Unit 4

1) **Compare R and SAS for data analytics jobs.**

SAS	R
A market leader in commercial analytics space	An effective data handling and storage facility
Provides a graphical point-and-click user interface	A comprehensive and integrated collection of intermediate tools for data analysis
Data can be published in HTML, PDF, Excel, and other formats using the Output Delivery System Can retrieve data from various sources and perform statistical analysis	Graphical facilities for data analysis and display can be done either in soft or hard copy An effective programming language which includes conditionals, loops, user-defined recursive functions, and input and output facilities
Availability Expensive	Open Source
Improvement Tools Added with New Version	Improvement Tools Quickly Added
Debugging is Easy	Debugging is Difficult
Graphical Capabilities Limited	Graphical Capabilities Advanced
Ease of Learning SAS is very good when it comes to picking a new tool to learn without any prior programming language experience	R is bit tougher to learn as compared to SAS. Before learning R, you must have a basic knowledge of programming.
Managing Data(Data Handling Capabilitites) In terms of handling and managing data, SAS is in a better position since the data is increasing at a huge pace day by day and SAS is better at handling data	R works only on RAM, and increasing the RAM as and when the data increases is not a feasible option.
Graphics SAS is not great at graphical capabilities. Though Base SAS has some graphical capabilities improvisation, these capabilities are not widely known, and so R gets a clear lead in this aspect.	Graphics is a very important aspect of any Data Science or Data Analytics capabilities. Ability to visualize and analyze data is a crucial part. R is the winner in this area, thanks to the availability of various packages like ggplot, Latice, and RGIS.
Working with Big Data SAS is taking fast strides to execute analytics within Hadoop without the need to move cluster data. But still, SAS lags R when it comes to integrating successfully with Big Data tools like Hadoop and others. capability. If you are looking for deploying	While working with Big Data, R has some very good features which can be utilized by Big Data, Data Science, and Data Analytics communities. R has very good integration with Hadoop, and it also has a parallelization analytics at scale for Machine Learning
Availabily and Cost SAS is widely used in most private organizations as it is a commercial software. It is more expensive than any other data analytics tool available. It might thus be a bit difficult buying the software	R is an open source software and is completely free to use. Anyone can begin using it right away without having to spend a penny. So, regarding availability and cost, R is hands down

if you are an individual professional or a student starting out.	the better tool.
Join Scenario Currently, large corporations insist on using SAS, but SMEs The current job trend seems to show that while SAS is losing its momentum,	start-ups are increasingly opting for R, given that it’s free. R is gaining potential. The job scenario is on the cusp of change, and both the tools seem strong, but since R is on an uphill path, it can probably witness more jobs in the future, albeit not in huge corporates.
Deep Learning Support hile SAS has just begun work on adding deep learning support,	R has added support for a few packages which enable deep learning capabilities in the tool. You can use KerasR and keras package in R which are mere interfaces for the original Keras package built on Python. Although none of the tools are excellent facilitators of deep learning, R has seen some recent active developments on this front.
Customer Support As one would expect from full-fledged commercial software, SAS offers excellent customer service support as well as the backing of a helpful community.	Since R is free open-source software, expecting customer support will be hard to justify. However, it has a vast online community that can help you with almost everything.

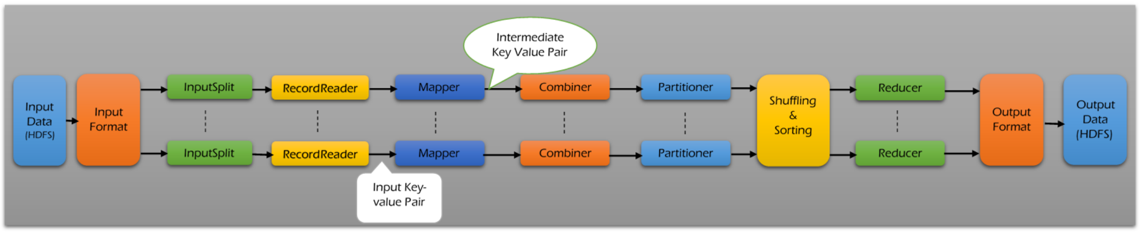
2) **Explain partitioners and combiners with reference to MapReduce.**

Partitioner: -

- ➔ Partitioner allows distributing how outputs from the map stage are send to the reducers.
- ➔ Partitioner controls the keys partition of the intermediate map-outputs.
- ➔ The key or a subset of the key is used to derive the partition by a hash function.
- ➔ The total number of partitions is almost same as the number of reduce tasks for the job.
- ➔ Partitioner runs on the same machine where the mapper had completed its execution by consuming the mapper output.
- ➔ Entire mapper output sent to partitioner.
- ➔ Partitioner forms number of reduce task groups from the mapper output.
- ➔ By default, Hadoop framework is hash based partitioner.
- ➔ The Hash partitioner partitions the key space by using the hash code.

Combiner: -

- ➔ Combiner acts as a mini reducer in MapReduce framework.
- ➔ This is an optional class provided in MapReduce driver class.
- ➔ Combiner process the output of map tasks and sends it to the Reducer.
- ➔ For every mapper, there will be one Combiner.
- ➔ Combiners are treated as local reducers.
- ➔ Hadoop does not provide any guarantee on combiner’s execution.
- ➔ Hadoop may not call combiner function if it is not required.
- ➔ Hadoop may call one or many times for a map output based on the requirement.



3) **Justify that "Hadoop is becoming the corner stone of big data". What aspects of Hadoop are especially relevant to the management and analysis of big data**

Hadoop is the rising star of the business technology agenda for a simple reason it disrupts the economics of data, analytics, and someday soon, all enterprise applications; it is secretly becoming an application platform too.

- ➔ **Hadooponomics makes enterprise adoption mandatory**:- Hadoop has been found not guilty of being a hyped-up open source platform. Hadoop has proven real value in any number of use cases including data lakes, traditional and advanced analytics, transactional data, and more. "Hadooponomics" — its ability to linearly scale both data storage and data processing and leverage pay-per-use public cloudonomics
- ➔ **SQL becomes Hadoop's killer app for 2015**:- SQL is the primary lingua franca for structured enterprise data and is used by application developers to read and write data to databases. It is often used by business intelligence professionals to explore data.SQL on Hadoop creates an instant, easy-to-use and justify use case for enterprises
- ➔ **Enterprise software vendors close Hadoop's data management and governance gaps:-** Hadoop is a general-purpose platform. That means it can store and process any kind or data. The bad news is that all parties agree that Hadoop - based data management and governance solutions have a ways to go before they provide the functionality sophisticated enterprises expect from their app platforms.
- ➔ **The Hadoop skills shortage disappears**:- Hadoop is not that hard to understand. It is a file system, albeit distributed, and it is a computing platform, albeit distributed. The APIs are Java. CEO's won't have to hire high-priced Hadoop consultants to get projects done.
- ➔ **Enterprises will let thousands of Hadoop clusters bloom in the cloud**:- Hadoop is both a data storage and data processing system. That means storage, compute, and network resources are required to run a Hadoop cluster. Early adopters of Hadoop will increasingly use Hadoop in the cloud to optimize the cost of Hadoop clusters and to meet demand for ad hoc analytics on Hadoop
- ➔ **Hadoop won't be just for analytics anymore**:- Hadoop is as much a computing platform that can become the foundational component of enterprise applications. With better esource management features provided by YARN, database options such as HBase, and in- memory overlay Apache Spark, Hadoop becomes an application platform.

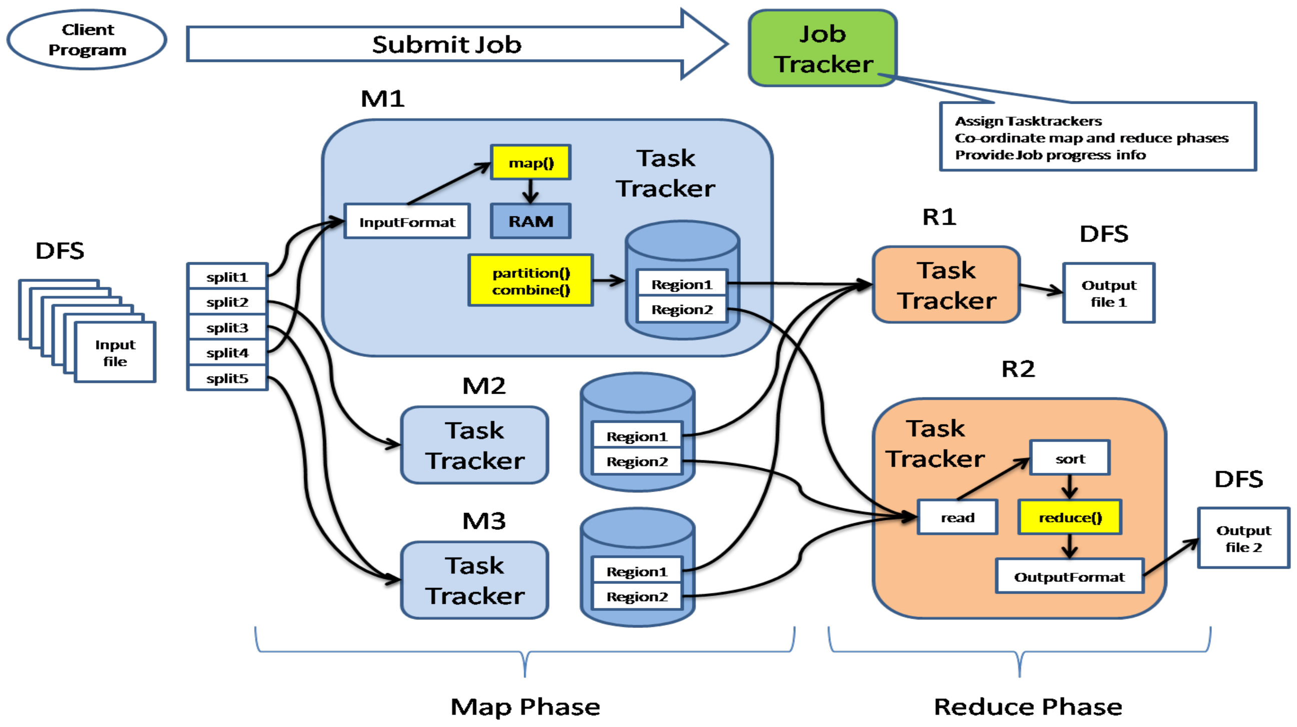
4) **i) Write a function in R that calculates the central tendency and spread of data objects. The function should give you a choice between parametric (mean and standard deviation) and nonparametric (median and median absolute deviation) statistics. The results should be returned as a named list. Additionally, the user should have the choice of automatically printing the results, or not. Unless otherwise specified, the function's default behavior should be to calculate parametric statistics and not print the results.**

```
list <- c(2,3,4,1,8,7,4,3,9)
para<- function(r)
{
 m = mean(r)
 print(m)
 s = sd(r)
 print(s)
}
non_para <- function(r)
{
 medianval <- median(r)
 actmedialval <- median(abs(r-medianval))
 print(actmedialval)
}
para(list)
non_para(list)
```

ii) Write a R function mydate() that uses the switch construct to print today's date. This function gives the user a choice regarding the format of today's date as either short or long. Example, mydate("long") prints] "Thursday December 02 2010" whereas mydate("short") prints "12-02-10". Any other type is not to be recognized. "long" is the default format for dates if type isn't specified.

```
mydate <- function(r="long"){
 switch(r,
   long={
    format(Sys.time(), "%A %B %d %Y")
   },
   short={
    format(Sys.time(), "%d-%m-%y")
   },
   {
    print('type is not recognized')
   }
 )
}
mydate("long")
mydate("short")
mydate()
mydate("f")
```

5) Explain the architecture of MapReduce.



➔ MapReduce is mainly used for parallel processing of large sets of data stored in Hadoop cluster. Initially, it is a hypothesis specially designed by Google to provide parallelism, data distribution and fault-tolerance. MR processes data in the form of key-value pairs. A key-value (KV) pair is a mapping element between two linked data items - key and its value.

➔ Map reduce architecture consists of mainly two processing stages. First one is the map stage and the second one is reduce stage. The actual MR process happens in task tracker. In between map and reduce stages, Intermediate process will take place. Intermediate process will do operations like shuffle and sorting of the mapper output data. The Intermediate data is going to get stored in local file system.

Mapper Phase

In Mapper Phase the input data is going to split into 2 components, Key and Value. The key is writable and comparable in the processing stage. Value is writable only during the processing stage. Suppose, client submits input data to Hadoop system, the Job tracker assigns tasks to task tracker. The input data that is going to get split into several input splits.

Input splits are the logical splits in nature. Record reader converts these input splits in Key-Value (KV) pair. This is the actual input data format for the mapped input for further processing of data inside Task tracker. The input format type varies from one type of application to another. So the programmer has to observe input data and to code according.

Suppose we take Text input format, the key is going to be byte offset and value will be the entire line. Partition and combiner logics come in to map coding logic only to perform special data operations. Data localization occurs only in mapper nodes.

Combiner is also called as mini reducer. The reducer code is placed in the mapper as a combiner. When mapper output is a huge amount of data, it will require high network bandwidth. To solve this bandwidth issue, we will place the reduced code in mapper as combiner for better performance. Default partition used in this process is Hash partition.

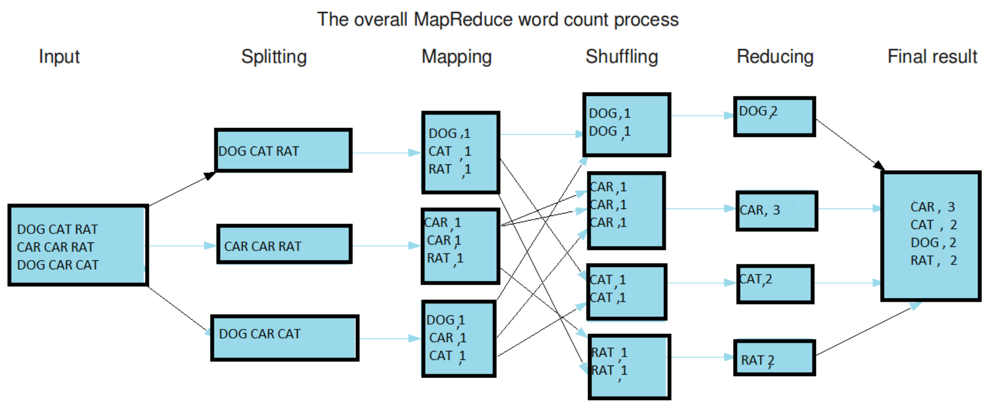
A partition module in Hadoop plays a very important role to partition the data received from either different mappers or combiners. Petitioner reduces the pressure that builds on reducer and gives more performance. There is a customized partition which can be performed on any relevant data on different basis or conditions.

Intermediate Process

The mapper output data undergoes shuffle and sorting in intermediate process. The intermediate data is going to get stored in local file system without having replications in Hadoop nodes. This intermediate data is the data that is generated after some computations based on certain logics. Hadoop uses a Round-Robin algorithm to write the intermediate data to local disk. There are many other sorting factors to reach the conditions to write the data to local disks.

Reducer Phase

Shuffled and sorted data is going to pass as input to the reducer. In this phase, all incoming data is going to combine and same actual key value pairs is going to write into hdfs system. Record writer writes data from reducer to hdfs. The reducer is not so mandatory for searching and mapping purpose. Reducer logic is mainly used to start the operations on mapper data which is sorted and finally it gives the reducer outputs like part-r-0001etc,. Options are provided to set the number of reducers for each job that the user wanted to run. In the configuration file mapred-site.xml, we have to set some properties which will enable to set the number of reducers for the particular task. Speculative Execution plays an important role during job processing. If two or more mappers are working on the same data and if one mapper is running slow then Job tracker assigns tasks to the next mapper to run the program fast. The execution will be on FIFO (First In First Out).



6)

A group of students have taken exams in Math, Science, and English. The data are presented in table below. Write the sequence of R statements to perform the following and explain the rationale:

(i) combine these scores in order to determine a single performance indicator for each student.

(ii) assign an A to the top 20 percent of students, a B to the next 20 percent, and so on.

(iii) sort the students alphabetically.

Student exam data

Student	**Maths**	**Science**	**English**
John Davis	502	95	25
Angela Williams	600	99	22
Bullwinkle Moose	412	80	18
David Jones	358	82	15
Janice Markhammer	495	75	20
Cheryl Cushing	512	85	28
Reuven Ytzrhak	410	80	15
Greg Knox	625	95	30
Joel England	573	89	27
Mary Rayburn	522	86	18

```
Student <- c("John Davis","Angela Williams","Bullwinkle Moose","David Jones","Janice Markhammer","Cheryl Cushing","Reuven Ytzrhak","Greg Knox","Joel England","Mary Rayburn")
Maths <- c(502,600,412,358,495,512,410,625,573,522)
Science <- c(95,99,80,82,75,85,80,95,89,86)
English <- c(25,22,18,15,20,28,15,30,27,18)
Data <- data.frame(Student, Maths, Science, English)
Total <- Data$Maths + Data$Science + Data$English
Data <- data.frame(Data, Total)
NewData <-Data[order(Data$Total, decreasing = TRUE),]
n1 <- nrow(NewData)
da <- n1/5
ggad <- c()
for (i in 1:n1) {
 if(i<=da){
  ggad[i] <- c("A")
 }
 da1 <- da * 2
 if(i+da<=da1){
  ggad[i+da] <- c("B")
 }
 da2 <- da * 3
 if(i+da1<=da2){
  ggad[i+da1] <- c("C")
 }
 da3 <- da * 4
 if(i+da2<=da3){
  ggad[i+da2] <- c("D")
 }
```

```
  da4 <- da * 5
  if(i+da3<=da4){
    ggad[i+da3] <- c("E")
  }
}
ggad <- as.factor(ggad)
DataGrade <- data.frame(NewData, ggad)
Data[order(DataGrade$Student),]
```

7. Explain the methods provided by R for aggregation and reshaping of data. Give examples.

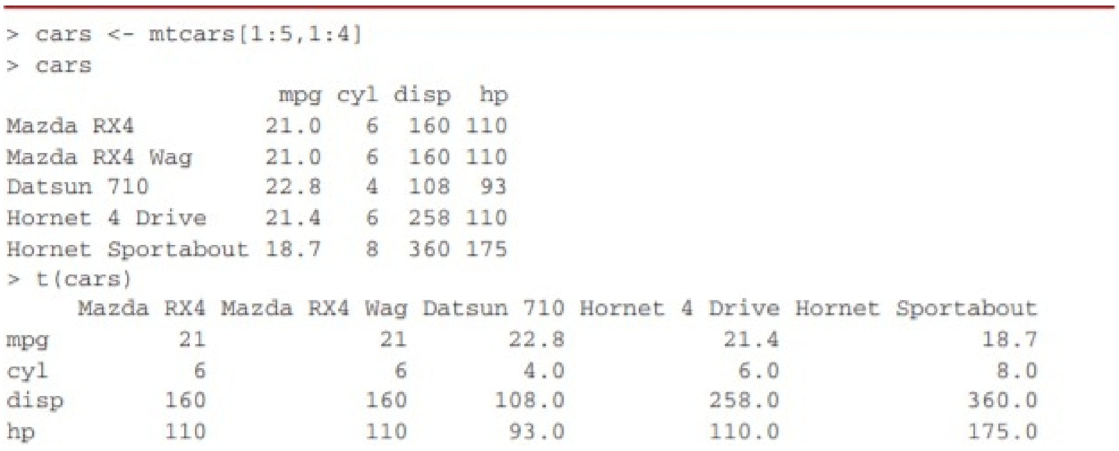
Aggregation and restructuring

R provides a number of powerful methods for aggregating and reshaping data. When we aggregate data, we replace groups of observations with summary statistics based on those observations.

When reshaping data, we alter the structure (rows and columns) determining how the data is organized. We'll use the mtcars data frame. This dataset, describes the design and performance characteristics (number of cylinders, displacement, horsepower, mpg, and so on) of automobiles.

Transpose

The transpose (reversing rows and columns) is perhaps the simplest method of reshaping a dataset. Use the t() function to transpose a matrix or a data frame. In the latter case, row names become variable (column) names. An example is below.

