**BIG DATA ANALYTICS LAB**

Week 1

i) Perform setting up and installing Vmware for Hadoop and Linux.

**Installing VMware Workstation**

VMware Workstation is a program that allows you to run a virtual computer within your physical computer. The virtual computer runs as if it was its own machine. A virtual machine is great for trying out new operating systems such as Linux, visiting websites you don't trust, creating a computing environment specifically for children, testing the effects of computer viruses, and much more. You can even print and plug in USB drives. Read this guide to get the most out of VMware Workstation.

First we'll need to download the VMware Workstation software, which can be done here:

<http://downloads.vmware.com/d/info/desktop_downloads/vmware_workstation/7_0>

**STEP:1**

**Make sure that your computer meets the minimum requirements to run VMware Workstation .**

**Quick overview**:

* Operating System : Windows or Linux 64-bit
* To run 32-bit virtual machines: 64-bit; 1.3 GHz or higher
* To run 64-bit virtual machines: the above requirements along with VT-x support if you have an Intel processor (make sure that it is enabled in the BIOS), or the above requirements along with long mode support if you have an AMD processor.
* RAM : 6 GB is minimum, but 8GB is recommended
* GPU : at least a 16 or 32 bit display adapter (you probably have a 32-bit display adapter). If you want Windows Aero graphics to work in Windows virtual machines, then you should have either an NVIDIA GeForce 8800GT graphics card or newer or an ATI Radeon HD 2600 graphics or newer.
* HDD space : 5 GB is required to install the program alone, but virtual machines will take up even more space.

**STEP:2**

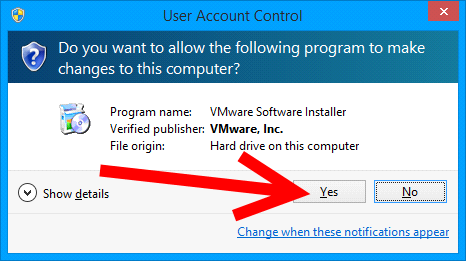
**Log into your My VMware account or create a new one (if you are not logged in).**Click [here](https://my.vmware.com/web/vmware/login) to open the log-in page, and either enter the log-in details of an account and click the Log In button, or click Register to create a new account.

* You might be able to find the log-in details of a public account on [this website](http://bugmenot.com/view/vmware.com).

**STEP:3  
Download VMware Workstation.** Once you have logged into an account, click [here](https://goo.gl/Irlhsq)to open the download page for VMware Workstation and download it.

**STEP:4**

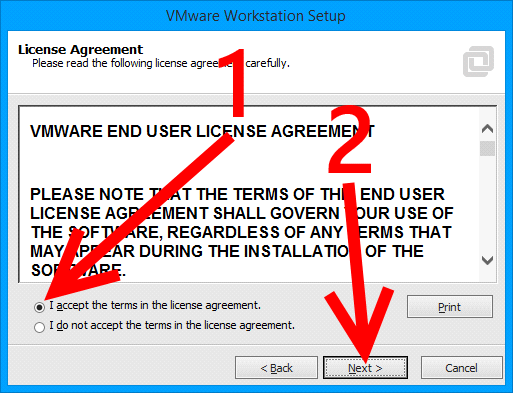
**Run the setup file.** Double-click the file that you downloaded. If the *User Account Control* or *Open File - Security Warning* dialog appears, click yes or Run respectively.



**Click Next to dismiss the Welcome dialog box.**

**STEP:5 Select the *I accept the terms in the license agreement* option then**

**click next.**

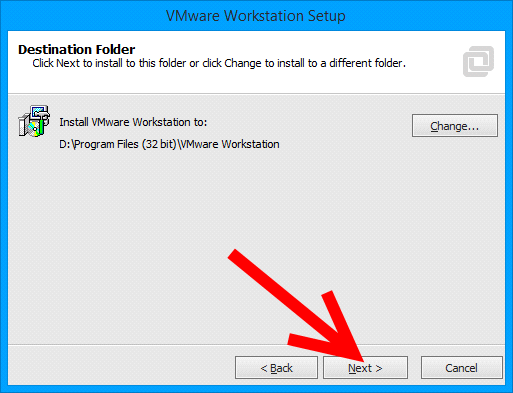


**Click the *Typical* button.**

**STEP:6**

**Choose the directory in which you want to install VMware Workstation.** To install it into a directory other than the default one, click Change... and browse to the desired directory. After deciding on where to install VMware Workstation, click next.

* If you install VMware Workstation on a network drive, you cannot run it when the network drive is inaccessible.



**STEP:7  
Decide if you want VMware Workstation to check for updates every time it is opened, then click Next .**

**STEP:8**

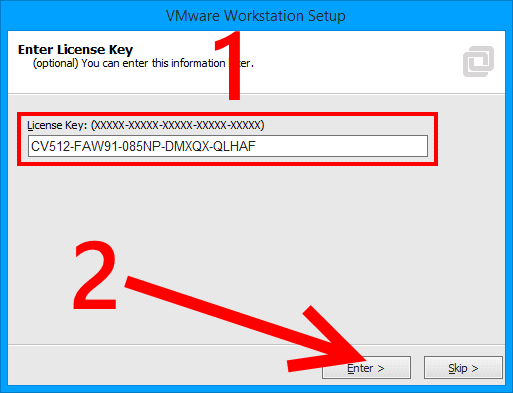
**Decide if want to send information about VMware Workstation's performance on your computer to VMware.** This information is used to help improve the program. Click Next after you have made your decision.

**STEP:9**

**Select the locations that you want setup to create shortcuts to VMware Workstation in.** Choices include the Desktop and the Start menu. Deselect any shortcuts you do not want the installer to create and then click next.

**STEP:10**

**The installer is ready to begin installing VMware Workstation.** If you want to change any options, click Back to return to them, change them, then click Next  until you reach this screen again once you've finished. Once you are ready to begin installing VMware Workstation, click Continue.



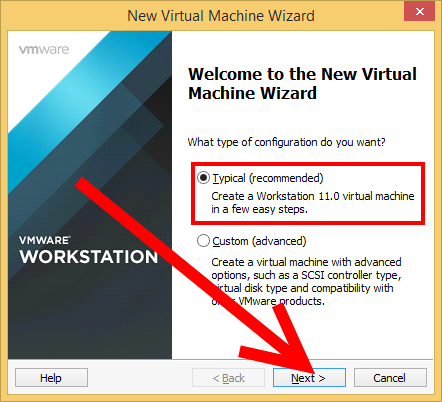
**Register VMware Workstation.** Type a license key and click Enter .

**STEP:11**

**Close setup.** Click the Finish button once setup completes its operations.

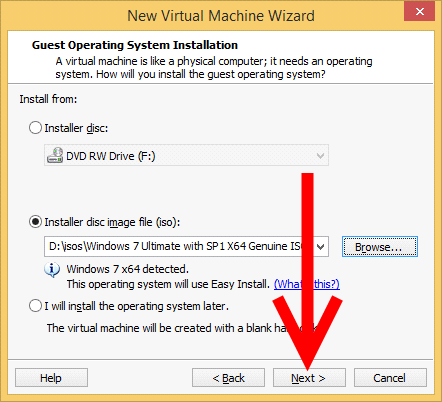
**STEP:12**

**Start the New Virtual Machine Wizard.** Choose File New Virtual Machine to begin creating your virtual machine.



**STEP:13**

**Choose the *Typical* virtual machine configuration option, then click next.**



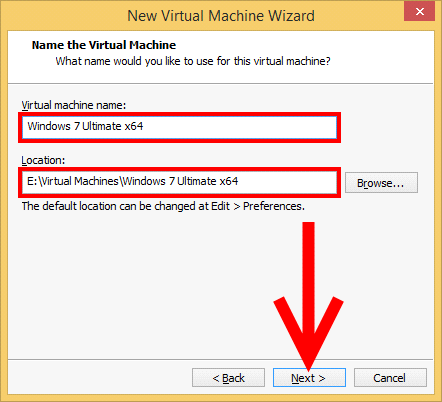
**STEP:14**

**Select the installation media that you want to use to install the virtual machine's operating system.** Click *Installer disc* then select a disc drive if you have a physical setup disc inserted into your computer. If you have an ISO file (copy of a physical disc), click *Installer disc image file (iso)* then select it. Click Next  after choosing your options.

* If VMware cannot recognize which operating system the disc/ISO file is supposed to install, you may have to select it manually after performing this step.
* If VMware Workstation recognizes that the installation media you chose is for Windows XP or later, you might be asked to enter the product key (optional), username (required), and password (optional), and select the version of Windows to install (some Windows setup discs enable you to select one out of several versions of the operating system to install). After providing the information and selecting the version of Windows to be installed, click next .
* VMware Workstation asks for this information because Easy Install (automatic installation) is supported for Windows XP and later. If you want to skip Easy Install, scroll down to the Tips section before performing Step 6.

