```java
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {
  public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer extends
Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context
contexts) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
```

```
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

.

Steps for hive installation

- Download and Unzip Hive
- Edit **.bashrc** file
- Edit **hive-config.sh** file
- Create **Hive directories** in HDFS
- Initiate **Derby database**
- Configure **hive-site.xml** file

download and unzip Hive

==========

wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz

tar xzf apache-hive-3.1.2-bin.tar.gz

Edit **.bashrc** file

==========

sudo nano .bashrc

export HIVE_HOME= /home/hdoop/apache-hive-3.1.2-bin

export PATH=$PATH:$HIVE_HOME/bin

source ~/.bashrc

Edit **hive-config.sh** file

==========

sudo nano $HIVE_HOME/bin/hive-config.sh

export HADOOP_HOME=/home/hdoop/hadoop-3.2.1

Create **Hive directories** in HDFS

==========

hdfs dfs -mkdir /tmp

hdfs dfs -chmod g+w /tmp

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

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

Fixing guava problem – Additional step

==========

rm $HIVE_HOME/lib/guava-19.0.jar

cp $HADOOP_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE_HOME/lib/

Initialize Derby and hive

==========

schematool -initSchema -dbType derby

hive

optional Step – Edit hive-site.xml

==========

cd $HIVE_HOME/conf

cp hive-default.xml.template hive-site.xml

sudo nano hive-site.xml – change metastore location to above created hdfs path(/user/hive/warehouse)

Cretae and Alter table

==========

```
Create table iris_tab_managed(length float, width float,
length1 float, width1 float, typeof string)
row format delimited
fields terminated by ‘,’

Create external table iris_tab_ext(ernallength float, width
float, length1 float, width1 float, typeof string)
row format delimited
fields terminated by ‘,’

ALTER table iris_tab_managed rename to iris_managed;
```

Load data in table

==========

```
load data local inpath ‘/home/aman/Downloads/iris.csv’ into
table iris;
```

Query Table

```
select * from iris_managed;
```

Create partitioned tables

==========

```
set hive.exec.dynamic.partition=True;
set hive.exec.dynamic.partition.mode=nonstrict;

Create table iris_partitioned(length float, width float, length1
float, width1 float)
partitioned by (typeof string)
```

row format delimited
fields terminated by ‘,’

insert into iris_partitioned partition(typeof)
Select length, width, length1, width1,typeof from iris;

Viewring files

Create Bucketing in tables
==========

set hive.enforce.bucketing=True;

Create table iris_bucketed(length float, width float, length1 float, width1 float)
clustered by (typeof string)
row format delimited
fields terminated by ‘,’

Ctaegory – Reg/green/blue
where category = “green”