1. Implement word count / frequency programs using MapReduce/Hive/Pig
2. To find the Length of the word program using MapReduce/Hive/Pig
3. WIKI datamining program to find the page counts using PIG(take 3 fields like language(en,us), Search engine name(yahoo,google), pageclicks)
4. To find the Length of the Voter data with city using Hive
5. Take 6 subjects marks of the students , find the max and min marks in the each subject and also find the count of pass and fail students in each the subject using Pig/Hive
6. Write Pig Latin scripts sort, group, join
7. Write Pig Latin scripts to cross, and filter your data.
8. Use Hive to create, insert, alter, and drop databases, tables.

II. R Programming:

1. Implement Linear Regression
2. Implement logistic Regression
3. Visualize data using R-Pie Charts plotting framework
4. Visualize data using R-Barcharts plotting framework
5. Visualize data using R-BoxPlots plotting framework
6. Visualize data using R-Histograms plotting framework
7. Visualize data using R-Line Charts plotting framework
8. Visualize data using R-Scatter Plots plotting framework
9. Word Count In Hive

CREATE TABLE FILES (line STRING);

LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE FILES;

CREATE TABLE word\_counts AS

SELECT word, count(1) AS count FROM

(SELECT explode(split(line, ' ')) AS word FROM FILES) w

GROUP BY word

ORDER BY word;

WordCount :--

i/p:--

sasi

siva

ram

som

sasi

siva

thina

o/p:--

ram 1

sasi 2

siva 2

som 1

thina 1

**Word Count In PIG**

records = LOAD '/words' as (names:chararray);

grp\_records = GROUP records by names;

ram ({ram})

sasi({sasi}, {sasi}

........

results = FOREACH grp\_records generate group,COUNT(records.names);

ram 1

sasi 2

...

sorted\_results = ORDER results by $0 ASC;

STORE sorted\_results into '/wordsop1';

single line comments --

multiple line comments /\* .... \*/

gedit abcd.pig

copy paste code (exclude my explanation lines)

pig -f /home/vssit/abcd.pig

WordCount Program :--

i/p:--

sasi

siva

ram

som

sasi

siva

thina

o/p:--

ram 1

sasi 2

siva 2

som 1

thina 1

1. **To find the Length of the word program using MapReduce /Pig**

grunt> employee\_data = LOAD ‘/home/vssit/Desktop/college’ USING PigStorage(',') as (id:int, name:chararray);

grunt> size = FOREACH employee\_data GENERATE SIZE(name);

grunt> Dump size;

input

123,abc

566,hjdhfjsgf,

234,fhfkhg

Output

(3)

(9)

(6)

1. WIKI datamining program to find the page counts using PIG(take 3 fields like language(en,us), Search engine name(yahoo,google), pageclicks)

See your lab manual

1. To find the Length of the Voter data with city using Hive

See your lab manual

1. Take details of the students , find the max and min marks .

A = LOAD 'student' AS (name:chararray, session:chararray, gpa:float);

DUMP A;

(John,fl,3.9F)

(John,wt,3.7F)

(John,sp,4.0F)

(John,sm,3.8F)

(Mary,fl,3.8F)

(Mary,wt,3.9F)

(Mary,sp,4.0F)

(Mary,sm,4.0F)

B = GROUP A BY name;

DUMP B;

(John,{(John,fl,3.9F),(John,wt,3.7F),(John,sp,4.0F),(John,sm,3.8F)})

(Mary,{(Mary,fl,3.8F),(Mary,wt,3.9F),(Mary,sp,4.0F),(Mary,sm,4.0F)})

X = FOREACH B GENERATE group, MAX(A.gpa);

DUMP X;

(John,4.0F)

(Mary,4.0F)

Y = FOREACH B GENERATE group, MIN(A.gpa);

DUMP Y;

(John,3.7F)

(Mary,3.8F)

1. Find the count of tuples by using Pig

A = LOAD 'data' AS (f1:int,f2:int,f3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

B = GROUP A BY f1;

DUMP B;

(1,{(1,2,3)})

(4,{(4,2,1),(4,3,3)})

(7,{(7,2,5)})

(8,{(8,3,4),(8,4,3)})

X = FOREACH B GENERATE COUNT(A);

DUMP X;

(1L)

(2L)

(1L)

(2L)

Take dtudent details , find the Average marks for each student using pig

A = LOAD 'student.txt' AS (name:chararray, term:chararray, gpa:float);

DUMP A;

(John,fl,3.9F)

(John,wt,3.7F)

(John,sp,4.0F)

(John,sm,3.8F)

(Mary,fl,3.8F)

(Mary,wt,3.9F)

(Mary,sp,4.0F)

(Mary,sm,4.0F)

B = GROUP A BY name;

DUMP B;

(John,{(John,fl,3.9F),(John,wt,3.7F),(John,sp,4.0F),(John,sm,3.8F)})

(Mary,{(Mary,fl,3.8F),(Mary,wt,3.9F),(Mary,sp,4.0F),(Mary,sm,4.0F)})

C = FOREACH B GENERATE A.name, AVG(A.gpa);

DUMP C;

({(John),(John),(John),(John)},3.850000023841858)

({(Mary),(Mary),(Mary),(Mary)},3.925000011920929)

6. Write Pig Latin scripts sort, group, join, cross, and filter your data.

A = load 'data' AS (f1:chararray,f2:int,f3:chararray);

DUMP A;

(David,1,N)

(Tete,2,N)

(Ranjit,3,M)

(Ranjit,3,P)

(David,4,Q)

(David,4,Q)

(Jillian,8,Q)

(JaePak,7,Q)

(Michael,8,T)

(Jillian,8,Q)

(Jose,10,V)

**SORT:**

C = rank A by f1 DESC, f2 ASC;

dump C;

(1,Tete,2,N)

(2,Ranjit,3,M)

(2,Ranjit,3,P)

(4,Michael,8,T)

(5,Jose,10,V)

(6,Jillian,8,Q)

(6,Jillian,8,Q)

(8,JaePak,7,Q)

(9,David,1,N)

(10,David,4,Q)

(10,David,4,Q)

**GROUP:** When using the GROUP operator with a single relation, records with a null group key are grouped together.

A = load 'student' as (name:chararray, age:int, gpa:float);

dump A;

(joe,18,2.5)

(sam,,3.0)

(bob,,3.5)

X = group A by age;

dump X;

(18,{(joe,18,2.5)})

(,{(sam,,3.0),(bob,,3.5)})

**JOIN :** The JOIN operator - when performing inner joins - adheres to the SQL standard and disregards (filters out) null values.

A = load 'student' as (name:chararray, age:int, gpa:float);

B = load 'student' as (name:chararray, age:int, gpa:float);

dump B;

(joe,18,2.5)

(sam,,3.0)

(bob,,3.5)

X = join A by age, B by age;

dump X;

(joe,18,2.5,joe,18,2.5)

#### CROSS:

Use the CROSS operator to compute the cross product (Cartesian product) of two or more relations.

CROSS is an expensive operation and should be used sparingly.

#### Example

Suppose we have relations A and B.

A = LOAD 'data1' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

B = LOAD 'data2' AS (b1:int,b2:int);

DUMP B;

(2,4)

(8,9)

(1,3)

In this example the cross product of relation A and B is computed.

X = CROSS A, B;

DUMP X;

(1,2,3,2,4)

(1,2,3,8,9)

(1,2,3,1,3)

(4,2,1,2,4)

(4,2,1,8,9)

(4,2,1,1,3)

**FILTER**

#### Usage

Use the FILTER operator to work with tuples or rows of data (if you want to work with columns of data, use the FOREACH...GENERATE operation).

FILTER is commonly used to select the data that you want; or, conversely, to filter out (remove) the data you don’t want.

#### Examples

Suppose we have relation A.

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

In this example the condition states that if the third field equals 3, then include the tuple with relation X.

X = FILTER A BY f3 == 3;

DUMP X;

(1,2,3)

(4,3,3)

(8,4,3)

In this example the condition states that if the first field equals 8 or if the sum of fields f2 and f3 is not greater than first field, then include the tuple relation X.

X = FILTER A BY (f1 == 8) OR (NOT (f2+f3 > f1));

DUMP X;

(4,2,1)

(8,3,4)

(7,2,5)

(8,4,3)

1. Use Hive to create, insert, alter, join and drop databases, tables.

## Create Database Statement

Create Database is a statement used to create a database in Hive. A database in Hive is a **namespace** or a collection of tables. The **syntax** for this statement is as follows:

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named **userdb**:

hive> CREATE DATABASE [IF NOT EXISTS] userdb;

**or**

hive> CREATE SCHEMA userdb;

The following query is used to verify a databases list:

hive> SHOW DATABASES;

default

userdb

hive> DROP DATABASE IF EXISTS userdb;

## Create Table Statement

Create Table is a statement used to create a table in Hive. The syntax and example are as follows:

hive> CREATE TABLE IF NOT EXISTS employee ( eid int, name String,

salary String, destination String)

COMMENT ‘Employee details’

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ‘\t’

LINES TERMINATED BY ‘\n’;

**Insert Statement**

**Hive> insert into table employee values(123,’ravi’,20000,’hyd’);**

## Alter Table Statement

It is used to alter a table in Hive.

hive> ALTER TABLE employee RENAME TO emp;

**DROP**

When you drop a table from Hive Metastore, it removes the table/column data and their metadata.

hive> DROP TABLE IF EXISTS employee;

On successful execution of the query, you get to see the following response:

OK

Time taken: 5.3 seconds

hive> SELECT \* FROM employee WHERE salary>30000;

hive> SELECT Id, Name, Dept FROM employee ORDER BY DEPT;

hive> SELECT Dept,count(\*) FROM employee GROUP BY DEPT;

## JOIN

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

There are different types of joins given as follows:

* JOIN
* LEFT OUTER JOIN
* RIGHT OUTER JOIN
* FULL OUTER JOIN
* We will use the following two tables in this chapter. Consider the following table named CUSTOMERS..
* +----+----------+-----+-----------+----------+
* | ID | NAME | AGE | ADDRESS | SALARY |
* +----+----------+-----+-----------+----------+
* | 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
* | 2 | Khilan | 25 | Delhi | 1500.00 |
* | 3 | kaushik | 23 | Kota | 2000.00 |
* | 4 | Chaitali | 25 | Mumbai | 6500.00 |
* | 5 | Hardik | 27 | Bhopal | 8500.00 |
* | 6 | Komal | 22 | MP | 4500.00 |
* | 7 | Muffy | 24 | Indore | 10000.00 |
* +----+----------+-----+-----------+----------+
* Consider another table ORDERS as follows:
* +-----+---------------------+-------------+--------+
* |OID | DATE | CUSTOMER\_ID | AMOUNT |
* +-----+---------------------+-------------+--------+
* | 102 | 2009-10-08 00:00:00 | 3 | 3000 |
* | 100 | 2009-10-08 00:00:00 | 3 | 1500 |
* | 101 | 2009-11-20 00:00:00 | 2 | 1560 |
* | 103 | 2008-05-20 00:00:00 | 4 | 2060 |
* +-----+---------------------+-------------+--------+

hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE

FROM CUSTOMERS c

LEFT OUTER JOIN ORDERS o

ON (c.ID = o.CUSTOMER\_ID);

hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER\_ID);

hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE

FROM CUSTOMERS c

FULL OUTER JOIN ORDERS o

ON (c.ID = o.CUSTOMER\_ID);

R- Programming:

* 1. Linear regression
* 2. Logistic Regression

1. gaussian regression
2. binomial regression
3. poison Regression
4. Quasi Regression

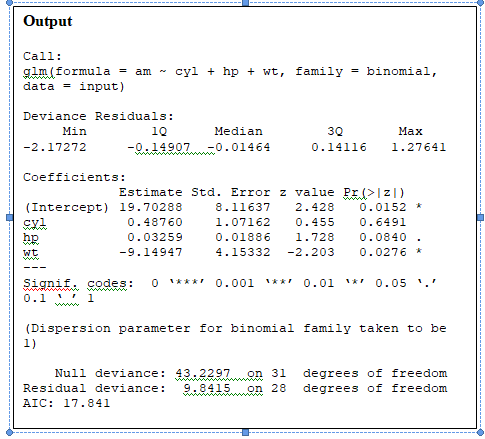
* 3. polynomial Regression

1.linear repression program

* The basic function for fitting ordinary multiple models is lm()
* Syntax: > fitted.model <- lm(formula, data = data.frame)
* Example1:multiple regression
* > fm2 <- lm(y ~ x1 + x2, data = production) #multiple regression
* Example:linear regression
* x=c(150,130,140,150,160) #height
* y= c(50,60,70,80,90,100) #weight
* relations = lm(y~x) #trained model
* relations
* summary(relations)
* testdata= data.frame(x=c(170,180,190,200)) #test data ---build a table
* predict(relations,testdata) #test the model
* Plot(x,y, main=“ Linear regression”,xlab=height,ylab=weight,abline(lm(y~x)))
* output

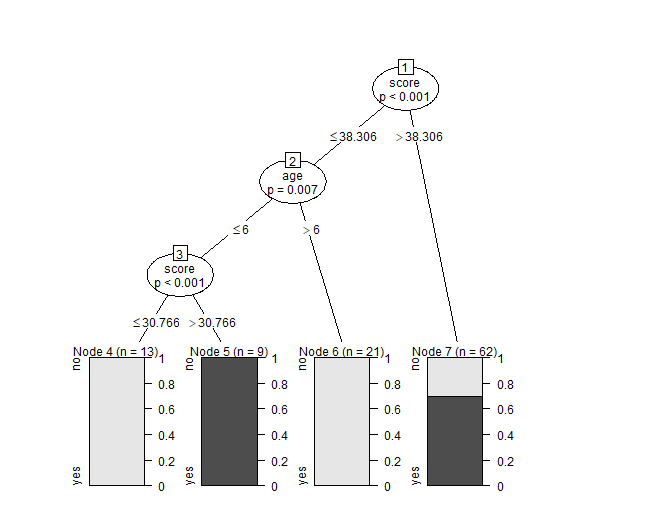
[1] 50, 79.5, 80.5,90.5

2.logistic regression program

* The R function to fit a generalized linear model is glm() which uses the form
* Syntax: fitted.model <- glm(formula, family=family.generator, data=data.frame)
* # where Family-> Gaussian, binomial(), quasi (),poission()
* Program
* We use the **glm()** function to create the regression model and get its summary for analysis.
* input <- mtcars[ , c("am","cyl","hp","wt")]
* inputdata = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)
* print(summary(inputdata))
* 

3. Decision Tree program

* install.packages("party") #install packages
* # Load the party package. It will automatically load other dependent packages.
* library(party)
* # Create the input data frame.
* input.dat <- readingSkills[c(1:105), ]
* # Create the tree.
* output.tree <- ctree(   nativeSpeaker ~ age + shoeSize + score,   data = input.dat)
* # Plot the tree.
* plot(output.tree)