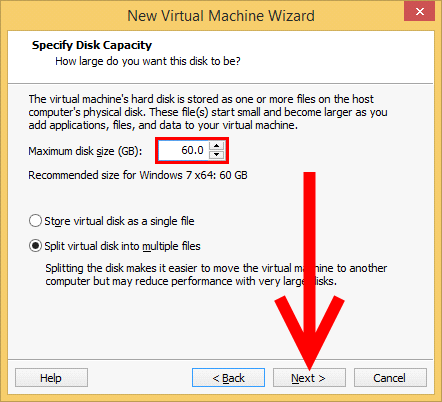
1. If the selected installation media installs a support Linux distro (such as Ubuntu), you might be asked to enter your name, username, and password. After providing the information, click Next .

* VMware Workstation asks for this information because Easy Install (automatic installation) is supported for the operating system. If you want to skip Easy Install, scroll down to the Tips section before performing Step 6.



**STEP:15**

**Name the virtual machine and choose a location to create the virtual machine in.** If you want to change the directory that the virtual machine is to be installed in, click Browse... and select it, or edit the path manually. Click Next after choosing your options.



**STEP:16**

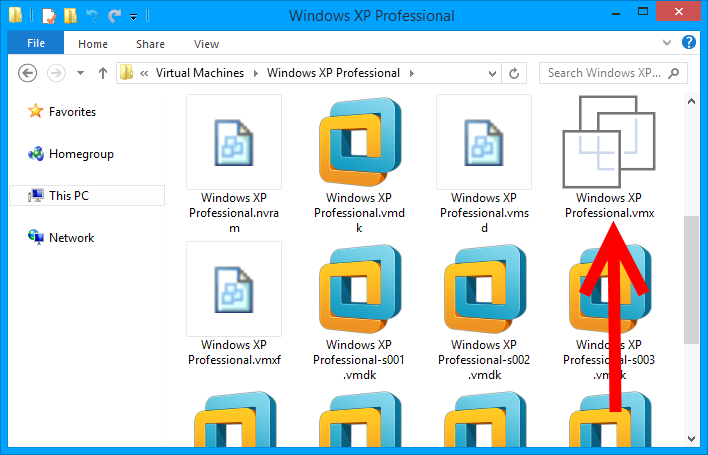
**Select the size that you want the virtual machine's hard drive to be and click Next .**

* The amount of space that you give to the virtual hard disk will not be immediately allocated. The space taken up by the virtual hard drive's file on your actual hard drive equals the total size of the files written to it. For example, if you create a 100 GB virtual HDD, its file will take up no space on your real HDD, but if you install an operating system that takes up 5 GB of space in the virtual HDD, the file's size will increase to 5 GB



**STEP:17**

**Navigate to the directory of the virtual machine..**



ii) **Basic Linux Commands:**

**$ cal**

June 2018

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30

**Change Password** :

All Unix systems require passwords to help ensure that your files and data remain your own and that the system itself is secure from hackers and crackers. Following are the steps to change your password – Step 1: To start, type password at the command prompt as shown below. Step 2: Enter your old password, the one you're currently using.

Step 3: Type in your new password. Always keep your password complex enough so that nobody can guess it. But make sure, you remember it.

Step 4: You must verify the password by typing it again.

**$ passwd**

Changing password for user (current)

Unix password:\*\*\*\*\*\*

New Unix password:\*\*\*\*\*\*\*

Retype new Unix password:\*\*\*\*\*\*\*

passwd: all authentication tokens updated successfully

$

**Listing Directories and Files** :

All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.

You can use the ls command to list out all the files or directories available in a directory. Following is the example of using ls command with -l option.

**$ ls -l**

total 19621

$ls

To list the invisible files, specify the -a option to ls –a

$ ls -a

$ whoami

**Creating Files**

$ vi filename

You can use the cat command to see the content of a file.

$ cat filename

**Counting Words in a File**

$ wc filename

$ wc filename1 filename2 filename3

Following is the example to create a copy of the existing file filename.

$ cp filename copyfile

**Renaming Files**

$ mv old\_file new\_file

**Deleting Files**

$ rm filename

**Creating Directories**

$mkdir dirname

**Removing Directories**

$rmdir dirname

**Changing Directories**

$cd dirname

Week 2

**Hadoop shell commands are invoked by the bin/hdfs script.**

**Version**

**Hadoop HDFS version Command Usage**

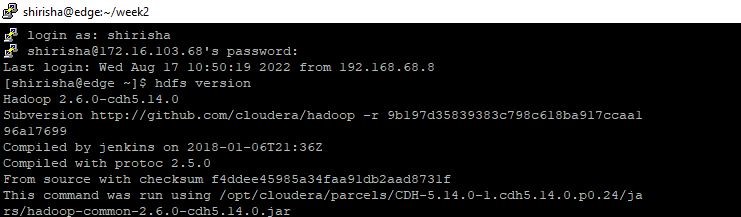
**1.version**

Hadoop HDFS version Command Example

Hdfs version

Hadoop HDFS version Command Description

This Hadoop command prints the Hadoop version



**2.mkdir**

Hadoop HDFS mkdir Command Usage

mkdir <path>

Hadoop HDFS mkdir Command Example

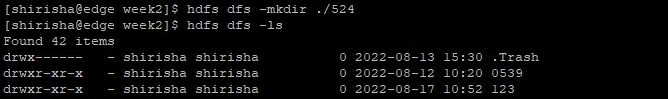
hdfs dfs -mkdir /user/dataflair/dir1

Hadoop HDFS mkdir Command Description

This HDFS command takes path URI’s as an argument and creates directories.

Creates any parent directories in path that are missing (e.g., mkdir -p in Linux).

Learn various features of Hadoop HDFS from this HDFS features guide.



**3.ls**

Hadoop HDFS ls Command Usage

ls <path>

Hadoop HDFS ls Command Example

hdfs dfs -ls /user/dataflair/dir1

Hadoop HDFS ls Commnad Description

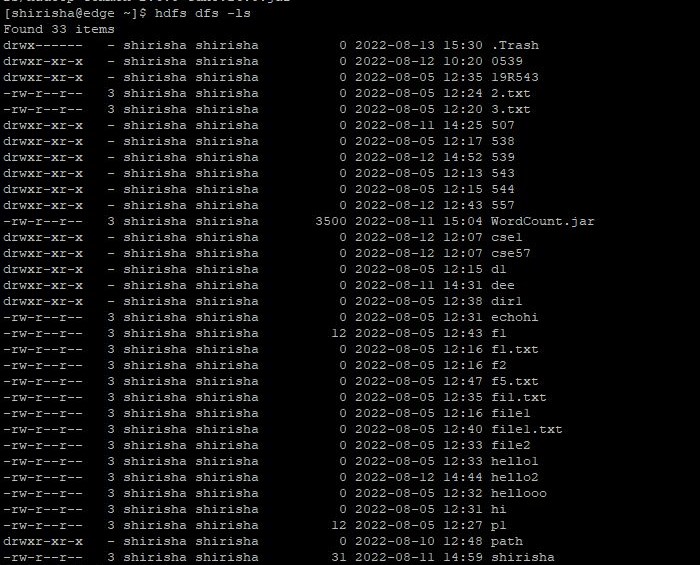
This Hadoop HDFS ls command displays a list of the contents of a directory specified by path provided by the user, showing the names, permissions, owner, size and modification date for each entry.

Hadoop HDFS ls Command Example

hdfs dfs -ls -R

Hadoop HDFS ls Description

This Hadoop fs command behaves like -ls, but recursively displays entries in all subdirectories of a path.



**4.put**

Hadoop HDFS put Command Usage

put <localSrc> <dest>

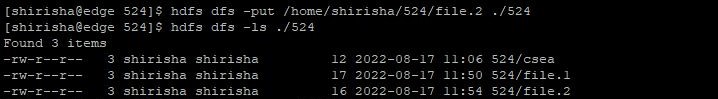
Hadoop HDFS put Command Example

hdfs dfs -put /home/dataflair/Desktop/sample /user/dataflair/dir1

Hadoop HDFS put Command Description

This hadoop basic command copies the file or directory from the local file system to the destination within the DFS.

Learn Internals of HDFS Data Write Pipeline and File write execution flow.



5.**copyFromLocal**

Hadoop HDFS copyFromLocal Command Usage

copyFromLocal <localSrc> <dest>

Hadoop HDFS copyFromLocal Command Example

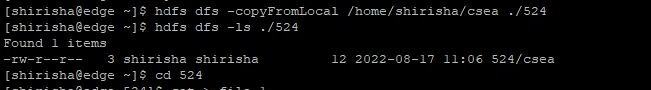
hdfs dfs -copyFromLocal /home/dataflair/Desktop/sample /user/dataflair/dir1

Hadoop HDFS copyFromLocal Command Description

This hadoop shell command is similar to put command, but the source is restricted to a local file reference.

Learn Internals of HDFS Data Read Operation, How Data flows in HDFS while reading the file.

Any Doubt yet in Hadoop HDFS Commands? Please Comment.



**6.get**

Hadoop HDFS get Command Usage

get [-crc] <src> <localDest>

Hadoop HDFS get Command Example

hdfs dfs -get /user/dataflair/dir2/sample /home/dataflair/Desktop

Hadoop HDFS get Command Description

This HDFS fs command copies the file or directory in HDFS identified by the source to the local file system path identified by local destination.



**HDFS commands in detail. Hadoop File System navigation and manipulation using commands. File Permission commands.**

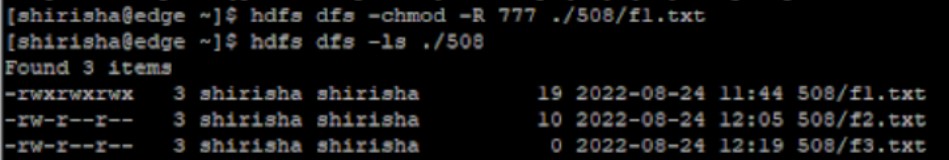
|  |  |
| --- | --- |
| **Sr.No** | **Command & Description** |
| 1 | **-ls <path>**  Lists the contents of the directory specified by path, showing the names, permissions, owner, size and modification date for each entry. |
| 2 | **-lsr <path>**  Behaves like -ls, but recursively displays entries in all subdirectories of path. |
| 3 | **-du <path>**  Shows disk usage, in bytes, for all the files which match path; filenames are reported with the full HDFS protocol prefix. |
| 4 | **-dus <path>**  Like -du, but prints a summary of disk usage of all files/directories in the path. |
| 5 | **-mv <src><dest>**  Moves the file or directory indicated by src to dest, within HDFS. |
| 6 | **-cp <src> <dest>**  Copies the file or directory identified by src to dest, within HDFS. |
| 7 | **-rm <path>**  Removes the file or empty directory identified by path. |
| 9 | **-put <localSrc> <dest>**  Copies the file or directory from the local file system identified by localSrc to dest within the DFS. |
| 10 | **-copyFromLocal <localSrc> <dest>**  Identical to -put |
| 11 | **-moveFromLocal <localSrc> <dest>**  Copies the file or directory from the local file system identified by localSrc to dest within HDFS, and then deletes the local copy on success. |
| 12 | **-get [-crc] <src> <localDest>**  Copies the file or directory in HDFS identified by src to the local file system path identified by localDest. |
| 14 | **-cat <filen-ame>**  Displays the contents of filename on stdout. |
| 15 | **-copyToLocal <src> <localDest>**  Identical to -get |
| 17 | **-mkdir <path>**  Creates a directory named path in HDFS.  Creates any parent directories in path that are missing (e.g., mkdir -p in Linux). |
| 8. | **-rmr <path>**  Removes the file or directory identified by path. Recursively deletes any child entries (i.e., files or subdirectories of path). |
| 13. | **-getmerge <src> <localDest>**  Retrieves all files that match the path src in HDFS, and copies them to a single, merged file in the local file system identified by localDest. |
| 16. | **-moveToLocal <src> <localDest>**  Works like -get, but deletes the HDFS copy on success. |
| 18. | **-setrep [-R] [-w] rep <path>**  Sets the target replication factor for files identified by path to rep. (The actual replication factor will move toward the target over time) |
| 19. | **-touchz <path>**  Creates a file at path containing the current time as a timestamp. Fails if a file already exists at path, unless the file is already size 0. |
| 20. | **-test -[ezd] <path>**  Returns 1 if path exists; has zero length; or is a directory or 0 otherwise. |
| 21. | **-stat [format] <path>**  Prints information about path. Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y). |
| 22. | **-tail [-f] <file2name>**  Shows the last 1KB of file on stdout. |

**Hadoop Commands for File Permissions**

**Hadoop fs – chmod** – Just like in Linux shell, changes access permission of files and directories.

hadoop fs -chmod -R 777 /user/nasa/results

Changes permissions of nasa/results directory and sub-directories to 777.



Hadoop fs -chown – Change ownership of file or directories.

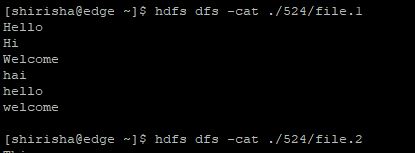
hadoop fs -chown -R hdfs /user/nasa/results

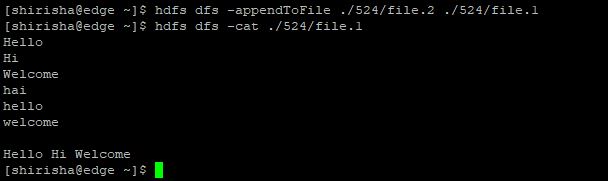
....

Change ownership of nasa/results directories and sub-directories to hdfs.

**appendToFile**

Usage: hdfs dfs -appendToFile <localsrc> ... <dst>





**Week 3**

i) Implement the following file management tasks in Hadoop:

* Adding files and directories
* Retrieving files
* Deleting files and directories.

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, HFTP FS, S3 FS, and others. The FS shell is invoked by:

bin/hadoop fs <args>

**appendToFile**

Usage: hdfs dfs -appendToFile <localsrc> ... <dst>

**cat**

Usage: hdfs dfs -cat URI [URI ...]

**CopyFromLocal**

Usage: hdfs dfs -copyFromLocal <localsrc> URI

Similar to put command, except that the source is restricted to a local file reference.

Options:

The -f option will overwrite the destination if it already exists.

**copyToLocal**

Usage: hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

Similar to get command, except that the destination is restricted to a local file reference.

**mkdir**

Usage: hdfs dfs -mkdir [-p] <paths>

**-rm <path>**

Removes the file or empty directory identified by path.

**-rmr <path>**

Removes the file or directory identified by path. Recursively deletes any child entries (i.e., files or subdirectories of path).

**1. Create a directory in HDFS at given path(s).**

Usage:

hadoop fs -mkdir <paths>

Example:

hadoop fs -mkdir /user/saurzcode/dir1 /user/saurzcode/dir2

**2.  List the contents of a directory.**

Usage :

hadoop fs -ls <args>

Example:

hadoop fs -ls /user/saurzcode

**3. Upload and download a file in HDFS.**

***Upload*:**

**hadoop fs -put:**

Copy single src file, or multiple src files from local file system to the Hadoop data file system

Usage:

hadoop fs -put <localsrc> ... <HDFS\_dest\_Path>

Example:

hadoop fs -put /home/saurzcode/Samplefile.txt /user/saurzcode/dir3/

**Download:**

**hadoop fs -get:**

Copies/Downloads files to the local file system

Usage:

hadoop fs -get <hdfs\_src> <localdst>

Example:

hadoop fs -get /user/saurzcode/dir3/Samplefile.txt /home/

**4. See contents of a file**

Same as unix cat command:

Usage:

hadoop fs -cat <path[filename]>

Example:

hadoop fs -cat /user/saurzcode/dir1/abc.txt

**5. Copy a file from source to destination**

This command allows multiple sources as well in which case the destination must be a directory.

Usage:

hadoop fs -cp <source> <dest>

Example:

hadoop fs -cp /user/saurzcode/dir1/abc.txt /user/saurzcode/dir2

**6. Copy a file from/To Local file system to HDFS**

**copyFromLocal**

Usage:

hadoop fs -copyFromLocal <localsrc> URI

Example:

hadoop fs -copyFromLocal /home/saurzcode/abc.txt /user/saurzcode/abc.txt

Similar to put command, except that the source is restricted to a local file reference.

**copyToLocal**

Usage:

hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

Similar to get command, except that the destination is restricted to a local file reference.

**7. Move file from source to destination.**

Note:- Moving files across filesystem is not permitted.

Usage :

hadoop fs -mv <src> <dest>

Example:

hadoop fs -mv /user/saurzcode/dir1/abc.txt /user/saurzcode/dir2

**8. Remove a file or directory in HDFS.**

Remove files specified as argument. Deletes directory only when it is empty

Usage :

hadoop fs -rm <arg>

Example:

hadoop fs -rm /user/saurzcode/dir1/abc.txt

WEEK- 4

i) Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

Program:

package com.mlrit.mapreduce;

import java.io.IOException;

import java.util.Iterator;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

public class WordCount{

public static class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

private static final IntWritable one = new IntWritable(1);

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context)

throws IOException, InterruptedException {

String word;

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

word = tokenizer.nextToken();

context.write(new Text(word), one);

}

}

}

public static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

@Override

protected void reduce(Text word, Iterable<IntWritable> list,

Reducer<Text, IntWritable, Text, IntWritable>.Context context)

throws IOException, InterruptedException {

int sum = 0;

Iterator<IntWritable> iterator = list.iterator();

while (iterator.hasNext()) {

sum += iterator.next().get();

}

context.write(word, new IntWritable(sum));

}

}

public static void main(String[] args) {

try {

Job job = new Job(new Configuration(),"WordCount");

job.setJarByClass(WordCount.class);

job.setInputFormatClass(TextInputFormat.class);

job.setMapperClass(WordCountMapper.class);

job.setCombinerClass(WordCountReducer.class);

job.setReducerClass(WordCountReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setNumReduceTasks(2);

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

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

job.waitForCompletion(true);

} catch (Exception e) {

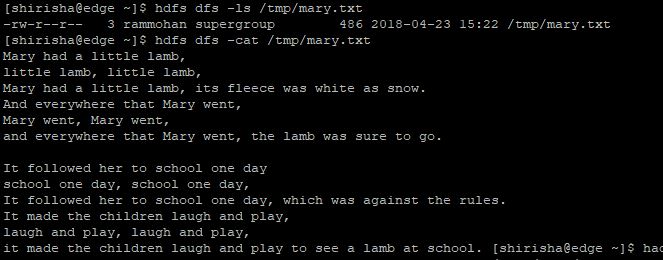
e.printStackTrace();

}

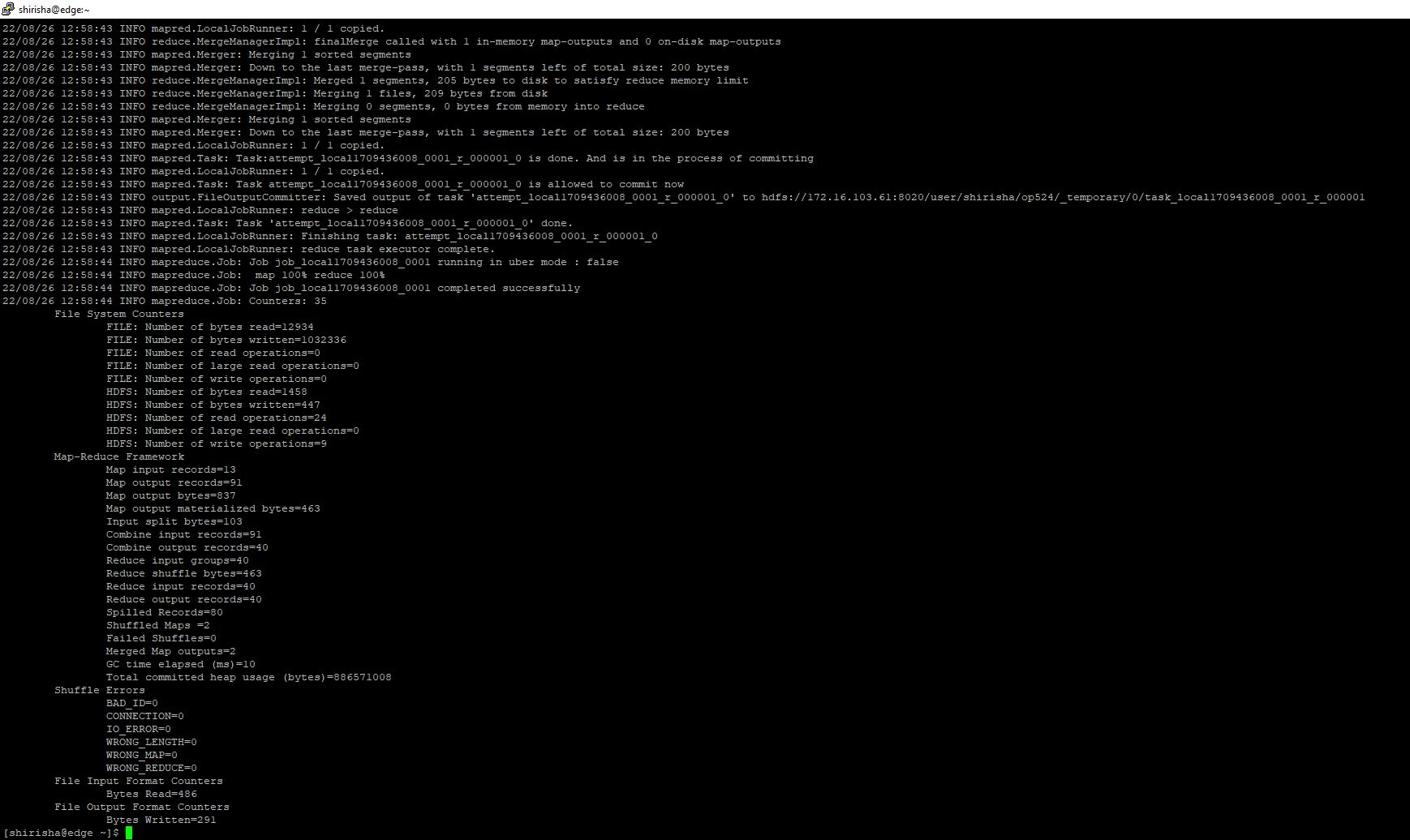
}

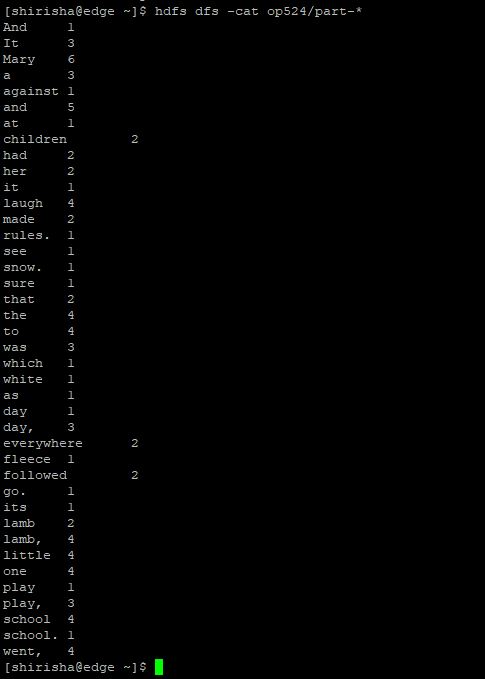
}

Output:









WEEK- 5

**MapReduce program to mine weather data**

**import** java.io.IOException;

**import** org.apache.hadoop.fs.Path;

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

**import** org.apache.hadoop.io.LongWritable;

**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** WEEK5524 {

**enum** Temperature{

***MISSING***,

***MALFORMED***,

***TOTAL***

}

**private** **static** **class** MaxTemperatureMapper **extends** Mapper<LongWritable, Text, Text, IntWritable> {

**private** **static** **final** **int** ***MISSING*** = 9999;

@Override

**public** **void** map(LongWritable key, Text value, Context context)**throws** IOException, InterruptedException {

String line = value.toString();

String year = line.substring(15, 19);

**int** airTemperature;

**if** (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs

airTemperature = Integer.*parseInt*(line.substring(88, 92));

} **else** {

airTemperature = Integer.*parseInt*(line.substring(87, 92));

}

String quality = line.substring(92, 93);

**if** (airTemperature != ***MISSING*** &&quality.matches("[01459]")) {

context.write(**new** Text(year), **new** IntWritable(airTemperature));

}**else**{

context.getCounter(Temperature.***MISSING***).increment(1);

}

context.getCounter(Temperature.***TOTAL***).increment(1);

}

}

**private** **static** **class** MaxTemperatureReducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {

@Override

**public** **void** reduce(Text key, Iterable<IntWritable> values,

Context context)

**throws** IOException, InterruptedException {

**int** maxValue = Integer.***MIN\_VALUE***;

**for** (IntWritable value : values) {

maxValue = Math.*max*(maxValue, value.get());

}

context.write(key, **new** IntWritable(maxValue));

}

}

**public** **static** **void** main(String[] args) **throws** Exception {

**if** (args.length != 2) {

System.***err***.println("Usage: maxtemp<input path><output path>");

System.*exit*(-1);

}

Job job = **new** ~~Job~~();

job.setJarByClass(WEEK5524.**class**);

job.setJobName("Max temperature");

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

job.setMapperClass(MaxTemperatureMapper.**class**);

job.setCombinerClass(MaxTemperatureReducer.**class**);

job.setReducerClass(MaxTemperatureReducer.**class**);

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

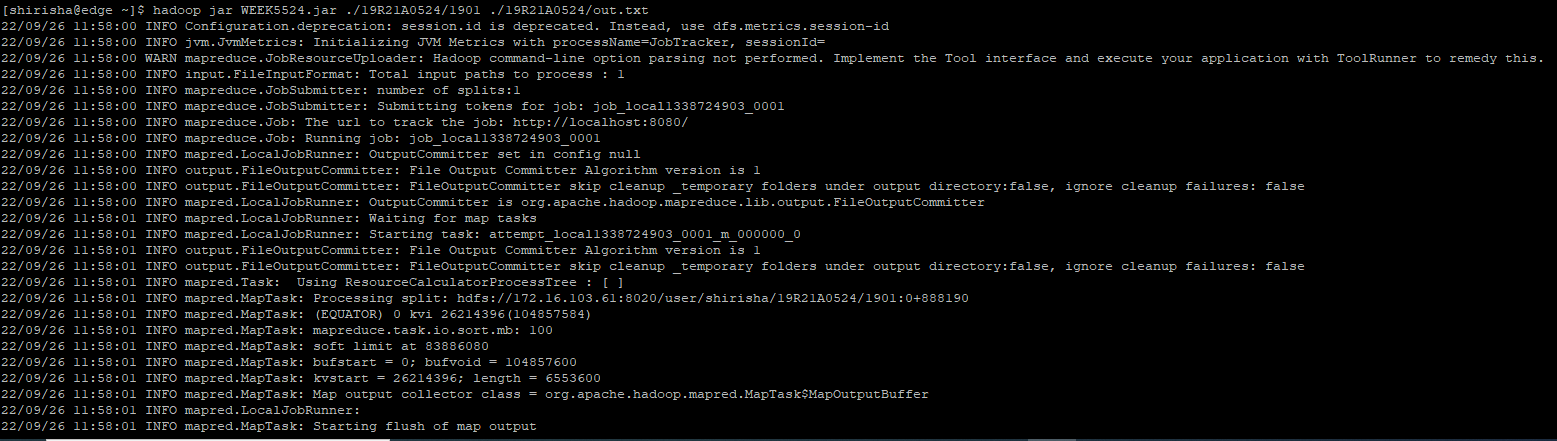
job.setOutputValueClass(IntWritable.**class**);

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

}

}

Output:-



C:\Users\M LRIT\Desktop\524\2.PNG

**WEEK – 6**

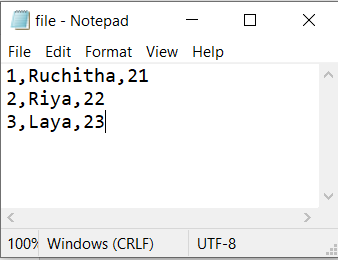
**Run PIG and perform basic PIG Commands**

**Local Mode:**

**To enter into local mode**

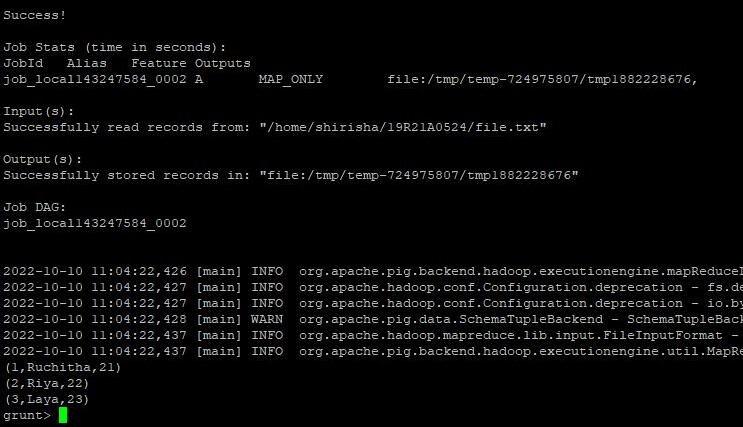
****

**file.txt**

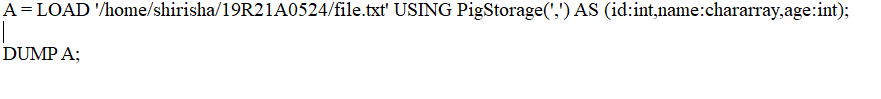


**1. Write pig command line script for student data**

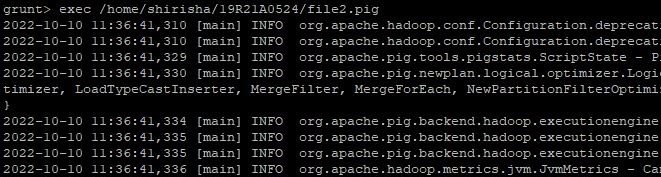


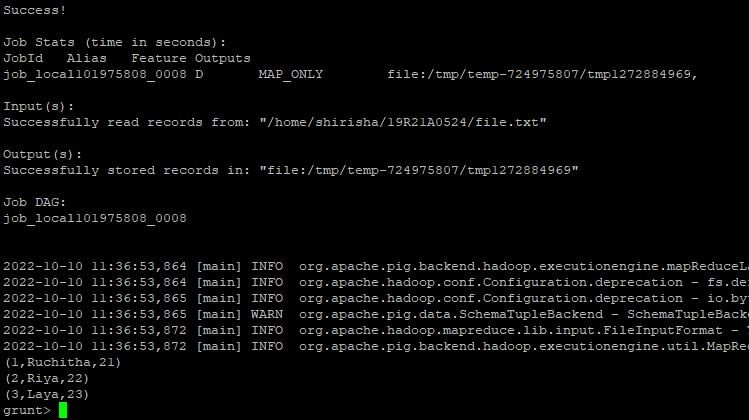


**2. Write pig script in a file for student data**



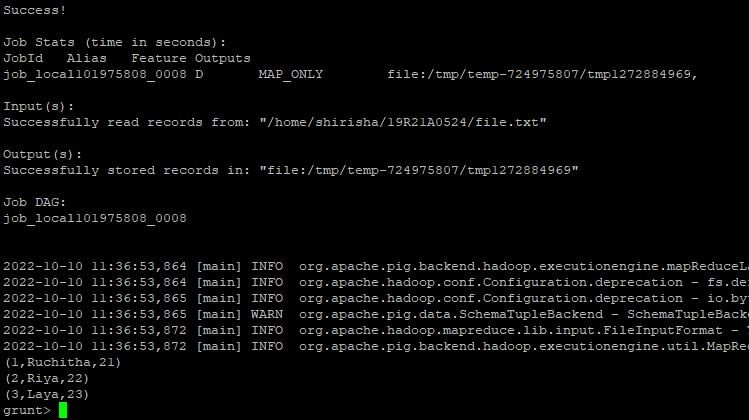
**3. Execute the pig script by execute command**



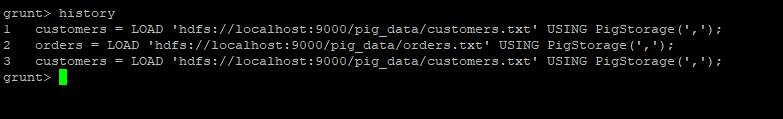


**4. Execute the pig script by run command**

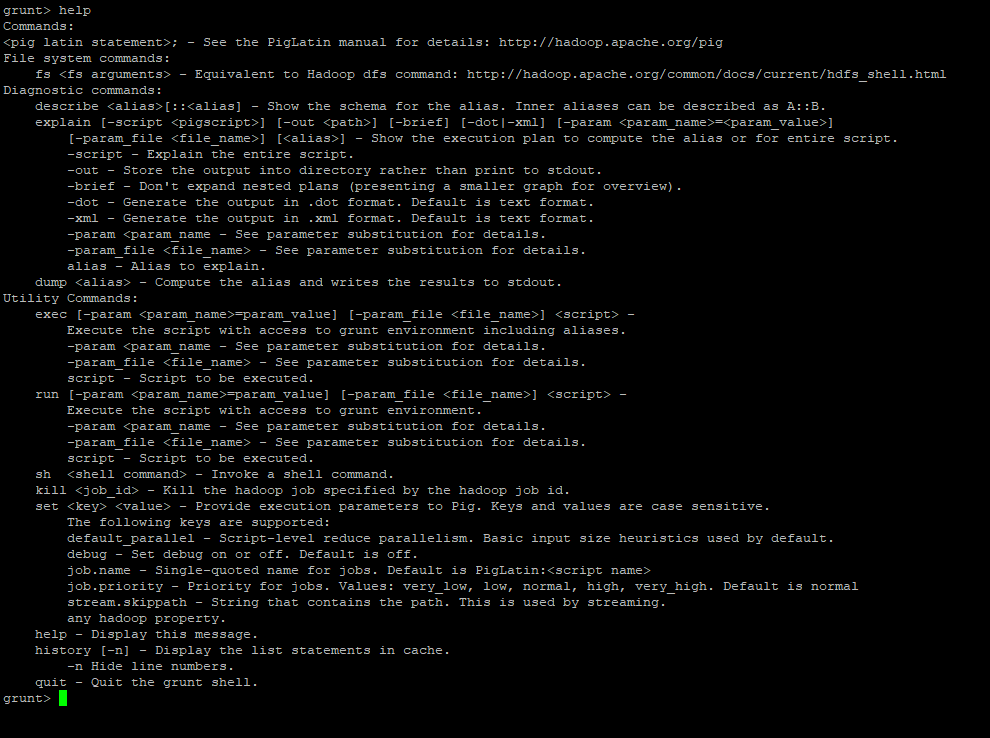
****



**5. Execute the history command**



**6. Execute the help command**



**7. Execute the kill command**

****

**8. Execute the clear command**



**9. Execute the quit command**



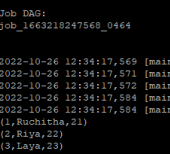
**Mapreduce Mode:**

**To enter into map reduce mode**

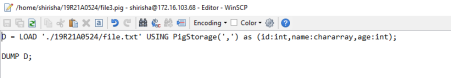
****

**First move the f1.txt and p1.pig files to hadoop/cluster**

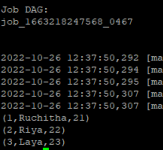
**1. Write pig command line script for student data**

****

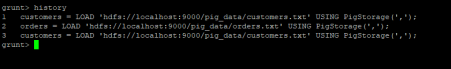
**2. Write pig script in a file for student data**

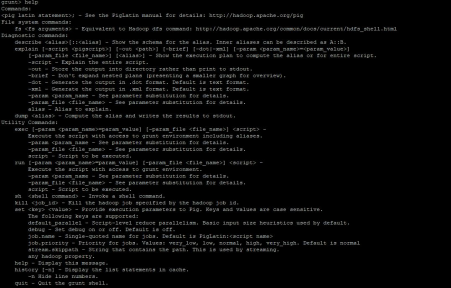
****

****

****

**3. Execute the history command**

**4.Execute the help command**

****

**5. Execute the kill command**

**6.Execute the clear command**

****

**7. Execute the quit command**

****

**WEEK – 7**

**Pig Latin scripts to sort, group, join, and filter your data**

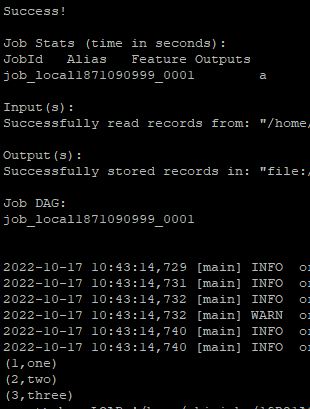
**Local Mode:**

**To enter into local mode**

****

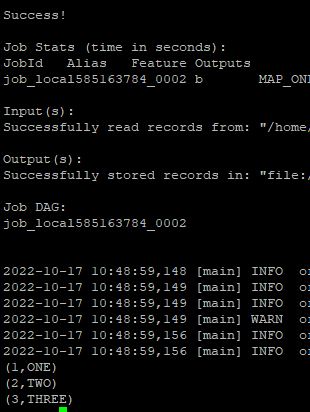
**file1.csv**

**C:\Users\MLRIT\Desktop\524\week 7\1.JPG**

****

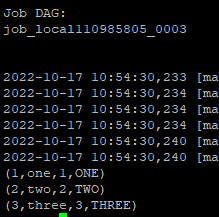
**file2.csv**

**C:\Users\MLRIT\Desktop\524\week 7\3.JPG**

****

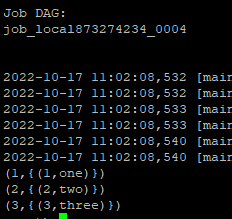
**JOIN**

****

****

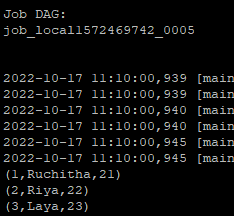
**Group**

****

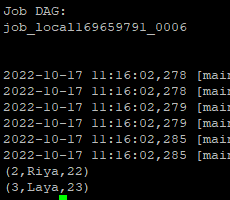
****

**Filter**

****

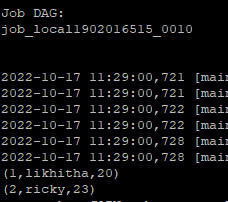
****

****

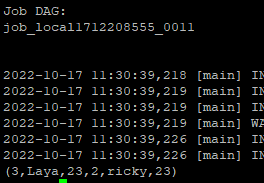
****

**Inner join**

****

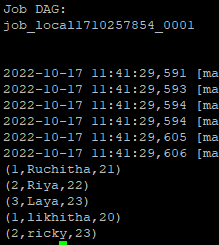
****

****

****

**Union**

****

****

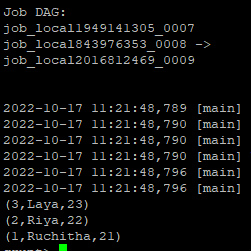
**Foreach**

****

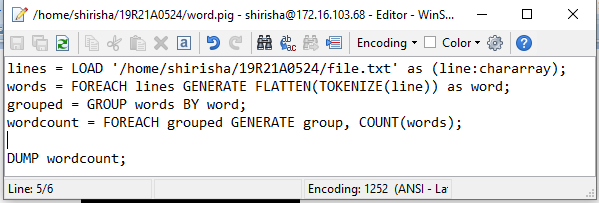
****

**Sort / Order**

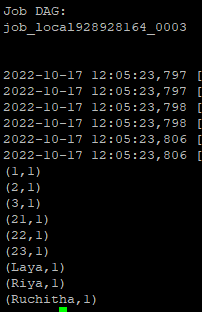
****

****

**WORD COUNT**

****

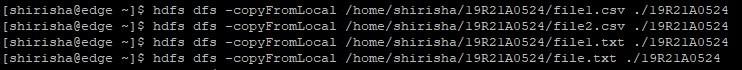
****

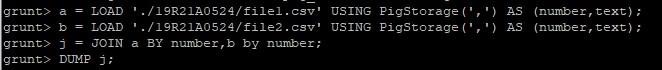
****

**Mapreduce Mode:**

# To enter into map reduce mode



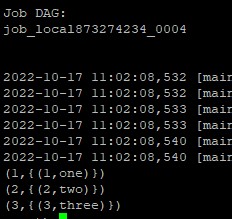




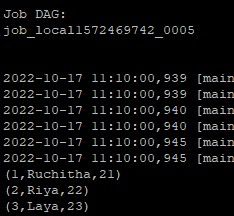
****

**Group**

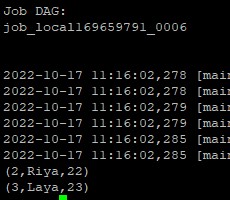




**Filter**



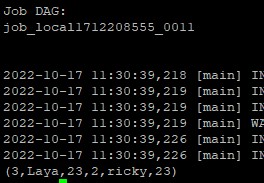




# Inner join

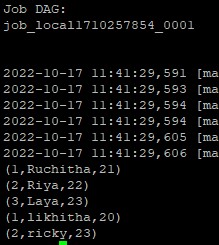


|  |
| --- |
|  |



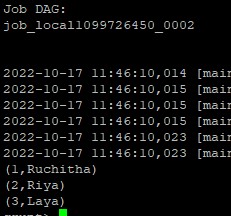
# Union





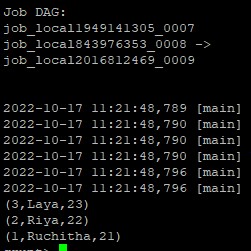
# Foreach





Sort / Order

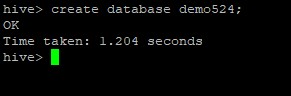


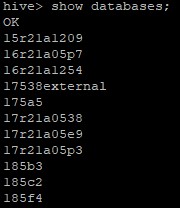


**WEEK – 8**

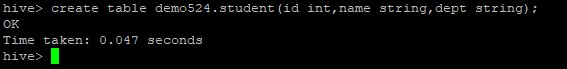
Run HIVE and perform basic HIVE commands to create a table and enter data into tables

1.Create Database:

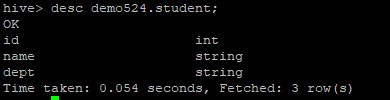




2.Create table:

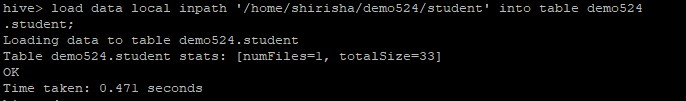


3.Describe:

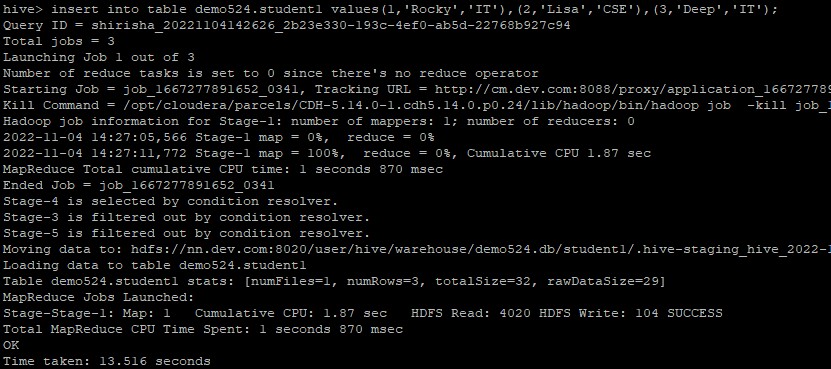


4.Load:

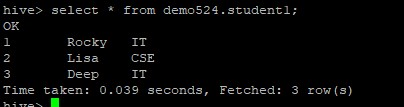
Once the internal table has been created, the next step is to load the data into it. So, in Hive, we can easily load data from any file to the database.



5.Insert:



Select:



Alter:

In Hive, we can perform modifications in the existing table like changing the table name, column name, comments, and table properties. It provides SQL like commands to alter the table.

Add columns:



Rename:

If we want to change the name of an existing table, we can rename that table by using the following

signature

Alter table old\_table\_name rename to new\_table\_name;



Change column name:

In Hive, we can add one or more columns in an existing table by using the following signature:

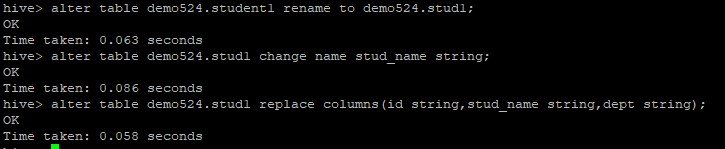
Alter table table\_name add columns(column\_name datatype);



Replace column name:

Hive allows us to delete one or more columns by replacing them with the new columns. Thus, we cannot drop the column directly.

alter table employee\_data replace columns( id string, first\_name string, age int);



Drop:

Hive facilitates us to drop a table by using the SQL **drop table** command. Let's follow the below steps to drop the table from the database.

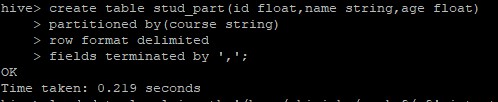


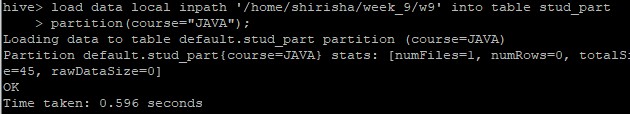
WEEK – 9

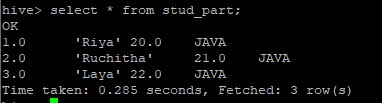
Run Hive Functions, Views and Indexes

Partitioning:

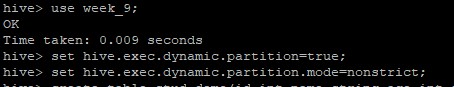
Static Partitioning

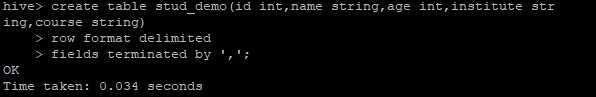


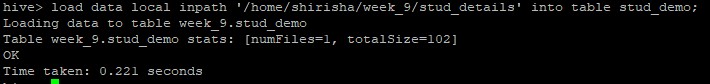


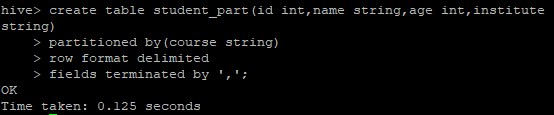


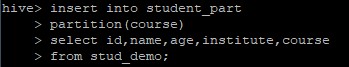
Dynamic Partitioning

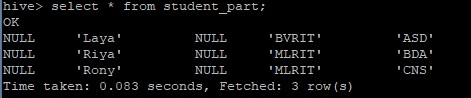




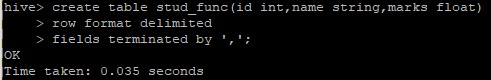


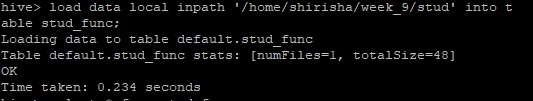


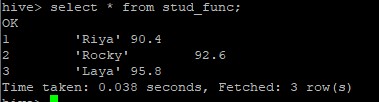




Functions







Mathematical functions in hive:

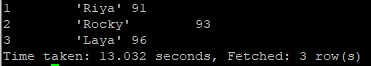
Round:



Floor:

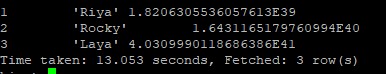


**Ceil and Celing:**



**Exponential:**

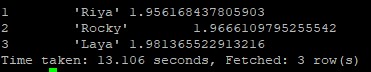




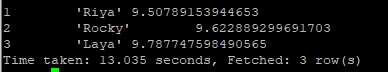
**Logorithm:**



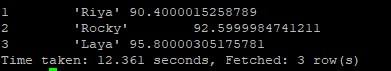
**Log10:**



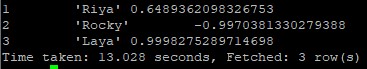
**Sqrt:**



**Abs:**

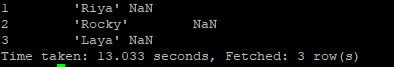


**Sin:**



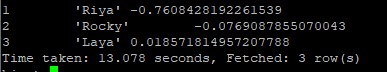
**asin():**



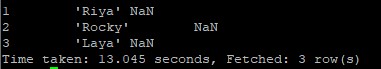


**Cos:**



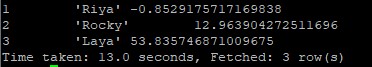


**acos:**



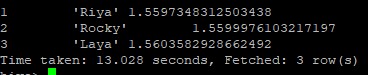
**tan:**





**atan:**





**Aggregate Functions:**

**Max:**



**Min:**





**Count:**





**Sum:**





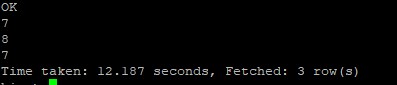
**Avg:**





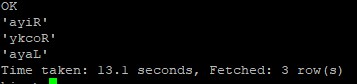
**Length:**





**Reverse:**





**Concat:**





**Substr:**





**Upper:**





**Lower:**





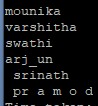
**Trim:**





**Rtrim:**





**VIEWS:**

**Creating a view:**

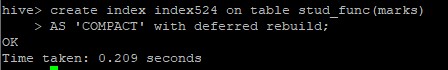


**Dropping a view:**

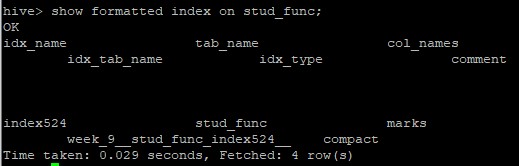


**Index:**

**Create Index:**

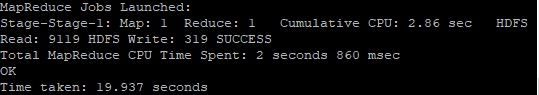


**Show Index:**



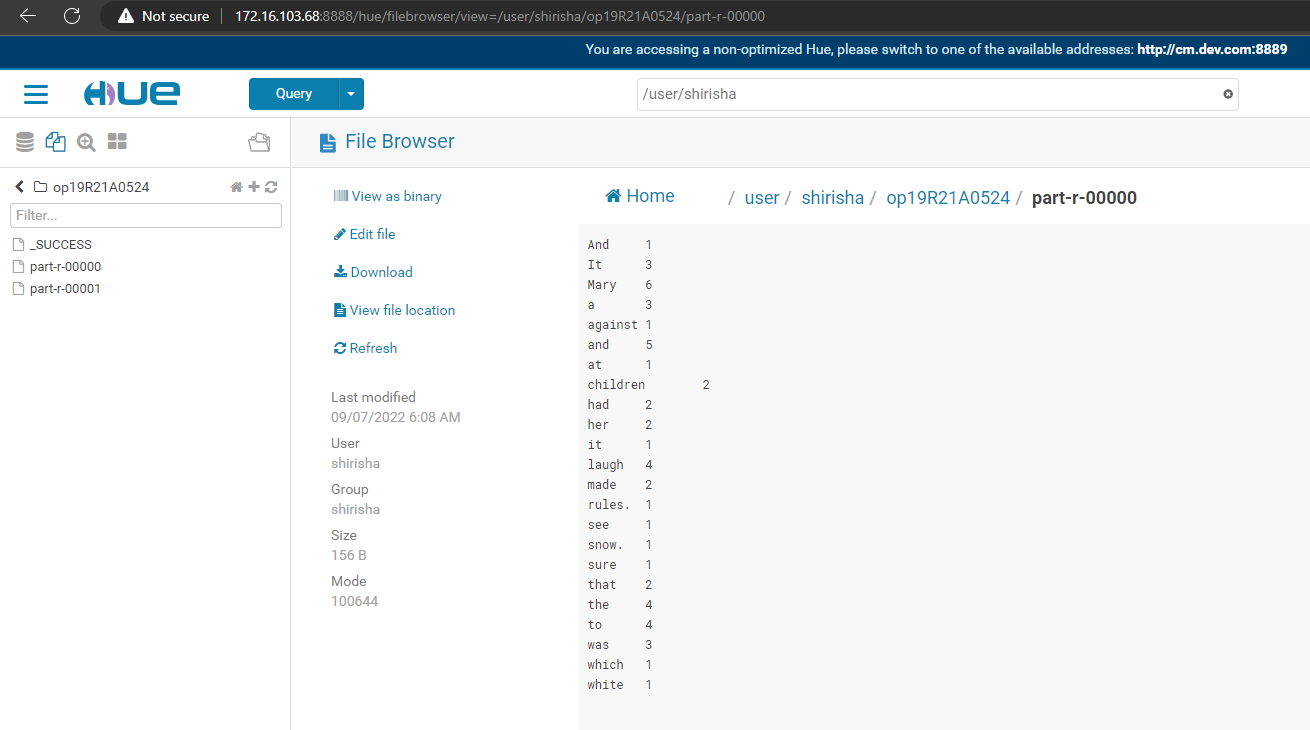
**Alter a Index:**

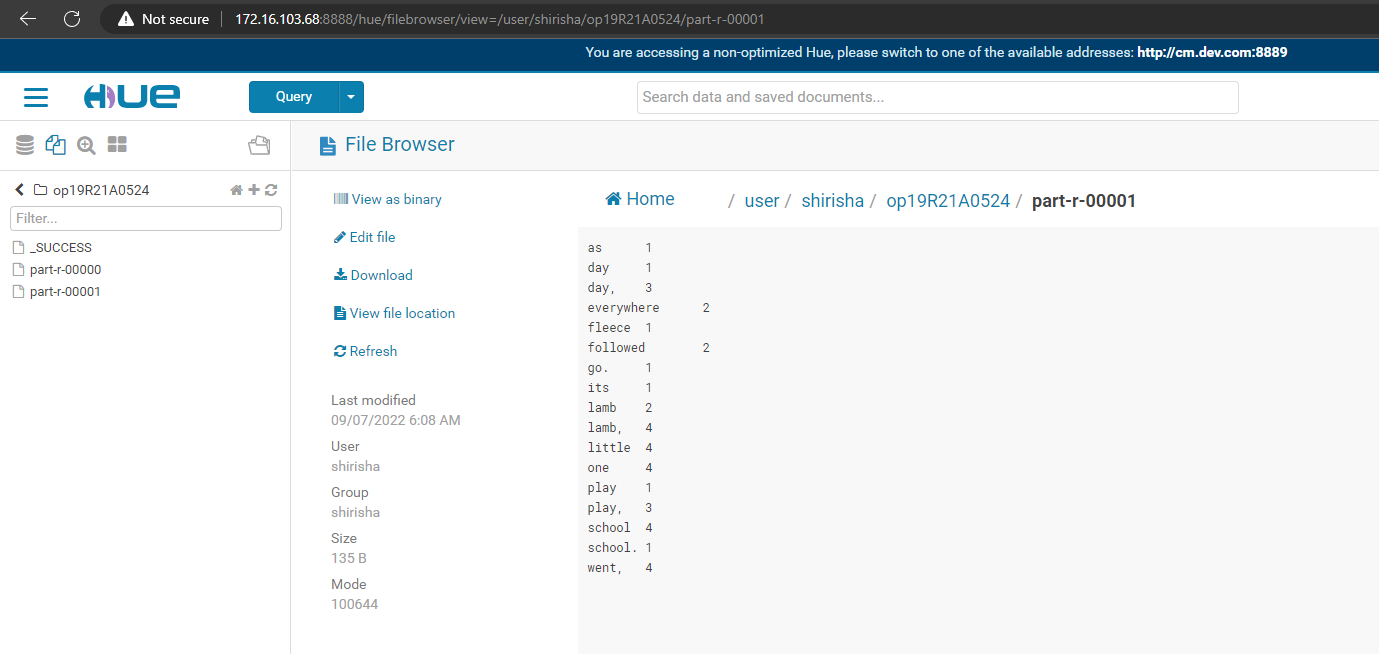




WEEK – 10

Use CDH and HUE to analyze data and generate reports for sample datasets





WEEK – 12

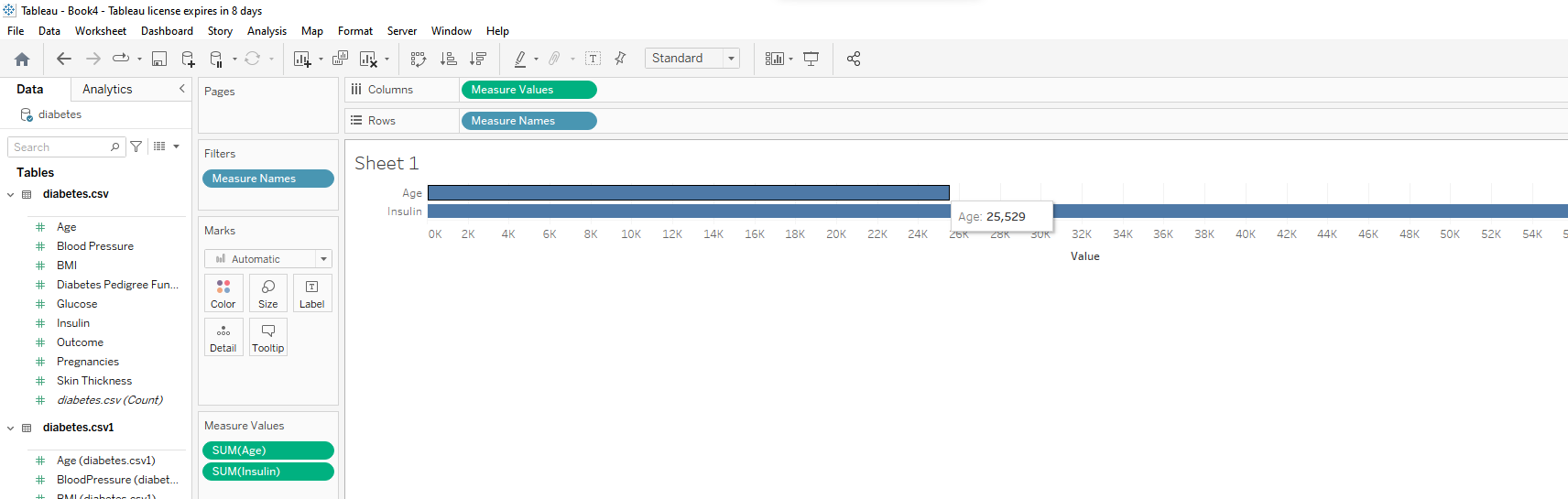
Use Tableau data visualization tool to generate reports on sample datasets.

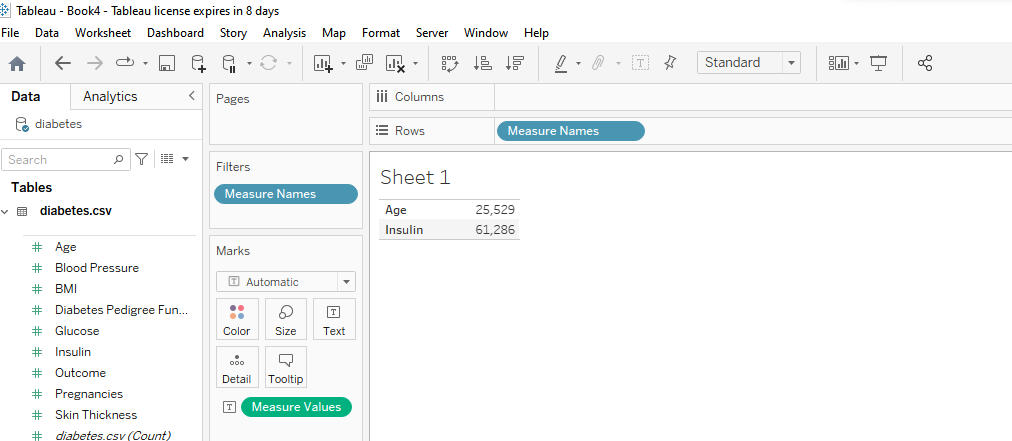
European Parliament Election 2019:

Every five years, EU citizens vote on who will represent them in the European Parliament. Explore the visualization to see which political party won the most seats in 2019.



Diabetes.csv





Sample Superstore:

