

PRACTICAL NO: 1

TO UNDERSTAND THE HDFS COMMANDS

PRACTICAL NO: 1 TO UNDERSTAND THE HDFS COMMANDS

Question 1: HDFS: List of Commands (ls, mkdir, touchz, copy from local/put, copy to local/get, move from local, viewing file content(cat, head, tail),cp, rmr, du, dus, stat)

Code :-

Steps:

Step 1: Download oracle virtual machine and install it as run as administrator

Step 2: Download Cloudera-quickstart-vm-5.4.2-0-Virtualbox and unzip it use 7 zip.

Step 3: Import Cloudera-quickstart-vm-5.4.2-0-Virtualbox.ovf in oracle Virtual Machine.

Step 4: Provide the Memory and processor to machine. It requires minimum 4 gb ram and 2 processor.

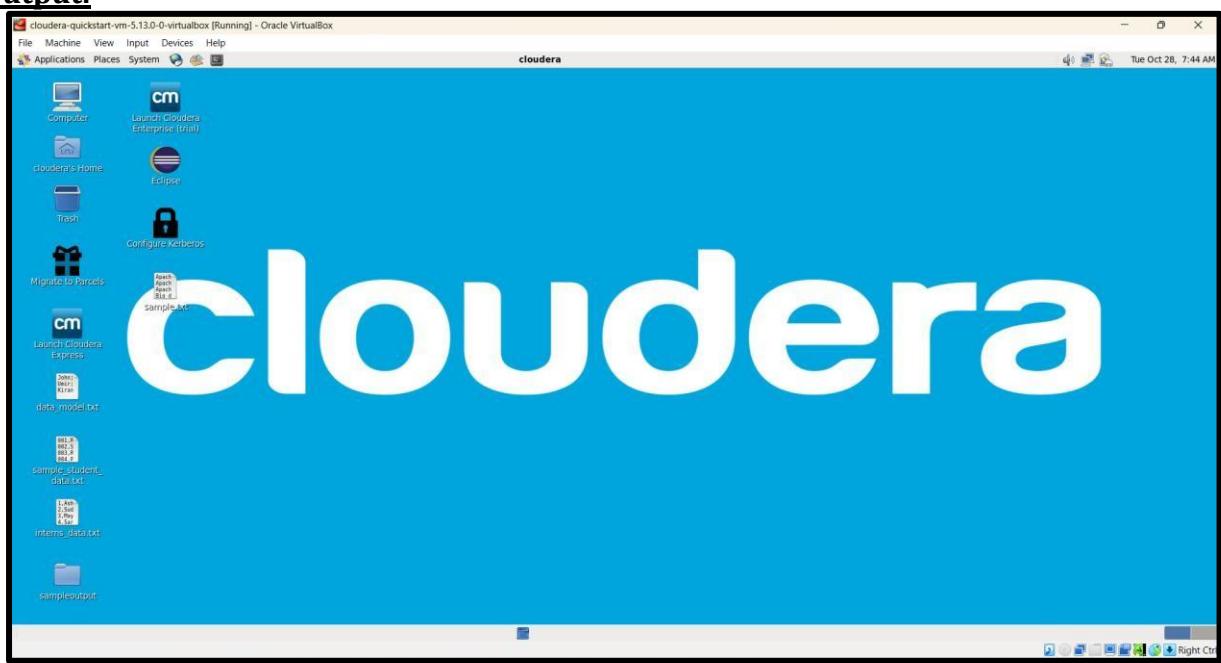
Step 5: Enable Network Adapter and attach it to NAT.

Step 6: Start the Machine.

Step 7: Launch Cloudera express. After launch we get command i.e sudo /home/cloudera- manager - -force We have to run this command by adding - - express command.

Step 8: we have to login the cloudera manager and after that we have to start the services of Hadoop system.

Output:



```
cloudera@quickstart:~
```

File Edit View Search Terminal Help

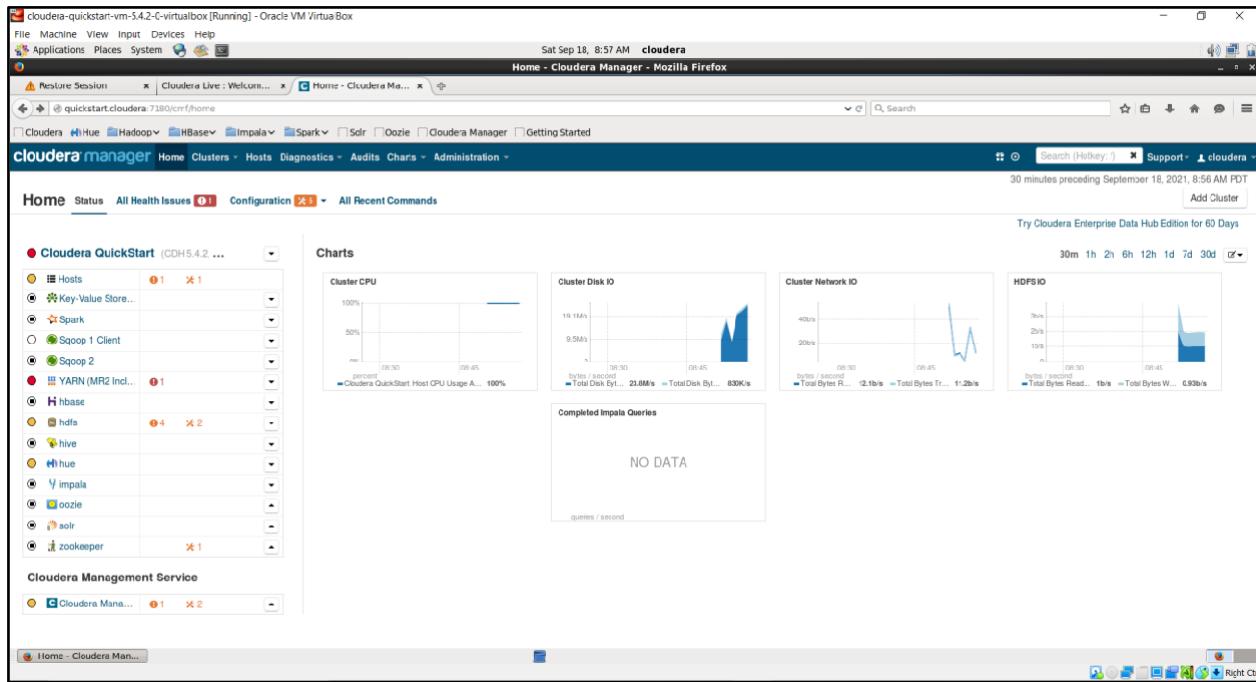
```
[cloudera@quickstart ~]$ sudo /home/cloudera/cloudera-manager --force
You must specify --express or --enterprise
[cloudera@quickstart ~]$ sudo /home/cloudera/cloudera-manager --force --express
[QuickStart] Shutting down CDH services via init scripts...
JMX enabled by default
Using config: /etc/zookeeper/conf/zoo.cfg
[QuickStart] Disabling CDH services on boot...
[QuickStart] Starting Cloudera Manager daemons...
[QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Configuring deployment...
[QuickStart] Deploying client configuration...
[QuickStart] Starting Cloudera Management Service...
[QuickStart] Enabling Cloudera Manager daemons on boot...
```

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:

<http://quickstart.cloudera:7180>

Username: cloudera
Password: cloudera

```
[cloudera@quickstart ~]$ d
```



1. Perform the following task on HDFS cluster using HDFS commands
 - a. Check Linux login detail and list of all files available in the current directory their size and date

The command for login detail:

>Whoami

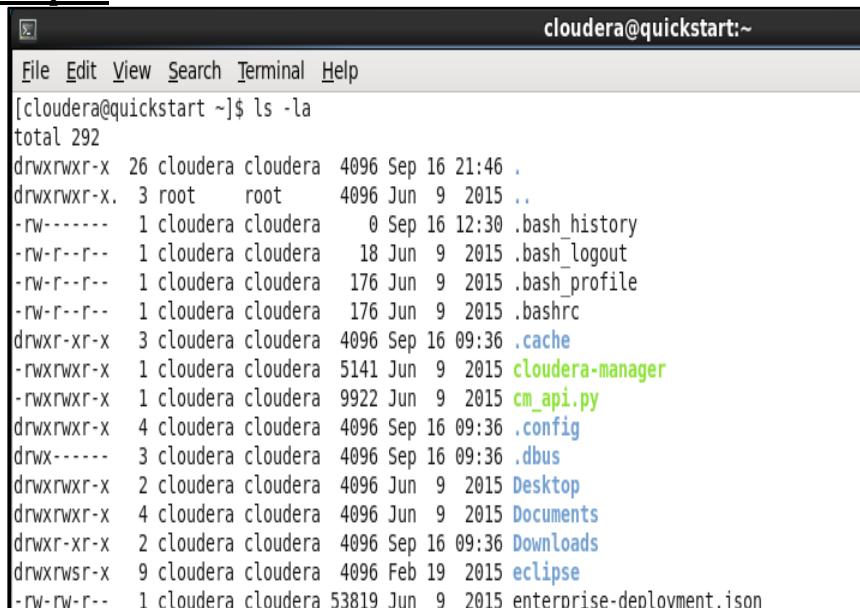
Output:

```
[cloudera@quickstart ~]$ whoami  
cloudera  
[cloudera@quickstart ~]$ █
```

The command for available files in the current directory their size and date:

>ls -la

Output:



A screenshot of a terminal window titled "cloudera@quickstart:~". The window shows the output of the "ls -la" command. The output lists numerous files and directories in the current directory, including ".bash_history", ".bash_logout", ".bash_profile", ".bashrc", ".cache", "cloudera-manager", "cm_api.py", ".config", ".dbus", "Desktop", "Documents", "Downloads", "eclipse", and "enterprise-deployment.json". The terminal interface includes a menu bar with File, Edit, View, Search, Terminal, and Help.

```
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ ls -la  
total 292  
drwxrwxr-x 26 cloudera cloudera 4096 Sep 16 21:46 .  
drwxrwxr-x 3 root      root      4096 Jun  9  2015 ..  
-rw----- 1 cloudera cloudera    0 Sep 16 12:30 .bash_history  
-rw-r--r-- 1 cloudera cloudera 18 Jun  9  2015 .bash_logout  
-rw-r--r-- 1 cloudera cloudera 176 Jun  9  2015 .bash_profile  
-rw-r--r-- 1 cloudera cloudera 176 Jun  9  2015 .bashrc  
drwxr-xr-x 3 cloudera cloudera 4096 Sep 16 09:36 .cache  
-rwxrwxr-x 1 cloudera cloudera 5141 Jun  9  2015 cloudera-manager  
-rwxrwxr-x 1 cloudera cloudera 9922 Jun  9  2015 cm_api.py  
drwxrwxr-x 4 cloudera cloudera 4096 Sep 16 09:36 .config  
drwx----- 3 cloudera cloudera 4096 Sep 16 09:36 .dbus  
drwxrwxr-x 2 cloudera cloudera 4096 Jun  9  2015 Desktop  
drwxrwxr-x 4 cloudera cloudera 4096 Jun  9  2015 Documents  
drwxr-xr-x 2 cloudera cloudera 4096 Sep 16 09:36 Downloads  
drwxrwsr-x 9 cloudera cloudera 4096 Feb 19  2015 eclipse  
-rw-rw-r-- 1 cloudera cloudera 53819 Jun  9  2015 enterprise-deployment.json
```

b. List all files of HDFS default directory

Command:

```
hdfs dfs -ls /user/cloudera
```

After create some file and directory

```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/FirstDir/Firstfile1.txt
cat: Permission denied: user=cloudera, access=EXECUTE, inode="/user/cloudera/FirstDir":cloudera:cloudera:-rw-r--r-- (Ancestor /user/cloudera/FirstDir is not a directory).
[cloudera@quickstart ~]$ hdfs dfs -rm /user/cloudera/FirstDir
25/09/06 07:12:07 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstart.cloudera:8020/user/cloudera/FirstDir' to trash at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -put Firstfile1.txt /user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/FirstDir
Found 1 items
```

c. Create new directory

Command:

```
hdfs dfs -mkdir FirstDir
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/FirstDir/Firstfile1.txt
cat: Permission denied: user=cloudera, access=EXECUTE, inode="/user/cloudera/FirstDir":cloudera:cloudera:-rw-r--r-- (Ancestor /user/cloudera/FirstDir is not a directory).
[cloudera@quickstart ~]$ hdfs dfs -rm /user/cloudera/FirstDir
25/09/06 07:12:07 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstart.cloudera:8020/user/cloudera/FirstDir' to trash at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -put Firstfile1.txt /user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/FirstDir
Found 1 items
```

d. Create file

Command:

File created in Hadoop:

```
hdfs dfs -touchz /user/cloudera/FirstDir/Firstfile.txt
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera
[cloudera@quickstart ~]$ nano Firstfile1.txt
[cloudera@quickstart ~]$ cat Firstfile1.txt
This is my first HDFS practical file.
Stored from Linux to HDFS.
```

e. Copy data from Linux to HDFS

Command:

```
hdfs dfs -put Firstfile1.txt /user/cloudera/FirstDir
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -put Firstfile1.txt /user/cloudera/FirstDir
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/FirstDir
-rw-r--r-- 1 cloudera cloudera 66 2025-09-06 07:10 /user/cloudera/FirstDir
```

f. Copy HDFS file in Linux default directory

Command:

```
hdfs dfs -get /user/cloudera/FirstDir/Firstfile1.txt /home/cloudera/FirstDir
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -get /user/cloudera/FirstDir/Firstfile2.txt /home/cloudera/
get: `/home/cloudera/Firstfile2.txt': File exists
```

g. Move Linux data in HDFS default folder

Command:

```
hdfs dfs -moveFromLocal /home/cloudera/Firstfile.txt /user/cloudera
```

Output:

```
[cloudera@quickstart ~]$ ls
cloudera-manager Desktop Downloads enterprise-deployment.json FirstDir Firstfile.txt kerberos Music Public Videos
cm_api.py Documents eclipse express-deployment.json Firstfile1.txt FirstFir lib Pictures Templates workspace
[cloudera@quickstart ~]$
```

```
[cloudera@quickstart ~]$ hdfs dfs -moveFromLocal /home/cloudera/Firstfile2.txt /user/c
loudera
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 4 items
drwxr-xr-x - cloudera cloudera 0 2021-11-18 07:19 FirstDir
-rw-r--r-- 1 cloudera cloudera 0 2021-11-18 07:30 Firstfile.txt
-rw-r--r-- 1 cloudera cloudera 13 2021-11-18 07:30 Firstfile2.txt
-rw-r--r-- 1 cloudera cloudera 0 2021-11-18 06:49 jagrit
[cloudera@quickstart ~]$
```

h. Remove created file from Linux and HDFS cluster too

Command:

Remove file from Hadoop:

```
hdfs dfs -rm /user/cloudera/FirstDir/Firstfile1.txt
```

Remove file from hdfs:

```
[cloudera@quickstart ~]$ hdfs dfs -rm -skipTrash /user/cloudera/FirstDir/Fi
le1.txt
Deleted /user/cloudera/FirstDir/Firstfile1.txt
```

i. Remove Directory from HDFS cluster which is having at least one file.

Command:

```
hdfs dfs -rmdir FirstDir
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -rmdir FirstDir
rmdir: DEPRECATED: Please use 'rm -r' instead.
21/09/20 14:28:30 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 1440 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://quickstart.cloudera:8020/user/cloudera/FirstDir' to trash at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current
```

j. Check the size of each file in directory of HDFS

Command:

```
hdfs dfs -du /user/cloudera
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -du /user/cloudera
66 66 /user/cloudera/.Trash
0 0 /user/cloudera/FirstDir
38 38 /user/cloudera/Firstfile.txt
```

k. Check the last modified time of current directory

Command:

```
hdfs dfs -stat /user
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -stat /user
2017-10-23 16:17:33
```

l. HDFS Command to know the usage of any command on HDFS

Command:

```
hdfs dfs -usage put
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -usage put
Usage: hadoop fs [generic options] -put [-f] [-p] [-l] <localsrc> ... <dst>
[cloudera@quickstart ~]$ █
```

m. HDFS Command to get help for any command on HDFS

Command:

```
hdfs dfs -help put
```

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -help put
-put [-f] [-p] [-l] <localsrc> ... <dst> :
  Copy files from the local file system into fs. Copying fails if the file already
  exists, unless the -f flag is given.
Flags:
  -p Preserves access and modification times, ownership and the mode.
  -f Overwrites the destination if it already exists.
  -l Allow DataNode to lazily persist the file to disk. Forces
        replication factor of 1. This flag will result in reduced
        durability. Use with care.
[cloudera@quickstart ~]$ █
```

PRACTICAL NO: 2

**TO UNDERSTAND THE USE OF
MAPREDUCE FOR
WORDCOUNT
AND
UNION OPERATIONS**

PRACTICAL NO: 2
TO UNDERSTAND THE USE OF MAPREDUCE FOR
WORDCOUNT AND UNION OPERATIONS

Question a) To implement a Date Time Server using RPC concept. (Make use of Server Socket).

Objective:

To write and execute a Hadoop MapReduce program in Java for counting the occurrences of each word in a text file.

Software Used:

- Cloudera VM 5.13.0
- Eclipse Luna (pre-installed)
- Hadoop 2.6.0 (CDH 5.13.0)

Algorithm / Procedure:

Step 1 – Open Eclipse Luna

Double-click the **Eclipse** icon on the Cloudera desktop.

When prompted for workspace, keep the default:

/home/cloudera/workspace

Click **OK**.

Step 2 – Create New Java

Project File → New → Java

Project Project Name:

WordCount

Finish

Step 3 – Add Hadoop JAR Libraries

Right-click WordCount → Properties

Java Build Path → Libraries tab → Add External JARs...

Add all JARs from the following folders:

/usr/lib/hadoop/*.jar
/usr/lib/hadoop-mapreduce/*.jar
/usr/lib/hadoop/common/*.jar
/usr/lib/hadoop/hdfs/*.jar
/usr/lib/hadoop/yarn/*.jar

Click **Apply and Close**.

Step 4 – Create 3 Java Files inside src folder

WordCountDriver.java

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.conf.Configuration;

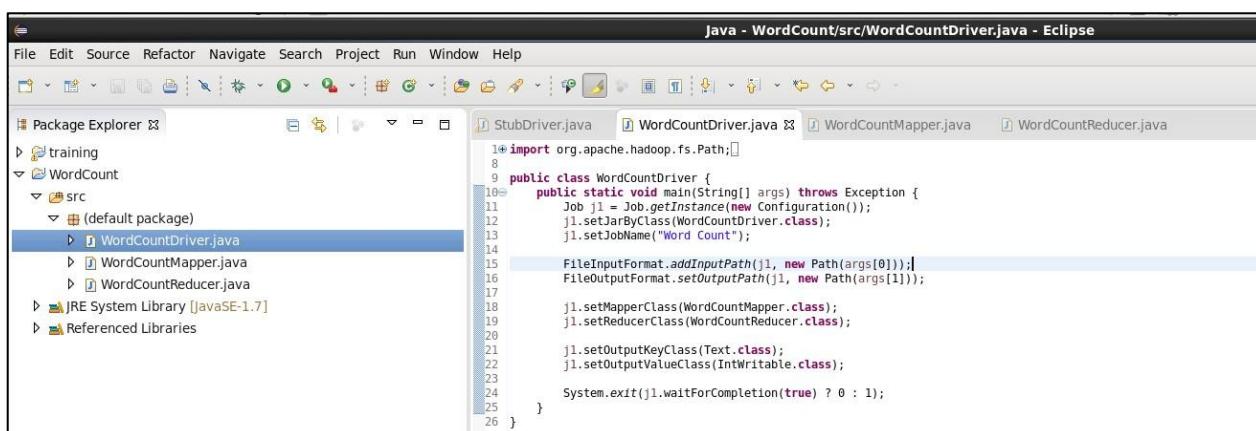
public class WordCountDriver {
    public static void main(String[] args) throws
        Exception { Job j1 = Job.getInstance(new
        Configuration());
        j1.setJarByClass(WordCountDriver.class);
        j1.setJobName("Word Count");

        FileInputFormat.addInputPath(j1, new Path(args[0]));
        FileOutputFormat.setOutputPath(j1, new Path(args[1]));

        j1.setMapperClass(WordCountMapper.class);
        j1.setReducerClass(WordCountReducer.class);

        j1.setOutputKeyClass(Text.class);
        j1.setOutputValueClass(IntWritable.class);

        System.exit(j1.waitForCompletion(true) ? 0 : 1);
    }
}
```



The screenshot shows the Eclipse IDE interface with the title bar "Java - WordCount/src/WordCountDriver.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, etc. The left sidebar is the "Package Explorer" showing a project structure with "training", "WordCount", "src", and files "WordCountDriver.java", "WordCountMapper.java", and "WordCountReducer.java". The right pane is the code editor with the content of WordCountDriver.java. The code is highlighted with syntax coloring, showing imports, class definitions, and method implementations.

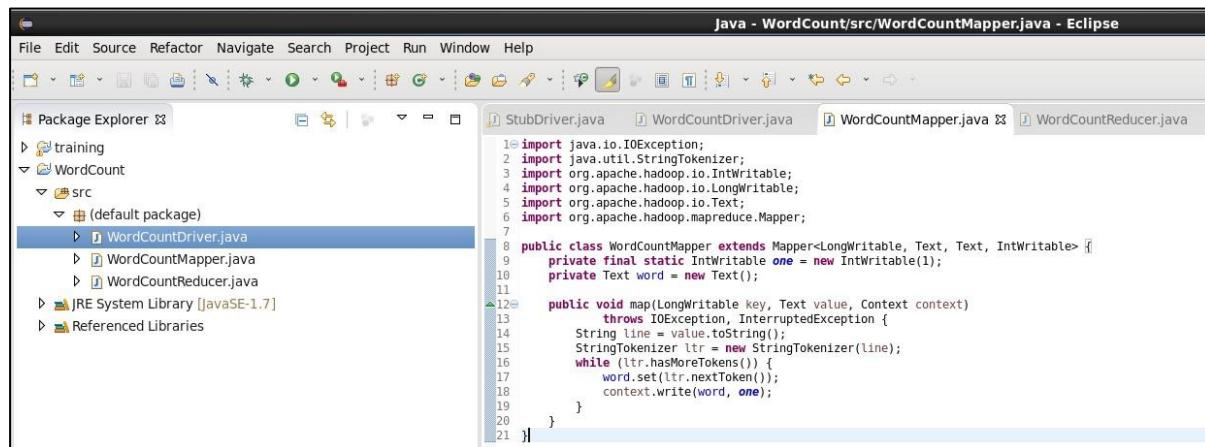
```
1 import org.apache.hadoop.fs.Path;
2
3 public class WordCountDriver {
4     public static void main(String[] args) throws Exception {
5         Job j1 = Job.getInstance(new Configuration());
6         j1.setJarByClass(WordCountDriver.class);
7         j1.setJobName("Word Count");
8
9         FileInputFormat.addInputPath(j1, new Path(args[0]));
10        FileOutputFormat.setOutputPath(j1, new Path(args[1]));
11
12        j1.setMapperClass(WordCountMapper.class);
13        j1.setReducerClass(WordCountReducer.class);
14
15        j1.setOutputKeyClass(Text.class);
16        j1.setOutputValueClass(IntWritable.class);
17
18        System.exit(j1.waitForCompletion(true) ? 0 : 1);
19
20    }
21
22 }
```

WordCountMapper.java

```
import java.io.IOException;
import
java.util.StringTokenizer;
import
org.apache.hadoop.io.IntWritable;
import
org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new
IntWritable(1); private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer ltr = new StringTokenizer(line);
        while (ltr.hasMoreTokens()) {
            word.set(ltr.nextToken());
            context.write(word, one);
        }
    }
}
```



The screenshot shows the Eclipse IDE interface with the title bar "Java - WordCount/src/WordCountMapper.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, etc. The left side features the "Package Explorer" view, which displays a project structure under "WordCount". The "src" folder contains four files: WordCountDriver.java, WordCountMapper.java (which is currently selected and highlighted in blue), and WordCountReducer.java, along with JRE System Library [javaSE-1.7] and Referenced Libraries. The right side of the interface is the code editor window, showing the Java code for WordCountMapper.java. The code is identical to the one provided above, with line numbers 1 through 21 visible on the left.

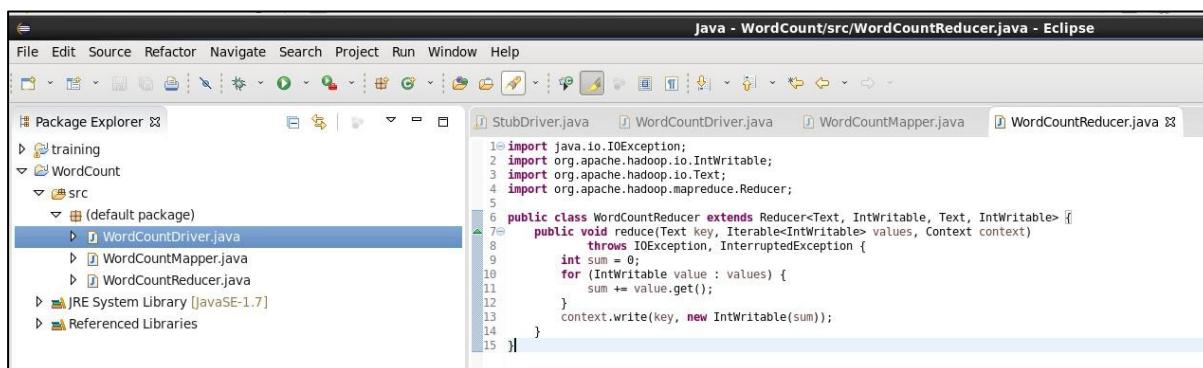
WordCountReducer.java

```

import java.io.IOException;
import
org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
    public void reduce(Text key, Iterable<IntWritable> values, Context
        context) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values)
            { sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }
}

```

**Step 5 – Export Project as JAR**

Right-click project → Export → JAR file JAR

file path: /home/cloudera/wordcount.jar

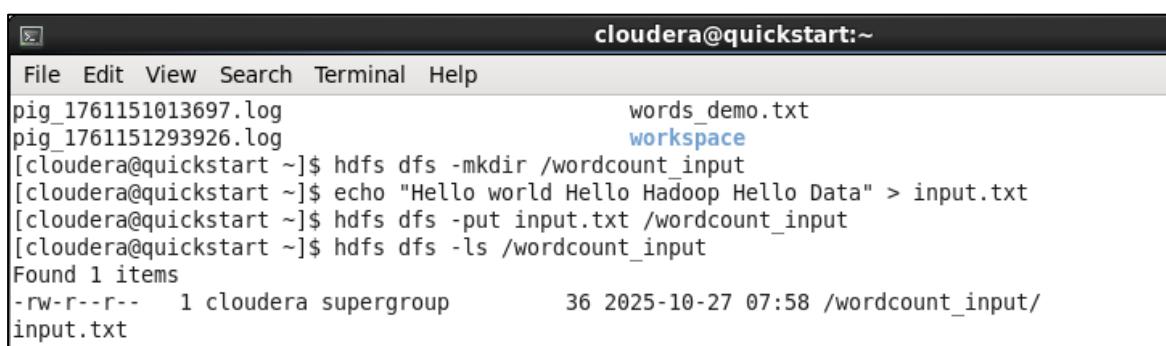
Finish

Step 6 – Create Input in HDFS

hdfs dfs -mkdir /wordcount_input

echo "Hello world Hello Hadoop Hello Data" >

input.txt hdfs dfs -put input.txt /wordcount_input



Step 7 – Run the MapReduce Job

```
hadoop jar /home/cloudera/wordcount.jar WordCountDriver /wordcount_input  
/wordcount_output
```

```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/wordcount.jar WordCountDriver  
/wordcount_input /wordcount_output  
25/10/27 07:58:47 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0  
:8032  
25/10/27 07:58:48 WARN mapreduce.JobResourceUploader: Hadoop command-line option  
parsing not performed. Implement the Tool interface and execute your applicatio  
n with ToolRunner to remedy this.  
25/10/27 07:58:48 INFO input.FileInputFormat: Total input paths to process : 1  
25/10/27 07:58:48 INFO mapreduce.JobSubmitter: number of splits:1  
25/10/27 07:58:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_17  
61571560830_0001  
25/10/27 07:58:49 INFO impl.YarnClientImpl: Submitted application application_17  
61571560830_0001  
25/10/27 07:58:49 INFO mapreduce.Job: The url to track the job: http://quickstar  
t.cloudera:8088/proxy/application_1761571560830_0001/  
25/10/27 07:58:49 INFO mapreduce.Job: Running job: job_1761571560830_0001  
25/10/27 07:59:15 INFO mapreduce.Job: Job job_1761571560830_0001 running in uber  
mode : false  
25/10/27 07:59:15 INFO mapreduce.Job: map 0% reduce 0%  
25/10/27 07:59:29 INFO mapreduce.Job: map 100% reduce 0%  
25/10/27 07:59:37 INFO mapreduce.Job: map 100% reduce 100%  
25/10/27 07:59:38 INFO mapreduce.Job: Job job_1761571560830_0001 completed succe  
ssfully  
25/10/27 07:59:38 INFO mapreduce.Job: Counters: 49  
File System Counters  
FILE: Number of bytes read=78  
FILE: Number of bytes written=286469
```

Step 8 – View the Output

```
hdfs dfs -cat /wordcount_output/part-r-00000
```

```
[cloudera@quickstart ~]$ hdfs dfs -cat /wordcount_output/part-r-00000  
Data 1  
Hadoop 1  
Hello 3  
world 1  
[cloudera@quickstart ~]$ █
```

Question b) Write a program in Map Reduce for Union operation.**Objective:**

To write and execute a Hadoop MapReduce program in Java to perform a **Union operation** on two datasets and output all unique records.

Software Used:

- Cloudera VM 5.13.0
- Eclipse Luna (pre-installed)
- Hadoop 2.6.0 (CDH 5.13.0)

Algorithm / Procedure:**Step 1 – Create New Java Project**

Open Eclipse Luna → File → New → Java Project

Project Name: UnionOperation

Finish

Step 2 – Add Hadoop JAR Libraries

Add the following libraries just like in WordCount:

```
/usr/lib/hadoop/*.jar
/usr/lib/hadoop-mapreduce/*.jar
/usr/lib/hadoop-hdfs/*.jar
/usr/lib/hadoop-yarn/*.jar
/usr/lib/hadoop/common/*.jar
```

Then click **Apply and Close** 

Step 3 – Create 3 Java Files inside src**UnionDriver.java**

```
import
org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;

public class UnionDriver {
    public static void main(String[] args) throws
        Exception { if (args.length != 3) {
            System.err.println("Usage: UnionDriver <input1> <input2> <output>");
            System.exit(1);
        }
        Configuration conf = new Configuration();
```

```
Job job = Job.getInstance(conf, "Union Operation");
```

```
job.setJarByClass(UnionDriver.class);
```

```

job.setReducerClass(UnionReducer.class);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);

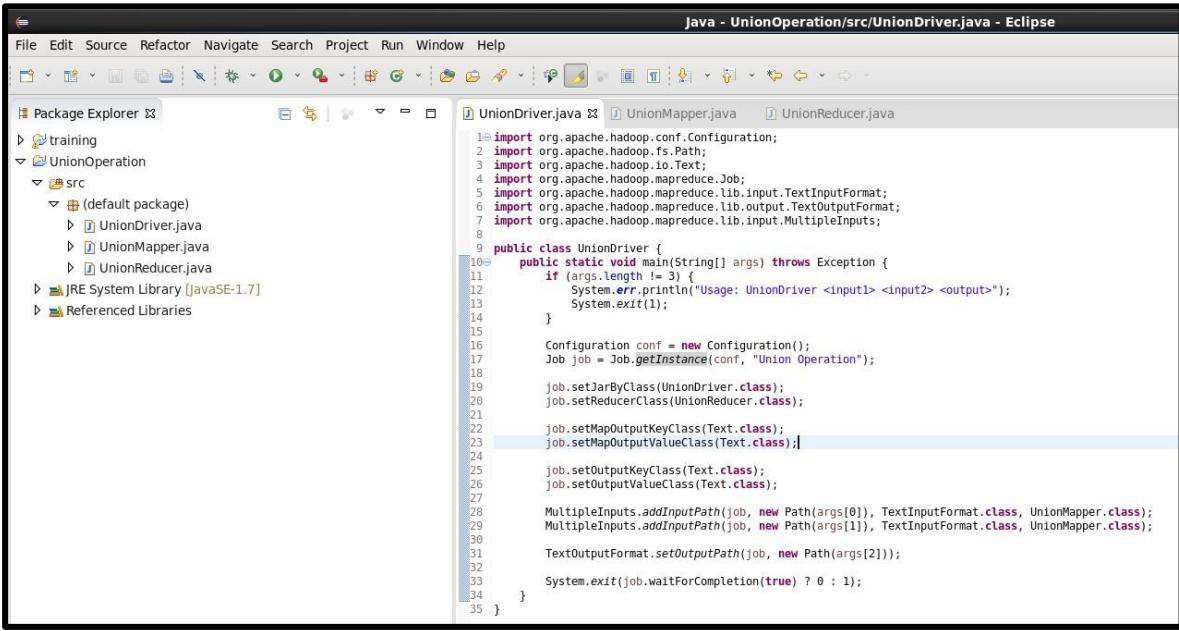
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);

MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,
UnionMapper.class);
MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
UnionMapper.class);

TextOutputFormat.setOutputPath(job, new Path(args[2]));

System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}

```



The screenshot shows the Eclipse IDE interface with the title bar "Java - UnionOperation/src/UnionDriver.java - Eclipse". The left side features the "Package Explorer" view, which displays a project structure under "src": "UnionDriver.java", "UnionMapper.java", and "UnionReducer.java". The right side is the code editor window showing the Java code for "UnionDriver.java". The code is as follows:

```

1 import org.apache.hadoop.conf.Configuration;
2 import org.apache.hadoop.fs.Path;
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Job;
5 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
6 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
7 import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
8
9 public class UnionDriver {
10     public static void main(String[] args) throws Exception {
11         if (args.length != 3) {
12             System.err.println("Usage: UnionDriver <input1> <input2> <output>");
13             System.exit(1);
14         }
15
16         Configuration conf = new Configuration();
17         Job job = Job.getInstance(conf, "Union Operation");
18
19         job.setJarByClass(UnionDriver.class);
20         job.setReducerClass(UnionReducer.class);
21
22         job.setMapOutputKeyClass(Text.class);
23         job.setMapOutputValueClass(Text.class);
24
25         job.setOutputKeyClass(Text.class);
26         job.setOutputValueClass(Text.class);
27
28         MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, UnionMapper.class);
29         MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, UnionMapper.class);
30
31         TextOutputFormat.setOutputPath(job, new Path(args[2]));
32
33         System.exit(job.waitForCompletion(true) ? 0 : 1);
34     }
35 }

```

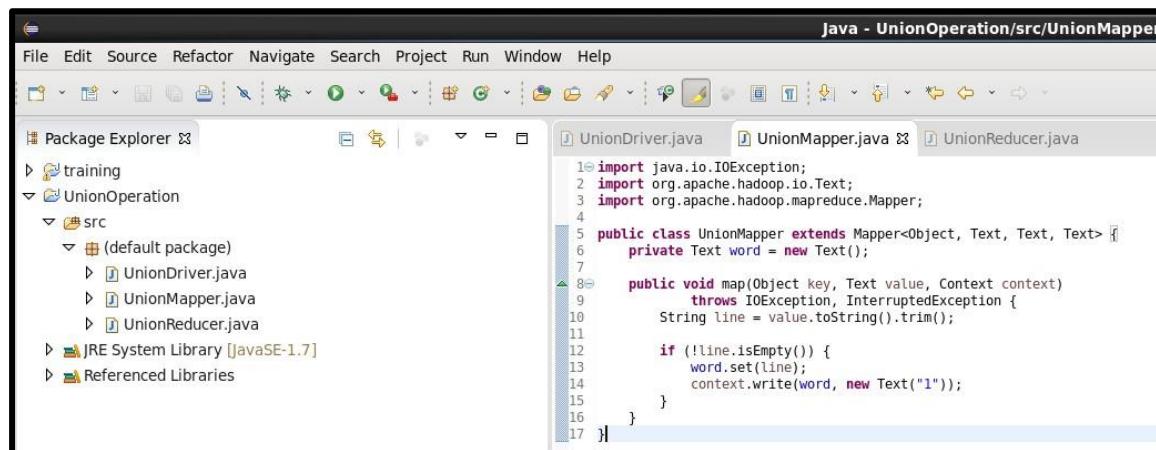
UnionMapper.java

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class UnionMapper extends Mapper<Object, Text, Text, Text> {
    private Text word = new Text();

    public void map(Object key, Text value, Context
        context) throws IOException,
        InterruptedException {
        String line = value.toString().trim();

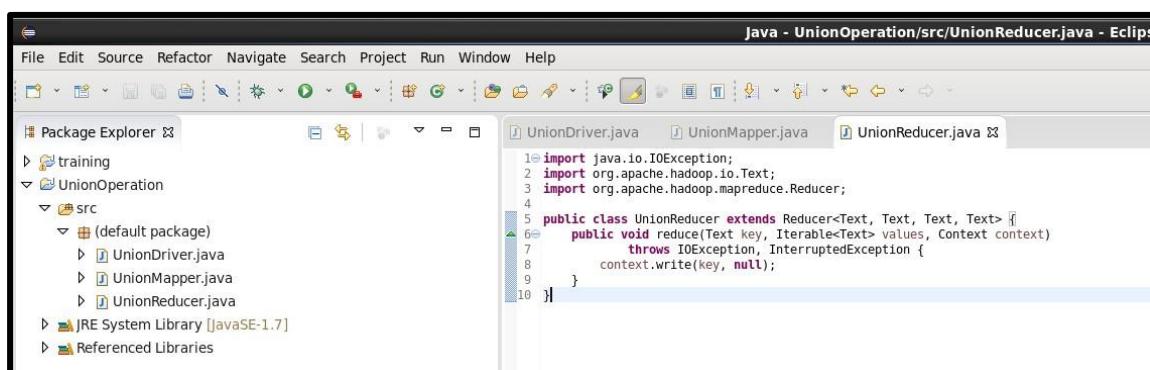
        if (!line.isEmpty()) {
            word.set(line);
            context.write(word, new Text("1"));
        }
    }
}
```



UnionReducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class UnionReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context
context)
        throws IOException, InterruptedException {
        context.write(key, null);
    }
}
```



Step 4 – Export Project as JAR

Right-click project → Export → JAR file
JAR file path: /home/cloudera/union.jar

Finish 

Step 5 – Create Input Files and Upload to HDFS

```
echo "nginx
CopyEdit
apple
banana
cherry" > file1.txt

echo "bash
CopyEdit
banana
date
apple" > file2.txt
```

Now create HDFS directories and upload:

```
hdfs dfs -mkdir /union_input1
hdfs dfs -mkdir /union_input2
hdfs dfs -put file1.txt /union_input1
```



```
[cloudera@quickstart Desktop]$ mkdir -p union_classes
[cloudera@quickstart Desktop]$ javac -classpath `hadoop classpath` -d union_classes UnionMapper.java UnionReducer.java UnionDriver.java
[cloudera@quickstart Desktop]$ ls union_classes
UnionDriver.class UnionMapper.class UnionReducer.class
[cloudera@quickstart Desktop]$ jar -cvf union.jar -C union_classes/ .
added manifest
adding: UnionMapper.class(in = 1474) (out= 637)(deflated 56%)
adding: UnionDriver.class(in = 1737) (out= 887)(deflated 48%)
adding: UnionReducer.class(in = 1263) (out= 488)(deflated 61%)
[cloudera@quickstart Desktop]$ jar tf union.jar
META-INF/
META-INF/MANIFEST.MF
UnionMapper.class
UnionDriver.class
UnionReducer.class
[cloudera@quickstart Desktop]$ hadoop jar union.jar UnionDriver /union_input/ufile1.txt /union_input/ufile2.txt /union_output
25/09/06 05:19:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
```

Step 6 – Run MapReduce Job

hadoop jar /home/cloudera/union.jar UnionDriver /union_input1 /union_input2 /union_output
If output folder already exists:
hdfs dfs -rm -r /union_output then re-run the job.

```
[cloudera@quickstart Desktop]$ hadoop jar union.jar UnionDriver /union_input/ufile1.txt /union_input/ufile2.txt /union_output
25/09/06 05:19:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
25/09/06 05:19:58 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
25/09/06 05:19:59 INFO input.FileInputFormat: Total input paths to process : 2
25/09/06 05:19:59 INFO mapreduce.JobSubmitter: number of splits:2
25/09/06 05:20:00 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1757159799958_0001
```

Step 7 – View Output

hdfs dfs -cat /union_output/part-r-00000

Expected Output:

```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /union_output/part-r-00000
CopyEdit
apple
banana
bash
cherry
date
nginx
[cloudera@quickstart Desktop]$ s
```

PRACTICAL NO: 3

**TO UNDERSTAND THE USE OF
MAPREDUCE FOR
INTERACTION
AND MATRIX OPERATIONS**

PRACTICAL NO: 3
TO UNDERSTAND THE USE OF MAPREDUCE FOR
INTERACTION
AND MATRIX OPERATIONS

Question a) Write a program in Map Reduce for Intersection operation.

Objective:

To write and execute a Hadoop MapReduce program in Java to perform an **Intersection operation** on two datasets and display common records.

Software Used:

- Cloudera VM 5.13.0
- Eclipse Luna (pre-installed)
- Hadoop 2.6.0 (CDH 5.13.0)

Algorithm / Procedure:

Step 1 – Create New Java Project

Open Eclipse Luna

File → New → Java Project

Project Name: Intersection

Finish

Step 2 – Add Hadoop JAR Libraries

Add the following libraries (same as before):

```
/usr/lib/hadoop/*.jar  
/usr/lib/hadoop-mapreduce/*.jar  
/usr/lib/hadoop-hdfs/*.jar  
/usr/lib/hadoop-yarn/*.jar  
/usr/lib/hadoop/common/*.jar
```

Then click **Apply and Close**.

Step 3 – Create 3 Java Files inside src folder

IntersectionMapper.java

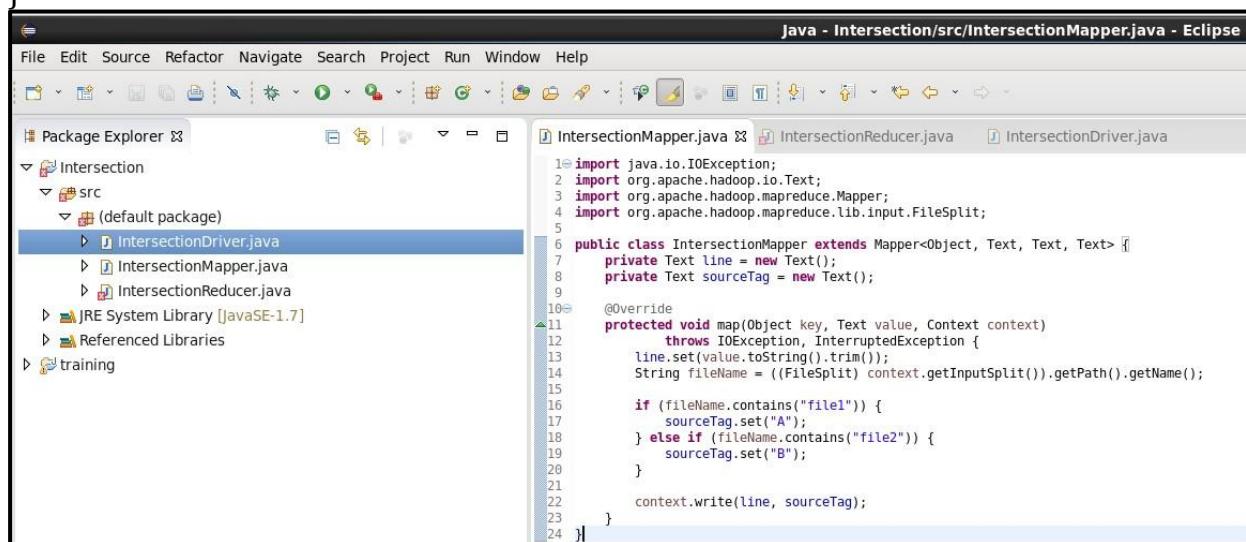
```
import java.io.IOException;
import
org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

public class IntersectionMapper extends Mapper<Object, Text, Text, Text> {
    private Text line = new Text();
    private Text sourceTag = new Text();

    @Override
    protected void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        line.set(value.toString().trim());
        String fileName = ((FileSplit) context.getInputSplit()).getPath().getName();

        if (fileName.contains("file1")) {
            sourceTag.set("A");
        } else if (fileName.contains("file2")) {
            sourceTag.set("B");
        }

        context.write(line, sourceTag);
    }
}
```



IntersectionReducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class IntersectionReducer extends Reducer<Text, Text, Text, Text> {
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
        boolean fromA = false;
        boolean fromB = false;

        for (Text val : values) {
            if (val.toString().equals("A")) fromA = true;
            else if (val.toString().equals("B")) fromB =
                true;
        }

        if (fromA && fromB) {
            context.write(key, null);
        }
    }
}
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under the "Intersection" package. The "src" folder contains three files: "IntersectionDriver.java" (selected), "IntersectionMapper.java", and "IntersectionReducer.java".
- Code Editor:** Displays the content of "IntersectionReducer.java". The code is identical to the one shown in the previous text block.
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and run.
- MenuBar:** Standard Eclipse menu bar with options like File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.

IntersectionDriver.java

```
import  
org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;  
import  
org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import  
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
  
public class IntersectionDriver {  
    public static void main(String[] args) throws  
        Exception { if (args.length != 3) {  
            System.err.println("Usage: IntersectionDriver <file1> <file2> <output>");  
            System.exit(2);  
        }  
  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "Intersection Operation");  
        job.setJarByClass(IntersectionDriver.class);  
  
        job.setReducerClass(IntersectionReducer.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(Text.class);  
  
        MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,  
        IntersectionMapper.class);  
        MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,  
        IntersectionMapper.class);  
  
        TextOutputFormat.setOutputPath(job, new Path(args[2]));  
  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}
```

```

java - Intersection/src/IntersectionDriver.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Intersection src (default package) IntersectionMapper.java IntersectionReducer.java IntersectionDriver.java
1 import org.apache.hadoop.conf.Configuration;
2 import org.apache.hadoop.fs.Path;
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Job;
5 import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
6 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
7 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
8
9 public class IntersectionDriver {
10     public static void main(String[] args) throws Exception {
11         if (args.length != 3) {
12             System.err.println("Usage: IntersectionDriver <file1> <file2> <output>");
13             System.exit(2);
14         }
15
16         Configuration conf = new Configuration();
17         Job job = Job.getInstance(conf, "Intersection Operation");
18         job.setJarByClass(IntersectionDriver.class);
19
20         job.setReducerClass(IntersectionReducer.class);
21         job.setOutputKeyClass(Text.class);
22         job.setOutputValueClass(Text.class);
23
24         MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, IntersectionMapper.class);
25         MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, IntersectionMapper.class);
26
27         TextOutputFormat.setOutputPath(job, new Path(args[2]));
28
29         System.exit(job.waitForCompletion(true) ? 0 : 1);
30     }
31 }

```

Step 4 – Export Project as JAR

Right-click project → Export → JAR file

Save as:

/home/cloudera/intersection.jar Finish

Step 5 – Create Input Files and Upload to HDFS

```

echo "apple
banana
orange
mango" > file1.txt

echo "banana
mango
grape" > file2.txt

```

Now create HDFS directories and upload:

hdfs dfs -mkdir

/inter_input1 hdfs dfs -

mkdir /inter_input2

hdfs dfs -put file1.txt /inter_input1

hdfs dfs -put file2.txt /inter_input2

Step 6 – Run MapReduce Job

hadoop jar /home/cloudera/intersection.jar IntersectionDriver /inter_input1 /inter_input2 /inter_output

Step 7 – View Output

```
hdfs dfs -cat /inter_output/part-r-00000
```

Expected Output:

```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /inter_output/part-r-00000
banana
mango
[cloudera@quickstart Desktop]$ █
```

Question b) Write a program in Map Reduce for Matrix Multiplication

Objective:

To write and execute a Hadoop MapReduce program in Java to perform **Matrix Multiplication ($C = A \times B$)** where:

- Matrix A has dimensions $m \times n$
- Matrix B has dimensions $n \times p$
- The resultant Matrix C has dimensions $m \times p$

Software Used:

- Cloudera VM 5.13.0
- Eclipse Luna (pre-installed)
- Hadoop 2.6.0 (CDH 5.13.0)

Algorithm / Logic

Input Format:

Each line of input represents one matrix element.

A,i,k,value

B,k,j,value

A/B → Matrix identifier

i, k, j → Indices

value → Numeric value

Mapper Logic:

- For every element in **A**, emit (i,j) for all possible columns j of **B**
→ $(A,k,value)$
- For every element in **B**, emit (i,j) for all possible rows i of **A**
→ $(B,k,value)$

Reducer Logic:

- Joins A and B records on same k (column of A = row of B)
- Computes
- $C[i][j] = \sum (A[i][k] \times B[k][j])$

Driver Logic:

- Configures Mapper and Reducer
- Sets input and output paths
- Submits and executes the job

Step 1 – Create Eclipse Project

Open Eclipse Luna →

File → New → Java Project → Name it: MatrixMultiplication → Finish 

Step 2 – Create 3 Java Files inside src folder

MatrixMultiplyDriver.java

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

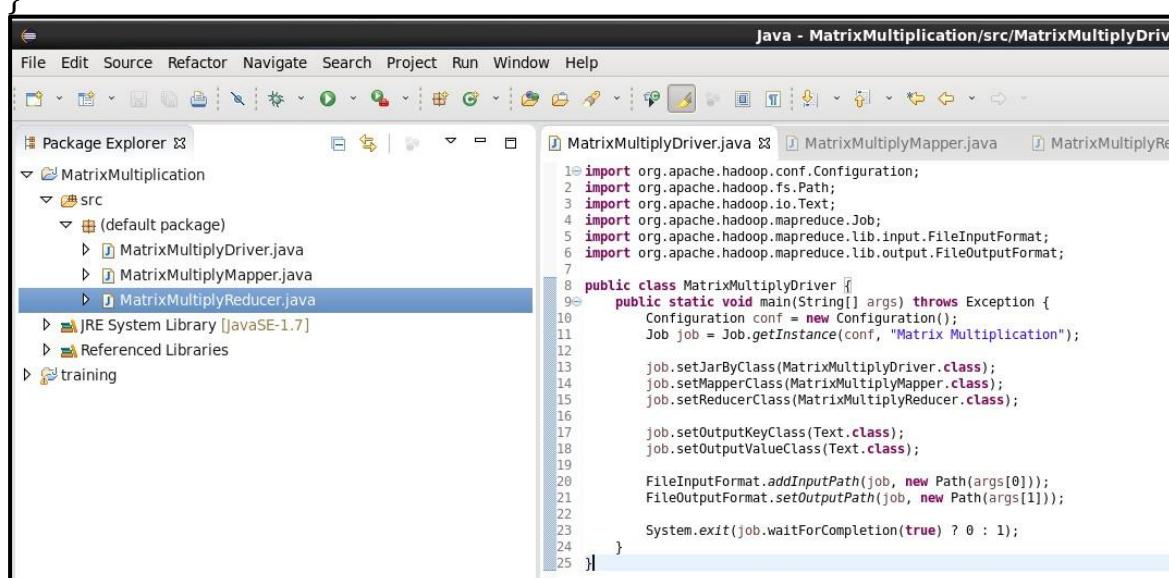
public class MatrixMultiplyDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Matrix Multiplication");

        job.setJarByClass(MatrixMultiplyDriver.class);
        job.setMapperClass(MatrixMultiplyMapper.class);
        job.setReducerClass(MatrixMultiplyReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```



MatrixMultiplyMapper.java

```

import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MatrixMultiplyMapper extends Mapper<Object, Text, Text,
Text> { private int m = 2; // rows in A
private int p = 2; // columns in B
public void map(Object key, Text value, Context
context) throws IOException,
InterruptedException {

String[] parts =
value.toString().split(",");
String matrix
= parts[0];
int row =
Integer.parseInt(parts[1]); int col
= Integer.parseInt(parts[2]); int
val = Integer.parseInt(parts[3]); if
(matrix.equals("A")) {
for (int j = 0; j < p; j++) {
context.write(new Text(row + "," + j), new Text("A," + col + "," + val));
}
} else if (matrix.equals("B")) {
for (int i = 0; i < m; i++) {
context.write(new Text(i + "," + col), new Text("B," + row + "," + val));
}
}
}
}

```

The screenshot shows the Eclipse IDE environment with the Java - MatrixMultiplication/src/MatrixMultiplyMapper.java - Eclipse tab active. The code in the editor matches the provided snippet. The Package Explorer on the left shows the project structure with files MatrixMultiplyDriver.java, MatrixMultiplyMapper.java, and MatrixMultiplyReducer.java. The JRE System Library [JAVASE-1.7] and Referenced Libraries are also listed.

```

1 import java.io.IOException;
2 import org.apache.hadoop.io.Text;
3 import org.apache.hadoop.mapreduce.Mapper;
4
5 public class MatrixMultiplyMapper extends Mapper<Object, Text, Text, Text> {
6     private int m = 2; // rows in A
7     private int p = 2; // columns in B
8
9     public void map(Object key, Text value, Context context)
throws IOException, InterruptedException {
10
11         String[] parts = value.toString().split(",");
12         String matrix = parts[0];
13         int row = Integer.parseInt(parts[1]);
14         int col = Integer.parseInt(parts[2]);
15         int val = Integer.parseInt(parts[3]);
16
17         if (matrix.equals("A")) {
18             for (int j = 0; j < p; j++) {
19                 context.write(new Text(row + "," + j), new Text("A," + col + "," + val));
20             }
21         } else if (matrix.equals("B")) {
22             for (int i = 0; i < m; i++) {
23                 context.write(new Text(i + "," + col), new Text("B," + row + "," + val));
24             }
25         }
26     }
27 }
28
29

```

MatrixMultiplyReducer.java

```

import java.io.IOException;
import java.util.HashMap;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MatrixMultiplyReducer extends Reducer<Text, Text, Text, Text>
{
    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {

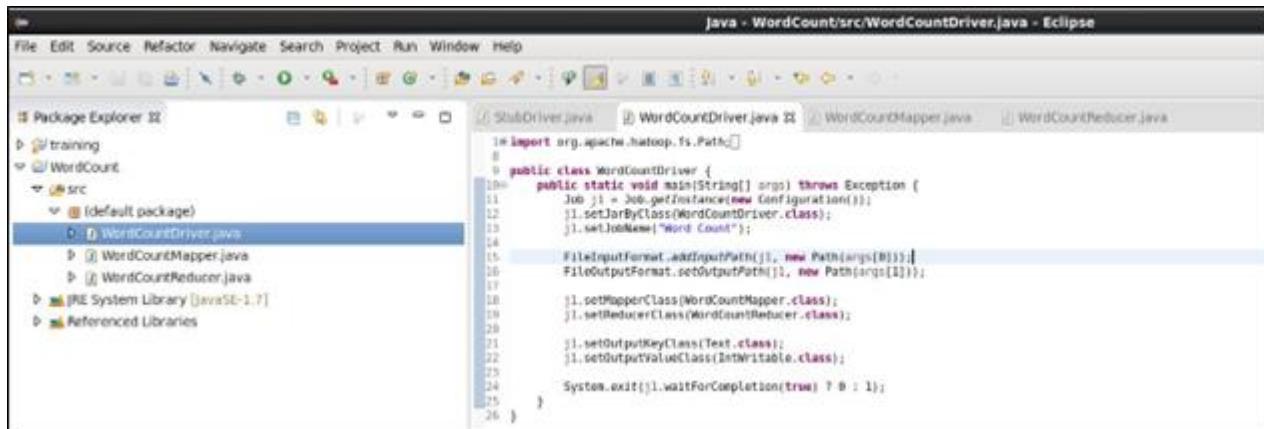
        HashMap<Integer, Integer> mapA = new HashMap<>();
        HashMap<Integer, Integer> mapB = new HashMap<>();

        for (Text val : values) {
            String[] parts =
                val.toString().split(",");
            if
                (parts[0].equals("A")) {
                    mapA.put(Integer.parseInt(parts[1]), Integer.parseInt(parts[2]));
                } else {
                    mapB.put(Integer.parseInt(parts[1]), Integer.parseInt(parts[2]));
                }
        }

        int sum = 0;
        for (int k : mapA.keySet()) {
            if (mapB.containsKey(k))
            {
                sum += mapA.get(k) * mapB.get(k);
            }
        }

        context.write(key, new Text(Integer.toString(sum)));
    }
}

```



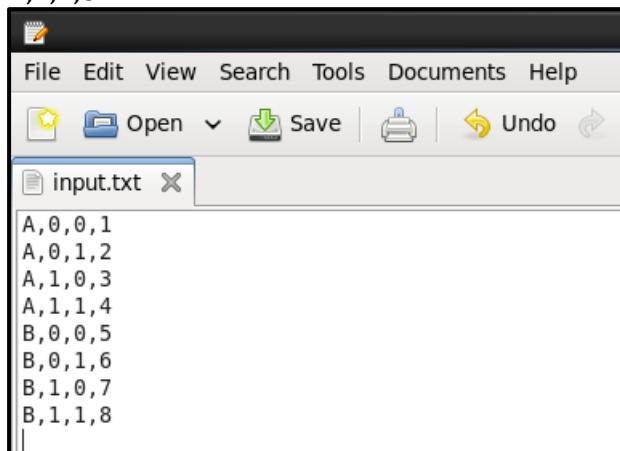
Step 3 - Export as JAR

Right-click project → Export → JAR file
Save as: /home/cloudera/matrixmul.jar
Finish

Step 4 – Prepare Input Data

Create input.txt:

A,0,0,1
A,0,1,2
A,1,0,3
A,1,1,4
B,0,0,5
B,0,1,6
B,1,0,7
B,1,1,8



Upload to HDFS:

```
hdfs dfs -mkdir /matrix_input
hdfs dfs -put input.txt /matrix_input
```

Step 5 – Run the MapReduce Job

hadoop jar /home/cloudera/matrixmul.jar MatrixMultiplyDriver /matrix_input /matrix_output then re-run the job.

Step 6 – View Output

```
hdfs dfs -cat /matrix_output/part-r-00000
```

Expected Output:

```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /matrix_output/part-r-00000
0,0      19
0,1      22
1,0      43
1,1      50
[cloudera@quickstart Desktop]$
```

PRACTICAL NO: 4

TO GET FAMILIAR WITH MONGODB

PRACTICAL NO: 4 TO GET FAMILIAR WITH MONGODB

Question a) Installation

1. Installation of MongoDB Steps:

1. Download and install **MongoDB Server** from the official website.
2. Install **MongoDB Shell (mongosh)** for command-line interaction.
3. Start the MongoDB service and open mongosh from CMD or PowerShell.



Command:

```
mongosh
```

```
C:\Windows\System32>mongosh
Current Mongosh Log ID: 69015d850bff9ca894cebea3
Connecting to:      mongod://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB:     8.2.1
Using Mongosh:    2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
The server generated these startup warnings when booting
2025-10-29T05:38:25.686+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
```

Question b) Sample Database Creation

Commands:

```
use sampleDB
```

This command creates and switches to a new database named sampleDB.

```
db.createCollection("students")
```

Creates a new collection named students.

```
test> use sampleDB
switched to db sampleDB
sampleDB> db.createCollection("students")
{ ok: 1 }
```

Insert Sample Data into Collection

Commands:

```
db.students.insertMany([
  { name: "Aakash", age: 23, course: "MCA", city: "Navi Mumbai" },
  { name: "Priya", age: 22, course: "BCA", city: "Mumbai" },
  { name: "Rohan", age: 24, course: "MSc", city: "Thane" }
])
```

```
sampleDB> db.students.insertMany([
...   { name: "Aakash", age: 23, course: "MCA", city: "Navi Mumbai" },
...   { name: "Priya", age: 22, course: "BCA", city: "Mumbai" },
...   { name: "Rohan", age: 24, course: "MSc", city: "Thane" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69015dd60bff9ca894cebea4'),
    '1': ObjectId('69015dd60bff9ca894cebea5'),
    '2': ObjectId('69015dd60bff9ca894cebea6')
  }
}
```

Question c) Query the Sample Database using MongoDB querying commands**Display All Documents:**

```
db.students.find()  
sampleDB> db.students.find()  
[  
  {  
    _id: ObjectId('69015dd60bff9ca894cebea4'),  
    name: 'Aakash',  
    age: 23,  
    course: 'MCA',  
    city: 'Navi Mumbai'  
  },  
  {  
    _id: ObjectId('69015dd60bff9ca894cebea5'),  
    name: 'Priya',  
    age: 22,  
    course: 'BCA',  
    city: 'Mumbai'  
  },  
  {  
    _id: ObjectId('69015dd60bff9ca894cebea6'),  
    name: 'Rohan',  
    age: 24,  
    course: 'MSc',  
    city: 'Thane'  
  }  
]
```

Filter Query (Example):

```
db.students.find({ city: "Mumbai" })  
sampleDB> db.students.find({ city: "Mumbai" })  
[  
  {  
    _id: ObjectId('69015dd60bff9ca894cebea5'),  
    name: 'Priya',  
    age: 22,  
    course: 'BCA',  
    city: 'Mumbai'  
  }  
]
```

Sorting (Ascending by Age):

```
db.students.find().sort({ age: 1 })
```

```
sampleDB> db.students.find().sort({ age: 1 })
[
  {
    _id: ObjectId('69015dd60bff9ca894cebea5'),
    name: 'Priya',
    age: 22,
    course: 'BCA',
    city: 'Mumbai'
  },
  {
    _id: ObjectId('69015dd60bff9ca894cebea4'),
    name: 'Aakash',
    age: 23,
    course: 'MCA',
    city: 'Navi Mumbai'
  },
  {
    _id: ObjectId('69015dd60bff9ca894cebea6'),
    name: 'Rohan',
    age: 24,
    course: 'MSc',
    city: 'Thane'
  }
]
```

5. Update a**Document****Command:**

```
db.students.updateOn
```

```
e(
```

```
  { name: "Aakash" },
  { $set: { city: "Panvel" } }
```

```
)
```

```
sampleDB> db.students.updateOne(
```

```
...  { name: "Aakash" },
...  { $set: { city: "Panvel" } }
```

```
... )
```

```
...
```

```
{
```

```
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
```

```
}
```

6. Delete a Document Command:

```
db.students.deleteOne({ name: "Rohan" })
```

```
sampleDB> db.students.deleteOne({ name: "Rohan" })
{ acknowledged: true, deletedCount: 1 }
sampleDB>
```

7. Display Total Number of Documents Command:

```
db.students.countDocuments()
```

```
sampleDB> db.students.deleteOne({ name: "Rohan" })
{ acknowledged: true, deletedCount: 1 }
sampleDB> db.students.countDocuments()
2
```

8. Drop the Collection (Optional) Command:

```
db.students.drop()
```

```
sampleDB> db.students.drop()
true
```

PRACTICAL NO: 5

**TO WORK WITH THE VARIOUS CRUD
OPERATIONS
USING MONGODB**

PRACTICAL NO: 5
TO WORK WITH THE VARIOUS CRUD OPERATIONS USING
MONGODE

Question a) Create Collection

1. create a database named salesdb and switch to salesdb
2. We have two collections namely sales and sales_profile. Let us populate them. First run a query to insert data in sales_profile.

```
test> use salesdb;
switched to db salesdb
salesdb> db.sales_profile.insertMany([
... { id:1, invoice_no:"750-67-8428", card:"Member", gender:"Female" } ,
... { id:2, invoice_no:"226-31-3081", card:"Normal", gender:"Female" } ,
... { id:3, invoice_no:"631-41-3108", card:"Normal", gender:"Male" } ,
... { id:4, invoice_no:"123-19-1176", card:"Member", gender:"Male" } ,
... { id:5, invoice_no:"373-73-7910", card:"Normal", gender:"Male" } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("633896ee748becd9a0a5da6c"),
    '1': ObjectId("633896ee748becd9a0a5da6d"),
    '2': ObjectId("633896ee748becd9a0a5da6e"),
    '3': ObjectId("633896ee748becd9a0a5da6f"),
    '4': ObjectId("633896ee748becd9a0a5da70")
}
```

Question b) Insert Document

```
test> use salesdb;
switched to db salesdb
salesdb> db.sales_profile.insertMany([
... { id:1, invoice_no: "750-67-8428", card:"Member", gender:"Female"} ,
... { id:2, invoice_no: "226-31-3081", card:"Normal", gender:"Female"} ,
... { id:3, invoice_no: "631-41-3108", card:"Normal", gender:"Male"} ,
... { id:4, invoice_no: "123-19-1176", card:"Member", gender:"Male"} ,
... { id:5, invoice_no: "373-73-7910", card:"Normal", gender:"Male"} ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("633896ee748becd9a0a5da6c"),
    '1': ObjectId("633896ee748becd9a0a5da6d"),
    '2': ObjectId("633896ee748becd9a0a5da6e"),
    '3': ObjectId("633896ee748becd9a0a5da6f"),
    '4': ObjectId("633896ee748becd9a0a5da70")
  }
}
```

Question c) Query Document

```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lab1-89\Downloads\mongodb-database-tools-windows-x86_64-100.6.1\bin>mongoimport --db=salesdb --collection=sales -
--type=csv --headerline --file=sales_profile.csv
2022-11-14T11:52:07.541+0530      connected to: mongodb://localhost/
2022-11-14T11:52:07.892+0530      2823 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\Lab1-89\Downloads\mongodb-database-tools-windows-x86_64-100.6.1\bin>
```

Find data query

Query : db.sales.find()

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000

salesdb> db.sales.find()
[
  {
    _id: ObjectId("6371de8f8976cbf6d0144748"),
    ORDERNUMBER: 10107,
    QUANTITYORDERED: 30,
    PRICEEACH: 95.7,
    ORDERLINENUMBER: 2,
    SALES: 2871,
    ORDERDATE: '2/24/2003 0:00',
    STATUS: 'Shipped',
    QTR_ID: 1,
    MONTH_ID: 2,
    YEAR_ID: 2003,
    PRODUCTLINE: 'Motorcycles',
    MSRP: 95,
    PRODUCTCODE: 'S10_1678',
    CUSTOMERNAME: 'Land of Toys Inc.',
    PHONE: 2125557818,
    ADDRESSLINE1: '897 Long Airport Avenue',
    ADDRESSLINE2: '',
    CITY: 'NYC',
    STATE: 'NY',
    POSTALCODE: 10022,
    COUNTRY: 'USA',
    TERRITORY: 'NA',
    CONTACTLASTNAME: 'Yu',
    CONTACTFIRSTNAME: 'Kwai',
    DEALSIZE: 'Small'
```

```
db.sales.find( { CITY : "Salzburg" , STATUS :"Shipped"}).limit(5)
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
salesdb> db.sales.find( { CITY : "Salzburg" , STATUS :"Shipped"}).limit(5)
[
  {
    _id: ObjectId("6371de8f8976cbf6d014475b"),
    ORDERNUMBER: 10341,
    QUANTITYORDERED: 41,
    PRICEEACH: 100,
    ORDERLINENUMBER: 9,
    SALES: 7737.93,
    ORDERDATE: '11/24/2004 0:00',
    STATUS: 'Shipped',
    QTR_ID: 4,
    MONTH_ID: 11,
    YEAR_ID: 2004,
    PRODUCTLINE: 'Motorcycles',
    MSRP: 95,
    PRODUCTCODE: 'S10_1678',
    CUSTOMERNAME: 'Salzburg Collectables',
    PHONE: '6562-9555',
    ADDRESSLINE1: 'Geislweg 14',
    ADDRESSLINE2: '',
    CITY: 'Salzburg',
    STATE: '',
    POSTALCODE: 5020,
    COUNTRY: 'Austria',
    TERRITORY: 'EMEA',
    CONTACTLASTNAME: 'Pipps',
    CONTACTFIRSTNAME: 'Georg',
    DEALSIZE: 'Large'
  },
]
```

Find data using greater than condition

```
db.sales.find( {QUANTITYORDERED: { $gt:33 }, PRICEEACH: { $gt:33 }}, { CITY : "Salzburg" , STATUS :"Shipped"}).limit(5)
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
salesdb> db.sales.find( {QUANTITYORDERED: { $gt:33 }, PRICEEACH: { $gt:33 }}, { CITY : "Salzburg" , STATUS :"Shipped"}).limit(5)
[
  {
    _id: ObjectId("6371de8f8976cbf6d0144749"),
    STATUS: 'Shipped',
    CITY: 'Reims'
  },
  {
    _id: ObjectId("6371de8f8976cbf6d014474a"),
    STATUS: 'Shipped',
    CITY: 'Paris'
  },
  {
    _id: ObjectId("6371de8f8976cbf6d014474b"),
    STATUS: 'Shipped',
    CITY: 'Pasadena'
  }
]
```

Question d) Delete Document

Update document

```
db.sales.updateMany( { CITY : "Salzburg"}, { $set :{ CITY:"MUMBAI" }})
salesdb> db.sales.updateMany( { CITY : "Salzburg"}, ...
...   { $set :{ CITY:"MUMBAI" }})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 40,
  modifiedCount: 40,
  upsertedCount: 0
}
salesdb>
```

Delete document

```
db.sales.deleteOne({CITY : "MUMBAI"})
salesdb> db.sales.deleteOne({CITY: "MUMBAI"})
{ acknowledged: true, deletedCount: 1 }
salesdb>
```

Question e) Indexing

Aggregate function

```
db.sales.aggregate([{$lookup: {from: "sales_profile", localField: "ID", foreignField: "ID", as: "QUNTTITY"} }])
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
salesdb> db.sales.aggregate([{$lookup: { from: "sales_profile", localField: "ID", foreignField: "ID", as: "QUNTTITY" } }])
[ {
    _id: ObjectId("6371de8f8976cbf6d0144748"),
    ORDERNUMBER: 10107,
    QUANTITYORDERED: 30,
    PRICEEACH: 95.7,
    ORDERLINENUMBER: 2,
    SALES: 2871,
    ORDERDATE: '2/24/2003 0:00',
    STATUS: 'Shipped',
    QTR_ID: 1,
    MONTH_ID: 2,
    YEAR_ID: 2003,
    PRODUCTLINE: 'Motorcycles',
    MSRP: 95,
    PRODUCTCODE: 'S10_1678',
    CUSTOMERNAME: 'Land of Toys Inc.',
    PHONE: 2125557818,
    ADDRESSLINE1: '897 Long Airport Avenue',
    ADDRESSLINE2: '',
    CITY: 'NYC',
    STATE: 'NY',
    POSTALCODE: 10022,
    COUNTRY: 'USA',
    TERRITORY: 'NA',
    CONTACTLASTNAME: 'Yu',
    CONTACTFIRSTNAME: 'Kwai',
    DEALSIZE: 'Small',
    QUNTTITY: []
}
```

9. Extract the salesdb database in json

```
mongoexport --collection=sales --db=salesdb --out=sales.json
```

```
C:\Users\Lab1-89\Downloads\mongodb-database-tools-windows-x86_64-100.6.1\bin>mongoexport --collection=sales --db=salesdb --out=sales.json
2022-11-14T12:59:26.100+0530    connected to: mongodb://localhost/
2022-11-14T12:59:26.282+0530    exported 2822 records
C:\Users\Lab1-89\Downloads\mongodb-database-tools-windows-x86_64-100.6.1\bin>
```

PRACTICAL NO: 6

**TO UNDERTAND THE CONCEPTS
OF HIVEQL**

PRACTICAL NO: 6
TO UNDERTAND THE CONCEPTS OF HIVEQL

Question a) Hive Data Types

Apache Hive Practical Journal

Environment: Cloudera 5.13.0

(VMware) Tool: Apache Hive

Setup: Opening Hive Shell

1. Start your Cloudera VM in VMware.
2. Open the terminal on the Cloudera desktop and
type: hiv

This command opens the Hive shell. The prompt will look like this: hive>

1. Hive Data Types

Common Hive Data

Types:

- Numeric: TINYINT, INT, BIGINT, FLOAT, DOUBLE
- String: STRING, VARCHAR(n), CHAR(n)
- Date/Time: DATE, TIMESTAMP
- Misc: BOOLEAN, ARRAY, MAP, STRUCT

Example:

```
>CREATE TABLE datatype_demo (
> id INT,
> name STRING,
> salary DOUBLE,
> doj DATE,
> is_active BOOLEAN
>)
>ROW FORMAT DELIMITED
>FIELDS TERMINATED BY ','
>STORED AS TEXTFILE;
```

```
cloudera@quickstart:~$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE TABLE datatype_demo (
    >   id INT,
    >   name STRING,
    >   salary DOUBLE,
    >   doj DATE,
    >   is_active BOOLEAN
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.607 seconds
hive> █
```

```
cloudera@quickstart:~$ File Edit View Search Terminal Help
OK
Time taken: 0.607 seconds
hive> CREATE DATABASE company_db;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask.
Database company_db already exists
hive> SHOW DATABASES;
OK
company_db
default
Time taken: 0.102 seconds, Fetched: 2 row(s)
hive> USE company_db;
OK
Time taken: 0.028 seconds
hive> CREATE TABLE employees (
    >   emp_id INT,
    >   emp_name STRING,
    >   dept STRING,
    >   salary DOUBLE
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
```

```
cloudera@quickstart:~
```

```
File Edit View Search Terminal Help
```

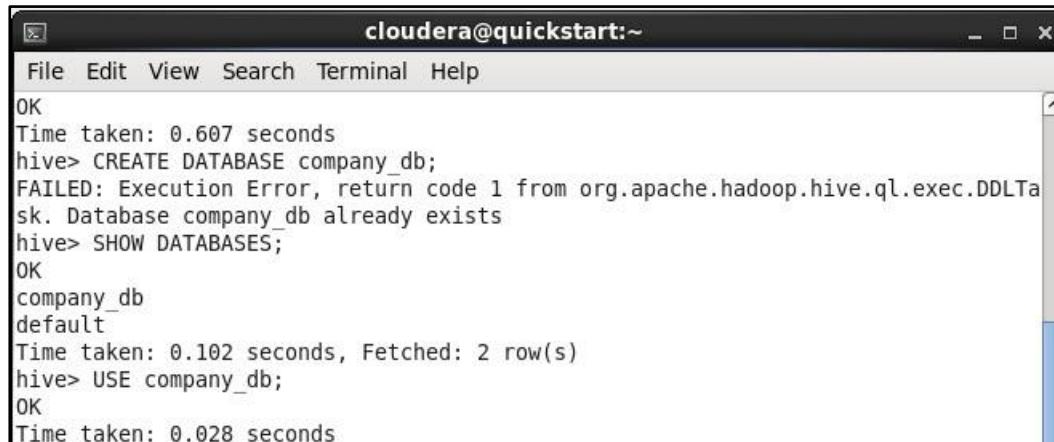
```
sales
Time taken: 0.191 seconds, Fetched: 4 row(s)
hive> LOAD DATA LOCAL INPATH '/home/cloudera/employees.txt' INTO TABLE employees
;
Loading data to table company_db.employees
Table company_db.employees stats: [numFiles=3, totalSize=213]
OK
Time taken: 0.85 seconds
hive> LOAD DATA LOCAL INPATH '/home/cloudera/departments.txt' INTO TABLE departments;
Loading data to table company_db.departments
Table company_db.departments stats: [numFiles=3, totalSize=99]
OK
Time taken: 0.789 seconds
hive> SELECT * FROM employees;
OK
1      Ravi    IT      50000.0
2      Amit    HR      45000.0
3      Suman   IT      60000.0
4      Neha    Finance 55000.0
NULL  NULL    NULL    NULL
```

Question b) Create Database & Table in Hive

2. Create Database and

Table Create Database:

```
>CREATE DATABASE company_db;  
>SHOW DATABASES;  
>USE company_db;
```

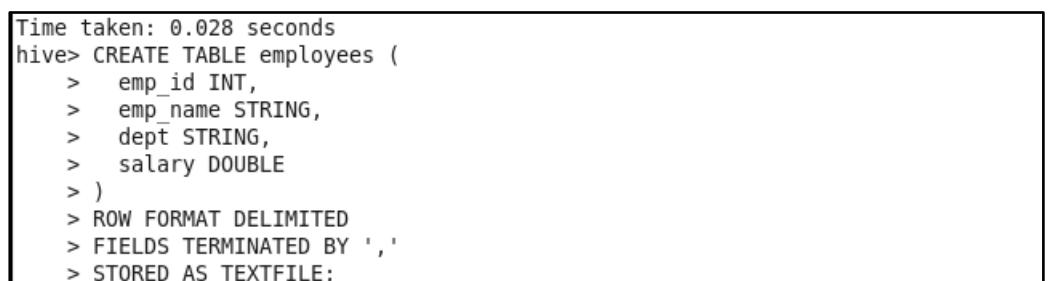


The screenshot shows a terminal window titled "cloudera@quickstart:~". The window contains the following text:

```
File Edit View Search Terminal Help  
OK  
Time taken: 0.607 seconds  
hive> CREATE DATABASE company_db;  
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask.  
Database company_db already exists  
hive> SHOW DATABASES;  
OK  
company_db  
default  
Time taken: 0.102 seconds, Fetched: 2 row(s)  
hive> USE company_db;  
OK  
Time taken: 0.028 seconds
```

Create Table:

```
CREATE TABLE employees (  
    emp_id INT,  
    emp_name STRING,  
    dept STRING,  
    salary DOUBLE  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
STORED AS TEXTFILE;
```



The screenshot shows a terminal window containing the following text:

```
Time taken: 0.028 seconds  
hive> CREATE TABLE employees (  
    >     emp_id INT,  
    >     emp_name STRING,  
    >     dept STRING,  
    >     salary DOUBLE  
    > )  
    > ROW FORMAT DELIMITED  
    > FIELDS TERMINATED BY ','  
    > STORED AS TEXTFILE;
```

Load Data:

Create a local file in terminal:

nano /home/cloudera/employees.txt

Add the following data:

1,Ravi,IT,50000
2,Amit,HR,45000
3,Suman,IT,60000
4,Neha,Finance,55000

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
GNU nano 2.0.9      File: /home/cloudera/employees.txt  
1,Ravi,IT,50000  
2,Amit,HR,45000  
3,Suman,IT,60000  
4,Neha,Finance,55000
```

Then execute in Hive:

>LOAD DATA LOCAL INPATH '/home/cloudera/employees.txt' INTO TABLE employees;

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
sales  
Time taken: 0.191 seconds, Fetched: 4 row(s)  
hive> LOAD DATA LOCAL INPATH '/home/cloudera/employees.txt' INTO TABLE employees  
;  
Loading data to table company_db.employees  
Table company_db.employees stats: [numFiles=3, totalSize=213]  
OK  
Time taken: 0.85 seconds
```

>SELECT * FROM employees;

```
hive> SELECT * FROM employees;  
OK  
1      Ravi    IT      50000.0  
2      Amit    HR      45000.0  
3      Suman   IT      60000.0  
4      Neha    Finance 55000.0  
NULL  NULL    NULL    NULL
```

Question c) Hive Partitioning

3. Hive Partitioning

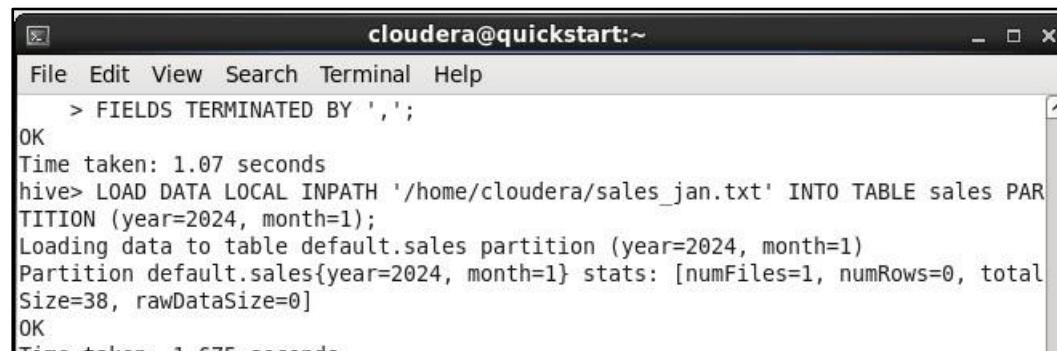
Partitioning helps improve query

performance. Example:

```
CREATE TABLE sales (
    id INT,
    item STRING,
    amount DOUBLE
)
>PARTITIONED BY (year INT, month INT)
>ROW FORMAT DELIMITED
>FIELDS TERMINATED BY ',';
```

```
hive> CREATE TABLE sales (
    >   id INT,
    >   item STRING,
    >   amount DOUBLE
    > )
    > PARTITIONED BY (year INT, month INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 1.07 seconds
```

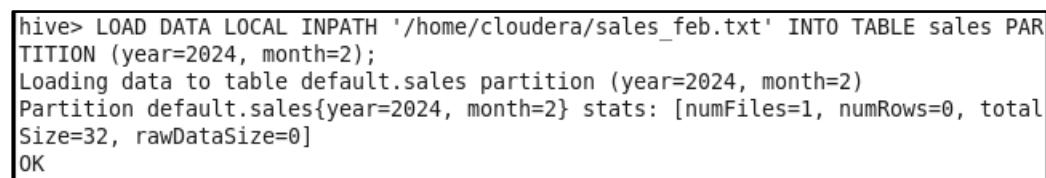
>LOAD DATA LOCAL INPATH '/home/cloudera/sales_jan.txt' INTO TABLE sales



```
cloudera@quickstart:~$ 
File Edit View Search Terminal Help
> FIELDS TERMINATED BY ',';
OK
Time taken: 1.07 seconds
hive> LOAD DATA LOCAL INPATH '/home/cloudera/sales_jan.txt' INTO TABLE sales PARTITION (year=2024, month=1);
Loading data to table default.sales partition (year=2024, month=1)
Partition default.sales{year=2024, month=1} stats: [numFiles=1, numRows=0, totalSize=38, rawDataSize=0]
OK
Time taken: 1.675 seconds
```

PARTITION (year=2024, month=1);

>LOAD DATA LOCAL INPATH '/home/cloudera/sales_feb.txt' INTO TABLE sales



```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/sales_feb.txt' INTO TABLE sales PARTITION (year=2024, month=2);
Loading data to table default.sales partition (year=2024, month=2)
Partition default.sales{year=2024, month=2} stats: [numFiles=1, numRows=0, totalSize=32, rawDataSize=0]
OK
Time taken: 0.686 seconds
```

PARTITION (year=2024, month=2);

```
>SHOW PARTITIONS sales;
```

```
hive>
    > SHOW PARTITIONS sales;
OK
year=2024/month=1
year=2024/month=2
Time taken: 0.248 seconds, Fetched: 2 row(s)
hive> █
```

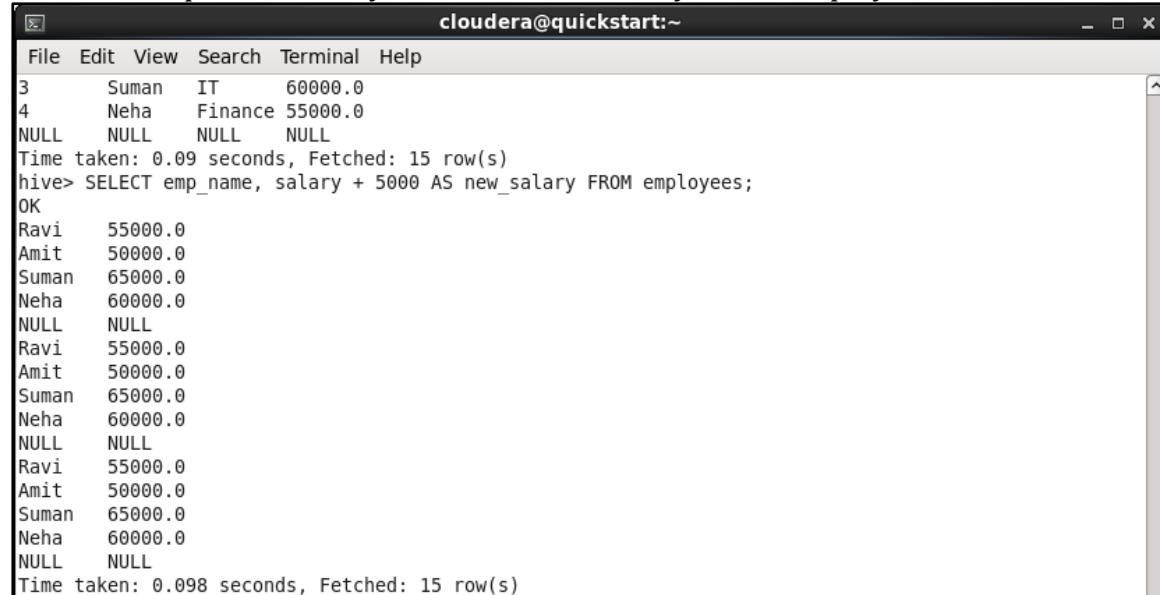
PRACTICAL NO: 7

**TO UNDERTAND THE USE OF OPERATORS,
FUNCTIONS,
VIEWS ETC USING HIVEQL**

PRACTICAL NO: 7
TO UNDERTAND THE USE OF OPERATORS, FUNCTIONS,
VIEWS ETC USING HIVEQL

Question a) Hive Built-In Operators

```
>SELECT emp_name, salary + 5000 AS new_salary FROM employees;
```

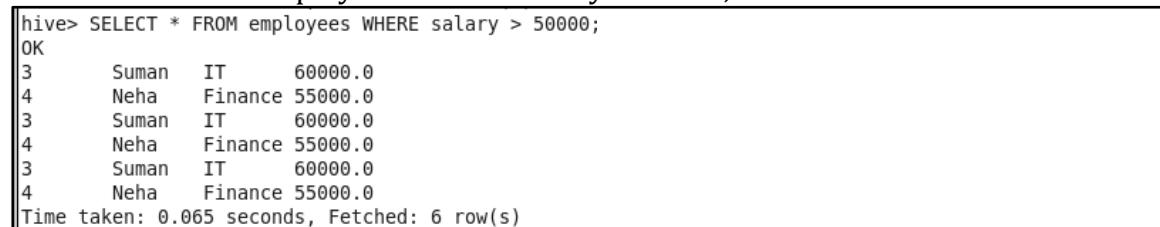


```

File Edit View Search Terminal Help
3      Suman    IT    60000.0
4      Neha     Finance 55000.0
NULL    NULL    NULL    NULL
Time taken: 0.09 seconds, Fetched: 15 row(s)
hive> SELECT emp_name, salary + 5000 AS new_salary FROM employees;
OK
Ravi    55000.0
Amit    50000.0
Suman   65000.0
Neha    60000.0
NULL    NULL
Ravi    55000.0
Amit    50000.0
Suman   65000.0
Neha    60000.0
NULL    NULL
Ravi    55000.0
Amit    50000.0
Suman   65000.0
Neha    60000.0
NULL    NULL
Time taken: 0.098 seconds, Fetched: 15 row(s)

```

```
>SELECT * FROM employees WHERE salary > 50000;
```

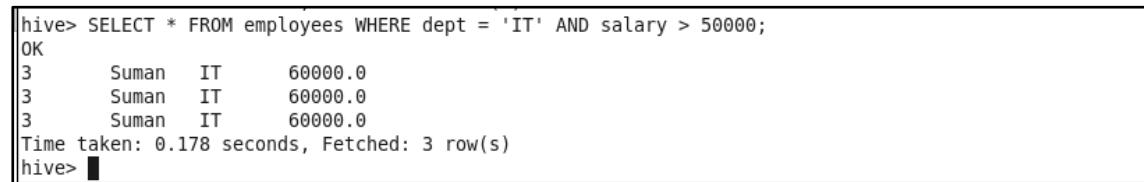


```

hive> SELECT * FROM employees WHERE salary > 50000;
OK
3      Suman    IT    60000.0
4      Neha     Finance 55000.0
3      Suman    IT    60000.0
4      Neha     Finance 55000.0
3      Suman    IT    60000.0
4      Neha     Finance 55000.0
Time taken: 0.065 seconds, Fetched: 6 row(s)

```

```
>SELECT * FROM employees WHERE dept = 'IT' AND salary > 55000;
```



```

hive> SELECT * FROM employees WHERE dept = 'IT' AND salary > 55000;
OK
3      Suman    IT    60000.0
3      Suman    IT    60000.0
3      Suman    IT    60000.0
Time taken: 0.178 seconds, Fetched: 3 row(s)
hive> █

```

Question b) Hive Built-In Functions

5. Hive Built-in Functions

-- String Functions

```
>SELECT UPPER(emp_name), LENGTH(emp_name) FROM employees;
```

```
hive> SELECT UPPER(emp_name), LENGTH(emp_name) FROM employees;
OK
RAVI      4
AMIT      4
SUMAN     5
NEHA      4
NULL      NULL
RAVI      4
AMIT      4
SUMAN     5
NEHA      4
NULL      NULL
RAVI      4
AMIT      4
SUMAN     5
NEHA      4
NULL      NULL
Time taken: 0.878 seconds, Fetched: 15 row(s)
hive> ■
```

-- Numeric Functions

```
>SELECT MAX(salary), MIN(salary), AVG(salary) FROM employees;
```

```
File Edit View Search Terminal Help
NULL      NULL
Time taken: 0.878 seconds, Fetched: 15 row(s)
hive> SELECT MAX(salary), MIN(salary), AVG(salary) FROM employees;
Query ID = cloudera_20251021004040_2d5b0599-9633-435e-b6dd-3e1a887306ae
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1761030184232_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761030184232_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1761030184232_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-21 00:40:53,169 Stage-1 map = 0%, reduce = 0%
2025-10-21 00:41:06,411 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.57 sec
2025-10-21 00:41:15,725 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.79 sec
MapReduce Total cumulative CPU time: 4 seconds 790 msec
Ended Job = job_1761030184232_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.79 sec HDFS Read: 9656 HDFS Write: 24 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 790 msec
OK
60000.0 45000.0 52500.0
Time taken: 37.582 seconds, Fetched: 1 row(s)
hive> ■
```

-- Date Function

>SELECT CURRENT_DATE();

```
hive> SELECT CURRENT_DATE();
OK
2025-10-21
Time taken: 1.018 seconds, Fetched: 1 row(s)
hive>
```

Question c) Hive Views

6. Hive Views

Views are virtual tables created from queries.

```
CREATE VIEW high_salary AS  
SELECT emp_name, dept, salary  
FROM employees  
WHERE salary > 50000;
```

```
>SELECT * FROM high_salary;
```

```
hive> CREATE VIEW high_salary AS  
> SELECT emp_name, dept, salary  
> FROM employees  
> WHERE salary > 50000;  
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. AlreadyExistsException(message:Table high_salary already exists)  
hive>  
> SELECT * FROM high_salary;  
OK  
Suman    IT      60000.0  
Neha     Finance 55000.0  
Suman    IT      60000.0  
Neha     Finance 55000.0  
Suman    IT      60000.0  
Neha     Finance 55000.0  
Time taken: 0.362 seconds, Fetched: 6 row(s)  
hive> █
```

Question d) HiveQL : Select Where, Select OrderBy, Select GroupBy, Select Joins

-- Select + Where Clause

```
>SELECT * FROM employees WHERE dept = 'IT';
```

```
hive> SELECT * FROM employees WHERE dept = 'IT';
OK
1      Ravi    IT    50000.0
3      Suman   IT    60000.0
1      Ravi    IT    50000.0
3      Suman   IT    60000.0
1      Ravi    IT    50000.0
3      Suman   IT    60000.0
Time taken: 0.087 seconds, Fetched: 6 row(s)
hive> █
```

-- Order By Clause

```
>SELECT * FROM employees ORDER BY salary DESC;
```

```
Total MapReduce CPU Time Spent: 3 seconds 930 msec
OK
3      Suman   IT    60000.0
3      Suman   IT    60000.0
3      Suman   IT    60000.0
4      Neha    Finance 55000.0
4      Neha    Finance 55000.0
4      Neha    Finance 55000.0
1      Ravi    IT    50000.0
1      Ravi    IT    50000.0
1      Ravi    IT    50000.0
2      Amit    HR    45000.0
2      Amit    HR    45000.0
2      Amit    HR    45000.0
NULL  NULL    NULL  NULL
NULL  NULL    NULL  NULL
NULL  NULL    NULL  NULL
Time taken: 26.908 seconds, Fetched: 15 row(s)
hive> █
```

-- Group By Clause

```
>SELECT dept, AVG(salary) AS avg_sal FROM employees GROUP BY dept;
```

```
Total MapReduce CPU Time Spent: 3 seconds 120 msec
OK
NULL  NULL
Finance 55000.0
HR    45000.0
IT    55000.0
Time taken: 21.36 seconds, Fetched: 4 row(s)
hive> █
```

-- Create and Load Departments Table

```
CREATE TABLE departments (
  dept STRING,
  location STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ''
STORED AS TEXTFILE;
```

Create local file:

```
nano /home/cloudera/departments.txt
```

Add data:
IT,Delhi
HR,Mumbai



The screenshot shows a terminal window titled "cloudera@quickstart:~". The menu bar includes File, Edit, View, Search, Terminal, and Help. The title bar shows "cloudera@quickstart:~". The status bar indicates "File: /home/cloudera/departments.txt" and "GNU nano 2.0.9". The main area of the terminal contains the following text:

```
IT,Delhi
HR,Mumbai
Finance,Pune
```

Finance,Pune

Load data:

```
>LOAD DATA LOCAL INPATH '/home/cloudera/departments.txt' INTO TABLE
departments;
```

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/departments.txt' INTO TABLE departments;
Loading data to table company_db.departments
Table company_db.departments stats: [numFiles=4, totalSize=132]
OK
Time taken: 0.912 seconds
```

-- Join Query

```
SELECT e.emp_name, e.dept, d.location
FROM employees e
JOIN departments d
ON (e.dept = d.dept);
```

```
hive> SELECT e.emp_name, e.dept, d.location
    > FROM employees e
    > JOIN departments d
    > ON (e.dept = d.dept);
Query ID = cloudera_20251021010000_082f3c80-2686-4e42-8922-9b4540bfbe35
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20251021010000_082f3c80-2686-4e42-8922-9b4540bfbe35.log
2025-10-21 01:00:15      Starting to launch local task to process map join;      maximum memory = 932184064
2025-10-21 01:00:16      Dump the side-table for tag: 1 with group count: 4 into file: file:/tmp/cloudera/aaf38c19-185
d-4f36-8bed-2dba0b973353/hive_2025-10-21_01-00-11_682_2040883646967824857-1/-local-10003/HashTable-Stage-3/MapJoin-ma
pfile01--.hashtable
2025-10-21 01:00:16      Uploaded 1 File to: file:/tmp/cloudera/aaf38c19-185d-4f36-8bed-2dba0b973353/hive_2025-10-21_0
1-00-11_682_2040883646967824857-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile01--.hashtable (479 bytes)
2025-10-21 01:00:16      End of local task; Time Taken: 1.063 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1761030184232_0004, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761030184232_0004/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1761030184232_0004
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2025-10-21 01:00:23,917 Stage-3 map = 0%, reduce = 0%
2025-10-21 01:00:35,084 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 2.56 sec
MapReduce Total cumulative CPU time: 2 seconds 560 msec
Ended Job = job_1761030184232_0004
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Cumulative CPU: 2.56 sec  HDFS Read: 7212 HDFS Write: 992 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 560 msec
OK
Ravi    IT    Delhi
Ravi    IT    Delhi
Ravi    IT    Delhi
Ravi    IT    Delhi
Amit    HR    Mumbai
Amit    HR    Mumbai
Amit    HR    Mumbai
Amit    HR    Mumbai
Suman   IT    Delhi
Suman   IT    Delhi
Suman   IT    Delhi
Suman   IT    Delhi
Neha    Finance Pune
Neha    Finance Pune
Neha    Finance Pune
Neha    Finance Pune
```

```
cloudera@quickstart:~
```

Ravi	IT	Delhi
Ravi	IT	Delhi
Amit	HR	Mumbai
Suman	IT	Delhi
Neha	Finance	Pune
Ravi	IT	Delhi
Amit	HR	Mumbai
Suman	IT	Delhi
Neha	Finance	Pune
Ravi	IT	Delhi
Amit	HR	Mumbai
Suman	IT	Delhi
Neha	Finance	Pune

```
Time taken: 25.61 seconds, Fetched: 64 row(s)
hive> 
```

PRACTICAL NO: 8

TO GET HANDSON EXPERIENCE

ON PIG LATIN

PRACTICAL NO: 8
TO GET HANDSON EXPERIENCE ON PIG LATIN

Question a) Pig Latin Basic

Step 1 – Start Hadoop and Pig

Start all Hadoop services:

>start-all.sh

Now open the Pig shell:

>pig

Expected Output:

You are now inside the Pig shell.

```
2025-10-28 11:56:31,249 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS  
2025-10-28 11:56:31,249 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address  
grunt> ■
```

Question b) Pig Data

Types Step 2 – Pig Data

Types

Pig supports various data types, as listed below:

Type	Description	Example
int	Integer number	5
long	Large integer	123456789
float	Decimal	10.5
number	Larger decimal number	999.99
chararray	String	'Aakash'
bytearray	Raw bytes	Binary data
tuple	Row of fields	(1,'Aakash',99)
bag	Collection of tuples	{ (1,'Aakash'), (2,'Sunny') }
map	Key-value pairs	['name' #'Aakash','marks' #99]

Example: Load and Display Data

```
grunt> a = LOAD 'student_data.txt' USING PigStorage(',') AS (id:int, name:chararray, marks:int);
```

```
grunt> a = LOAD 'student_data.txt' USING PigStorage(',') AS (id:int, name:chararray, marks:int);  
grunt>
```

```
grunt> DUMP a;
```

Question c) Download the data

Step 3 – Download the Data File

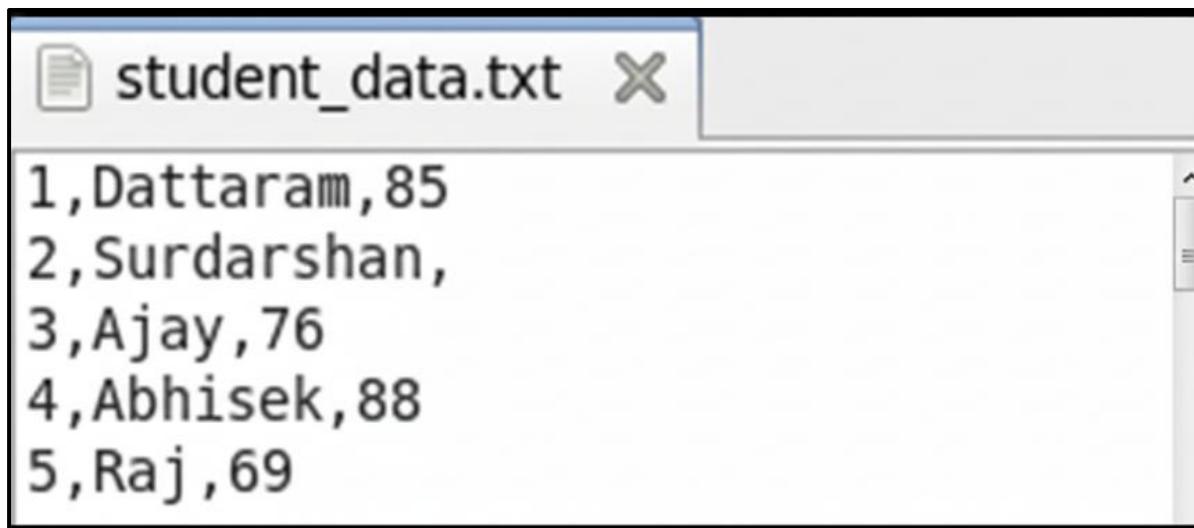
Step 1: Go to your home

directory: cd /home/cloudera

Step 2: Create a new file:

gedit student_data.txt

Step 3: Paste the following data and save:



The image shows a terminal window titled "student_data.txt". The window contains the following text:

```
1,Dattaram,85
2,Surdarshan,
3,Ajay,76
4,Abhisek,88
5,Raj,69
```

Question d) Create your Script

Step 4 – Create a Pig Script

Step 1: Create a new Pig script file:

gedit student_analysis.pig

Step 2: Paste the following code:

-- Load data

```
students = LOAD '/home/cloudera/student_data.txt' USING  
    PigStorage(',') AS (id:int, name:chararray, marks:int);
```

-- Display all records

```
DUMP students;
```

-- Filter students with marks > 80

```
top_students = FILTER students BY marks > 80;  
DUMP top_students;
```

-- Order students by marks descending

```
ordered = ORDER students BY marks DESC;  
DUMP ordered;
```

-- Group all students to find average

```
marks group_all = GROUP students ALL;  
avg_marks = FOREACH group_all GENERATE AVG(students.marks);
```

```
student_analysis.pig  
-- Load data  
students = LOAD '/home/cloudera/student_data.txt' USING PigStorage(',')  
    AS (id:int, name:chararray, marks:int);  
  
-- Display all records  
DUMP students;  
  
-- Filter students with marks > 80  
top_students = FILTER students BY marks > 80;  
DUMP top_students;  
  
-- Order by marks descending  
ordered = ORDER students BY marks DESC;  
DUMP ordered;  
  
-- Group all students to find average marks  
group_all = GROUP students ALL;  
avg_marks = FOREACH group_all GENERATE AVG(students.marks);  
DUMP avg_marks;  
  
DUMP avg_marks;
```

Question e) Save and Execute the**Script Step 5 – Execute the Pig Script****Run Pig in local mode:**

pig -x local student_analysis.pig

Expected Output:

```
s to process : 1
(1,Dattaram,85)
(2,Surdarshan
(3,Ajay,76)
(4,Abhisiek,88)
(5,Sraj,69)
```

```
Success!
Job Stats (time in seconds):
JobID Alias Feature Outputs
job_local852736765_0006 avg_marks,group_all,students GROUP_BY,COMBINER      file:/tmp/temp637870975/tmp1160779770
,
Input(s):
Successfully read records from: "/home/cloudera/student_data.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp637870975/tmp1160779770"
Job DAG:
job_local852736765_0006

2025-10-28 12:05:15,296 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
Success!
2025-10-28 12:05:15,296 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated
. Instead, use fs.defaultFS
2025-10-28 12:05:15,296 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is depreca
ted. Instead, use mapreduce.jobtracker.address
2025-10-28 12:05:15,297 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been ini
tialized
2025-10-28 12:05:15,366 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to pro
cess : 1
2025-10-28 12:05:15,366 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(81.6)
[cloudera@quickstart ~]$
```

PRACTICAL NO: 9

**TO WORK WITH VARIOUS OPERATIONS
IN PIG LATIN**

PRACTICAL NO: 9

TO WORK WITH VARIOUS OPERATIONS IN PIG LATIN

Question a) Pig Operations : Diagnostic Operators, Grouping and Joining, Combining & Splitting, Filtering, Sorting

Load the data:

```
>student_info = LOAD '/home/cloudera/Desktop/interns_data.txt' USING PigStorage(',')  
AS (id:chararray, fname:chararray, age:int, city:chararray);
```

--Check loaded data

>DUMP student_info;

```
Output(s):
Successfully stored records in: "file:/tmp/temp1272393649/tmp885491590"
```

```
Job DAG:  
job_local130138465_0004  
  
2025-10-23 00:30:10,885 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!  
2025-10-23 00:30:10,886 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS  
2025-10-23 00:30:10,886 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address  
2025-10-23 00:30:10,886 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
2025-10-23 00:30:10,886 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized  
2025-10-23 00:30:10,914 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1  
2025-10-23 00:30:10,914 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1  
(001,Rajiv,Patil,21,9848022337,Mumbai)  
(002,Siddarth,Bhattacharya,22,9848022338,Kolkata)  
(003,Rajesh,Khanna,22,9848022339,Mumbai)  
(004,Preeti,Agarwal,21,9848022330,Pune)  
(005,Trupti,Jain,23,9848022336,Mumbai)  
(006,Archana,Mishra,23,9848022335,Chennai)  
(007,Komal,Nayak,24,9848022334,Kolkata)  
(008,Bharathi,Nambiyar,24,9848022333,Chennai)  
grunt> ■  
... [Cloudera Live : Welco... cloudera@quickstart:~] interns_data.txt (~/D... ]
```

Describe schema

-- Describe schema

```
>DESCRIBE student_info;
```

```
grunt> describe student_info;
student info: {id: chararray, fname: chararray, age: int, city: chararray}
grunt> █
```

Filter data

```
>filterstudent = FILTER student_info BY age > 22;
>DUMP filterstudent;
```

```
cloudera@quickstart:~
```

```
File Edit View Search Terminal Help
2025-10-23 00:39:02,763 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2025-10-23 00:39:02,794 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2025-10-23 00:39:02,794 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2025-10-23 00:39:02,794 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2025-10-23 00:39:02,794 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2025-10-23 00:39:02,887 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2025-10-23 00:39:02,888 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(1,Ashok,22,Mumbai)
(2,Sudha,23,Delhi)
(3,Maya,23,Tokyo)
(4,Sara,25,NewYork)
(5,Suraj,23,Mumbai)
(6,Sandy,22,Chennai)
```

```
>EXPLAIN filterstudent;
```

```
cloudera@quickstart:~
```

```
File Edit View Search Terminal Help
(5,Suraj,23,Mumbai)
grunt> EXPLAIN filterstudent;
2025-10-23 00:46:08,513 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachFlatten, PushupFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimizer]}
#-----
# New Logical Plan:
#-----
filterstudent: (Name: LOStore Schema: id#298:chararray,fname#299:chararray,age#300:int,city#301:chararray)
|---filterstudent: (Name: LOFilter Schema: id#298:chararray,fname#299:chararray,age#300:int,city#301:chararray)
|   |---(Name: GreaterThan Type: boolean Uid: 305)
|   |---age:(Name: Project Type: int Uid: 300 Input: 0 Column: 2)
|   |---(Name: Constant Type: int Uid: 304)
|---student_info: (Name: LOForEach Schema: id#298:chararray,fname#299:chararray,age#300:int,city#301:chararray)
|   |---(Name: LOGenerate[false,false,false,false] Schema: id#298:chararray,fname#299:chararray,age#300:int,city#301:chararray)ColumnPrune:InputUids=[298, 299, 300, 301]ColumnPrune:OutputUids=[298, 299, 300, 301]
|       |---(Name: Cast Type: chararray Uid: 298)
|       |---id:(Name: Project Type: bytearray Uid: 298 Input: 0 Column: (*))
|       |---(Name: Cast Type: chararray Uid: 299)
|       |---fname:(Name: Project Type: bytearray Uid: 299 Input: 1 Column: (*))
|       |---(Name: Cast Type: int Uid: 300)
|       |---age:(Name: Project Type: bytearray Uid: 300 Input: 2 Column: (*))
|       |---(Name: Cast Type: chararray Uid: 301)
|       |---city:(Name: Project Type: bytearray Uid: 301 Input: 3 Column: (*))
|---(Name: LOInnerLoad[0] Schema: id#298:bytearray)
|---(Name: LOInnerLoad[1] Schema: fname#299:bytearray)
|---(Name: LOInnerLoad[2] Schema: age#300:bytearray)
|---(Name: LOInnerLoad[3] Schema: city#301:bytearray)
```

```
cloudera@quickstart:~$ 
File Edit View Search Terminal Help
rray)RequiredFields:null
#-----
# Physical Plan:
#-----
filterstudent: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-152
|---filterstudent: Filter[bag] - scope-148
|   |
|   | Greater Than[boolean] - scope-151
|   |
|   |---Project[int][2] - scope-149
|   |
|   |---Constant(22) - scope-150
|---student_info: New For Each(false,false,false,false)[bag] - scope-147
|   |
|   | Cast[chararray] - scope-136
|   |
|   |---Project[bytearray][0] - scope-135
|   |
|   | Cast[chararray] - scope-139
|   |
|   |---Project[bytearray][1] - scope-138
|   |
|   | Cast[int] - scope-142
|   |
|   |---Project[bytearray][2] - scope-141
|   |
|   | Cast[chararray] - scope-145
|   |
|   |---Project[bytearray][3] - scope-144
|---student_info: Load(/home/cloudera/Desktop/interns_data.txt:PigStorage(',')) - scope-134
2025-10-23 00:46:08,521 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2025-10-23 00:46:08,521 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2025-10-23 00:46:08,522 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
#-----
# Map Reduce Plan
#-----
MapReduce node scope-153
Map Plan
filterstudent: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-152
|---filterstudent: Filter[bag] - scope-148
```

```
cloudera@quickstart:~$ 
File Edit View Search Terminal Help
|---Project[bytearray][3] - scope-144
|---student_info: Load(/home/cloudera/Desktop/interns_data.txt:PigStorage(',')) - scope-134
2025-10-23 00:46:08,521 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2025-10-23 00:46:08,521 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2025-10-23 00:46:08,522 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
#-----
# Map Reduce Plan
#-----
MapReduce node scope-153
Map Plan
filterstudent: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-152
|---filterstudent: Filter[bag] - scope-148
|   |
|   | Greater Than[boolean] - scope-151
|   |
|   |---Project[int][2] - scope-149
|   |
|   |---Constant(22) - scope-150
|---student_info: New For Each(false,false,false,false)[bag] - scope-147
|   |
|   | Cast[chararray] - scope-136
|   |
|   |---Project[bytearray][0] - scope-135
|   |
|   | Cast[chararray] - scope-139
|   |
|   |---Project[bytearray][1] - scope-138
|   |
|   | Cast[int] - scope-142
|   |
|   |---Project[bytearray][2] - scope-141
|   |
|   | Cast[chararray] - scope-145
|   |
|   |---Project[bytearray][3] - scope-144
|---student_info: Load(/home/cloudera/Desktop/interns_data.txt:PigStorage(',')) - scope-134-----
Global sort: false
-----
grunt> 
```

Foreach (select columns)

>foreachstudent = FOREACH filterstudent GENERATE id, fname, age, city;
>DUMP foreachstudent;

```
2025-10-23 00:49:29,765 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(2,Sudha,23,Delhi)
(3,Maya,23,Tokyo)
(4,Sara,25,NewYork)
(5,Suraj,23,Mumbai)
grunt> ■
```

-- Illustrate data flow

>ILLUSTRATE foreachstudent;

student_info	id:chararray	fname:chararray	age:int	city:chararray
	3	Maya	23	Tokyo
	1	Ashok	22	Mumbai

filterstudent	id:chararray	fname:chararray	age:int	city:chararray
	3	Maya	23	Tokyo

foreachstudent	id:chararray	fname:chararray	age:int	city:chararray
	3	Maya	23	Tokyo

```
grunt> ■
```

Grouping

-- Group by city

>DESCRIBE groupstudent;

```
groupstudent: {group: chararray,student_info: {((id: chararray,fname: chararray,age: int,city: chararray))}}
grunt> ■
```

-- Group by city and age

>groupstudent2 = GROUP student_info BY (city, age);

>DUMP groupstudent2;

```
2025-10-23 00:53:44,529 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
((Delhi,23),{(2,Sudha,23,Delhi)})
((Tokyo,23),{(3,Maya,23,Tokyo)})
((Mumbai,22),{(1,Ashok,22,Mumbai)})
((Mumbai,23),{(5,Suraj,23,Mumbai)})
((Chennai,22),{(6,Sandy,22,Chennai)})
((NewYork,25),{(4,Sara,25,NewYork)})
grunt> ■
```

-- Group all

>groupstudent3 = GROUP student_info ALL;

>DUMP groupstudent3;

```
2025-10-23 00:54:48,164 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(all,{(6,Sandy,22,Chennai),(5,Suraj,23,Mumbai),(4,Sara,25,NewYork),(3,Maya,23,Tokyo),(2,Sudha,23,Delhi),(1,Ashok,22,M
umbai)})
grunt> ■
```

Store output

>STORE student_info INTO '/home/cloudera/Desktop/sampleoutput' USING PigStorage('|');

```

Success!
Job Stats (time in seconds):
JobID    Alias    Feature Outputs
job_local1724561446_0012      student_info    MAP_ONLY      /home/cloudera/Desktop/sampleoutput,
Input(s):
Successfully read records from: "/home/cloudera/Desktop/interns_data.txt"
Output(s):
Successfully stored records in: "/home/cloudera/Desktop/sampleoutput"
Job DAG:
job_local1724561446_0012

2025-10-23 00:58:08,196 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
Success!
grunt> █

```

Load data again into another alias for COGROUP

```

>I = LOAD '/home/cloudera/Desktop/interns_data.txt'
USING PigStorage(',')
AS (id:chararray, fname:chararray, age:int, city:chararray);

```

>DUMP I;

```

2025-10-23 01:00:05,043 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(1,Ashok,22,Mumbai)
(2,Sudha,23,Delhi)
(3,Maya,23,Tokyo)
(4,Sara,25,NewYork)
(5,Suraj,23,Mumbai)
(6,Sandy,22,Chennai)
grunt> █

```

COGROUP

```

>CG = COGROUP student_info BY age, I BY age;
>DUMP CG;

```

```

2025-10-23 01:01:08,713 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(22,{(6,Sandy,22,Chennai),(1,Ashok,22,Mumbai)},{(6,Sandy,22,Chennai),(1,Ashok,22,Mumbai)})
(23,{(5,Suraj,23,Mumbai),(3,Maya,23,Tokyo),(2,Sudha,23,Delhi)},{(5,Suraj,23,Mumbai),(3,Maya,23,Tokyo),(2,Sudha,23,Delhi)})
(25,{(4,Sara,25,NewYork)},{(4,Sara,25,NewYork)})
grunt> █

```

Self Join

```

>student_infoB = LOAD '/home/cloudera/student_data.txt'
>USING PigStorage(',') AS
(id:chararray,fname:chararray,lname:chararray,age:int,phone:chararray,city:chararray);
>SelfJoin = JOIN student_info BY age, student_infoB BY age;
>DUMP SelfJoin;

```

```

2025-10-22 10:07:32,645 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(2,Neha,Sharma,21,9988776655,Pune,2,Neha,Sharma,21,9988776655,Pune)
(4,Kiran,Doe,22,7766554433,Nashik,4,Kiran,Doe,22,7766554433,Nashik)
(1,Aakash,Patil,23,9876543210,Mumbai,1,Aakash,Patil,23,9876543210,Mumbai)
(3,Raj,Verma,24,8877665544,Mumbai,3,Raj,Verma,24,8877665544,Mumbai)
(5,Sneha,Naik,25,6655443322,Pune,5,Sneha,Naik,25,6655443322,Pune)
grunt> █

```

Inner Join

>InnerJoin = JOIN student_info BY city, I BY city;
>DUMP InnerJoin;

```
Output(s):
Successfully stored 0 records in: "hdfs://quickstart.cloudera:8020/tmp/temp-1826155470/tmp255345062"

Counters:
Total records written : 0
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1761126620207_0019

2025-10-22 10:13:27,171 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
Success!
2025-10-22 10:13:27,176 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated
. Instead, use fs.defaultFS
2025-10-22 10:13:27,176 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is depreca
ted. Instead, use mapreduce.jobtracker.address
2025-10-22 10:13:27,177 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... wi
ll not generate code.
2025-10-22 10:13:27,217 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to pro
cess : 1
2025-10-22 10:13:27,224 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
grunt> ■
```

Left Join

>LeftJoin = JOIN student_info BY city LEFT OUTER, I BY city;
>DUMP LeftJoin;

```
2025-10-22 10:19:30,371 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(5,Sneha,Naik,25,6655443322,Pune,,,,)
(2,Neha,Sharma,21,9988776655,Pune,,,,)
(3,Raj,Verma,24,8877665544,Mumbai,,,,)
(1,Aakash,Patil,23,9876543210,Mumbai,,,,)
(4,Kiran,Doe,22,7766554433,Nashik,,,,)
grunt> ■
```

Right Join

>RightJoin = JOIN student_info BY city RIGHT OUTER, I BY city;
>DUMP RightJoin;

```
2025-10-22 11:01:58,383 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(,,,,,,11,Neha,21,)
(,,,,,,13,Aakash,23,)
(,,,,,,10,Ravi,23,)
(,,,,,,14,Raj,24,)
(,,,,,,12,Pratik,25,)
grunt>
grunt> d■
```

Full Join

>FullJoin = JOIN student_info BY city FULL OUTER, I BY city;
>DUMP FullJoin;

```
2025-10-22 11:04:23,604 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(,,,,,,11,Neha,21,)
(,,,,,,13,Aakash,23,)
(,,,,,,10,Ravi,23,)
(,,,,,,14,Raj,24,)
(,,,,,,12,Pratik,25,)
(5,Sneha,Naik,25,6655443322,Pune,„„)
(2,Neha,Sharma,21,9988776655,Pune,„„)
(3,Raj,Verma,24,8877665544,Mumbai,„„)
(1,Aakash,Patil,23,9876543210,Mumbai,„„)
(4,Kiran,Doe,22,7766554433,Nashik,„„)
grunt> ■
```

Split Data

>SPLIT student_info INTO younger_students IF age < 23, older_students IF age >= 23;
>DUMP younger_students;

```
2025-10-22 11:07:01,674 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(2,Neha,Sharma,21,9988776655,Pune)
(4,Kiran,Doe,22,7766554433,Nashik)
grunt> ■
```

>DUMP older_students;

```
2025-10-22 11:08:21,251 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(1,Aakash,Patil,23,9876543210,Mumbai)
(3,Raj,Verma,24,8877665544,Mumbai)
(5,Sneha,Naik,25,6655443322,Pune)
grunt> ■
```

Filter by City

>filter_city = FILTER student_info BY city == 'Mumbai';
>DUMP filter_city;

```
2025-10-22 11:09:43,653 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(1,Aakash,Patil,23,9876543210,Mumbai)
(3,Raj,Verma,24,8877665544,Mumbai)
grunt> ■
```

Distinct Cities

>all_city = FOREACH student_info GENERATE city;
>distinct_cities = DISTINCT all_city;

```
2025-10-22 11:10:58,042 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(Mumbai)
(Pune)
(Mumbai)
(Nashik)
(Pune)
grunt> ■
```

>DUMP distinct_cities;

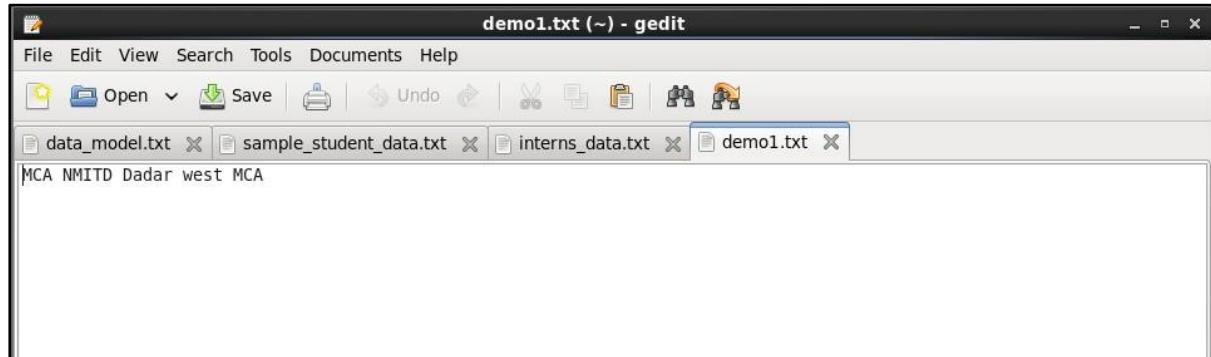
```
2025-10-22 11:12:06,474 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input path
s to process : 1
(Pune)
(Mumbai)
(Nashik)
grunt> ■
```

Write a Pig Latin Script to get the count of each word of a text file, save it in WordCount.pig and execute script to get the required output.(Use any text file as input file. It should have a few statements).

Create files demo1.txt and wordcount.pig in /home/cloudera

demo1.txt

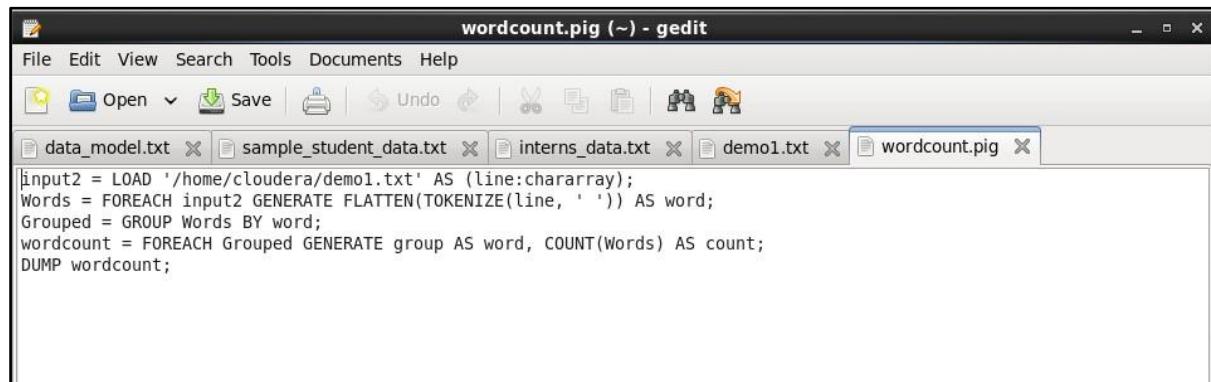
MCA NMITD Dadar west MCA



```
MCA NMITD Dadar west MCA
```

wordcount.pig

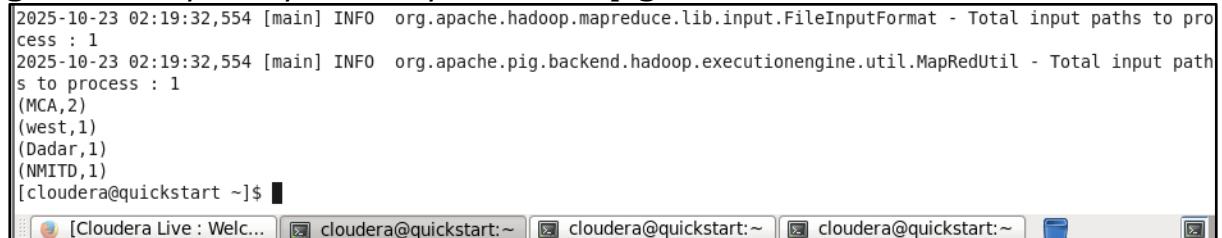
```
input2 = LOAD '/home/cloudera/demo1.txt' AS
(line:Chararray) ; Words = FOREACH input2 GENERATE
FLATTEN(TOKENIZE(line, ' ')) AS word;
Grouped = GROUP Words BY word;
wordcount = FOREACH Grouped GENERATE COUNT(Words), group;
```



```
input2 = LOAD '/home/cloudera/demo1.txt' AS (line:chararray);
Words = FOREACH input2 GENERATE FLATTEN(TOKENIZE(line, ' ')) AS word;
Grouped = GROUP Words BY word;
wordcount = FOREACH Grouped GENERATE group AS word, COUNT(Words) AS count;
DUMP wordcount;
```

dump wordcount;

grunt> exec /home/cloudera/wordcount.pig



```
2025-10-23 02:19:32,554 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2025-10-23 02:19:32,554 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(MCA,2)
(west,1)
(Dadar,1)
(NMITD,1)
[cloudera@quickstart ~]$
```

PRACTICAL NO: 10

WORKING WITH SPARK

PRACTICAL NO: 10 WORKING WITH SPARK

Question a) Downloading Data Set and Processing it Spark

Downloading Data Set and Processing it Spark

Word Count in Apache Spark

Aim:

To download a dataset and perform word count processing using Apache Spark (Scala).

Software Requirement:

Cloudera 5.13.0 with Apache Spark pre-installed.

Procedure:

Step 1: Check Spark Installation

Command:

spark-shell

```

cloudera@quickstart:~$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to

          \   ^__/
           \  V _\
             ^  )\/\_
               ||----\
               ||     \
               \  \  /  /
                 \  \
                   \  \
                     \  \
                       \  \
                         \  \
                           \  \
                             \  \
                               \  \
                                 \  \
                                   \  \
                                     \  \
                                       \  \
                                         \  \
                                           \  \
                                             \  \
                                               \  \
                                                 \  \
                                                   \  \
                                                     \  \
                                                       \  \
                                                         \  \
                                                           \  \
                                                             \  \
                                                               \  \
                                                                 \  \
                                                                   \  \
                                                                     \  \
                                                                       \  \
                                                                         \  \
                                                                           \  \
                                                                             \  \
                                                                               \  \
                                                                                 \  \
                                                                                   \  \
                                                                                     \  \
                                                                                       \  \
                         version 1.6.0

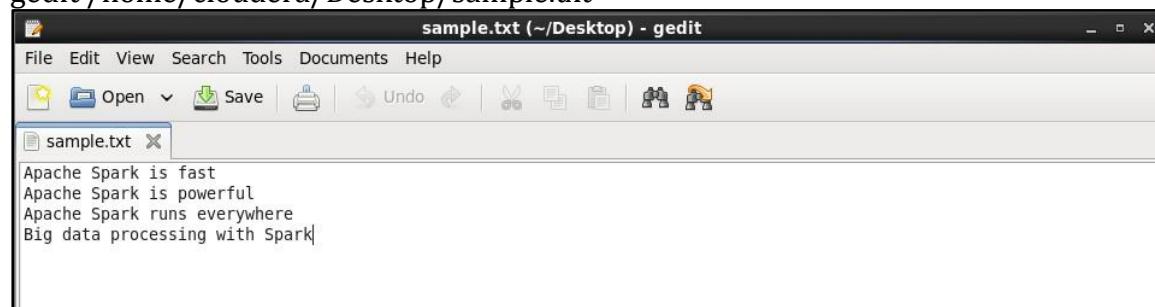
Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
25/10/23 09:49:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context available as sc (master = local[*], app id = local-1761238185589).
25/10/23 09:49:50 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
SQL context available as sqlContext.

```

Step 2: Create Dataset File

Command:

gedit /home/cloudera/Desktop/sample.txt



Step 3: Open Spark Shell

Command:

spark-shell

```
cloudera@quickstart:~$ spark-shell
[cloudera@quickstart ~]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to

    / \ \ / \ \ / \ \ / \ \ / \
   / \ \ / \ \ / \ \ / \ \ / \ \ / \
  / \ \ / \ \ / \ \ / \ \ / \ \ / \
 / \ \ / \ \ / \ \ / \ \ / \ \ / \
version 1.6.0

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
25/10/23 09:49:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context available as sc (master = local[*], app id = local-1761238185589).
25/10/23 09:49:50 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
SQL context available as sqlContext.
```

Step 4: Load the Dataset in Spark

Command:

```
val textFile = sc.textFile("file:///home/cloudera/Desktop/sample.txt")
```

```
scala> val textFile = sc.textFile("file:///home/cloudera/Desktop/sample.txt")
textFile: org.apache.spark.rdd.RDD[String] = file:///home/cloudera/Desktop/sampl
e.txt MapPartitionsRDD[1] at textFile at <console>:27
```

Step 4a: Verify Dataset Loaded

Command:

```
textFile.collect()
```

```
scala> textFile.collect()
res0: Array[String] = Array(Apache Spark is fast, Apache Spark is powerful, Apac
he Spark runs everywhere, Big data processing with Spark)
```

Step 5: Split Lines into Words

Command:

```
val words = textFile.flatMap(line => line.split(" "))
```

```
scala> val words = textFile.flatMap(line => line.split(" "))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <con
sole>:29
```

Question b) Word Count in Apache Spark.

Command:

```
words.collect()
```

```
scala> words.collect()
res1: Array[String] = Array(Apache, Spark, is, fast, Apache, Spark, is, powerful
, Apache, Spark, runs, everywhere, Big, data, processing, with, Spark)
```

Step 6: Create (word, 1) Pair

Command:

```
val wordPairs = words.map(word => (word, 1))
```

```
scala> val wordPairs = words.map(word => (word, 1))
wordPairs: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map
at <console>:31
```

Step 7: Reduce to Count Words

Command:

```
val wordCounts = wordPairs.reduceByKey(_ + _)
```

```
scala> val wordCounts = wordPairs.reduceByKey(_ + _)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey
Key at <console>:33
```

Step 8: Show the Final Output

Command:

```
scala> wordCounts.collect().foreach(println)
(is,2)
(runs,1)
(Big,1)
(fast,1)
(Apache,3)
(with,1)
(data,1)
(everywhere,1)
(powerful,1)
(Spark,4)
(processing,1)
scala> ■
```

```
wordCounts.collect().foreach(println)
```

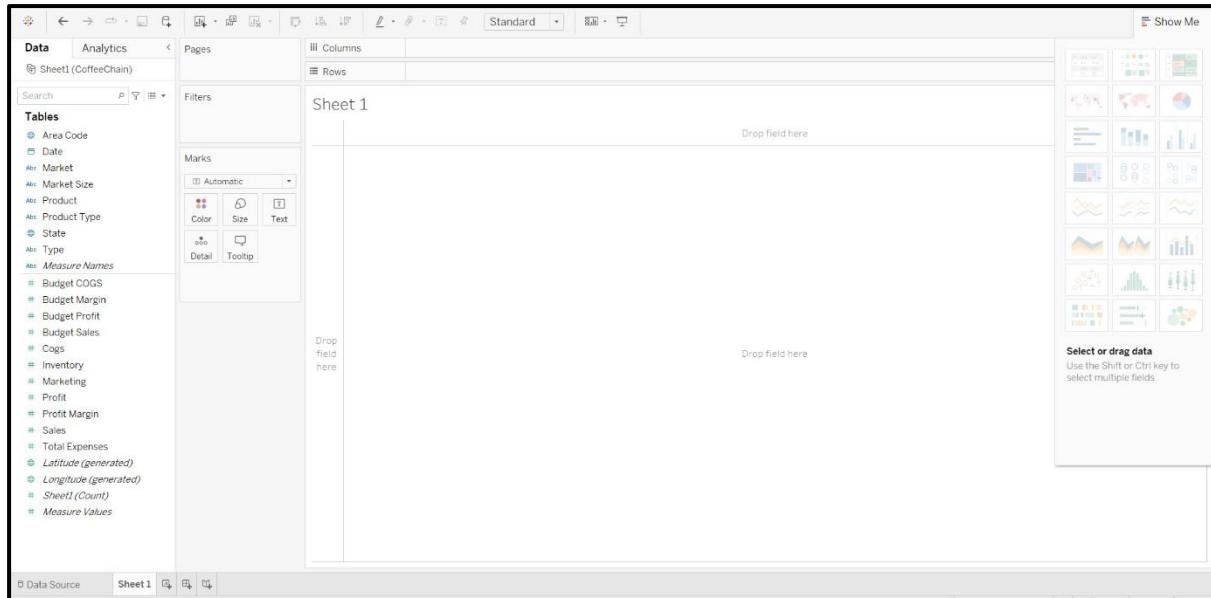
PRACTICAL NO: 11

**TO UNDERSTAND WORKING OF TABLEAU TOOL FOR
VISUALISATION**

PRACTICAL NO: 11
TO UNDERTAND WORKING OF TABLEAU TOOL FOR
VISUALISATION

Question a) Tool

Overview Tool Overview:



This screenshot shows the **Tableau Start Page**, which is the first screen you see when you open the application. It acts as the main hub or "home base" for all your projects. It is organized into three main sections:

The "Connect" Pane (The Start)

On the **left** side is the **Connect** pane. This is the starting point for any analysis. It's where you tell Tableau where your data lives.

The "Open" Pane (Your Projects)

In the **center** is the **Open** pane. This area shows all the recent workbooks

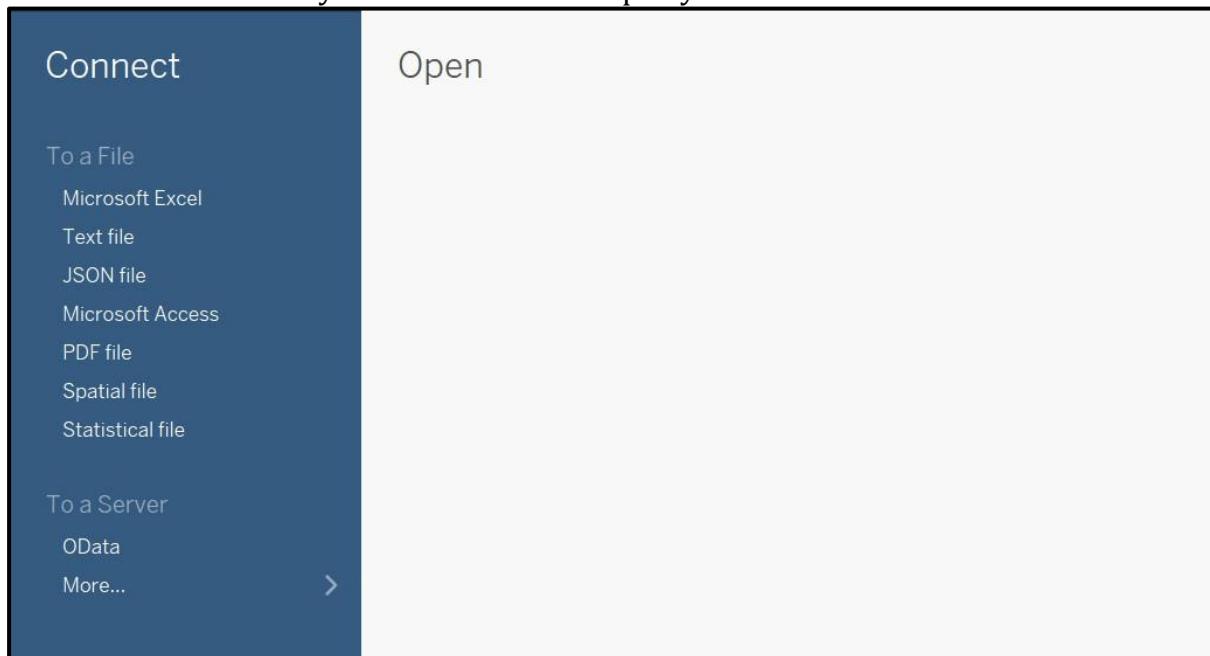
The "Discover" Pane (Learning & Inspiration)

On the **right** side is the **Discover** pane. This is a great resource for learning. It provides training videos, sample workbooks, and links to the "Viz of the Week," which shows amazing dashboards created by other users. It's a perfect place to get inspiration for new chart ideas.

Question b) Importing

Data Importing Data:

Select the extension of your dataset file and open your dataset:



Drag the sheet to the right panel to extract the data:

The screenshot shows the 'Data Extract' interface. On the left, the 'Sheets' pane lists 'Sheet1'. The main area displays a table with 19 fields and 4248 rows. The table has columns for Profit, Profit Margin, Sales, Cogs, Total Expenses, Marketing, Inventory, and Budget Profit. Below the table, a 'Fields' section shows mappings for Profit, Profit Margin, and Sales. At the bottom, there are buttons for 'Data Source' and 'Sheet1'.

#	Sheet1 Profit	Sheet1 Profit Margin	Sheet1 Sales	Sheet1 Cogs	Sheet1 Total Expenses	Sheet1 Marketing	Sheet1 Inventory	Sheet1 Budget Profit
94	130	219	89		36	24	777	
68	107	190	83		39	27	623	
101	139	234	95		38	26	821	
30	56	100	44		26	14	623	
54	80	134	54		26	15	456	
53	108	180	72		55	23	558	

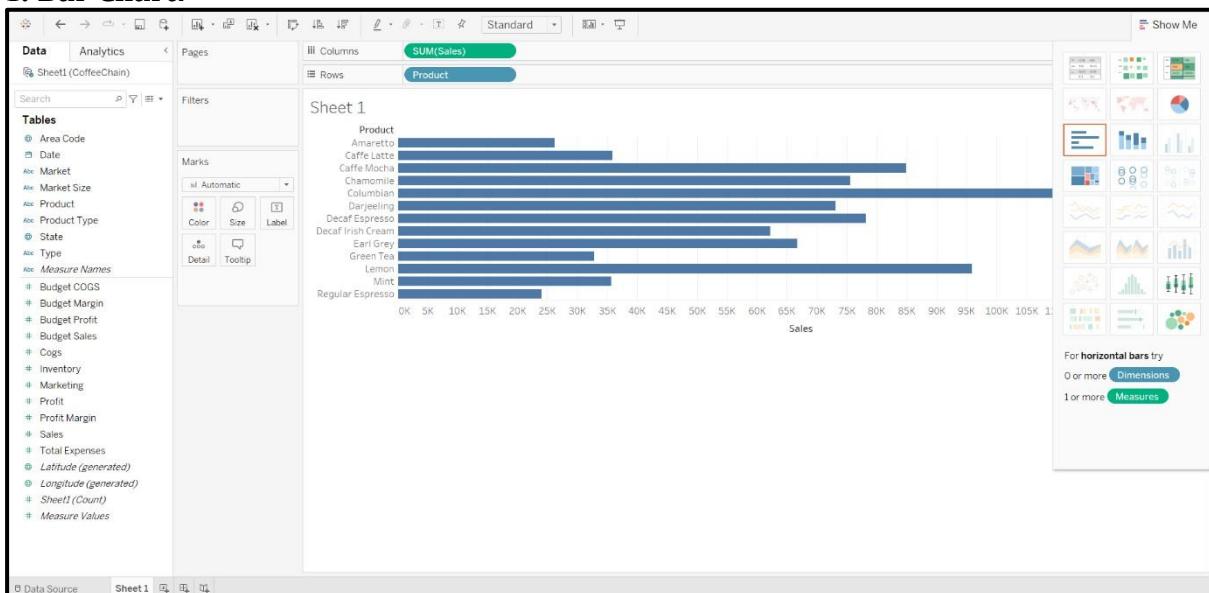
Question c) Analyzing with Charts

Analyzing with Charts:

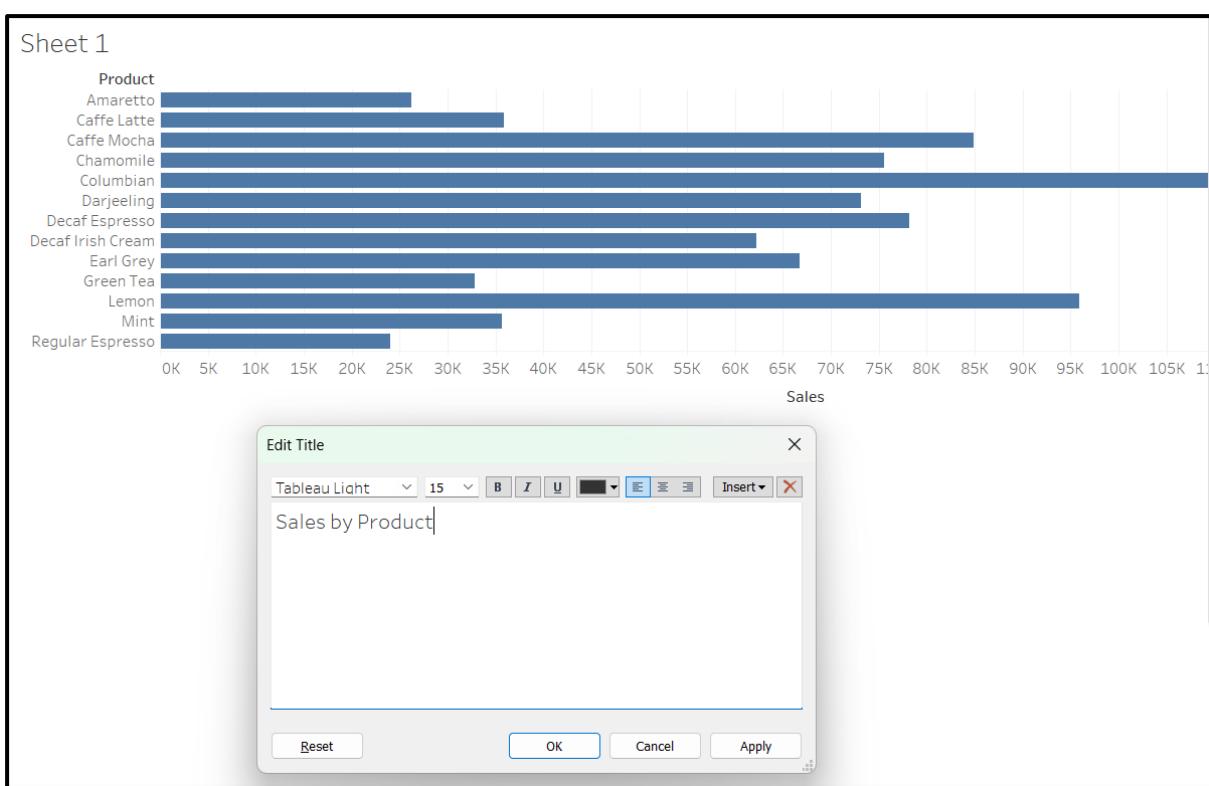
Analysis is done by dragging and dropping data fields from the **Data pane** (on the left) onto the **shelves** (at the top).

- **Dimensions** (categories, like Product or Market) are usually dragged to **Columns**.
- **Measures** (numbers, like Sales or Profit) are usually dragged to **Rows**.

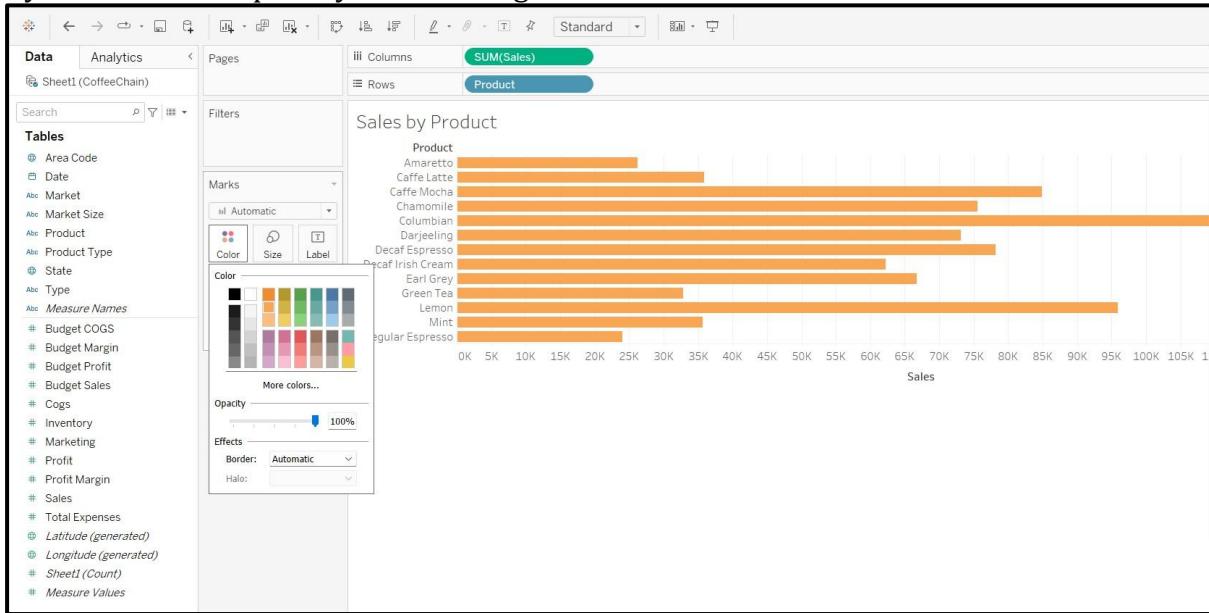
1. Bar Chart:



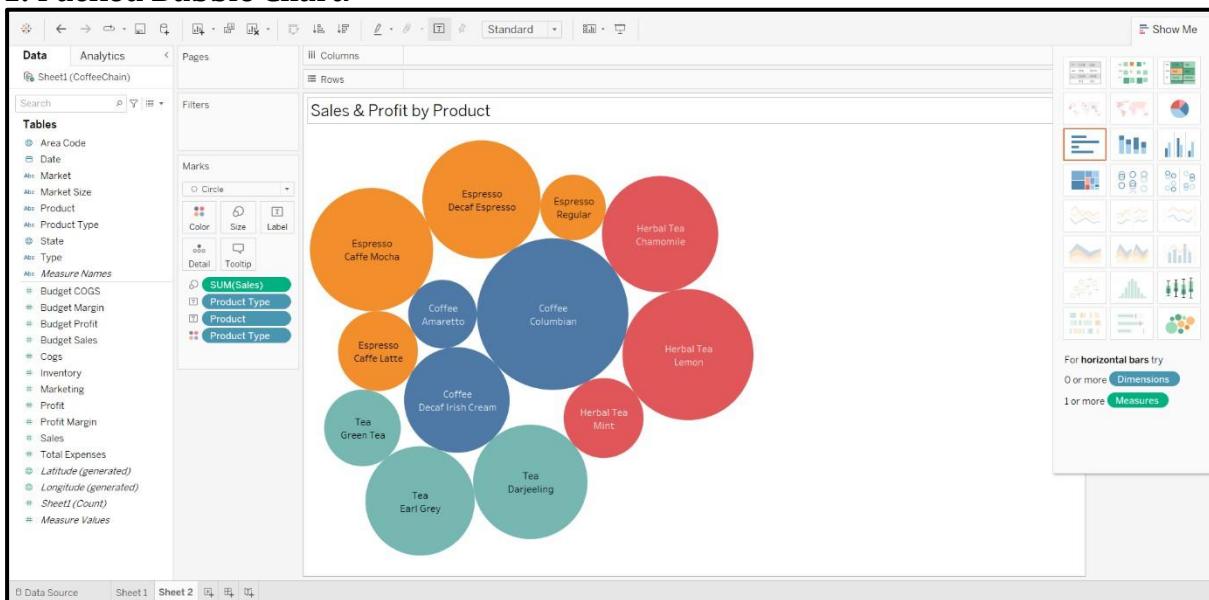
By clicking on sheet name, you can change the sheet name to your desired name:



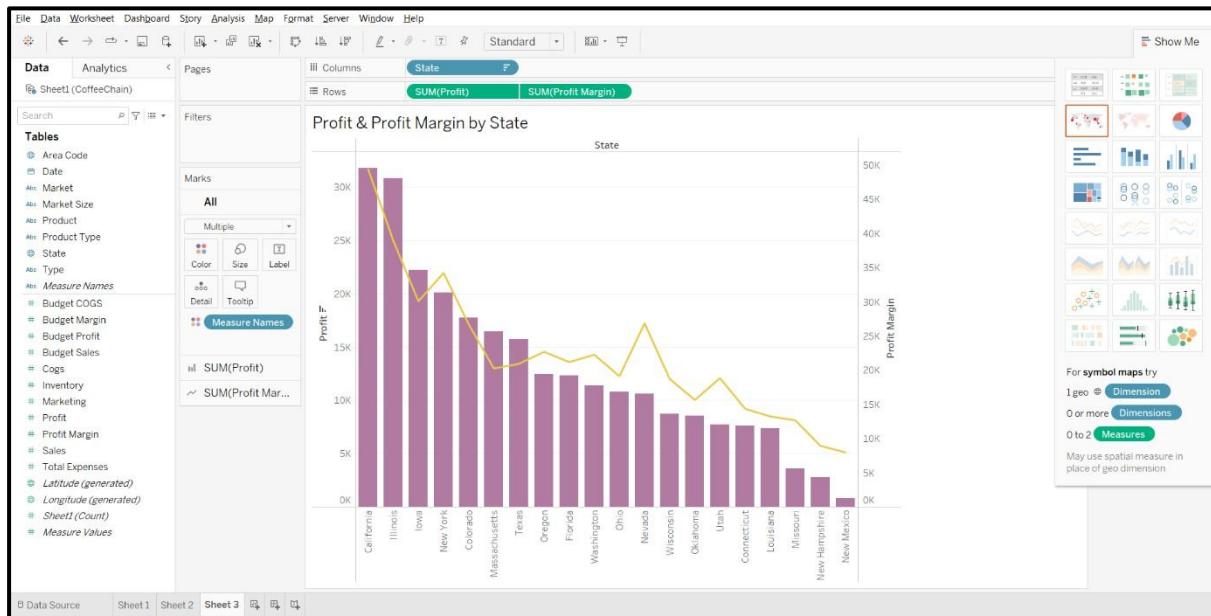
By Click on colors panel you can change the colors:



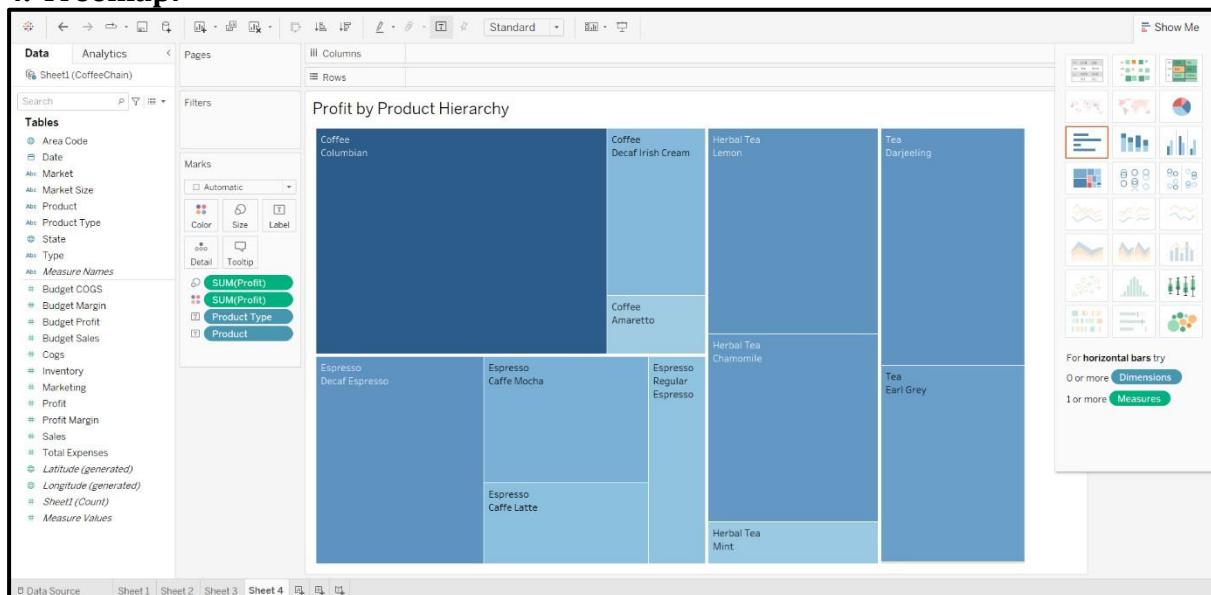
2. Packed Bubble Chart:



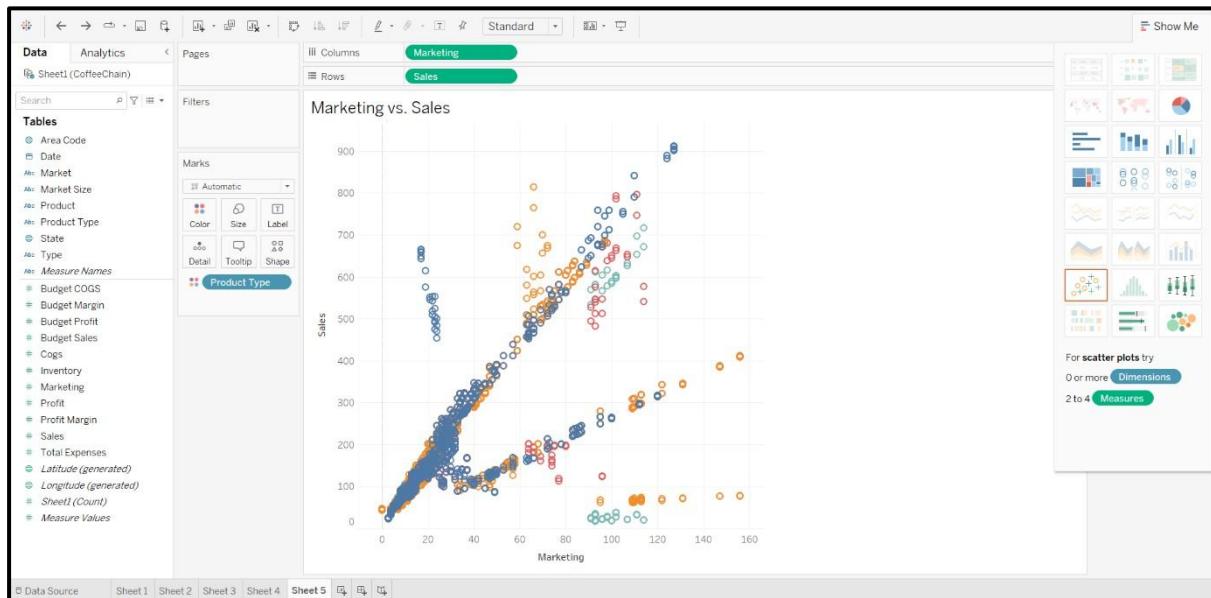
3. Dual-Combination Chart:



4. Treemap:



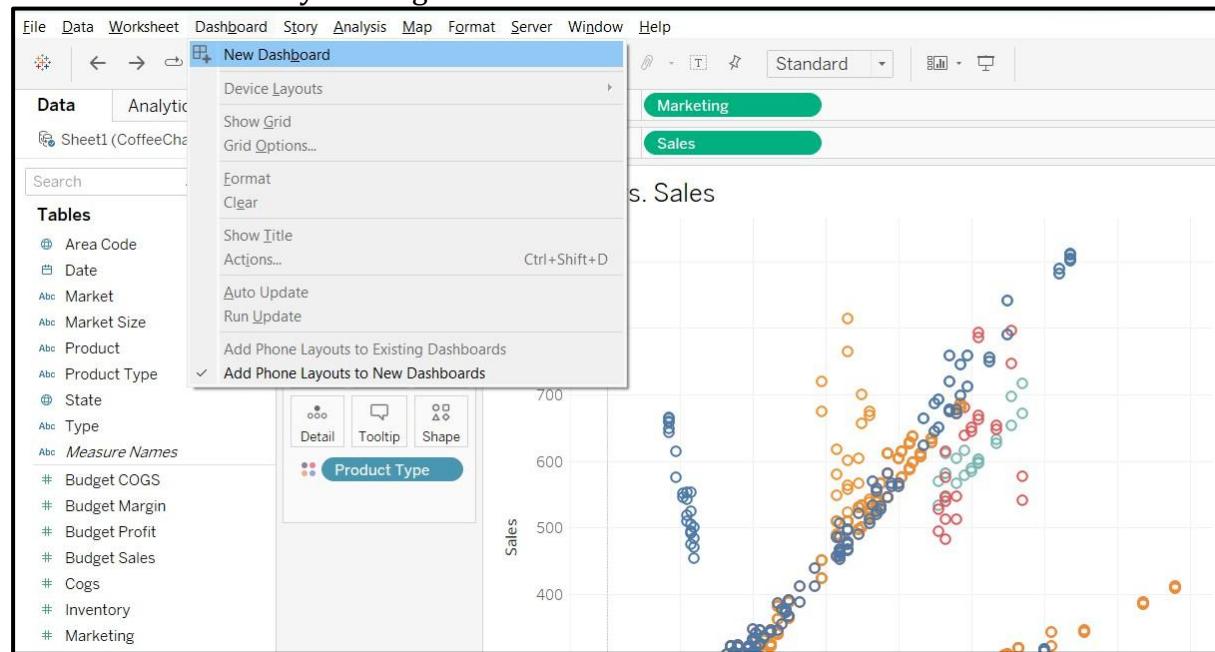
5. ScatterPlot:



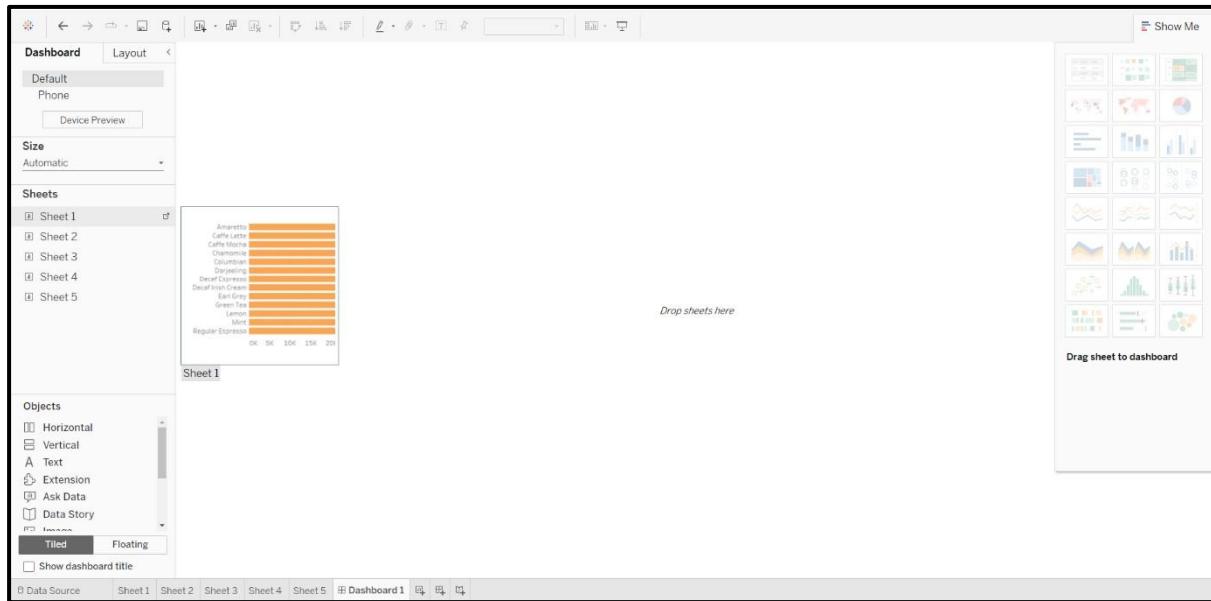
Question d) Creating Dashboards

Creating Dashboards:

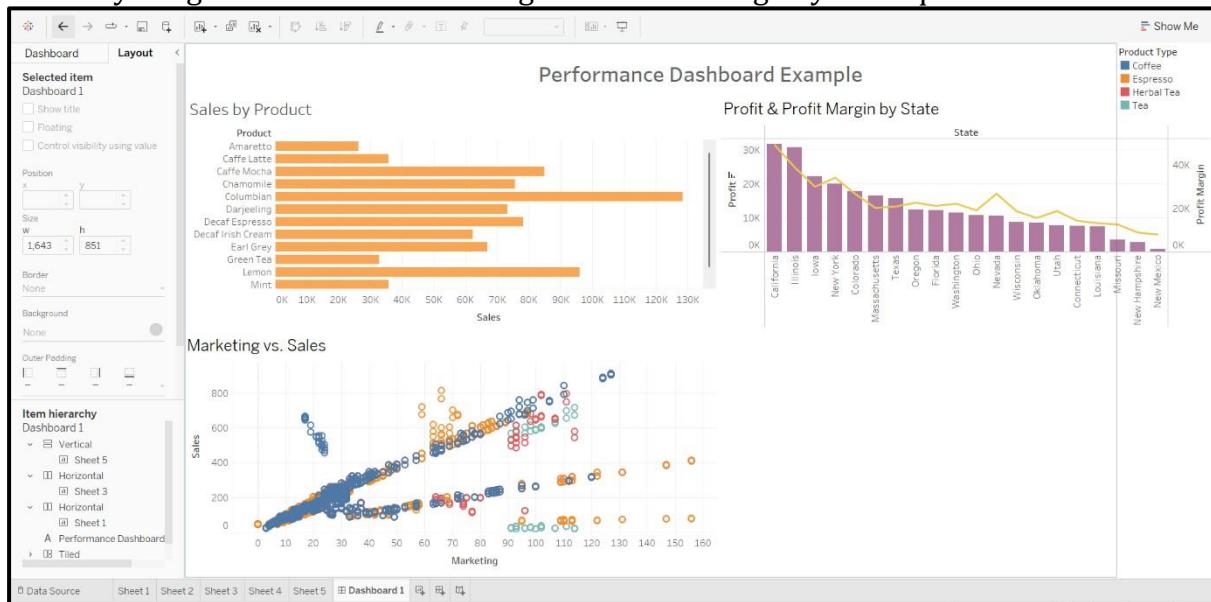
Create a dashboard by clicking the "New Dashboard" icon.



On the left, you see a list of all your completed charts. You simply **drag and drop** your worksheets from this list onto the main dashboard canvas.



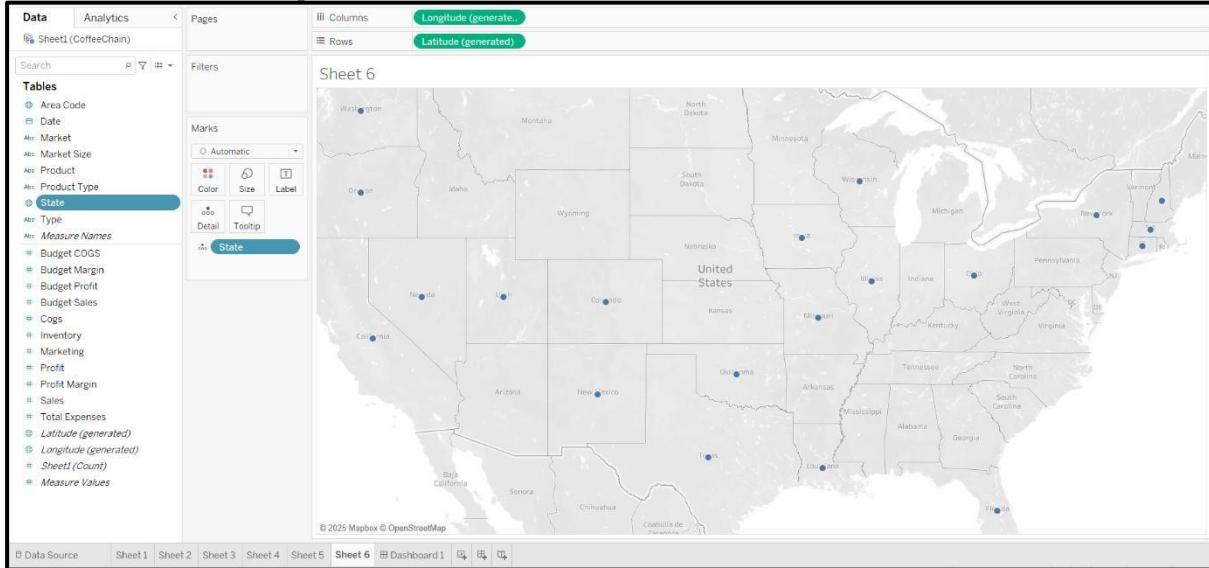
Similarly Drag other sheets and arrange them according to your requirement:



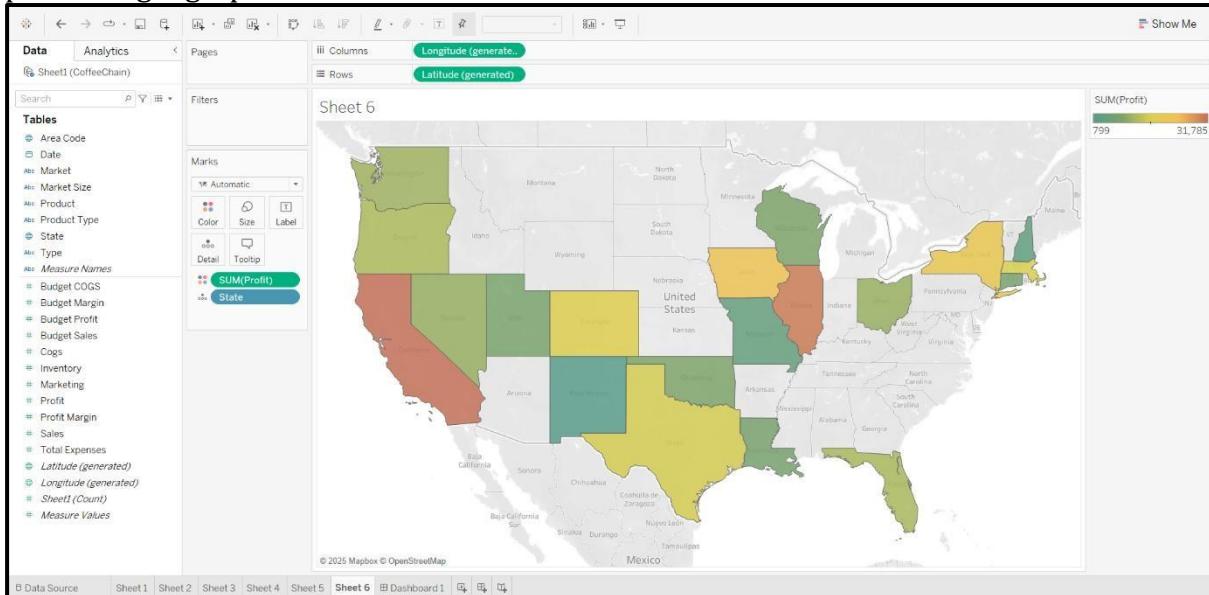
Question e) Working with maps

Working with maps:

Tableau makes it easy to create maps because it automatically recognizes geographic data. The State field had a small globe icon next to it. By simply **double-clicking State**, Tableau created a map with a dot for each state.



We then dragged the Profit measure onto the **Color** card. This instantly colored each state based on its profitability (as seen in the screenshot), turning our data into a powerful geographic visual.

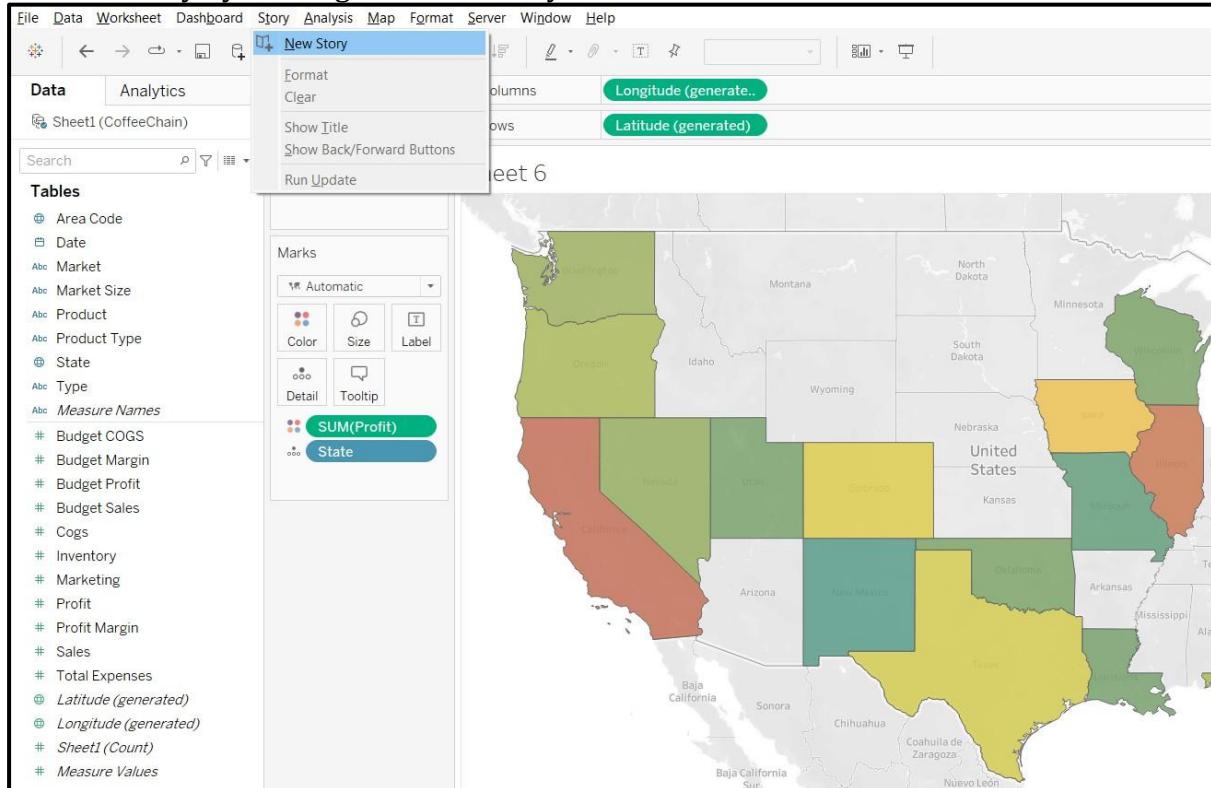


Question e) Telling Stories with

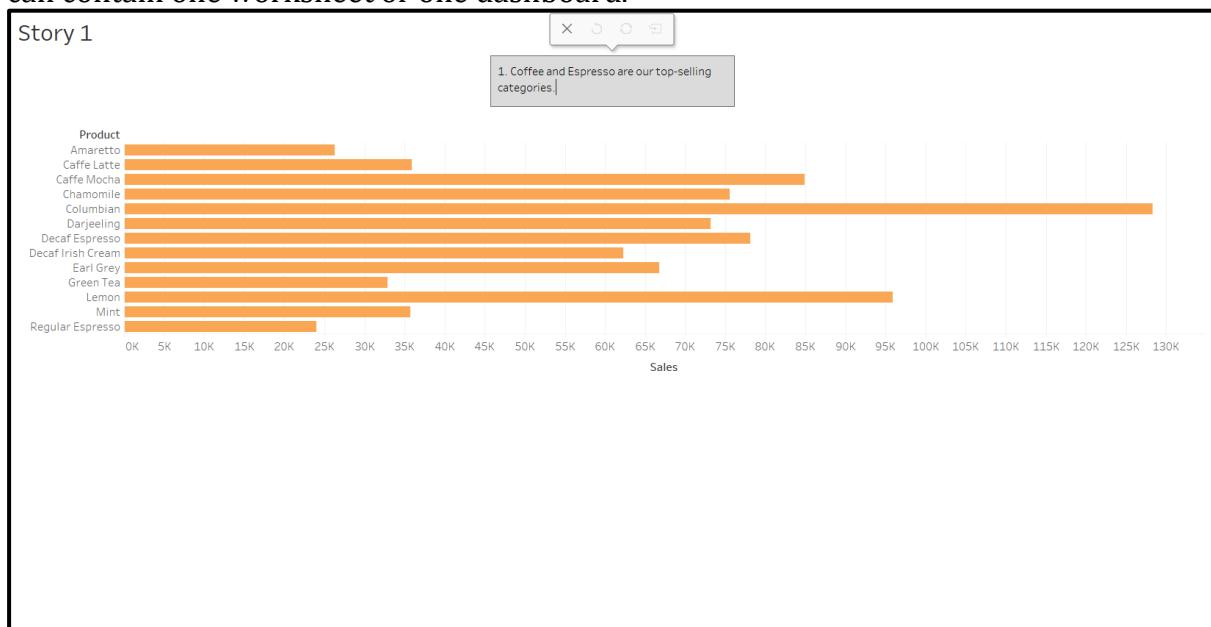
tableau Telling Stories with tableau:

The **Story** feature is like creating a PowerPoint presentation directly inside Tableau. It lets you guide an audience through your findings step-by-step.

Create a Story by clicking the "New Story" icon.



You create a "Story" by adding "**Story Points**" (which are like slides). Each story point can contain one worksheet or one dashboard.



You then write a **caption** for each point to explain what the audience should see. This is a powerful way to present your analysis and "tell a story" with your data.