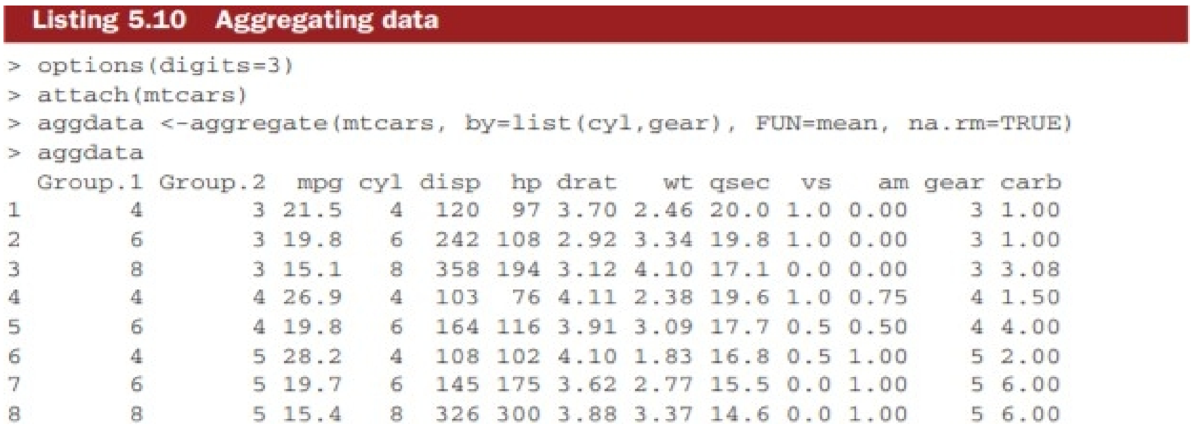


```
> cars <- mtcars[1:5,1:4]
> cars
                   mpg cyl disp  hp
Mazda RX4         21.0   6  160 110
Mazda RX4 Wag     21.0   6  160 110
Datsun 710        22.8   4  108  93
Hornet 4 Drive    21.4   6  258 110
Hornet Sportabout 18.7   8  360 175
> t(cars)
     Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive Hornet Sportabout
mpg         21            21       22.8           21.4              18.7
cyl          6             6        4.0            6.0               8.0
disp       160           160      108.0          258.0             360.0
hp         110           110       93.0          110.0             175.0
```

Listing above uses a subset of the mtcars dataset in order to conserve space on the page.

Aggregating data

It's relatively easy to collapse data in R using one or more by variables and a defined function. The format is aggregate(x, by, FUN) where x is the data object to be collapsed, by is a list of variables that will be crossed to form the new observations, and FUN is the scalar function used to calculate summary statistics that will make up the new observation values.As an example, we'll aggregate the mtcars data by number of cylinders and gears, returning means on each of the numeric variables.


Listing 5.10 Aggregating data

```
> options(digits=3)
> attach(mtcars)
> aggdata <-aggregate(mtcars, by=list(cyl,gear), FUN=mean, na.rm=TRUE)
> aggdata
  Group.1 Group.2  mpg cyl disp  hp drat   wt qsec  vs   am gear carb
1       4       3 21.5   4  120  97 3.70 2.46 20.0 1.0 0.00    3 1.00
2       6       3 19.8   6  242 108 2.92 3.34 19.8 1.0 0.00    3 1.00
3       8       3 15.1   8  358 194 3.12 4.10 17.1 0.0 0.00    3 3.08
4       4       4 26.9   4  103  76 4.11 2.38 19.6 1.0 0.75    4 1.50
5       6       4 19.8   6  164 116 3.91 3.09 17.7 0.5 0.50    4 4.00
6       4       5 28.2   4  108 102 4.10 1.83 16.8 0.5 1.00    5 2.00
7       6       5 19.7   6  145 175 3.62 2.77 15.5 0.0 1.00    5 6.00
8       8       5 15.4   8  326 300 3.88 3.37 14.6 0.0 1.00    5 6.00
```

In these results, Group.1 represents the number of cylinders (4, 6, or 8) and Group.2 represents the number of gears (3, 4, or 5). For example, cars with 4 cylinders and 3 gears have a mean of 21.5 miles per gallon (mpg).

When we're using the aggregate() function , the by variables must be in a list (even if there's only one). You can declare a custom name for the groups from within the list, for instance, using by=list(Group.cyl=cyl, Group.gears=gear) . The function specified can be any built-in or user-provided function. This gives the aggregate command a great deal of power. But when it comes to power, nothing beats the reshapepackage.

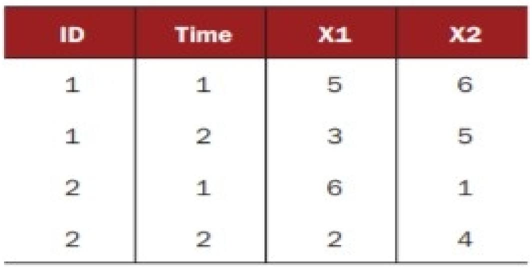
The Reshape package

The reshape package can be used for both restructuring and aggregating datasets.Reshape can be installed using**install.packages("reshape").**

Basically, we'll "melt" data so that each row is a unique ID-variable combination. Then we'll "cast" the melted data into any shape of desire. During the cast, one can aggregate the data with any function of our wish.

The dataset we'll be working with is shown in table 5. In this dataset, the measurements are the values in the last two columns (5, 6, 3, 5, 6, 1, 2, and 4). Each measurement is uniquely identified by a combination of ID variables (in this case ID, Time, and whether the measurement is on X1 or X2). For example, the measured value 5 in the first row is uniquely identified by knowing that it's from observation (ID) 1, at Time 1, and on variableX1.

Table 5.8 The original dataset (`mydata`)



ID	Time	X1	X2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

MELTING

Melting a dataset means to restructure it into a format where each measured variable is in its own row, along with the ID variables needed to uniquely identify it.

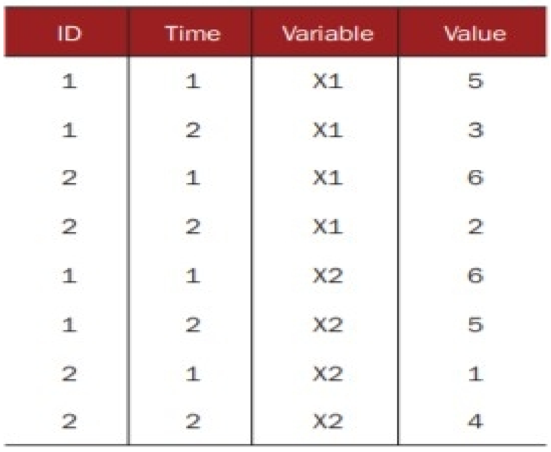
If we melt the data from table 5.8, using the following

code library(reshape)

md <- melt(mydata, id=(c("id", "time")))

we end up with the structure shown in table 5.9.

Table 5.9 The melted dataset



ID	Time	Variable	Value
1	1	X1	5
1	2	X1	3
2	1	X1	6
2	2	X1	2
1	1	X2	6
1	2	X2	5
2	1	X2	1
2	2	X2	4

We must specify the variables needed to uniquely identify each measurement (ID and Time) and that the variable indicating the measurement variable names (X1 or X2) is created automatically.

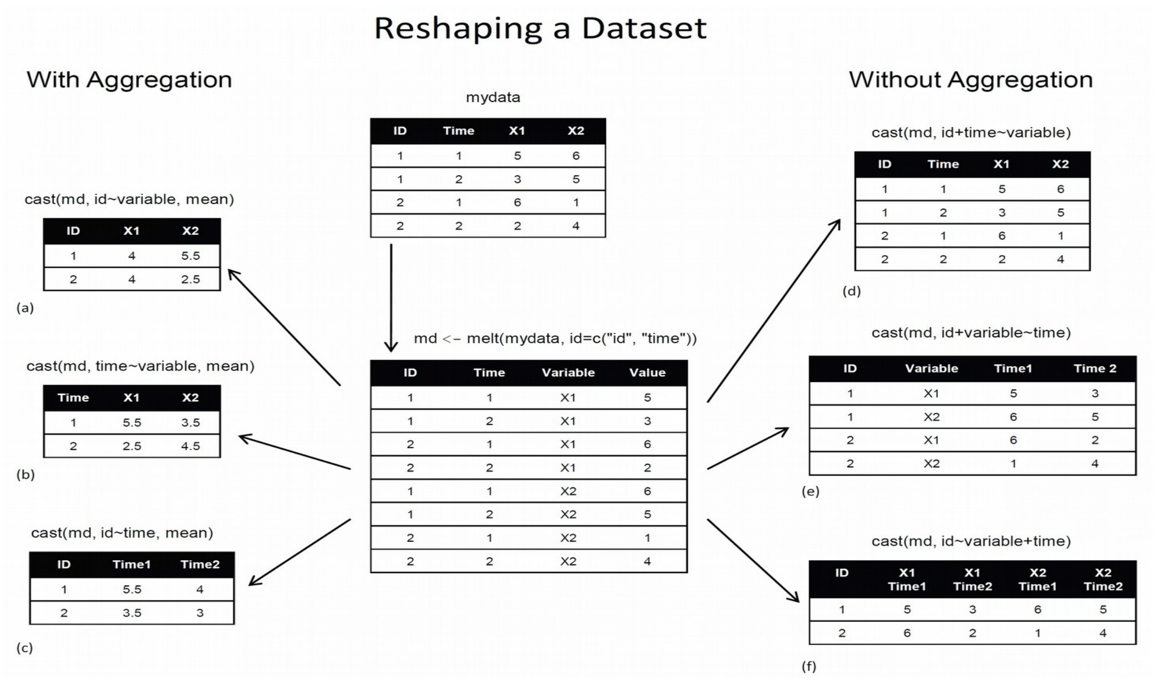
Now that we haves data in a melted form, we can recast it into any shape, using the cast() function.

CASTING

The cast() function starts with melted data and reshapes it using a formula that we provide and an (optional) function used to aggregate the data. The format is

newdata <- cast(md, formula, FUN)

where md is the melted data, formula describes the desired end result, and FUN is the (optional) aggregating function.


Reshaping a Dataset
With Aggregation
Without Aggregation
mydata
md <- melt(mydata, id=c("id", "time"))
cast(md, id~variable, mean)
(a)
cast(md, time~variable, mean)
(b)
cast(md, id~time, mean)
(c)
cast(md, id+time~variable)
(d)
cast(md, id+variable~time)
(e)
cast(md, id~variable+time)
(f)

mydata

ID	Time	X1	X2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

md <- melt(mydata, id=c("id", "time"))

ID	Time	Variable	Value
1	1	X1	5
1	2	X1	3
2	1	X1	6
2	2	X1	2
1	1	X2	6
1	2	X2	5
2	1	X2	1
2	2	X2	4

cast(md, id~variable, mean)

ID	X1	X2
1	4	5.5
2	4	2.5

(a)

cast(md, time~variable, mean)

Time	X1	X2
1	5.5	3.5
2	2.5	4.5

(b)

cast(md, id~time, mean)

ID	Time1	Time2
1	5.5	4
2	3.5	3

(c)

cast(md, id+time~variable)

ID	Time	X1	X2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

(d)

cast(md, id+variable~time)

ID	Variable	Time1	Time 2
1	X1	5	3
1	X2	6	5
2	X1	6	2
2	X2	1	4

(e)

cast(md, id~variable+time)

ID	X1 Time1	X1 Time2	X2 Time1	X2 Time2
1	5	3	6	5
2	6	2	1	4

(f)

7) What is MADlib? How is it helpful in performing in-database analytics?

MADlib is a free, open-source library of in-database analytic methods.It provides an evolving suite of SQL-based algorithms for machine learning, data mining and statistics that run at scale within a database engine, with no need for data import/export to other tools.

8)Explain HDFS Diagram architecture withdiagram

The Hadoop Distributed File System (HDFS) is designed to provide a fault-tolerant file system designed to run on commodity hardware. The primary objective of HDFS is to store data reliably even in the presence of failures including Name Node failures, Data Node failures and network partitions.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallelprocessing.

Features of HDFS

- Itis suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- Thebuilt-inserversofnamenodeanddatanodehelpuserstoeasilycheckthestatusofcluster.
- Streaming access to file system data.

HDFS provides file permissions and authentication.

HDFS uses a master/slave architecture in which one device (the master) controls one or more other devices (the slaves). The HDFS cluster consists of a single Name Node and a master server manages the file system namespace and regulates access to files.

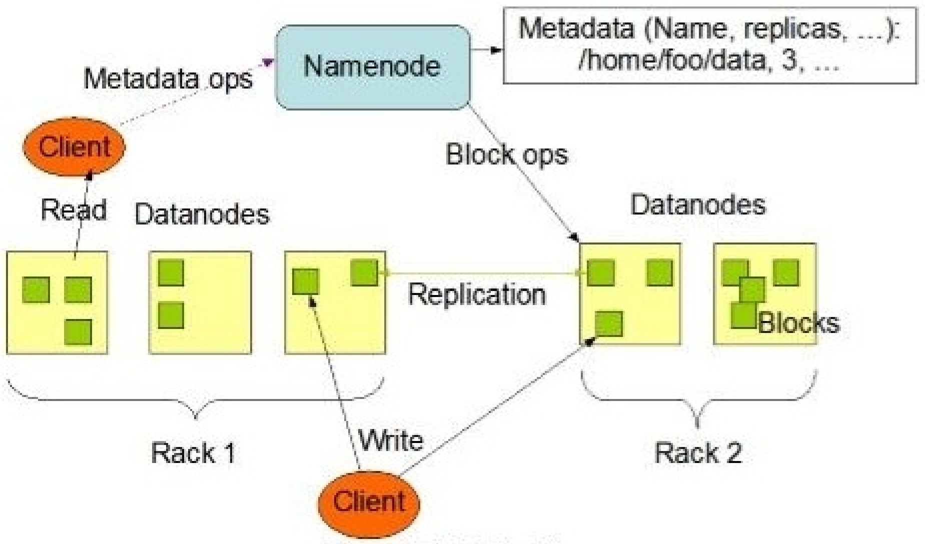


Figure: HDFS Architecture

The main components of HDFS are as described below:

Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware.

The system having the namenode acts as the master server and it does the following tasks

➔ Manages the file syste namespace

➔ regulates client access to files

➔ it also executes file system operations such as renaming closing opening files and directories

Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

➔ They also perform operations such as block creation, deletion, and replication accordingto the instructions of the namenode.

➔ Data Nodes Perfors read-write Operations on the file syste as per client requests

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks.

The File System Namespace:

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode.

Data Replication:

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later.

Unit 3

6) **Explain the nearest neighbor algorithm for classification. Explain the characteristics of the algorithm and importance of choosing the right value for k.**
   Reference P Unit (3)- 6

7) **Explain how PageRank algorithm is implemented in MapReduce with relevant pseudo code. How are random jump factor and dangling nodes handled?**
   Reference P Unit (3)- 8

8) **Explain the k-means algorithm for clustering. Discuss its limitations.**

➔ Partitional clustering approach

➔ Each cluster is associated with a centroid (center point)

➔ Each point is assigned to the cluster with the closest centroid.

➔ Number of clusters, K, must be specified.

➔ The basic algorithm is very simple

➔ Initial centroids are often chosen randomly. – Clusters produced vary from one run to another.

➔ The centroid is (typically) the mean of the points in the cluster.

➔ 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.

➔ K-means will converge for common similarity measures mentioned above.

➔ Most of the convergence happens in the first few iterations. – Often the stopping condition is changed to 'Until relatively few points change clusters'

➔ Complexity is O(n * K * I * d) – n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

```
1: Select $K$ points as the initial centroids.
2: **repeat**
3:     Form $K$ clusters by assigning all points to the closest centroid.
4:     Recompute the centroid of each cluster.
5: **until** The centroids don't change
```

Limitations of K-means

• K-means has problems when clusters are of differing – Sizes – Densities – Non-globular shapes

• K-means has problems when the data contains outliers.

• One solution is to use many clusters. – Find parts of clusters, but need to put together.

Reference P Unit (3)- 4

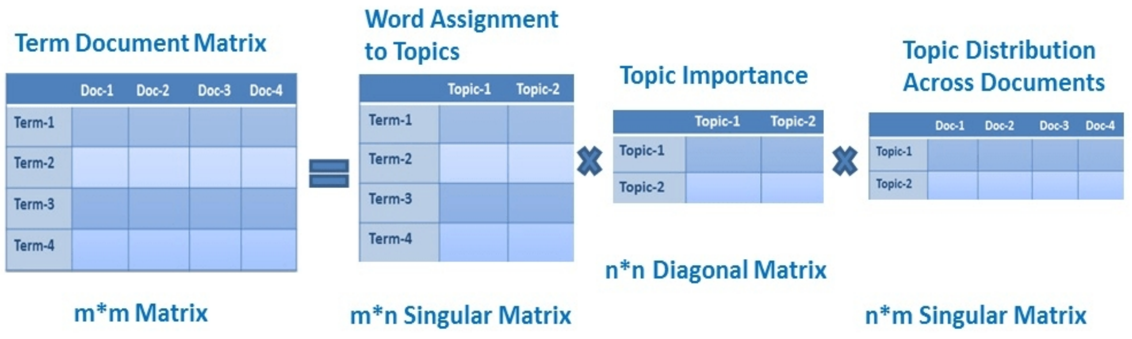
9) **Discuss latent semantic analysis with reference to text analytics.**

Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text.

LSA is an information retrieval technique which analyzes and identifies the pattern in unstructured collection of text and the relationship between them.

LSA itself is an unsupervised way of uncovering synonyms in a collection of documents.

LSA (Latent Semantic Analysis) also known as LSI (Latent Semantic Index) LSA uses bag of word(BoW) model, which results in a term-document matrix(occurrence of terms in a document). Rows represent terms and columns represent documents. LSA learns latent topics by performing a matrix decomposition on the document-term matrix using Singular value decomposition. LSA is typically used as a dimension reduction or noise reducing technique.



Reference P Unit (3)- 11

10) **Identify the challenges of visual analytics.**

Reference P Unit (3)- 10

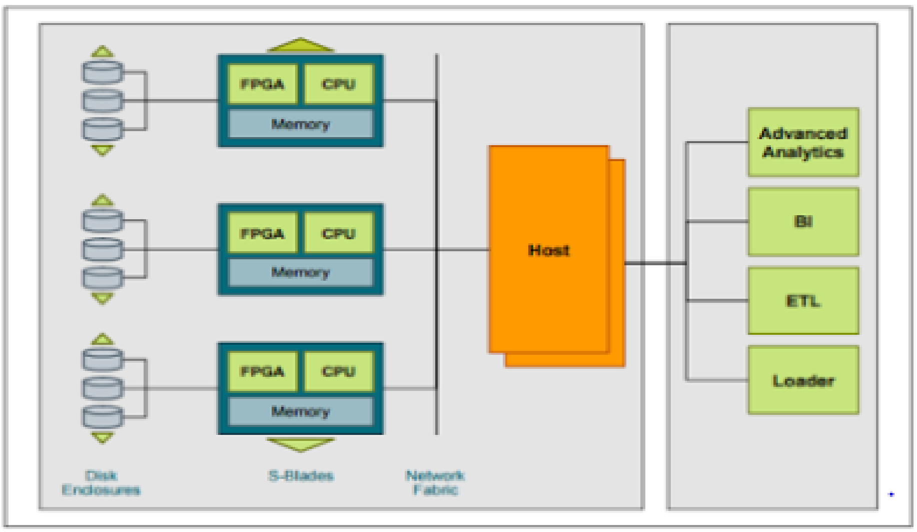
11) Desribe POS tagging, lemmatization and stemming with examples.

Reference P Unit (3)- 13

UNIT 5

12) Explain the software architecture of IBM Netezza and how do they work together for fast query execution.

Netezza is a data warehouse and big data analytics appliance. It uses Asymmetric Massively Parallel Processing (AMPP) architecture, which combines an SMP front end with a shared MPP back end for query processing. Netezza is a result of database integration , processing engine and storage in a system. Netezza architecture resembles Hadoop cluster design in may ways. e.g. Distribution, active-passive node, data storing methods, replications etc.



Netezza has four major components:

Netezza Architecture – Hosts

The Netezza hosts are high-performance Linux servers that are set up in an active-passive mode for high availability. In case of active server failure, the passive host will take over the processing tasks. It just requires very small time to passive node to take over. The active host is an interface to external tools and client applications such BI, ETL, JDBC, ODBC tools. Client submits SQL requests via ODBC/JDBC. Number of tools such as Aginity, Squirrel, nzsql utility are used to submit SQL query to Netezza host. The Netezza compiles them into executable code segments called snippets (usually C/C++ codes) , and creates optimised query plans by distributing the snippets across to all the nodes for execution. FPGA fetches the required data and snippet execution takes place.

Field Programmable Ggate Arrays – FPGA

The FPGA is a Netezza proprietary hardware tool developed to filters out unwanted data as early as possible when SQL query is submitted to hosts. The data will be eliminated as early as when reading from disks. This process of data elimination removes IO bottlenecks and frees up downstream components such as the CPU, memory and network from processing extra data hence notably improves performance. The FPGA always rely on the zone maps to eliminate the unwanted data. Zone maps are created to every column in the tables during certain Netezza operations.

Snippet Blades (S-Blades)

S-Blades are intelligent processing nodes that make up the MPP engine of the Netezza data warehouse appliance. Each S-Blade is an independent server that contains powerful multi-core CPUs, multi-engine FPGAs and gigabytes of RAM, all working in parallel to deliver high performance. FPGA in each s-blade is important Netezza architecture hardware that improves the performance.

Disk Enclosure

Finally, other important Netezza architecture hardware is high performance Disks.The disk enclosures contain high density and high performance disks that are RAID protected. Each disk contains a slice of the data in a database tables. Either hash or random algorithm will be used by host to distributes the data across all the disks evenly. A mirror copy of each slice of data is maintained on a different disk drive if the mirroring is enabled. The disk enclosures are connected to the S-Blades via high-speed interconnects that allow all the disks simultaneously stream data to the S-Blades at the maximum rate possible. The data distribution and the storage is based on the distribution key which we use while creating table.

Reference VB Unit (5)- 1

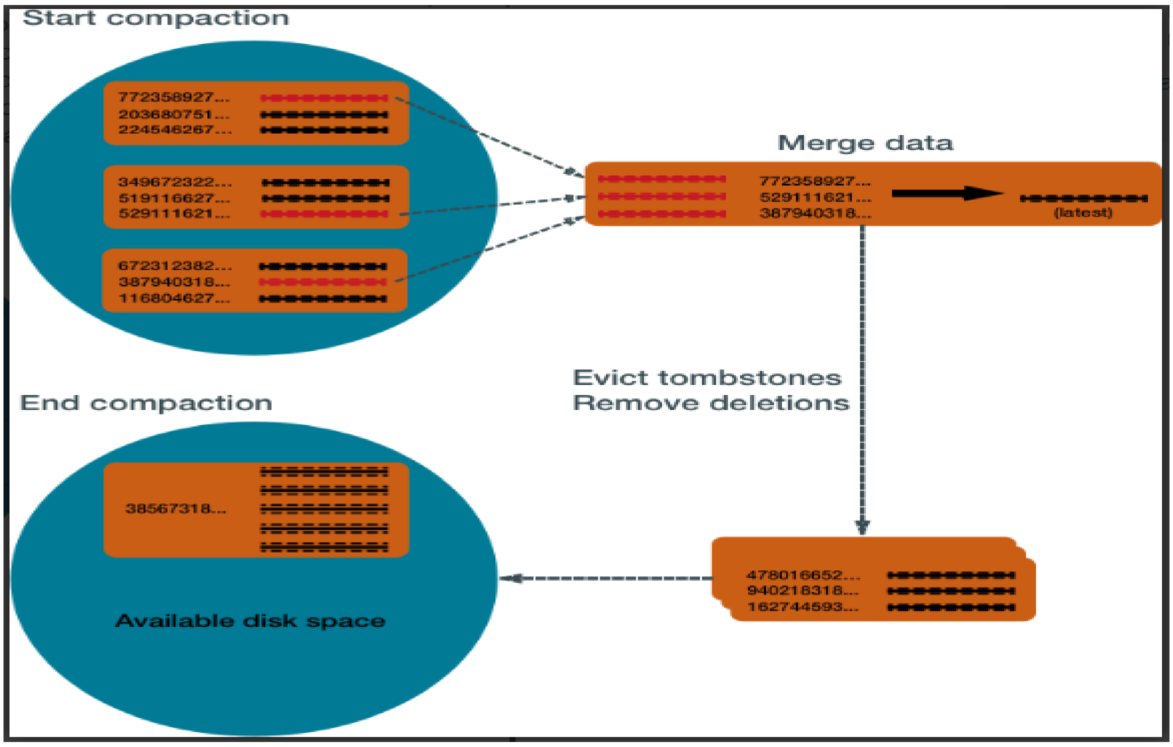
13) **Explain how data is maintained in Cassandra database.**

SSTable stands for Sorted Strings Table a concept borrowed from Google BigTable which stores a set of immutable row fragments in sorted order based on row keys.

The Cassandra write process stores data in files called SSTables. SSTables are immutable. Instead of overwriting existing rows with inserts or updates, Cassandra writes new time stamped versions of the inserted or updated data in new SSTables. Cassandra does not perform deletes by removing the deleted data: instead, Cassandra marks it with tombstones. Over time, Cassandra may write many versions of a row in different SSTables. Each version may have a unique set of columns stored with a different timestamp. As SSTables accumulate, the distribution of data can require accessing more and more SSTables to retrieve a complete row.To keep the database healthy, Cassandra periodically merges SSTables and discards old data. This process is called compaction.

Compaction

Compaction works on a collection of SSTables. From these SSTables, compaction collects all versions of each unique row and assembles one complete row, using the most up-to-date version (by timestamp) of each of the row's columns. The merge process is performant, because rows are sorted by partition key within each SSTable, and the merge process does not use random I/O. The new versions of each row is written to a new SSTable. The old versions, along with any rows that are ready for deletion, are left in the old SSTables, and are deleted as soon as pending reads are completed. Compaction causes a temporary spike in disk space usage and disk I/O while old and new SSTables co-exist. As it completes, compaction frees up disk space occupied by old SSTables. It improves read performance by incrementally replacing old SSTables with compacted SSTables. Cassandra can read data directly from the new SSTable even before it finishes writing, instead of waiting for the entire compaction process to finish.As Cassandra processes writes and reads, it replaces the old SSTables with new SSTables in the page cache. The process of caching the new SSTable, while directing reads away from the old one, is incremental — it does not cause a the dramatic cache miss. Cassandra provides predictable high performance even under heavy load.



Reference VB Unit (5)- 4

14) Explain the features, capabilities and deployment options of Rainstor.

RainStor is an online, SQL-based, enterprise-class data archive solution, which radically reduces the footprint and costs of archiving, provides immutable data storage, and helps meet regulatory, and compliance needs. It can store massive amounts of structured and semi-structured data from data warehouses, machines, other relational databases, or major data sources. RainStor provides a cost effective means of storing data in a minimal hardware footprint. By adding the Teradata RainStor archiving solution to their Teradata Unified Data Architecture, organizations can tap a rich source of archived data to support users' advanced analytics, The Rainstor Archive solution supports advanced analytics with data previously stored on tapes, in ungoverned data lakes, or deleted because of a lack of storage. Teradata RainStor's features and capabilities enable customers to better store and extract value from their archived data:

Footprint - The archive compression technology radically reduces the storage footprint by as much as 10 to 40 times, depending on the type of data.

Secured Data - RainStor provides an immutable data storage model. Once saved, data cannot be changed, which provides a tamper-proof audit trail. New data or changed data is appended to a file, not replaced. In addition, RainStor offers security for access authorization, data lineage tracking, and encryption

Runs on Hadoop - RainStor provides an enterprise online data archiving solution on top of Hadoop. It runs natively on the Hadoop distributed file system (HDFS) on the Teradata Appliance for Hadoop or commodity Hadoop clusters. In addition to Hadoop, RainStor is also available on non-Hadoop platforms.

Multiple Data Sources - In addition to the Teradata Database, Rainstor can archive data from Oracle, SQL Server, Sybase IQ, and IBM Netezza. The ability to consolidate data from across the enterprise into a single archive saves money and improves analytics.

Teradata RainStor System Archive - The RainStor System Archive retains "glacial data" or data that would normally be aged-out of a relational database to historical archive repositories, such as tape or disk. It is an enterprise archive solution with data governance, lineage, and auditing capabilities. It stores the data in the most efficient, scalable and secure manner, while maintaining full access to the data from the source system.

Teradata RainStor Online Archive - The RainStor Online Archive solution is targeted for analytics on a deep historical data repository. It builds on RainStor System Archive features, but it provides greater accessibility with standard SQL through the RainStor query interface, and Hive or Pig if RainStor is deployed on Hadoop.
RainStor Regulatory Archive - The RainStor Regulatory Archive application provides the features of the RainStor Online Archive solution with added compliance capabilities. This solution is a fit for highly regulated data, such as the Payment Card Industry Data Security Standard (PCI DSS), Health Insurance Portability and Accountability Act of 1996 (HIPAA) or the U.S. Securities and Exchange Commission (SEC).

Ingest: Data is rapidly ingested in a number of different formats from sources including source databases, network logs or flat files (CSV, BCP format). RainStor easily scales to ingest extreme volumes up to 30,000 records per second per core.
Reduce: By only storing the unique field and pattern values contained within each imported record, RainStor de-duplicates the data, resulting in extreme compression – up to 40x compared to raw data.
Comply: RainStor's auto-expiry of data from the repository is based on configurable retention policies that also support legal hold.
Analyse: The data in RainStor, while highly compressed, remains accessible using standard SQL, popular BI tools and MapReduce on Hadoop. The data requires no re-inflation to run the analysis and results are returned consistently at high performance. **Manage**: RainStor does not require any specialist DBA skills to install and maintain and with no special tuning or indexes, requires low to zero maintenance over time. **Scale**: As data volumes grow, the underlying hardware and storage platforms can be easily extended to meet additional demand.

The RainStor database actually runs natively on Hadoop's Distributed File System (HDFS) and so all features and benefits that you benefit from running RainStor in a NAS environment are the same as when you run it on Hadoop. Your data footprint is reduced because of the granular level compression and therefore you require less nodes. Additionally, queries run 10-100x faster because of built-in filtering and the ability to dynamically eliminate files that do not contain the result-set. And with RainStor, you have the benefit of choosing SQL or MapReduce depending on the query and therefore productivity levels improve because SQL is a standard for all IT departments. RainStor also provides built-in security and availability, which are expected requirements for large enterprises that deploy mission critical environments on Hadoop. RainStor can be deployed alongside the chosen Hadoop distribution (supporting all of them – Apache, Cloudera, Hortonworks, IBM and MapR) and in fact can be deployed after a customer deploys a Hadoop environment.

Reference VB Unit (5)- 6

15) **Briefly explain the main components of Cassandra.**
key components of Cassandra are as follows

- **Node** − It is the place where data is stored.
- **Data center** − It is a collection of related nodes.
- **Cluster** − A cluster is a component that contains one or more data centers.
- **Commit log** − The commit log is a crash-recovery mechanism in Cassandra. Every write operation is written to the commit log.
- **Mem-table** − A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.

- **SSTable** − It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.
- **Bloom filter** − These are nothing but quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.

Reference VB Unit (5)- 3

16) **Discuss the different type of views provided by Jigsaw.**
Reference VB Unit (5)- 7

17) Describe how IBM BigSheets can help with the business and technical challenges of big data.
BigSheets is a browser-based analytic tool included in the InfoSphere® BigInsights™ Console that you use to break large amounts of unstructured data into consumable, situation-specific business contexts.When you develop applications by using InfoSphere BigInsights Console, you can view the output data in its raw format in the console. From the same console, you can use BigSheets to collect data from multiple sources, including crawling the Internet, local files or files on your network. After you collect data, you load data into a master workbook. Then, you format and explore the data by building sheets, which resemble spreadsheets, in workbooks that are based on the master workbook. You can combine columns from different
workbooks, run formulas, and filter data. These manipulations form the basis of your analysis. You can also combine data with functions that are designed for InfoSphere BigInsights text analytics to drill further into information and derive content out of raw data. These deep insights help you to filter and manipulate data from sheets even further. When you finish building sheets and refining data, you apply the analysis to the workbook. After you run the analysis, you can apply visualizations such as tag clouds, bar charts, maps, and pie charts. These visualizations provide a consumable output for your data that highlights relationships and distill insights from previously disconnected data. In BigSheets, you work with collections of data in master workbooks, workbooks, and sheets. Master workbooks are the initial collection of data that is created from an output results file. The output results file can be created by uploading a file or by
using applications to gather the data. Data in master workbooks is read-only so that you can explore the data set by building on top of the original data. You can visualize the data in a master workbook through a map or a chart. But if you want to further explore the data in the master workbook, you create new workbooks from the master workbook. Workbooks contain data from one or more master or child workbooks, but you can tailor the format, content, and structure of the data. Create workbooks to save a particular set of data results, refine the data, and explore the data. You can create workbooks from either a master workbook or from other workbooks. If the workbook is created from a master workbook, the workbook is said to be a child of the master workbook. If the workbook was created from another workbook, the new workbook is a child of the existing workbook (the parent workbook). A related workbook relationship is created between those workbooks and all of their descendants. Workbooks can have one or more sheets. Sheets within workbooks are representations of data that apply a different function to analyze and view subsets of the data. Each row in a sheet represents a record of data and each column represents a property of the record. Add sheets to workbooks to progressively edit and
explore the data. In addition to showing the relationships between workbooks, the Workbook diagram also displays all the sheets and charts that are associated with the related

workbooks. By default, the last sheet in your workbook is named the Result sheet. When you save and run the
workbook, the data in the Result sheet is the output or the results for that workbook. You can then create maps and charts to visualize the results data from the Results sheet. You can see all of the workbooks related to the current workbook by viewing the Workflow diagram. The Workflow diagram charts the relationships between the current workbook, the workbooks that the current workbook was created from (the master workbook and any parent workbooks), and, all workbooks created from this workbook (children of the current workbook).The Workbook diagram displays all the sheets and processes that created the current workbook. Youcan click any node on the diagram to quickly go to the workbook that contains the sheet or chart represented by the node.

UNIT-2

18) What is meant by data cleaning? Explain the basic methods for data cleaning.

Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies