    name: 'Alice',
    age: 22,
    course: 'CS'
  }
]
university> db.students.insertMany([{name:"Bob",age: 24, course:"IT"},{name:"Charlie",age: 21,course: "AI"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('688e18fefbc3cf4400eec4aa'),
    '1': ObjectId('688e18fefbc3cf4400eec4ab')
  }
}
university> db.students.updateOne({name:"Charlie"},{$set:{age: 23}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
university> db.students.deleteOne({name: "Charlie"})
{ acknowledged: true, deletedCount: 1 }
university>
```

**Conclusion: Thus, we have successfully installed and configured MongoDB and Executed NoSQL Commands.**

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 4

**Date of Performance :**

**Date of Submission :**

**Aim:** a) Installation of Hive & Data aggregation using Hive.

b) Installation of Pig & Implementation of Pig Script for displaying contents of file stored in local system.

**Objectives:**

To learn different technique for data aggregation using Hive and displaying content of files stored in local system using Pig.

**Software Used:** Hadoop 2.x

**Theory:**

**a) Installation of Hive & Data aggregation using Hive. What is Apache Hive?**

**Apache Hive** is a data warehouse infrastructure that facilitates querying and managing large data sets which resides in distributed storage system. It is built on top of Hadoop and developed by Facebook. **Hive** provides a way to query the data using a SQL-like query language called **HiveQL(Hive query Language).** Internally, a compiler translates **HiveQL** statements into **MapReduce** jobs, which are then submitted to **Hadoop framework** for execution.

**Difference between Hive and SQL:**

**Hive** looks very much similar like traditional database with **SQL** access. However, because **Hive** is based on **Hadoop** and **MapReduce** operations, there are several key differences: As Hadoop is intended for long sequential scans and **Hive** is based on **Hadoop**, you would expect queries to have a very high latency. Itmeans that **Hive** would not be appropriate for those applications that need very fast response times, as you can expect with a traditional RDBMS database. Finally, **Hive** is read-based and therefore not appropriate for transaction processing that typically involves a high percentage of write operations.

**Hive Installation on Ubuntu:**

Please follow the below steps to install **Apache Hive** on Ubuntu: **Step 1:** Download **Hive tar.**

**Command:** wget http://archive.apache.org/dist/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz **Step 2:** Extract the **tar** file.

**Command:** tar -xzf apache-hive-2.1.0-bin.tar.gz

**Command:** ls


Step 3: Edit the ".bashrc" file to update the environment variables for user.

**Command:** sudo gedit .bashrc

Add the following at the end of the file:

# Set HIVE_HOME ***export HIVE_HOME=/home/edureka/apache-hive-2.1.0-bin export PATH=$PATH:/home/edureka/apache-hive-2.1.0-bin/bin*** Also, make sure that hadoop path is also set.

```
# Set Hadoop-related environment variables

export HADOOP_HOME=/home/edureka/hadoop-2.7.3
export HADOOP_CONF_DIR=/home/edureka/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=/home/edureka/hadoop-2.7.3
export HADOOP_COMMON_HOME=/home/edureka/hadoop-2.7.3
export HADOOP_HDFS_HOME=/home/edureka/hadoop-2.7.3
export YARN_HOME=/home/edureka/hadoop-2.7.3
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

# Set JAVA_HOME

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386
export PATH=$PATH:/usr/lib/jvm/java-8-openjdk-i386/bin

# Add Hadoop bin/ directory to PATH

export PATH=$PATH:/home/edureka/hadoop-2.7.3/bin
export HADOOP_PID_DIR=/home/edureka/hadoop-2.7.3/hadoop2_data/hdfs/pid
```

Run below command to make the changes work in same terminal.

**Command:** source .bashrc

**Step 4:** Check hive version.

```
edureka@localhost:~$ hive --version
Hive 2.1.0
Subversion git://jcamachguezrMBP/Users/jcamachorodriguez/src/workspaces/
hive/HIVE-release2/hive -r 9265bc24d75ac945bde9ce1a0999fddd8f2aae29
Compiled by jcamachorodriguez on Fri Jun 17 01:03:25 BST 2016
From source with checksum 1f896b8fae57fbd29b047d6d67b75f3c
edureka@localhost:~$
```

**Step 5:** Create **Hive** directories within **HDFS**. The directory **'warehouse'** is the location to store the table or data related to hive.

**Command:**

- hdfs dfs -mkdir -p /user/hive/warehouse
- hdfs dfs -mkdir /tmp

**Step 6:** Set read/write permissions for table.

**Command:**

In this command, we are giving write permission to the group:

- hdfs dfs -chmod g+w /user/hive/warehouse
- hdfs dfs -chmod g+w /tmp

**Step 7:** Set **Hadoop** path in **hive-env.sh**

**Command:** cd apache-hive-2.1.0-bin/

**Command:** gedit conf/hive-env.sh

```
edureka@localhost:~$ cd apache-hive-2.1.0-bin/
edureka@localhost:~/apache-hive-2.1.0-bin$ cp conf/hive-env.sh.template
conf/hive-env.sh
edureka@localhost:~/apache-hive-2.1.0-bin$ gedit conf/hive-env.sh
```

Set the parameters as shown in the below snapshot.

```
# Set HADOOP_HOME to point to a specific hadoop install directory
export HADOOP_HOME=/home/edureka/hadoop-2.7.3

export HADOOP_HEAPSIZE=512

# Hive Configuration Directory can be controlled by:
export HIVE_CONF_DIR=/home/edureka/apache-hive-2.1.0-bin/conf
```

**Step 8:** Edit **hive-site.xml**
**Command:** gedit conf/hive-site.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:derby:;databaseName=/home/edureka/apache-hive-2.1.0-
bin/metastore_db;create=true</value>
<description>
JDBC connect string for a JDBC metastore.
To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in the
connection URL.
For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
</description>
</property>
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive/warehouse</value>
<description>location of default database for the warehouse</description>
</property>
<property>
<name>hive.metastore.uris</name>
<value/>
<description>Thrift URI for the remote metastore. Used by metastore client to connect to remote
metastore.</description>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>org.apache.derby.jdbc.EmbeddedDriver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>javax.jdo.PersistenceManagerFactoryClass</name>
<value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>
<description>class implementing the jdo persistence</description>
</property>
</configuration>
```

**Step 9:** By default, Hive uses **Derby** database. Initialize Derby database.
**Command:** bin/schematool -initSchema -dbType derby

```
edureka@localhost:~$ cd apache-hive-2.1.0-bin/
edureka@localhost:~/apache-hive-2.1.0-bin$ bin/schematool -initSchema -d
bType derby
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/edureka/apache-hive-2.1.0-bin/li
b/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/edureka/hadoop-2.7.3/share/hadoo
p/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an expl
anation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFa
ctory]
Metastore connection URL:        jdbc:derby:;databaseName=/home/edureka/
apache-hive-2.1.0-bin/metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 2.1.0
Initialization script hive-schema-2.1.0.derby.sql
Initialization script completed
schemaTool completed
edureka@localhost:~/apache-hive-2.1.0-bin$
```

**Step 10**: Launch **Hive. (In case of Cloudera , we can lunch hive directly on terminal by writing hive command)**

**Command:** hive

```
edureka@localhost:~/apache-hive-2.1.0-bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/edureka/apache-hive-2.1.0-bin/li
b/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/edureka/hadoop-2.7.3/share/hadoo
p/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an expl
anation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFa
ctory]

Logging initialized using configuration in jar:file:/home/edureka/apache
-hive-2.1.0-bin/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async:
 true
Hive-on-MR is deprecated in Hive 2 and may not be available in the futur
e versions. Consider using a different execution engine (i.e. spark, tez
) or using Hive 1.X releases.
hive>
```

**Step 11**: Run few queries in Hive shell.

**Command:** show databases;

**Command:** create table employee (id string, name string, dept string) row format delimited fields terminated by ' ' stored as textfile;

**Command:** show tables;

```
hive> show databases;
OK
default
Time taken: 1.742 seconds, Fetched: 1 row(s)
hive> create table employee (id string, name string, dept string) row fo
rmat delimited fields terminated by '\t' stored as textfile;
OK
Time taken: 1.396 seconds
hive> show tables;
OK
employee
Time taken: 0.228 seconds, Fetched: 1 row(s)
hive>
```

**Step 12:** To exit from **Hive:**

**Command:** exit;

**Hive Commands :**

**Data Definition Language (DDL )**

DDL statements are used to build and modify the tables and other objects in the database.

*Example :* CREATE, DROP, TRUNCATE, ALTER, SHOW, DESCRIBE Statements.

**Data Manipulation Language (DML )**

DML statements are used to retrieve, store, modify, delete, insert and update data in the database.
*Example :* LOAD, INSERT Statements.

Syntax :
LOAD data <LOCAL> inpath <file path> into table [tablename]
The Load operation is used to move the data into corresponding Hive table. If the keyword **local** is specified, then in the load command will give the local file system path.
If the keyword local is not specified we have to use the HDFS path of the file.

**b)- Installation of Pig & Implementation of Pig Script for displaying contents of file stored in local system.**

*Apache Pig* is a tool/platform for creating and executing Map Reduce program used with Hadoop. It is a tool/platform for analyzing large sets of data. You can say, Apache Pig is an abstraction over MapReduce. Programmers who are not so good at Java used to struggle working on Hadoop, majorly while writing MapReduce jobs.

Below are the steps for Apache Pig Installation on Linux (ubuntu/centos/windows using Linux VM). I am using Ubuntu 16.04 in below setup. **Step 1:** Download **Pig tar** file.

**Command:** wget http://www-us.apache.org/dist/pig/pig-0.16.0/pig-0.16.0.tar.gz


**Step 2:** Extract the **tar** file using tar command. In below tar command, **x** means extract an archive file, **z** means filter an archive through gzip, **f** means filename of an archive file. **Command:** tar -xzf pig-0.16.0.tar.gz

**Command:** ls


**Step 3:** Edit the "**.bashrc**" file to update the environment variables of Apache Pig. We are setting it so that we can access pig from any directory, we need not go to pig directory to execute pig commands. Also, if any other application is looking for Pig, it will get to know the path of Apache Pig from this file.

**Command:** sudo gedit .bashrc

Add the following at the end of the file:

# Set PIG_HOME ***export***
***PIG_HOME=/home/edureka/pig-0.16.0 export***

***PATH=$PATH:/home/edureka/pig-0.16.0/bin export***
***PIG_CLASSPATH=$HADOOP_CONF_DIR***

Also, make sure that hadoop path is also set.
Run below command to make the changes get updated in same terminal. **Command:**
source .bashrc

**Step 4:** Check pig version. This is to test that Apache Pig got installed correctly. In case, you don't get the Apache Pig version, you need to verify if you have followed the above steps correctly.
**Command:** pig -version

```
edureka@localhost:~$ source .bashrc
edureka@localhost:~$ pig -version
Apache Pig version 0.16.0 (r1746530)
compiled Jun 01 2016, 23:10:49
```

**Step 5**: Check pig help to see all the pig command options.
**Command:** pig -help

```
edureka@localhost:~$ pig -help

Apache Pig version 0.16.0 (r1746530)
compiled Jun 01 2016, 23:10:49

USAGE: Pig [options] [-] : Run interactively in grunt shell.
       Pig [options] -e[xecute] cmd [cmd ...] : Run cmd(s).
       Pig [options] [-f[ile]] file : Run cmds found in file.
  options include:
    -4, -log4jconf - Log4j configuration file, overrides log conf
    -b, -brief - Brief logging (no timestamps)
    -c, -check - Syntax check
    -d, -debug - Debug level, INFO is default
    -e, -execute - Commands to execute (within quotes)
    -f, -file - Path to the script to execute
    -g, -embedded - ScriptEngine classname or keyword for the ScriptEngi
ne
```

**Step 6**: Run Pig to start the grunt shell. Grunt shell is used to run Pig Latin scripts.( on Cloudera we can run pig directly by writing pig in terminal.)

```
edureka@localhost:~$ pig
16/11/18 18:23:05 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/11/18 18:23:05 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
16/11/18 18:23:05 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2016-11-18 18:23:05,903 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (
r1746530) compiled Jun 01 2016, 23:10:49
2016-11-18 18:23:05,903 [main] INFO  org.apache.pig.Main - Logging error messages to:
/home/edureka/pig_1479473585894.log
2016-11-18 18:23:06,035 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup f
ile /home/edureka/.pigbootup not found
2016-11-18 18:23:07,666 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable
to load native-hadoop library for your platform... using builtin-java classes where ap
plicable
2016-11-18 18:23:07,748 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
- mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2016-11-18 18:23:07,748 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
- fs.default.name is deprecated. Instead, use fs.defaultFS
2016-11-18 18:23:07,749 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HEx
ecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2016-11-18 18:23:09,138 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the
session: PIG-default-09da455b-2398-4800-91f8-9e642ee4ebfe
2016-11-18 18:23:09,139 [main] WARN  org.apache.pig.PigServer - ATS is disabled since
yarn.timeline-service.enabled set to false
grunt>
```

**Command:** pig

If you look at the above image correctly, Apache Pig has two modes in which it can run, by default it chooses MapReduce mode. The other mode in which you can run Pig is Local mode. Let me tell you more about this.

**Execution modes in Apache Pig:**

- *MapReduce Mode* – This is the default mode, which requires access to a Hadoop cluster and HDFS installation. Since, this is a default mode, it is not necessary to specify -x flag ( you can execute *pig* OR *pig -x mapreduce*). The input and output in this mode are present on HDFS.
- *Local Mode* – With access to a single machine, all files are installed and run using a local host and file system. Here the local mode is specified using '-x flag' (*pig -x local*). The input and output in this mode are present on local file system.

**Command:** pig -x local

```
edureka@localhost:~$ pig -x local
16/11/22 18:53:12 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/11/22 18:53:12 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2016-11-22 18:53:12,375 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (
r1746530) compiled Jun 01 2016, 23:10:49
2016-11-22 18:53:12,375 [main] INFO  org.apache.pig.Main - Logging error messages to:
/home/edureka/pig_1479820992372.log
2016-11-22 18:53:12,458 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup f
ile /home/edureka/.pigbootup not found
2016-11-22 18:53:12,836 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
- mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2016-11-22 18:53:12,842 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
- fs.default.name is deprecated. Instead, use fs.defaultFS
2016-11-22 18:53:12,844 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HE
ecutionEngine - Connecting to hadoop file system at: file:///
2016-11-22 18:53:13,272 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
- io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-22 18:53:13,328 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the
session: PIG-default-e8d202a8-bf5a-4fb6-b765-777af4fe7b18
2016-11-22 18:53:13,328 [main] WARN  org.apache.pig.PigServer - ATS is disabled since
yarn.timeline-service.enabled set to false
grunt>
```

**MapReduce Mode**: In 'MapReduce mode', the data needs to be stored in HDFS file system and you can process the data with the help of pig Sript. **Apache**

**Pig Script in MapReduce Mode**

*****************s1.txt*************** 1**

**10000**

**2 20000**
**3 30000**

**4 40000**
**5 50000**

*************output.pig***************

```
A = LOAD '/home/user/input/s1.txt' using PigStorage (',') as (id: chararray, salary:
chararray);

B = FOREACH A generate id, salary; DUMP
B;
```

**Conclusion: Thus installed and implemented Hive and Pig**

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Mark) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 5

**Date of Performance :**

**Date of Submission :**

**Aim:** Implementation of Matrix Multiplication using MapReduce.

## Objective:

- To learn the key issues in big data management and its tools and techniques, specifically programming module of Hadoop.
- To understand need of multiple mappers and reducers in analytics.

**Software Required:** Hadoop 2.X version, eclipse

## Theory:

## Definitions:

M is a matrix with element $m_{ij}$ in row i and column j.N is a

matrix with element $n_{jk}$ in row j and column k.

P is a matrix = MN with element $p_{ik}$ in row i and column k, where $p_{ik} = \sum_j m_{ij} n_{jk}$

Mapper function does not have access to the i, j, and k values directly. An extra MapReduceJob has to be run initially in order to retrieve the values.

**The Map Function:**

For each element $m_{ij}$ of M, emit a key-value pair (i, k), (M, j, $m_{ij}$ ) for k = 1, 2, . . .number of columns of N.

For each element $n_{jk}$ of N, emit a key-value pair (i, k), (N, j, $n_{jk}$) for i = 1, 2, . . . number ofrows of M.

**The Reduce Function:**

For each key (i, k), emit the key-value pair (i, k), $p_{ik}$ where, $P_{ik} = \sum_j m_{ij} * n_{jk}$

The product MN is almost a natural join followed by grouping and aggregation. That is, the natural join of M(I, J, V ) and N(J,K,W), having only attribute J in common, would produce tuples (i, j, k, v,w) from each tuple(i, j, v) in M and tuple (j, k,w) in N.

This five-component tuple represents the pair of matrix elements (mij ,njk).What we want instead is the product of these elements, that is, the four-component tuple (i, j, k, v × w), because that represents the product mijnjk. Once we have this relation as the result of one MapReduce operation, we can perform grouping and aggregation, with me and K as the grouping attributes and the sum of V × W as the aggregation. That is, we can implement matrix multiplication as the cascade of two MapReduce operations, as follows.

**Flow of entire matrix multiplication using map-reduce**

The input file contains two matrices M and N. The entire logic is divided into twoparts:

Step1: Find the product.

Step2: Find sum of the products

**Algorithm 1:** The Map Function

1 **for** *each element* $m_{ij}$ *of* $M$ **do**
2 produce $(key, value)$ pairs as $((i,k),(M,j,m_{ij}))$, for $k = 1,2,3,..$ up to the number of columns of $N$
3 **for** *each element* $n_{jk}$ *of* $N$ **do**
4 produce $(key, value)$ pairs as $((i,k),(N,j,n_{jk}))$, for $i = 1,2,3,...$ up to the number of rows of $M$
5 **return** *Set of (key, value) pairs that each key,* $(i,k)$*, has a list with values* $(M,j,m_{ij})$ *and* $(N,j,n_{jk})$ *for all possible values of* $j$

**Algorithm 2:** The Reduce Function

1 **for** *each key (i,k)* **do**
2 sort values begin with $M$ by $j$ in $list_M$
3 sort values begin with $N$ by $j$ in $list_N$
4 multiply $m_{ij}$ and $n_{jk}$ for $j_{th}$ value of each list
5 sum up $m_{ij} * n_{jk}$
6 **return** $(i,k), \sum_{j=1} m_{ij} * n_{jk}$

## Program

**********matrixmultiplication.java**********

```
import java.io.IOException;
import java.util.*;
import java.io.*;
import java.lang.*;
import org.apache.hadoop.fs.Path;
import  org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class mm
 {
    public static class Map extends Mapper<LongWritable, Text, Text, Text>
 {
       public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException
{
```

```
try
{Configuration conf = context.getConfiguration();
int m = Integer.parseInt(conf.get("m"));int p =
Integer.parseInt(conf.get("p")); String line =
value.toString();
 String[] indicesAndValue = line.split(",");Text
 outputKey = new Text();
 Text outputValue = new Text();
 if (indicesAndValue[0].equals("A")) {for (int
    k = 0; k < p; k++) {
       outputKey.set(indicesAndValue[1] + "," + k);
       outputValue.set("A," + indicesAndValue[2] + "," + indicesAndValue[3]);
       context.write(outputKey, outputValue);}}
        else{
             for (int i = 0; i < m; i++) {
                outputKey.set(i + "," + indicesAndValue[2]);
               outputValue.set("B," + indicesAndValue[1] + "," + indicesAndValue[3]);
                context.write(outputKey, outputValue);}}}
catch(ArrayIndexOutOfBoundsException e){}}}
    public static class Reduce extends Reducer<Text, Text, Text, Text> {
       public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
          String[] value;
          HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
          HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
          for (Text val : values) {
             value = val.toString().split(",");
            if (value[0].equals("A")) {
                hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
             } else {
                hashB.put(Integer.parseInt(value[1]),   Float.parseFloat(value[2]));}}
          int n = Integer.parseInt(context.getConfiguration().get("n"));
          float result = 0.0f;
          float a_ij;
         float b_jk;
          for (int j = 0; j < n; j++) {
             a_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
            b_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
            result += a_ij * b_jk;}
          if (result != 0.0f) {
             context.write(null, new Text(key.toString() + "," + Float.toString(result)));}}}
   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      // A is an m-by-n matrix; B is an n-by-p matrix.
      conf.set("m", "2");
      conf.set("n", "5");
      conf.set("p", "3");
      Job job = new Job(conf, "MatrixMatrixMultiplicationOneStep");
      job.setJarByClass(mm.class);
```

```
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);}}
hadoop fs -rm -r /input
hadoop fs -rm -r /output
hadoop fs -mkdir /input
hadoop fs -put input.txt /input
javac -Xlint -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-
3.21.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar
mm.java
jar cvf  mm.jar *.class
hadoop jar mm.jar mm /input /output
hadoop fs -cat /output/part-r-00000
```

A-2x5 matrix and B-5x3 matrix

**********input**********

(A,0,0,1.0)(A,0,1,1.0)(A,0,2,2.0)(A,0,3,3.0)(A,0,4,4.0)(A,1,0,5.0)(A,1,1,6.0)(A,1,2,7.0)(A,1,3,8.0)
(A,1,4,9.0)(B,0,1,1.0)(B,0,2,2.0)(B,1,0,3.0)(B,1,1,4.0)(B,1,2,5.0)(B,2,0,6.0)(B,2,1,7.0)(B,2,2,8.0)
(B,3,0,9.0)(B,3,1,10.0)(B,3,2,11.0)(B,4,0,12.0)(B,4,1,13.0)(B,4,2,14.0)

**Output:**

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

**Conclusion: Thus, we have successfully Implemented of Matrix Multiplication using MapReduce.**

## Sign and Remark:

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Mark) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 6

**Date of Performance :**

**Date of Submission :**

**Aim:** Implement Bloom Filter and DGIM filter using python.

## Objective:

- To learn filtering methods used in Data mining techniques.

## Software Required: Python.

## Theory:

**Bloom Filter:**

Bloom filter introduced by Burton Bloom in 1970. The filter matches the membership of an element in a dataset. The filter is basically a bit vector of length m that represent a set S={x1, x2, …., xm} of m elements.

Steps for filtering process:

1. Initially all bits 0.
2. Then, define k independent hash function h1, h2, …, and hk.
3. Each of which maps some element x in set S to one of the m array positions with a uniform random distribution.
4. Number k is constant, and much smaller than m. that is for each element X ϵ S, the bits hi(x) are set to 1 for $1 \leq i \leq k$ [ϵ is symbol in set theory for 'contained in '].
5. Counting bloom filter is a variant of bloom filter. It maintain a counter for each bit in the bloom filter. The counter corresponding to the k hash values increment or decrement, whenever an element in the filter is added or deleted, respectively. As soon as a counter changes from 0 to 1, the corresponding bit in the bit vector is set to 1. When a counter changes from 1 to 0, the corresponding bit in the bit vector is set to 0. The counter basically maintain the number of elements that hashed.
6. For new input, find hash value and set corresponding bits and confirm that bit position with status of bloom filter.
7. If it is not match with bloom filter status, it is consider as true negative answer
8. If it is match with bloom filter status, it is might be correct positive or false positive.

**Example :**

Let m=5 size of bloom filter

Step 1: Decide State of bloom filter

H1(x)=x mod 5

H2(x)= (2x+3) mod 5

| X | H1(x) | H2(x) | B | | | | |
|---|---|---|---|---|---|---|---|
| 9 | 4 | 1 | 0 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |

State of Bloom Filter (B) is

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 2 | 3 | 4 |

Step 2: Find number using bloom filter

New X1=15

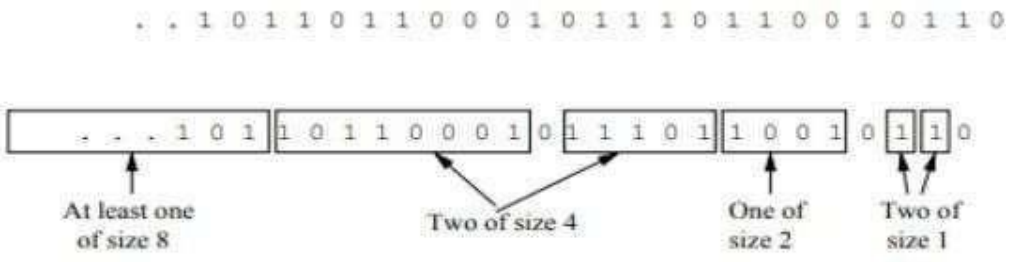
| X | H1(x) | H2(x) | B | Result | Remark |
|---|---|---|---|---|---|
| 15 | 0 | 3 | 1 0 0 1 0 | Not match with bloom filter, Hence 15 number is not in input stream | True Negative answer |
| 16 | 1 | 0 | 1 1 0 0 0 | Match with bloom filter, hence 16 is part of input stream | False positive answer |

Bloom filter able to give 100% correct negative answer, but incorrect positive answer.

**The Datar-Gionis-Indyk-Motwani Algorithm(DGIM):**

This version of the algorithm uses O(log2 N) bits to represent a window of N bits, and allows us to estimate the number of 1's in the window with an error of no more than 50%.
Steps for filtering process:

1. Divide window into buckets, consisting of timestamp of its right 1- the number of 1s in the bucket.
2. This number must be power of 2 and we refer number of 1s as the size of bucket.
3. Right end of bucket is always start with position with a 1.
4. The number of 1s must be power of 2Either
5. 1 or 2 buckets with same power of 2 number of 1s exist.
6. Bucket do not overlap in timestamp.
7. Buckets are stored by size.
8. Buckets disappear when their end time is N time units in the past.
9. As shown in figure below a bit stream divided into buckets following DGIM rules




A bit-stream divided into buckets following the DGIM rules

## Program

### a) Bloom filter:

```
frombloom_filter import BloomFilter
dna_bf = BloomFilter(max_elements=5, error_rate=0.01)
bases = ["A", "T", "G", "C"]
for dna in bases:
    assert (dna in dna_bf) == False
    dna_bf.add(dna)
for base in bases + ["N"]:
    if base in dna_bf:
        print("Base {} is very likely in the BF (0.01 error rate)".format(base))
    else:
        print("Base {} is for sure not in the BF".format(base))
```

### b) DGIM filter:

```
import IPython
import sys
import itertools
import time
import math

def checkAndMergeBucket(bucketList, t):
    bucketListLength = len(bucketList)
    for i in range (bucketListLength):
        if len(bucketList[i]) > 2:
            bucketList[i].pop(0)
            if i + 1 >= bucketListLength:
                bucketList[i].pop(0)
            else:
                bucketList[i+1].append(bucketList[i].pop(0))
K = 1000
N = 1000
k = int(math.floor(math.log(N, 2)))
t = 0
onesCount = 0
bucketList = []
for i in range(k+1):
    bucketList.append(list())
with open('engg5108_stream_data.txt') as f:
    while True:
        c = f.read(1)
        if not c:
            for i in range(k+1):
                for j in range(len(bucketList[i])):
                    print ("Size of bucket: %d, timestamp: %d" % (pow(2,i), bucketList[i][j]))
                    earliestTimestamp = bucketList[i][j]
            for i in range(k+1):
                for j in range(len(bucketList[i])):
                    if bucketList[i][j] != earliestTimestamp:
                        onesCount = onesCount + pow(2,i)
                    else:
                        onesCount = onesCount + 0.5 * pow(2,i)
            print ("Number of ones in last %d bits: %d" % (K, onesCount))
            break
        t = (t + 1) % N
        for i in range(k+1):
            for bucketTimestamp in bucketList[i]:
```

```
            if bucketTimestamp == t:
                bucketList[i].remove(bucketTimestamp)
        if c == '1':
            bucketList[0].append(t)
            checkAndMergeBucket(bucketList, t)
        elif c == '0':
            continue
```

'engg5108_stream_data.txt'

```
11000010110000101000010010011010001000110111111111111001111111011000100000101111000101
01101000100110110011010010111111011010111001100111111010001101010101001110110000010100
00100000100111110011001001101110001001100011111100001101110010101011011010110110000011
11100010011010000011001111000010001011100001000011110100000001111010010001010000010100
01011101101111001100010010000001100100010100011000101110011111010100101010111101010000
00100010011111101110000000001011110001110011001000001101100101011111001110000000011110
11101010100010010000110010111110101001010011110101111101100101000001001110010011111011
11111101000101101110110010111111111001110100001110000110111111101011111110101110110100
00100110101010100100010000001111101111111101001011000011000111011001101000110000000100
00000100101011100010100011000110100011011110001110011010100000101010010100011000110011
11011101111100100110001110000111110010111000011011001110001001011001011100110000101100
00100011101001111011100111010000010111101100110001010101111001111101000010110010011000
00010010101001101000010000001101110100101110101000011101011011101100001100111001000000
01000101101000101010101100110000101101101010011010100111110110010101010100100011011100
10000011001011010111011011100001100111001100001100011111000011001100111010111001100100
10011000000010100100100100010010000111110100011111001000000010110011010100000010000100
00101111000110001010111000000010100101010001111000111101001001000011011010110000011100
00100101100010100001010110001101001010001001000110001100001101001100110010011011110100
10111001100101001100111101110101110010101000001101000011111110101101011001110011110000
01000100110101001011010110111111000011000000110100000011110101011000010011111111010100
11010111111111100111111011010100101100110011000011011011001001010100101000011110001100
11011000111101001001100111100101001000110100111101000000000110001011110000000110001000
01101111100111011001110001010101110011011000010101001010000000010000101101110000100000
11101000100011000001110100100101000100101010000000000100101011011000011000100001000011
10101101111100101110111010010001101110001111110000000000010010101010010000010101101100
10010101100100000011000001101010101010110000100011001100000001101010101010111011010100
01011010110101000100111000111001011010100001001100011011101110101111111011111010000000
00001001110001000011000010101110010000100001100110000101001000110011100110100001010100
10100110100110001100001011101010100110001010010111010111101100000010010000011110001100
10010101010000000000111010110100011000100001010011000011001011111100101101110101010100
10101000100101011010001101011010110011011100011101011100000010100010010101110011111100
00111101111110111001110000001001011001000110111111011010110100110000011101100001101000
10100111010100101000100001001110111010001111111111101001111000001100111001100111111111
11110101110100101110011000000100011001000111010101100001001000101111010101011000010100
11011010000011101100100001111111010001001010000010111110101100101101111001011110010100
11110000011010100101101100111111010001000000001111001011010000011000001101101100101010
00000010001011101110100011010000001000110101001001100110010001000100111110010110110000
10110011011000111010001101011100000011011000100100100101111010100010100100111100100100
00101110011000110100000000010000100101001001010111001101001101100010100110100001001000
01010001110011000000001101011101011011011110001001011011001011011000011100110010011000
11000011110011100011001001111100100001001101000100100000101011010110110101101011111101
10111000011011111110010100000101111110101111100000111101101000001100000010011111100000
00011110111110001001010101010110110011111001001000111010000111000011010001100101110000
01001110101000110110000001100101011011110101111010110110110001001011111001100011011111
00010111100000101010000000000001111010101011001110001000111011110011000111000101010100
10011011010100001001001100010010010110010101010000011110011110001100011111111000110100
11011110111111100101110011011010100000111000110000110000011000111111000101001111110110
10000100111100100111000000001010011011010011100011000101011011110111001010000110110000
11110101101011100100101110111011001111101001001000111010101010100011000001101111001111
00100000111101110110111000001101010110000000011010001111110101000001110101110101001000
00010010100011010000000011010101110000110111000001000111110101110011100010000011011000
```

**Output:**

**a)** Bloom filter

```
Base A is for sure not in the BF
Base T is for sure not in the BF
Base G is for sure not in the BF
Base C is for sure not in the BF
Base N is for sure not in the BF
```

**b)** DGIM filter:

```
1 1 2 3 4 5 6 7 8 8
9 10 11 12 13 14 14 15 12 13
14 14 15 10 10 11 12 12 12 11
12 12 13 8 9 10 10 11 12 23
12 13 14 15 16 16 11 12 19 20
13 14 15 16 11 11 12 23 24 13
14 15 16 16 11 12 19 20 13 14
15 15 10 11 12 23 24 13 13 14
15 16 11 12 19 20 13 14 15 16
17 12 23 24 25 14 15 16 17 17
12 19 20 20 13 14 14 9 10 11
11 12 11 12 13 14 15 16 11 12
12 19 19 12 13 14 14 9 10 11
12 23 24 13 14 15 16 17 12 19
20 20 20 13 14 15 10 11 12 23
12 13 14 15 16 16 11 12 19 19
12 13 14 15 10 11 12 23 23 24
13 14 15 16 11 12 19 20 21 21
14 14 15 10 11 12 23 12 13 14
15 16 17 17 12 19 20 13 14 15
16 11 12 12 23 24 13 14 15 15
10 11 11 12 19 12 12 13 14 15
10 10 11 12 23 12 13 13 14 15
10 10 11 11 12 11 11 12 12 13
8 9 10 11 12 23 24 13 14 15
16 17 12 19 20 20 20 13 13 14
9 10 11 12 12 11 12 13 14 15
16 11 12 19 20 21 14 15 16 16
11
```

**Conclusion**: Thus we have successfully Implement Bloom Filter and DGIM filter using python.

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 7

**Date of Performance :**

**Date of Submission :**

**Aim:** Implementation of recommendation engine (CF) using python.

## Objective:

- To learn how to do recommendation using collaborative filtering.

**Software Required:** Python

## Theory:

**Collaborative filtering**

Collaborative filtering is used to find similar users or items and provide multiple ways to calculate rating based on ratings of similar users.

User-Based: The system finds out the users who have rated various items in the same way. Suppose User A likes 1,2,3 and B likes 1,2 then the system will recommend movie 3 to B.

Item Based: Here, the system tries to find users who bought similar items. For example, A and B like movie 1 and 3 and C likes 3 then, the system will recommend movie 1 to user C.

Here we used pearson or cosine correlation formula to calculate similarity between two users.

$$sim(u,v)^{PCC} = \frac{\sum_{p \in I}(r_{u,p} - \bar{r}_u)(r_{v,p} - \bar{r}_v)}{\sqrt{\sum_{p \in I}(r_{u,p} - \bar{r}_u)^2} \cdot \sqrt{\sum_{p \in I}(r_{v,p} - \bar{r}_v)^2}}$$

$$sim(u,v)^{COS} = \frac{\vec{r}_u \cdot \vec{r}_v}{\|\vec{r}_u\| \cdot \|\vec{r}_v\|}$$

## Program:

**********dataset-recommendation_data1.py**********

```
dataset={'Eric Anderson':{
'0001055178':4.0,
'0007189885':5.0,
'0020209851':5.0,
'0060004843':5.0,
'0060185716':3.0},
'H. Schneider':{
'0006511252':5.0,
```

```
'0020209851':5.0,

'0007156618':4.0,

'0007156634':5.0,

'0030565812':5.0,

'0020519001':4.0},

'Amanda':{

'0000031887':4.0,

'0002007770':2.0,

'0007164785':5.0,

'0007173687':5.0

}}
**********cf_pearson.py**********

from recommendation_data1 import dataset

from math import sqrt

def similarity_score(person1,person2):
        both_viewed = {}

        for item in dataset[person1]:

                if item in dataset[person2]:

                        both_viewed[item] = 1

        no = len(both_viewed)
        print("total no of common items in "+person1 +" and "+person2+ " are")

        print(no)

        if no == 0:
                return 0

        sum_of_eclidean_distance = 0

        a=0
        for item in dataset[person1]:
                if item in dataset[person2]:

                        a=pow(dataset[person1][item] - dataset[person2][item],2)
sum_of_eclidean_distance = sum_of_eclidean_distance +  a
```

```
    return  1/(1+sqrt(sum_of_eclidean_distance))
print(b)

def pearson_correlation(person1,person2):
    both_rated = {}
    for item in dataset[person1]:
        if item in dataset[person2]:
            both_rated[item] = 1
    number_of_ratings = len(both_rated)
    if number_of_ratings == 0:
        return 0
    person1_preferences_sum =  sum([dataset[person1][item]  for item in both_rated])
    person2_preferences_sum =  sum([dataset[person2][item]  for item in both_rated])

    person1_square_preferences_sum = sum([pow(dataset[person1][item],2) for item in
both_rated])
    person2_square_preferences_sum = sum([pow(dataset[person2][item],2) for item in
both_rated])
    product_sum_of_both_users = sum([dataset[person1][item] * dataset[person2][item] for
item in both_rated])

    numerator_value = product_sum_of_both_users -
(person1_preferences_sum*person2_preferences_sum/number_of_ratings)
    denominator_value = sqrt((person1_square_preferences_sum -
pow(person1_preferences_sum,2)/number_of_ratings) * (person2_square_preferences_sum -
pow(person2_preferences_sum,2)/number_of_ratings))
    if denominator_value == 0:
        return 0
    else:
        r = numerator_value/denominator_value
        print(r)
        return r
def most_similar_users(person,number_of_users):
    scores = [(pearson_correlation(person,other_person),other_person) for other_person in
dataset if other_person != person ]
    scores.sort()
```

```
        scores.reverse()
        return scores[0:number_of_users]

for person in dataset:

        print(" similar users for given user "+person)

        print (most_similar_users(person,3))

        print(" --- ")
def user_reommendations(person):
        totals = {}
        simSums = {}
        rankings_list =[]

        for other in dataset:

                if other == person:

                        continue
                sim = pearson_correlation(person,other)

                if sim <=0:

                        continue
                for item in dataset[other]:
                        if item not in dataset[person] or dataset[person][item] == 0:
                                totals.setdefault(item,0)
                                totals[item] += dataset[other][item]* sim

                                simSums.setdefault(item,0)

                                simSums[item]+= sim

        rankings = [(total/simSums[item],item) for item,total in totals.items()]

        rankings.sort()

        rankings.reverse()
        recommendataions_list = [recommend_item for score,recommend_item in rankings]

        return recommendataions_list

  print("**********Recommendation for all users**********")

  for person in dataset:

        print (" Recommendations for user "+person)

        print (user_reommendations(person))
```

**Output:**

```
Number of ratings: 100836
Number of unique movieId's: 9724
Number of unique users: 610
Average number of ratings per user: 165.3
Average number of ratings per movie: 10.37
========================================
# lowest rated
    movieId    title    genres
2689    3604    Gypsy (1962)    Musical

# highest rated
    movieId    title    genres
48    53    Lamerica (1994)    Adventure|Drama

# who rate highest rated movie
userId    movieId    rating    timestamp
13368    85    53    5.0    889468268
96115    603    53    5.0    963180003
```

```
# who rate lowest rated movie
userId    movieId    rating    timestamp
13633    89    3604    0.5    1520408880


Since you watched Grumpier Old Men (1995)
Grumpy Old Men (1993)
Striptease (1996)
Nutty Professor, The (1996)
Twister (1996)
Father of the Bride Part II (1995)
Broken Arrow (1996)
Bio-Dome (1996)
Truth About Cats & Dogs, The (1996)
Sabrina (1995)
Birdcage, The (1996
```

`

**Conclusion**: **Thus implemented of recommendation engine (CF) using python.**

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 8

**Date of Performance :**

**Date of Submission :**

**Aim:** Data Visualization using R

## Objective:

- To learn data visualization methods using R

**Software Required:** R language

## Theory:

**Data visualization** is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

R is a language that is designed for statistical computing, graphical data analysis, and scientific research. It is usually preferred for data visualization as it offers flexibility and minimum required coding through its packages.

Consider the following air quality data set for visualization in R:

| Ozone | Solar R. | Wind | Temp | Month | Day |
|---|---|---|---|---|---|
| 41 | 190 | 7.4 | 67 | 5 | 1 |
| Ozone | Solar R. | Wind | Temp | Month | Day |
| 36 | 118 | 8.0 | 72 | 5 | 2 |
| 12 | 149 | 12.6 | 74 | 5 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| 18 | 313 | 11.5 | 62 | 5 | 4 |
| NA | NA | 14.3 | 56 | 5 | 5 |
| 28 | NA | 14.9 | 66 | 5 | 6 |

**Types of Data Visualizations are as follows**

**Bar Plot**

There are two types of bar plots- horizontal and vertical which represent data points as horizontal or vertical bars of certain lengths proportional to the value of the data item. They are generally used for continuous and categorical variable plotting.

**Histogram**

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

**Box Plot**

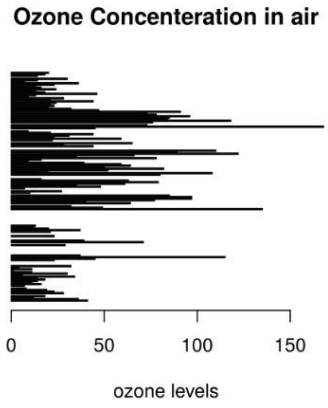
The statistical summary of the given data is presented graphically using a boxplot. A boxplot depicts information like the minimum and maximum data point, the median value, first and third quartile, and interquartile range.

**Scatter Plot**

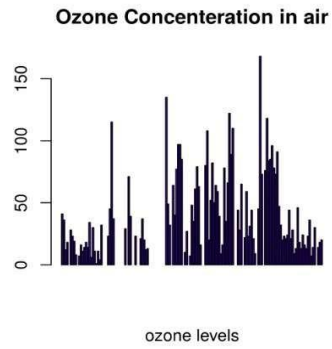
A scatter plot is composed of many points on a Cartesian plane. Each point denotes the value taken by two parameters and helps us easily identify the relationship between them.

**Program**

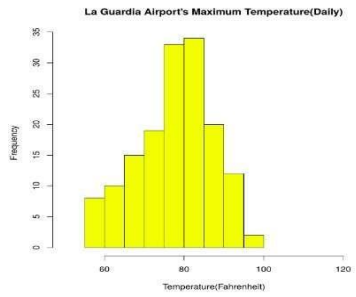
```
Bar Plot
barplot(airqualityOzone,
        main = 'Ozone Concentration in
        air', xlab = 'ozone levels', horizontal = TRUE)
```



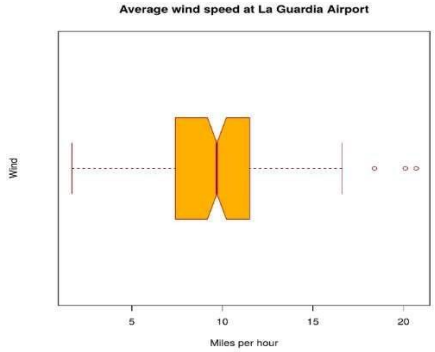

barplot(airquality$Ozone, main = 'Ozone Concenteration in air', xlab = 'ozone levels', col ='blue', horiz = FALSE)



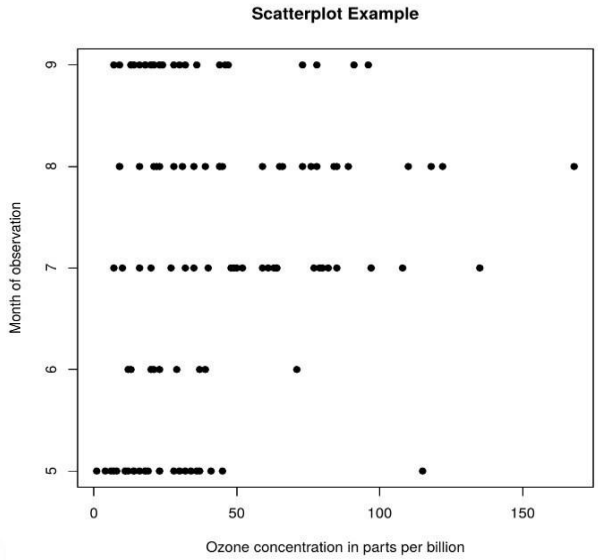

hist(airquality$Temp, main ="La Guardia Airport's\ Maximum Temperature(Daily)", xlab ="Temperature(Fahrenheit)", xlim = c(50, 125), col ="yellow", freq = TRUE)




boxplot(airquality$Wind, main = "Average wind speed\ at La Guardia Airport", xlab = "Miles per hour", ylab = "Wind", col = "orange", border = "brown", horizontal = TRUE, notch = TRUE)



data(airquality) plot(airquality$Ozone, airquality$Month, main="Scatterplot Example", xlab ="Ozone Concentration in parts per billion", ylab =" Month of observation ", pch =19)




## Conclusion:

Thus, we have successfully visualized data with R.

## Sign and Remark:

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 9

**Date of Performance :**

**Date of Submission :**

**Aim:** Use of Sqoop to Transfer Data between Hadoop and Mysql

## Objective:

- To learn data exchange in between Mysql and Hive using sqoop

**Software Required:** Hive, mysql, hadoop

## Theory:

Sqoop: Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases

**Program**

mysql> create table dmce(no int, branch varchar(30));

mysql> insert into dmce(no, branch) values(1, 'comp');

mysql> insert into dmce(no, branch) values(21, 'IT');

mysql> select * from dmce;

mysql> exit;

[root@localhost training]# exit;

[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/testdb --username root --table dmce --hive-import --hive-table dmce --m 1;[training@localhost ~]$ hive

hive> select * from dmce;

mysql> create table txnrecord(txno int,txdate varchar(30),custno int,amount double,category varchar(30));

hive> show databases;

hive> use books;

hive> show tables;

hive> describe txnrecords;

hive> load data local inpath '/home/training/Desktop/text.txt' into table txnrecords;

hive> describe formatted txnrecords;

hive> exit;

[training@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/db2 -m 1 --table txnrecord --export-dir 'hdfs://localhost/user/hive/warehouse/books.db/txnrecords' --input-fields-terminated-by ',' --username root;

mysql> use db2;

mysql> select * from txnrecord;

**Output:**

MySQL

```
+----------------+
| employee       |
+----------------+
1 row in set (0.00 sec)

mysql>
mysql>
mysql>
mysql> select * from employee;
+-------+--------+-------------+---------+------------+------+--------+
| empno | ename  | designation | manager | hire_date  | sal  | deptno |
+-------+--------+-------------+---------+------------+------+--------+
|  7369 | SMITH  | CLERK       |    7902 | 1980-12-17 |  800 |     20 |
|  7499 | ALLEN  | SALESMAN    |    7698 | 1981-02-20 | 1600 |     30 |
|  7521 | WARD   | SALESMAN    |    7698 | 1981-02-22 | 1250 |     30 |
|  7566 | JONES  | MANAGER     |    7839 | 1981-04-02 | 2975 |     20 |
|  7654 | MARTIN | SALESMAN    |    7698 | 1981-09-28 | 1250 |     30 |
|  7369 | SMITH  | CLERK       |    7902 | 1980-12-17 |  800 |     20 |
|  7369 | SMITH  | CLERK       |    7902 | 1980-12-17 |  800 |     20 |
|  7369 | SMITH  | CLERK       |    7902 | 1980-12-17 |  800 |     20 |
|  7369 | SMITH  | CLERK       |    7902 | 1980-12-17 |  800 |     20 |
```

Import data from MySQL to Hive

```
1 sqoop import --connect jdbc:mysql://127.0.0.1:3306/emp --username root --password hortonworks1 --m 1
  --table employee --target-dir /input/sqoop2 --fields-terminated-by "," --hive-import --create-hive-table
  --hive-table sqoop.employee_sqoop --hive-overwrite
2
3
4
```

```
59-a271-c8200a47c11a); Time taken: 0.013 seconds
INFO  : OK
+-----------------+
|    tab_name     |
+-----------------+
| employee_sqoop  |
+-----------------+
1 row selected (0.062 seconds)
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2>
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2>
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select * from employee_sqoop;
INFO  : Compiling command(queryId=hive_20220310221444_4e81c10c-046a-418f-ab36-b2
4f64612e84): select * from employee_sqoop
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:employee_sq
oop.empno, type:int, comment:null), FieldSchema(name:employee_sqoop.ename, type:
string, comment:null), FieldSchema(name:employee_sqoop.designation, type:string,
 comment:null), FieldSchema(name:employee_sqoop.manager, type:int, comment:null)
, FieldSchema(name:employee_sqoop.hire_date, type:string, comment:null), FieldSc
```

**Conclusion**: Thus, we have Used of Sqoop to Transfer Data between Hadoop and Mysql.

**Sign and Remark:**

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |

## EXPERIMENT NUMBER: 10

**Date of Performance :**

**Date of Submission :**

**Aim:** Implementation of Girvan-Newman Algorithm

## Objective:

- To learn Girvan-Newman Algorithm

**Software Required:** Python language

## Theory:

Girvan-Newman Algorithm

For the detection and analysis of community structures, the Girvan-Newman algorithm relies on the iterative elimination of edges that have the highest number of shortest paths between nodes passing through them. By removing edges from the graph one-by-one, the network breaks down into smaller pieces, so-called communities. The algorithm was introduced by Michelle Girvan and Mark Newman.

The idea is to find which edges in a network occur most frequently between other pairs of nodes by finding edge betweenness centrailities. The edges joining communities are then expected to have a high edge betweenness. The underlying community structure of the network will be much more fine-grained once the edges with the highest betweenness are eliminated which means that communities will be much easier to spot.

The Girvan-Newman algorithm can be divided into four main steps:

1. For every edge in a graph, calculate the edge betweenness centrality.
2. Remove the edge with the highest betweenness centrality.
3. Calculate the betweenness centrality for every remaining edge.
4. Repeat steps 2–4 until there are no more edges left.

## Pseudo code

```
REPEAT
  LET n BE number of edges in the graph
  FOR i=0 to n-1
    LET B[i] BE betweenness centrality of edge i
    IF B[i] > max_B THEN
      max_B = B[i]
      max_B_edge = i
    ENDIF
  ENDFOR
  REMOVE edge i FROM graph
UNTIL number of edges in graph is 0
```

**Community Detection Algorithms in NetworkX**

girvan_newman(G, most_valuable_edge=None)

**Method input**

The first input parameter of the method, G, is a NetworkX graph. The second parameter, most_valuable_edge, is a function that takes a graph as input and returns the edge that should be removed from the graph in each iteration. If no function is specified, the edge with the highest betweenness centrality will be chosen in each iteration.

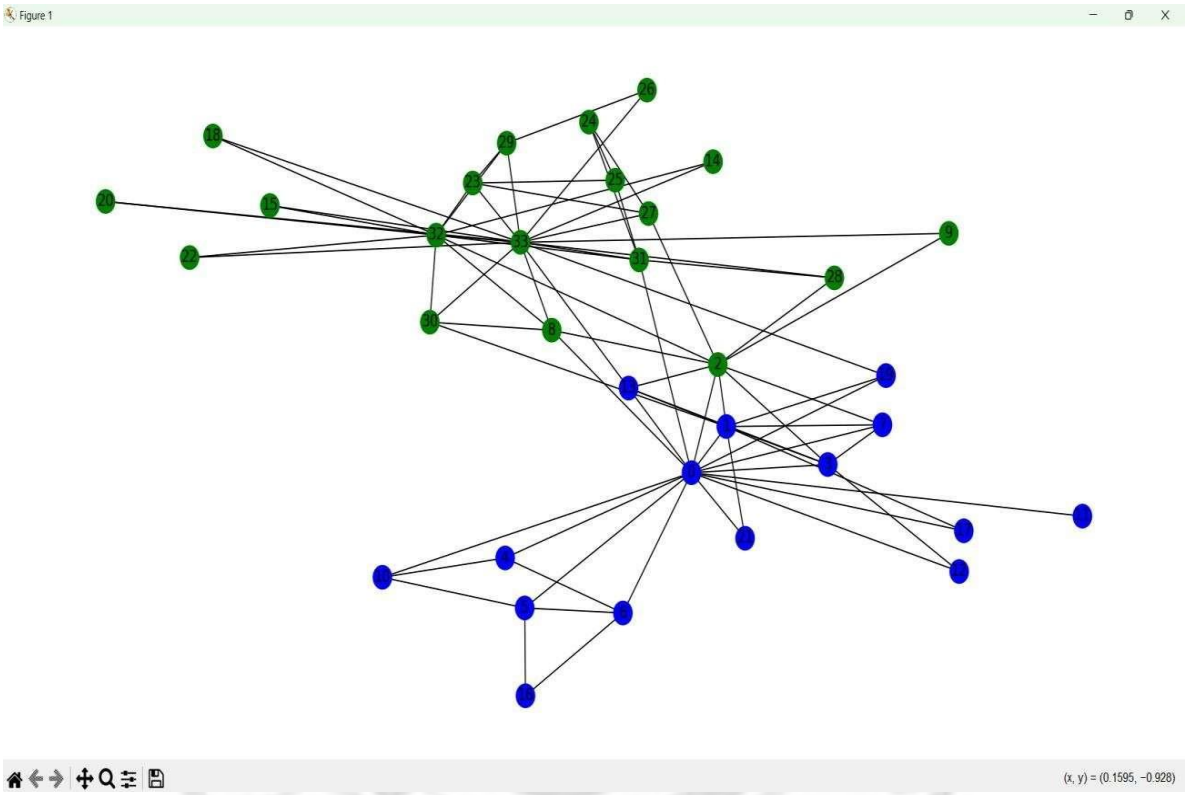
**Method output**

The output of the method is an iterator over tuples of sets of nodes in G. Each set of nodes represents a community and each tuple is a sequence of communities at a particular level (iteration) of the algorithm.

## Program:

```
import matplotlib.pyplot as plt
import networkx as nx
from networkx.algorithms.community.centrality import girvan_newman
 G = nx.karate_club_graph()
communities = girvan_newman(G)
 node_groups = []
for com in next(communities):
node_groups.append(list(com))
print(node_groups)
 color_map = []
for node in G:
    if node in node_groups[0]:
        color_map.append('blue')
    else:
        color_map.append('green')
nx.draw(G, node_color=color_map, with_labels=True)
plt.show()
```

**Output:**




## Conclusion:

Thus, we have Implemented of Girvan-Newman Algorithm

## Sign and Remark:

| R1 (4 Marks) | R2 (4 Marks) | R3 (4 Marks) | R4 (3 Marks) | Total Marks (15 Marks) | Signature |
|---|---|---|---|---|---|
| | | | | | |