4.K-means clustering program

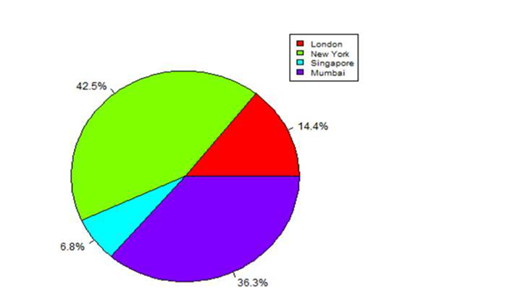
* **# Loading data**
* data(iris)
* **# Structure**
* str(iris)
* **# Installing Packages**
* install.packages("ClusterR")
* install.packages("cluster")
* **# Loading package**
* library(ClusterR)
* library(cluster)
* **# Removing initial label of  Species from original dataset**
* iris\_1 <- iris[, -5]
* # Fitting K-Means clustering Model to training dataset
* set.seed(240) # Setting seed
* kmeans.re <- kmeans(iris\_1, centers = 3, nstart = 20)
* kmeans.re
* **# Cluster identification for  each observation**
* kmeans.re$cluster
* **# Confusion Matrix**
* cm <- table(iris$Species, kmeans.re$cluster)
* cm
* **## Visualizing clusters**
* y\_kmeans <- kmeans.re$cluster
* clusplot(iris\_1[, c("Sepal.Length", "Sepal.Width")],   y\_kmeans,   lines = 0,  shade = TRUE,  color = TRUE,  labels =2,  plotchar = FALSE, span = TRUE,  main = paste("Cluster iris"),  xlab = 'Sepal.Length',  ylab = 'Sepal.Width')



5.graphical tools

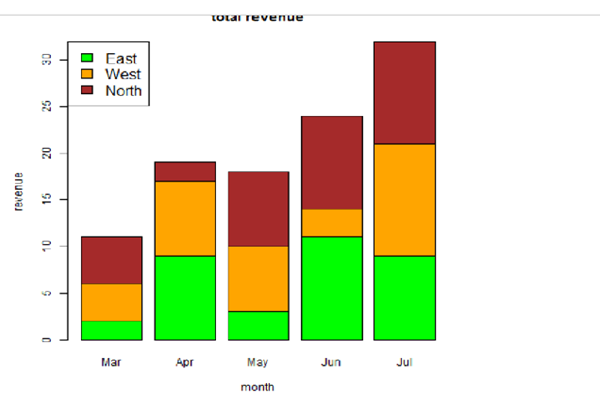
Pie-chart

* **A very simple pie-chart is created using just the input vector and labels.**
* **# Create data for the graph.**
* **x <- c(21, 62, 10,53)**
* **mylabels <- c("London","New York","Singapore","Mumbai")**
* **mycolors= c(“pink”,”yellow”,”orange”,”blue”)**
* **# Plot the chart.**
* **pie(x, labels=piepercent, main="City pie chart",mycolors)**
* **legend("topright", mylabels, fill=mycolors)**

****

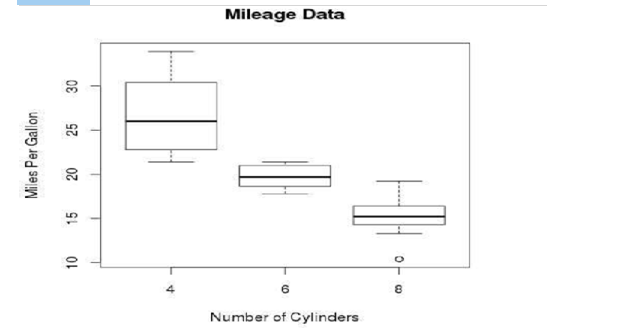
**6.Barcharts**

* **# Create the input vectors.**
* **colors <- c("green","orange","brown")**
* **months <- c("Mar","Apr","May","Jun","Jul")**
* **regions <- c("East","West","North")**
* **# Create the matrix of the values.**
* **Values <- matrix(c(2,9,3,11,9,4,8,7,3,12,5,2,8,10,11),nrow=3,ncol=5,byrow=TRUE)**
* **# Create the bar chart.**
* **barplot(Values,main="total**
* **revenue",names.arg=months,xlab="month",ylab="revenue",col=colors)**
* **# Add the legend to the chart.**
* **legend("topleft", regions, cex=1.3, fill=colors)**

****

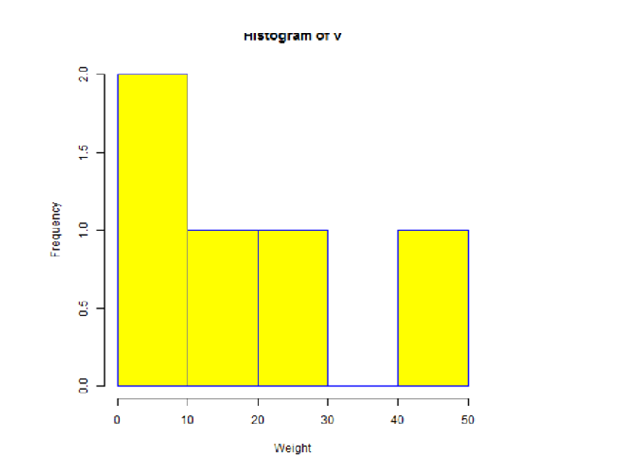
**7.Box plot**

* **We use the data set "mtcars" available in the R environment to create a basic boxplot.**
* **Let's look at the columns "mpg" and "cyl" in mtcars.**
* **input <- mtcars[,c('mpg','cyl')]**
* **print(head(input))**
* **# Plot the chart.**
* **boxplot(mpg ~ cyl, data=mtcars,xlab="Number of Cylinders“,ylab="Miles Per Gallon“,main="Mileage Data")**

****

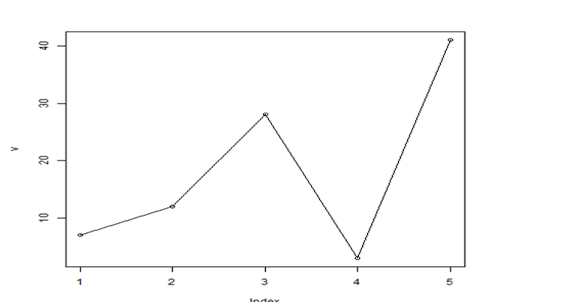
**8.histogram**

* **A simple histogram is created using input vector, label, col and border parameters.**
* **The script given below will create and save the histogram in the current R working**
* **directory.**
* **# Create data for the graph.**
* **v <- c(9,13,21,8,36,22,12,41,31,33,19)**
* **# Create the histogram.**
* **hist(v,xlab="Weight",col="yellow",border="blue")**

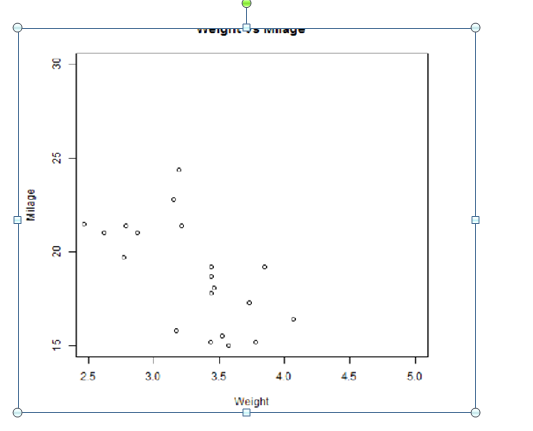
****

**9.R-line charts**

* **Single line Example**
* **A simple line chart is created using the input vector and the type parameter as "O". The**
* **below script will create and save a line chart in the current R working directory.**
* **Create the data for the chart.**
* **v <- c(7,12,28,3,41)**
* **# Plot the bar chart.**
* **plot(v,type="o")**

****

**10.scatter plot**

* **We use the data set "mtcars" available in the R environment to create a basic scatterplot.**
* **Let's use the columns "wt" and "mpg" in mtcars.**
* **input <- mtcars[,c('wt','mpg')]**
* **print(head(input))**
* **#The below script will create a scatterplot graph for the relation between wt(weight) and**
* **mpg(miles per gallon).**
* **# Get the input values.**
* **input <- mtcars[,c('wt','mpg')]**
* **# Plot the chart for cars with weight between 2.5 to 5 and mileage between 15**
* **and 30.**
* **plot(x=input$wt,y=input$mpg,**
* **xlab="Weight",**
* **ylab="Milage",**
* **xlim=c(2.5,5),**
* **ylim=c(15,30),**
* **main="Weight vs Milage")**
* ****