# Program:- 1

WCMapper.java

import java.io.IOException;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text, Text, IntWritable> { // Map function public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces for (String word : line.split(""))

{ if (word.length() > 0)

{ output.collect(new Text(word), new IntWritable(1)); } } } }

# WCReducer.java

// Importing libraries import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> { // Reduce function public void reduce(Text key, Iterator value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

{

int count = 0;

// Counting the frequency of each words while (value.hasNext())

{

IntWritable i = value.next(); count += i.get();

}

output.collect(key, new IntWritable(count));

}

}

# WCDriver.java

// Importing libraries import java.io.IOException;

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf; import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool { public int run(String args[])

throws IOException

{ if (args.length < 2)

{

System.out.println("Please give valid inputs"); return -1;

}

JobConf conf = new JobConf(WCDriver.class); FileInput

Format.setInputPaths(conf, new Path(args[0])); FileOutputFormat.setOutputPath(conf, new Path(args[1])); conf.setMapperClass(WCMapper.class); conf.setReducerClass(WCReducer.class); conf.setMapOutputKeyClass(Text.class); conf.setMapOutputValueClass(IntWritable.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); JobClient.runJob(conf);

return 0;

}

// Main Method

public static void main(String args[])

throws Exception { int exitCode = ToolRunner.run(new WCDriver(), args);

System.out.println(exitCode); } }

**Output:**

Create a text file

cloudera@quickstart ~]$ Cat> WCFile.txt

Amar2

Amar

Pramod Rakesh Chethan V Tejas

Tejas

[cloudera@quickstart ~]$ hadoop fs -put WCFile.txt WCFile.txt

[cloudera@quickstart ~]$hadoop jar /home/cloudera/wordcount.jar

[cloudera@quickstart ~]$hdfs dfs –ls /out1

[cloudera@quickstart ~]$ hadoop fs -cat /outfile/part-00000

Amar2

Pramod 1 Rakesh1 Chethan V1 Tejas2

# Programs:- 2

**$ pig**

grunt>stud = LOAD 'student1.txt' USING PigStorage(',') AS (id:int, firstname:chararray, age:int, phone:chararray, city:chararray);

grunt> STORE stud INTO 'pig\_Output.txt' USING PigStorage(''); cat hdfs:

//quickstart.cloudera:8020/user/cloudera/pig\_Output.txt Dump stud(Display the results)

stud = LOAD 'student2.txt' USING PigStorage(',') A S (id:int, firstname:chararray, age:int,

phone:chararray, city:chararray); group\_data = GROUP stud by age; grunt> dump group\_data

(20,{(4,Shivu,20,9980700776,Shimoga)}) (21,{(5,Manu,21,9980700776,Bangalore),

(1,siri,21,8164490454,Bangalore)}) (22,{(5,prajju,22,9902084476,Bangalore), (3,Bharathi,22,9902084476,Kanakapura), (2,shreyu,22,9886724104,Mysore)})

grunt> Describe group\_data;

group\_data: {group: int,stud: {(id: int,firstname: chararray,age: int,phone: chararray,city: chararray)}}

Illustrate group\_data;

group\_multiple = GROUP student\_details by (age, city);

group\_multiple = GROUP student\_details by (age, city);

((20,Shimoga),{(4,Shivu,20,9980700776,Shimoga)})

((21,Bangalore),{(5,Manu,21,9980700776,Bangalore),(1,siri,21,816 4490454,Bangalore)})

((22,Mysore),{(2,shreyu,22,9886724104,Mysore)}) ((22,Bangalore),{(5,prajju,22,9902084476,Bangalore)}) ((22,Kanakapura),{(3,Bharathi,22,9902084476,Kanakapura)})

grunt> group\_all = GROUP stud All; grunt> dump group\_all

(all,{(5,prajju,22,9902084476,Bangalore),(5,Manu,21,9980700776, Bangalore),(4,Shivu,20,9980700776,Shimoga),(3,Bharathi,22,99020 84476,Kanakapura),(2,shreyu,22,9886724104,Mysore),(1,siri,21,81 64490454,Bangalore)})

Union

stud1 = LOAD 'student3.txt' USING PigStorage(',') AS (id:int, firstname:chararray, age:int, phone:chararray, city:chararray);

grunt> res= UNION stud, stud1; grunt> dump res

Join

grunt> ord= LOAD 'order.txt' USING PigStorage(',') AS (oid:int, date:chararray, customer\_id:int, amt:int);

grunt> dump ord

word count Problem

grunt> in = LOAD 'input.txt' AS (line:chararray);

grunt> words = FOREACH in GENERATE FLATTEN(TOKENIZE(line, '')) as word;

grunt> grouped = GROUP words BY word;

grunt> wordcount = FOREACH grouped GENERATE group, COUNT(words);

grunt> DUMP wordcount;(a,2) (is,3)

(This,1)

(post,1)

(very,1) (hadoop,2) (bigdata,2) (Technology,1) (interesting,1)

grunt> wc = LOAD 'WCFile.txt' AS (line:chararray); grunt>

grunt> words = FOREACH wc GENERATE FLATTEN(TOKENIZE(line, '')) as word;

grunt> grouped = GROUP words BY word;

grunt> wordcount = FOREACH grouped GENERATE group, COUNT(words);

grunt> DUMP wordcount;

(bus,1)

(car,2)

(deer,2)

(river,2)

(train,2)

# Program:-3

Use Hive to create, alter and drop databases, tables, views, functions and indexes

First lets open hive in terminal

once hive is opened do the commands

# Creating Database

create database first\_db;

{Here test\_db is database name}

create database second\_db; Seeing Available Databasess **how databases;**

{displays all the databases in hive}

# Selecting Database

use first\_db;

{this tells hive which database we are going to use}

# Creating Table

create table t1 (x int, y float);

{Here t1 is the table name & (x int) is the column name with data type}

**create table t2 like t1;** Displaying tables in database show tables in first\_db;

{this show all the tables in first\_db database}

# Displaying Table details

describe t1;

{this gives table's column names and their data types} describe formatted t1;

{gives all the details about the table }

# Renaming tables

alter table t1 rename to t3;

{Here t1 is the current name of the table and s1 is the new name}

# show tables in first\_db;

ALTER TABLE can also be used to move a table from 1 database to another

alter table first\_db.t3 rename to second\_db.s1; show tables in second\_db;

# Deleting Table

drop table t2;

{Deletes the table}

# Deleting Database

drop database first\_db;

{Deletes database (database should be empty) }

# press ctrl+l to clear the terminal

Creating VIEW use second\_db;

{selecting second\_db database} create table t1 (x int, y int, s string);

create view v1 as select \* from t1 LIMIT 10;

{Creating view with first 10 rows, LIMIT is optional} create view v2 as select x, y, s from t1;

{Creating view for specific columns}

# Displaying VIEW

describe v1;

{same a describe in table} describe formatted v1;

Altering VIEW

create table t2 like t1;

alter view v1 as select \* from t2;

{altering view}

alter view v1 as select x, upper(s) s from t2; describe formatted v1;

# Creating INDEX

CREATE INDEX C\_Index ON TABLE s1 (x) AS 'COMPACT' WITH DEFERRED REBUILD;

{Compact indexing stores the pair of indexed column’s value and its block id}

CREATE INDEX B\_index ON TABLE s1 (x) AS 'BITMAP' WITH DEFERRED REBUILD;

{Bitmap indexing stores the combination of indexed column value and list of rows as a bitmap}

Bitmap indexing is a standard technique for indexing columns with few distinct values.

# Displaying the Index

SHOW FORMATTED INDEX ON s1;

Deleting the Index

DROP INDEX c\_index ON s1; DROP INDEX b\_index ON s1;

# Functions

To do functions, we need a table with data in it now lets create it

use , between the words to divide to column

for this example we are doing a employee table with slno, name, salary

now lets execute the statements

create table emp (slno int, name string, salary int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

{Create table}

load data local inpath '/home/cloudera/db.txt' into table emp;

{Load data into the table} select \* from emp;

{view data in table}

select count(salary) from emp;

{Number of rows}

select sum(salary) from emp;

{Total of salary}

select avg(salary) from emp;

{Average of salary}

select max(salary) from emp;

{highest value in column} select min(salary) from emp;

{Lowest value in column}

press ctrl+c to exit hive