1. Introduction about Big Data Analytics

Big Data Analytics uses advanced analytical methods that can extract important business insights from bulk datasets. Within these datasets lies both structured (organized) and unstructured (unorganized) data. Its applications cover different industries such as **healthcare, education, insurance, AI, retail, and manufacturing**.

What is Big-Data Analytics?

Big Data Analytics is all about crunching massive amounts of information to uncover hidden trends, patterns, and relationships. It's like sifting through a giant mountain of data to find the gold nuggets of insight.

Here's a breakdown of what it involves:

Collecting Data: Such data is coming from various sources such as social media, web traffic, sensors and customer reviews.

Cleaning the Data: Imagine having to assess a pile of rocks that included some gold pieces in it. You would have to clean the dirt and the debris first. When data is being cleaned, mistakes must be fixed, duplicates must be removed and the data must be formatted properly.

Analyzing the Data: It is here that the wizardry takes place. Data analysts employ powerful tools and techniques to discover patterns and trends. It is the same thing as looking for a specific pattern in all those rocks that you sorted through.

How does big data analytics work?

Big Data Analytics is a powerful tool which helps to find the potential of large and complex datasets. To get better understanding, let's break it down into key steps:

Data Collection: Data is the core of Big Data Analytics. It is the gathering of data from different sources such as the customers' comments, surveys, sensors, social media, and so on. The primary aim of data collection is to compile as much accurate data as possible. The more data, the more insights.

Data Cleaning (Data Preprocessing): The next step is to process this information. It often requires some cleaning. This entails the replacement of missing data, the correction of inaccuracies, and the removal of duplicates. It is like sifting through a treasure trove, separating the rocks and debris and leaving only the valuable gems behind.

Data Processing: After that we will be working on the data processing. This process contains such important stages as writing, structuring, and formatting of data in a way it will be usable for the analysis. It is like a chef who is gathering the ingredients before cooking. Data processing turns the data into a format suited for analytics tools to process.

Data Analysis: Data analysis is being done by means of statistical, mathematical, and machine learning methods to get out the most important findings from the processed data. For example, it can uncover customer preferences, market trends, or patterns in healthcare data.

Data Visualization: Data analysis usually is presented in visual form, for illustration charts, graphs and interactive dashboards. The visualizations provided a way to simplify the large amounts of data and allowed for decision makers to quickly detect patterns and trends.

Data Storage and Management: The stored and managed analyzed data is of utmost importance. It is like digital scrapbooking. May be you would want to go back to those lessons in the long run, therefore, how you store them has great importance. Moreover, data protection and adherence to regulations are the key issues to be addressed during this crucial stage.

Continuous Learning and Improvement: Big data analytics is a continuous process of collecting, cleaning, and analyzing data to uncover hidden insights. It helps businesses make better decisions and gain a competitive edge.

Types of Big Data Analytics

Big Data Analytics comes in many different types, each serving a different purpose:

Descriptive Analytics: This type helps us understand past events. In social media, it shows performance metrics, like the number of likes on a post.

Diagnostic Analytics: In Diagnostic analytics delves deeper to uncover the reasons behind past events. In healthcare, it identifies the causes of high patient re-admissions.

Predictive Analytics: Predictive analytics forecasts future events based on past data. Weather forecasting, for example, predicts tomorrow's weather by analyzing historical patterns.

Prescriptive Analytics: However, this category not only predicts results but also offers recommendations for action to achieve the best results. In e-commerce, it may suggest the best price for a product to achieve the highest possible profit.

Real-time Analytics: The key function of real-time analytics is data processing in real time. It swiftly allows traders to make decisions based on real-time market events.

Spatial Analytics: Spatial analytics is about the location data. In urban management, it

optimizes traffic flow from the data unde the sensors and cameras to minimize the traffic jam.

Text Analytics: Text analytics delves into the unstructured data of text. In the hotel business, it can use the guest reviews to enhance services and guest satisfaction.

Big Data Analytics Technologies and Tools

Big Data Analytics relies on various technologies and tools that might sound complex,

Hadoop: Imagine Hadoop as an enormous digital warehouse. It's used by companies like Amazon to store tons of data efficiently. For instance, when Amazon suggests products you might like, it's because Hadoop helps manage your shopping history.

Spark: Think of Spark as the super-fast data chef. Netflix uses it to quickly analyze what you watch and recommend your next binge-worthy show.

NoSQL Databases: NoSQL databases, like MongoDB, are like digital filing cabinets that Airbnb uses to store your booking details and user data. These databases are famous because of their quick and flexible, so the platform can provide you with the right information when you need it.

Tableau: Tableau is like an artist that turns data into beautiful pictures. The World Bank uses it to create interactive charts and graphs that help people understand complex economic data.

Python and R: Python and R are like magic tools for data scientists. They use these languages to solve tricky problems. For example, Kaggle uses them to predict things like house prices based on past data.

Machine Learning Frameworks (e.g., TensorFlow): In Machine learning frameworks are the tools who make predictions. Airbnb uses TensorFlow to predict which properties are most likely to be booked in certain areas. It helps hosts make smart decisions about pricing and availability. These tools and technologies are the building blocks of Big Data Analytics and helps organizations gather, process, understand, and visualize data, making it easier for them to make decisions based on information.

Benefits of Big Data Analytics

Big Data Analytics offers a host of real-world advantages, and let's understand with examples:

Informed Decisions: Imagine a store like Walmart. Big Data Analytics helps them make smart choices about what products to stock. This not only reduces waste but also keeps customers happy and profits high.

Enhanced Customer Experiences: Think about Amazon. Big Data Analytics is what makes those product suggestions so accurate. It's like having a personal shopper who knows your taste and helps you find what you want.

Fraud Detection: Credit card companies, like MasterCard, use Big Data Analytics to catch and stop fraudulent transactions. It's like having a guardian that watches over your money and keeps it safe.

Optimized Logistics: FedEx, for example, uses Big Data Analytics to deliver your packages faster and with less impact on the environment. It's like taking the fastest route to your destination while also being kind to the planet.

Challenges of Big data analytics

While Big Data Analytics offers incredible benefits, it also comes with its set of challenges:

Data Overload: Consider Twitter, where approximately 6,000 tweets are posted every second. The challenge is sifting through this avalanche of data to find valuable insights.

Data Quality: If the input data is inaccurate or incomplete, the insights generated by Big Data Analytics can be flawed. For example, incorrect sensor readings could lead to wrong conclusions in weather forecasting.

Privacy Concerns: With the vast amount of personal data used, like in Facebook's ad targeting, there's a fine line between providing personalized experiences and infringing on privacy.

Security Risks: With cyber threats increasing, safeguarding sensitive data becomes crucial. For instance, banks use Big Data Analytics to detect fraudulent activities, but they must also protect this information from breaches.

Costs: Implementing and maintaining Big Data Analytics systems can be expensive. Airlines like Delta use analytics to optimize flight schedules, but they need to ensure that the benefits outweigh the costs.

Usage of Big Data Analytics

Big Data Analytics has a significant impact in various sectors:

Healthcare: It aids in precise diagnoses and disease prediction, elevating patient care.

Retail: Amazon's use of Big Data Analytics offers personalized product recommendations based on your shopping history, creating a more tailored and enjoyable shopping experience.

Finance: Credit card companies such as Visa rely on Big Data Analytics to swiftly identify

and prevent fraudulent transactions, ensuring the safety of your financial assets.

Transportation: Companies like Uber use Big Data Analytics to optimize drivers' routes and predict demand, reducing wait times and improving overall transportation experiences.

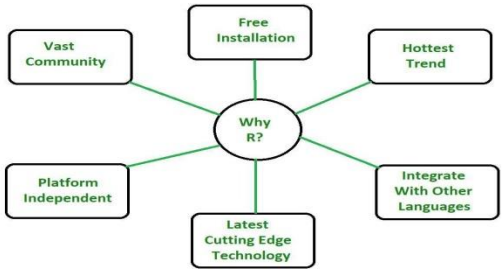
Agriculture: Farmers make informed decisions, boosting crop yields while conserving resources.

Manufacturing: Companies like General Electric (GE) use Big Data Analytics to predict machinery maintenance needs, reducing downtime and enhancing operational efficiency.

2. Introduction about R Programming

The R Language stands out as a powerful tool in the modern era of statistical computing and data analysis. Widely embraced by statisticians, data scientists, and researchers, the R Language offers an extensive suite of packages and libraries tailored for data manipulation, statistical modeling, and visualization. In this article, we explore the features, benefits, and applications of the R Programming Language, shedding light on why it has become an indispensable asset for data-driven professionals across various industries.

R programming language is an implementation of the S programming language. It also combines with lexical scoping semantics inspired by Scheme. Moreover, the project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000.



What is R Programming Language?

R programming is a leading tool for machine learning, statistics, and data analysis, allowing for the easy creation of objects, functions, and packages. Designed by Ross Ihaka and Robert Gentleman at the University of Auckland and developed by the R Development Core Team, R Language is platform-independent and open-source, making it accessible for use across all operating systems without licensing costs. Beyond its capabilities as a statistical package, R integrates with other languages like C and C++, facilitating interaction with various data

sources and statistical tools. With a growing community of users and high demand in the Data Science job market, R is one of the most sought-after programming languages today. Originating as an implementation of the S programming language with influences from Scheme, R has evolved since its conception in 1992, with its first stable beta version released in 2000.

Why Use R Language?

The **R Language** is a powerful tool widely used for data analysis, statistical computing, and machine learning. Here are several reasons why professionals across various fields prefer R:

1. Comprehensive Statistical Analysis:

R language is specifically designed for statistical analysis and provides a vast array of statistical techniques and tests, making it ideal for data-driven research.

2. Extensive Packages and Libraries:

The R Language boasts a rich ecosystem of packages and libraries that extend its capabilities, allowing users to perform advanced data manipulation, visualization, and machine learning tasks with ease.

3. Strong Data Visualization Capabilities:

R language excels in data visualization, offering powerful tools like ggplot2 and plotly, which enable the creation of detailed and aesthetically pleasing graphs and plots.

4. Open Source and Free:

As an open-source language, R is free to use, which makes it accessible to everyone, from individual researchers to large organizations, without the need for costly licenses.

5. Platform Independence:

The R Language is platform-independent, meaning it can run on various operating systems, including Windows, macOS, and Linux, providing flexibility in development environments.

6. Integration with Other Languages:

R can easily integrate with other programming languages such as C, C++, Python, and Java, allowing for seamless interaction with different data sources and statistical packages.

7. Growing Community and Support:

R language has a large and active community of users and developers who contribute to its continuous improvement and provide extensive support through forums, mailing lists, and

online resources.

8. High Demand in Data Science:

R is one of the most requested programming languages in the Data Science job market, making it a valuable skill for professionals looking to advance their careers in this field.

Features of R Programming Language

The R Language is renowned for its extensive features that make it a powerful tool for data analysis, statistical computing, and visualization. Here are some of the key features of R:

1. Comprehensive Statistical Analysis:

R langauge provides a wide array of statistical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, and clustering.

2. Advanced Data Visualization:

With packages like ggplot2, plotly, and lattice, R excels at creating complex and aesthetically pleasing data visualizations, including plots, graphs, and charts.

3. Extensive Packages and Libraries:

The Comprehensive R Archive Network (CRAN) hosts thousands of packages that extend R's capabilities in areas such as machine learning, data manipulation, bioinformatics, and more.

4. Open Source and Free:

R is free to download and use, making it accessible to everyone. Its open-source nature encourages community contributions and continuous improvement.

5. Platform Independence:

R is platform-independent, running on various operating systems, including Windows, macOS, and Linux, which ensures flexibility and ease of use across different environments.

6. Integration with Other Languages:

R language can integrate with other programming languages such as C, C++, Python, Java, and SQL, allowing for seamless interaction with various data sources and computational processes.

7. Powerful Data Handling and Storage:

R efficiently handles and stores data, supporting various data types and structures, including vectors, matrices, data frames, and lists.

8. Robust Community and Support:

R has a vibrant and active community that provides extensive support through forums, mailing lists, and online resources, contributing to its rich ecosystem of packages and documentation.

9. Interactive Development Environment (IDE):

RStudio, the most popular IDE for R, offers a user-friendly interface with features like syntax highlighting, code completion, and integrated tools for plotting, history, and debugging.

10. Reproducible Research:

R supports reproducible research practices with tools like R Markdown and Knitr, enabling users to create dynamic reports, presentations, and documents that combine code, text, and visualizations.

Advantages of R language

- R is the most comprehensive statistical analysis package. As new technology and concepts often appear first in R.
- As R programming language is an open source. Thus, you can run R anywhere and at any time.
- R programming language is suitable for GNU/Linux and Windows operating systems.
- R programming is cross-platform and runs on any operating system.

In R, everyone is welcome to provide new packages, bug fixes, and code enhancements.

Disadvantages of R language

- In the R programming language, the standard of some packages is less than perfect.
- Although, R commands give little pressure on memory management. So R programming language may consume all available memory.
- In R basically, nobody to complain if something doesn't work.
- R programming language is much slower than other programming languages such as Python and MATLAB.

Applications of R language

- We use R for Data Science. It gives us a broad variety of libraries related to statistics. It also provides the environment for statistical computing and design.
- R is used by many quantitative analysts as its programming tool. Thus, it helps in

data importing and cleaning.

- R is the most prevalent language. So many data analysts and research programmers use it. Hence, it is used as a fundamental tool for finance.
- Tech giants like Google, Facebook, Bing, Twitter, Accenture, Wipro, and many more using R nowadays.

3. Introduction About Apache Pig

Pig Represents Big Data as data flows. Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as Pig Latin which is used to develop the data analysis codes. First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language. Internally Pig Engine(a component of Apache Pig) converted all these scripts into a specific map and reduce task. But these are not visible to the programmers in order to provide a high-level of abstraction. Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

Note: Pig Engine has two type of execution environment i.e. a local execution environment in a single JVM (used when dataset is small in size)and distributed execution environment in a Hadoop Cluster.

Need of Pig: One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development using the multi-query approach. Also, Pig is beneficial for programmers who are not from Java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin.

It uses query approach which results in reducing the length of the code.

Pig Latin is SQL like language.

It provides many builtIn operators.

It provides nested data types (tuples, bags, map).

Evolution of Pig: Earlier in 2006, Apache Pig was developed by Yahoo’s researchers. At that time, the main idea to develop Pig was to execute the MapReduce jobs on extremely large

datasets. In the year 2007, it moved to Apache Software Foundation(ASF) which makes it an open source project. The first version(0.1) of Pig came in the year 2008. The latest version of Apache Pig is 0.18 which came in the year 2017.

Features of Apache Pig:

For performing several operations Apache Pig provides rich sets of operators like the filtering, joining, sorting, aggregation etc.

Easy to learn, read and write. Especially for SQL-programmer, Apache Pig is a boon.

Apache Pig is extensible so that you can make your own process and user-defined functions(UDFs) written in python, java or other programming languages .

Join operation is easy in Apache Pig.

Fewer lines of code.

Apache Pig allows splits in the pipeline.

By integrating with other components of the Apache Hadoop ecosystem, such as Apache Hive, Apache Spark, and Apache ZooKeeper, Apache Pig enables users to take advantage of these components' capabilities while transforming data.

The data structure is multivalued, nested, and richer.

Pig can handle the analysis of both structured and unstructured data.

Difference between Pig and MapReduce

Apache Pig	MapReduce
It is a scripting language.	It is a compiled programming language.
Abstraction is at higher level.	Abstraction is at lower level.
It have less line of code as compared to MapReduce.	Lines of code is more.
Less effort is needed for Apache Pig.	More development efforts are required for MapReduce.
Code efficiency is less as compared to	As compared to Pig efficiency of code is

Apache Pig	MapReduce
MapReduce.	higher.
Pig provides built in functions for ordering, sorting and union.	Hard to perform data operations.
It allows nested data types like map, tuple and bag	It does not allow nested data types

Applications of Apache Pig:

For exploring large datasets Pig Scripting is used.

Provides the supports across large data-sets for Ad-hoc queries.

In the prototyping of large data-sets processing algorithms.

Required to process the time sensitive data loads.

For collecting large amounts of datasets in form of search logs and web crawls.

Used where the analytical insights are needed using the sampling.

Types of Data Models in Apache Pig: It consist of the 4 types of data models as follows:

Atom: It is a atomic data value which is used to store as a string. The main use of this model is that it can be used as a number and as well as a string.

Tuple: It is an ordered set of the fields.

Bag: It is a collection of the tuples.

Map: It is a set of key/value pairs.

4. Introduction about cassendra

Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL database. Let us first understand what a NoSQL database does.

NoSQLDatabase

A NoSQL database (sometimes called as Not Only SQL) is a database that provides a

mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

The primary objective of a NoSQL database is to have

simplicity of design,

horizontal scaling, and

finer control over availability.

NoSql databases use different data structures compared to relational databases. It makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it must solve.

NoSQL vs. Relational Database

The following table lists the points that differentiate a relational database from a NoSQL database.

Relational Database	NoSql Database
Supports powerful query language.	Supports very simple query language.
It has a fixed schema.	No fixed schema.
Follows ACID (Atomicity, Consistency, Isolation, and Durability).	It is only "eventually consistent".
Supports transactions.	Does not support transactions.

Besides Cassandra, we have the following NoSQL databases that are quite popular −

Apache HBase − HBase is an open source, non-relational, distributed database modeled after Google's BigTable and is written in Java. It is developed as a part of Apache Hadoop project and runs on top of HDFS, providing BigTable-like capabilities for Hadoop.

MongoDB − MongoDB is a cross-platform document-oriented database system that avoids using the traditional table-based relational database structure in favor of JSON-like documents

with dynamic schemas making the integration of data in certain types of applications easier and faster.

What is Apache Cassandra?

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

Listed below are some of the notable points of Apache Cassandra −

It is scalable, fault-tolerant, and consistent.

It is a column-oriented database.

Its distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable.

Created at Facebook, it differs sharply from relational database management systems.

Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model.

Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Features of Cassandra

Cassandra has become so popular because of its outstanding technical features. Given below are some of the features of Cassandra:

Elastic scalability − Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.

Always on architecture − Cassandra has no single point of failure and it is continuously available for business-critical applications that cannot afford a failure.

Fast linear-scale performance − Cassandra is linearly scalable, i.e., it increases your throughput as you increase the number of nodes in the cluster. Therefore it maintains a quick response time.

Flexible data storage − Cassandra accommodates all possible data formats including: structured, semi-structured, and unstructured. It can dynamically accommodate changes to your data structures according to your need.

Easy data distribution − Cassandra provides the flexibility to distribute data where you need by replicating data across multiple data centers.

Transaction support − Cassandra supports properties like Atomicity, Consistency, Isolation,

and Durability (ACID).

Fast writes − Cassandra was designed to run on cheap commodity hardware. It performs blazingly fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

History of Cassandra

Cassandra was developed at Facebook for inbox search.

It was open-sourced by Facebook in July 2008.

Cassandra was accepted into Apache Incubator in March 2009.

It was made an Apache top-level project since February 2010.

WEEK - 1

Q1. Implement a simple map reduce job that builds an inverted index on the set of input documents (hadoop)?

Creating an inverted index using MapReduce in Hadoop is a common task for efficiently searching and retrieving documents. An inverted index maps each term to the list of documents (and positions) in which it occurs.
Here's a basic implementation:

Mapper Class
The Mapper reads each document, tokenizes it, and emits each token (word) along with the document ID as the key-value pair.

```
Import java.io.IOException;
Import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class InvertedIndexMapper extends Mapper<LongWritable, Text, Text, Text> {
   private Text word = new Text();
   private Text docID = new Text();
   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      String[] parts = line.split("\t", 2);
      String documentID = parts[0];
      String content = parts[1];
      docID.set(documentID);
      String[] words = content.split("\\s+");
      for (String w : words) {
         word.set(w.toLowerCase());
         context.write(word, docID);
      }
   }
}
```

Reducer Class
The Reducer receives all the values (document IDs) for each key (word) and writes them out.

```
import java.io.IOException;
import java.util.HashSet;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class InvertedIndexReducer extends Reducer<Text, Text, Text, Text> {
   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
      HashSet<String> docSet = new HashSet<>();
      for (Text val : values) {
         docSet.add(val.toString());
      }
      context.write(key, new Text(String.join(",", docSet)));
   }
}
```

Driver Class

The driver class to set up and run the job.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class InvertedIndex {
   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "Inverted Index");
      job.setJarByClass(InvertedIndex.class);
      job.setMapperClass(InvertedIndexMapper.class);
      job.setReducerClass(InvertedIndexReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(Text.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

Example Input

Input files should have the format docID<tab>document content. For example:

1 Hadoop is a framework

2 Hadoop allows for the distributed processing and an example of an inverted index

Example Output

The output will be the inverted index, mapping each term to a list of document IDs where it occurs and output in below table

a	1,2
allows	2
an	3
distributed	2
example	3
framework	1
hadoop	1,2
index	3
is	1,3
of	3
processing	2
the	2
this	3

WEEK - 2

Q2. Process BigData in HBase?

Processing Big Data in HBase involves using its distributed, column-oriented storage system to store, manage, and analyze large datasets. HBase is well-suited for real-time read/write access to large amounts of sparse data. Let's go through a simple example of how to process data in HBase.

Steps to Process Big Data in HBase

Set up HBase: Ensure you have HBase installed and running on your cluster. You can download HBase from the Apache HBase website and follow the installation instructions.

Create an HBase Table: Create a table in HBase to store your data. For example, let's create a table named BigData with a column family data.

```
hbase(main):001:0> create 'BigData', 'data'
```

Load Data into HBase: Load your data into the HBase table. For instance, let's assume we have some sample input data:

Input Data:

```
row1, value1
row2, value2
row3, value3
```

We can use the HBase shell to insert this data into the BigData table.

```
hbase(main):002:0> put 'BigData', 'row1', 'data:column1', 'value1'
hbase(main):003:0> put 'BigData', 'row2', 'data:column1', 'value2'
hbase(main):004:0> put 'BigData', 'row3', 'data:column1', 'value3'
```

Query Data from HBase: You can retrieve the data using the HBase shell to verify the insertion.

```
hbase(main):005:0> scan 'BigData'
```

Output:

```
Row   column+cell
row1  column=data:column1, timestamp=..., value=value1
row2  column=data:column1, timestamp=..., value=value2
row3  column=data:column1, timestamp=..., value=value3
```

Process Data Using a MapReduce Job: You can also process the data using a MapReduce job. Here is a simple example of a MapReduce job to read and process data from the BigData table.

Mapper Class:

```
import java.io.IOException;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
```

```
import org.apache.hadoop.hbase.mapreduce.TableMapper;
import org.apache.hadoop.io.Text;
public class HBaseMapper extends TableMapper<Text, Text> {
   @Override
   protected void map(ImmutableBytesWritable key, Result value, Context context) throws
IOException, InterruptedException {
      String rowKey = new String(key.get());
      String cellValue = new String(value.getValue("data".getBytes(), "column1".getBytes()));
      context.write(new Text(rowKey), new Text(cellValue));
   }
}
```

Reducer Class:

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class HBaseReducer extends Reducer<Text, Text, Text, Text> {
   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
      for (Text val : values) {
         context.write(key, val);
      }
   }
}
```

Driver Class:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
public class HBaseDriver {
   public static void main(String[] args) throws Exception {
      Configuration conf = HBaseConfiguration.create();
      Job job = Job.getInstance(conf, "HBase Processing");
      job.setJarByClass(HBaseDriver.class);
      TableMapReduceUtil.initTableMapperJob("BigData", null, HBaseMapper.class,
Text.class, Text.class, job);
      job.setReducerClass(HBaseReducer.class);
      job.setOutputKeyClass(Text.class);
```

```
        job.setOutputValueClass(Text.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Example Output

The MapReduce job will read the data from the BigData table and process it, producing the following output:

```
row1    value1
row2    value2
row3    value3
```

This example demonstrates how to set up an HBase table, load data into it, query the data, and process it using a MapReduce job. HBase is powerful for handling large datasets in real-time and is often used in combination with Hadoop for big data processing tasks.

WEEK - 3

Q3. Store and Retrieve data in Pig?

Using Apache Pig, you can analyze large datasets stored in Hadoop and perform various data manipulation operations. Let's look at an example of storing and retrieving data using Pig Latin scripts.
Example Scenario
Suppose we have a dataset of user information with fields: user_id, name, and age. We want to store this data in a file and then retrieve and process it using Pig.
Step 1: Store Data in a File
First, create an input file named userdata.txt with the following content:
Input Data (userdata.txt):

1,John,25
2,Jane,30
3,Mike,22
4,Alice,28

Step 2: Load Data into Pig
Write a Pig script to load the data from the file and perform some operations.
Pig Script (process_userdata.pig):

```
-- Load the data from the input file

userdata = LOAD 'userdata.txt' USING PigStorage(',') AS (user_id: int, name: chararray, age: int);

-- Filter users whose age is greater than 25

older_users = FILTER userdata BY age > 25;

-- Store the filtered data into an output file

STORE older_users INTO 'output' USING PigStorage(',');
```

Step 3: Run the Pig Script
Execute the Pig script using the pig command:

```
pig process_userdata.pig
```

Step 4: Retrieve Data from the Output
After running the Pig script, the filtered data will be stored in the output directory. You can

view the contents of the output file:

hadoop fs -cat output/part-r-00000

Output Data:

2,Jane,30
4,Alice,28

Explanation

Load: The LOAD statement loads the data from userdata.txt into a relation called userdata. The PigStorage(',') function specifies that the data is comma-separated.
Filter: The FILTER statement filters the users whose age is greater than 25 and stores the result in a relation called older_users.
Store: The STORE statement stores the filtered data into the output directory using comma-separated format.
This example demonstrates how to load data into Pig, perform data filtering, and store the processed data back to HDFS. Pig is highly versatile and can be used for various data processing tasks with simple scripts.

WEEK - 4

Q4. Perform Social Media Analysis using Cassandra?

Performing social media analysis using Apache Cassandra involves storing and analyzing large volumes of data generated by social media platforms. Cassandra is well-suited for this task due to its scalability and high write throughput. Let's walk through an example of how to set up Cassandra for social media analysis.

Step 1: Set Up Cassandra

Ensure Cassandra is installed and running on your cluster. You can download Cassandra from the Apache Cassandra website and follow the installation instructions.

Step 2: Create a Keyspace and Table

Create a keyspace and table to store social media data. For example, let's create a keyspace named social_media and a table named posts to store social media posts.

```
CREATE KEYSPACE social_media WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
USE social_media;
CREATE TABLE posts (
   user_id text,
   post_id text PRIMARY KEY,
   content text,
   timestamp timestamp
);
```

Step 3: Load Data into Cassandra

Load sample social media data into the posts table. For example:

```
INSERT INTO posts (user_id, post_id, content, timestamp) VALUES ('user123', 'post1', 'Loving the new features of Cassandra!', '2023-01-01 12:00:00');
INSERT INTO posts (user_id, post_id, content, timestamp) VALUES ('user456', 'post2', 'Just had a great day at the beach!', '2023-01-01 13:00:00');
INSERT INTO posts (user_id, post_id, content, timestamp) VALUES ('user789', 'post3', 'Excited about the upcoming concert!', '2023-01-01 14:00:00');
```

Step 4: Query Data from Cassandra

Retrieve and analyze the data using CQL (Cassandra Query Language). For example, to get all posts from a specific user:

SELECT * FROM posts WHERE user_id = 'user123';

Output:

```
 user_id | post_id | content                   | timestamp
---------+---------+---------------------------+---------------------
 user123 | post1   | Loving the new features of Cassandra! | 2023-01-01 12:00:00
```

Step 5: Analyze Data

You can perform various analyses on the data stored in Cassandra. For example, you can count the number of posts per user, analyze the frequency of certain keywords, or perform sentiment analysis on the content.

Example Analysis: Count Posts per User

SELECT user_id, COUNT(*) AS post_count FROM posts GROUP BY user_id;

Output:

```
 user_id | post_count
---------+------------
 user123 | 1
 user456 | 1
 user789 | 1
```

This example demonstrates how to set up Cassandra for social media analysis, load data, query it, and perform basic analysis. Cassandra's distributed architecture and efficient data handling make it an excellent choice for handling large-scale social media data.

WEEK-5

Q5. Buyer event analytics using Cassandra on suitable product sales data?

Sure! Let's walk through an example of how to perform buyer event analytics using Cassandra on product sales data.

Step 1: Set Up Cassandra

Ensure Cassandra is installed and running on your cluster.

Step 2: Create a Keyspace and Table

Create a keyspace and table to store product sales data. For example:

```
CREATE KEYSPACE sales_data WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
USE sales_data;

CREATE TABLE purchases (
   purchase_id text,
   user_id text,
   product_id text,
   quantity int,
   purchase_date timestamp,
   PRIMARY KEY (purchase_id)
);
```

Step 3: Load Data into Cassandra

Load sample product sales data into the purchases table. For example:

```
INSERT INTO purchases (purchase_id, user_id, product_id, quantity, purchase_date) VALUES ('p1', 'u1', 'prod1', 2, '2023-01-01 12:00:00');
INSERT INTO purchases (purchase_id, user_id, product_id, quantity, purchase_date) VALUES ('p2', 'u2', 'prod2', 1, '2023-01-01 13:00:00');
INSERT INTO purchases (purchase_id, user_id, product_id, quantity, purchase_date) VALUES ('p3', 'u1', 'prod3', 3, '2023-01-01 14:00:00');
```

Step 4: Query Data from Cassandra

Retrieve and analyze the data using CQL (Cassandra Query Language). For example, to get all purchases made by a specific user:

SELECT * FROM purchases WHERE user_id = 'u1';

Output:

```
 purchase_id | user_id | product_id | quantity | purchase_date
-------------+---------+------------+----------+---------------------
 p1          | u1      | prod1      | 2        | 2023-01-01 12:00:00
 p3          | u1      | prod3      | 3        | 2023-01-01 14:00:00
```

Step 5: Analyze Data

Perform various analyses on the data stored in Cassandra. For example, you can count the number of purchases per product:

SELECT product_id, COUNT(*) AS purchase_count FROM purchases GROUP BY product_id;

Output:

```
 product_id | purchase_count
------------+----------------
 prod1      | 1
 prod2      | 1
 prod3      | 1
```

Example Analysis: Calculate Total Quantity Sold per Product

SELECT product_id, SUM(quantity) AS total_quantity_sold FROM purchases GROUP BY product_id;

Output:

```
 product_id | total_quantity_sold
```

```
------------+--------------------
 prod1      | 2
 prod2      | 1
 prod3      | 3
```

This example demonstrates how to set up Cassandra for buyer event analytics, load data, query it, and perform basic analysis. Cassandra's distributed architecture and efficient data handling make it an excellent choice for handling large-scale product sales data.

WEEK - 6

Q6. Using Power Pivot (Excel) perform the following on any dataset?

Using Power Pivot in Excel, you can analyze large datasets and create powerful data visualizations. Let's walk through an example of how to perform Big Data Analytics and Big Data Charting using Power Pivot.

Step 1: Load Data into Power Pivot

Open Excel and go to the "Power Pivot" tab.
Click on "Manage" to open the Power Pivot window.
Click on "Get External Data" and choose your data source (e.g., Excel file, SQL Server, etc.).
Import the data into Power Pivot.

For this example, let's assume we have the following dataset in an Excel file named sales_data.xlsx:

Example Input (sales_data.xlsx):

Date	Product	Sales	Quantity
2023-01-01	Product A	1000	10
2023-01-02	Product B	1500	15
2023-01-03	Product A	2000	20
2023-01-04	Product C	2500	25
2023-01-05	Product B	3000	30

Step 2: Create Relationships

If your dataset spans multiple tables, create relationships between the tables in the Power Pivot window.

Q6a. Big Data Analytics

Use Power Pivot to create measures, calculated columns, and pivot tables for analysis.
Create Measures:

Click on "Home" > "New Measure" to create a new measure.

For example, create a measure to calculate the total sales:

DAX
Total Sales := SUM([Sales])

Create Calculated Columns:

Click on "Home" > "Calculated Column" to create a new calculated column.

For example, create a calculated column to calculate the average sales price:

DAX
Avg Sales Price := [Sales] / [Quantity]

Create Pivot Tables:

Go back to the Excel window.
Click on "Insert" > "Pivot Table" and choose to use an external data source from the Power Pivot model.
Use the fields from the Power Pivot model to create a pivot table.

Example Output (Pivot Table):

Product	Total Sales	Total Quantity	Avg Sales Price
Product A	3000	30	100
Product B	4500	45	100
Product C	2500	25	100

Q6b. Big Data Charting

Create visualizations using the pivot table data.

Create Charts:

Select the pivot table.
Click on "Insert" > "PivotChart" to create a chart based on the pivot table data.
Choose the desired chart type (e.g., column chart, line chart, etc.).
Example Output (Pivot Chart):

Date Product Sales Quantity Avg Sales Price

Date	Product	Sales	Quantity	Avg Sales Price
2023-01-01	Product A	1000	10	100
2023-01-02	Product B	1500	15	100
2023-01-03	Product A	2000	20	100
2023-01-04	Product C	2500	25	100
2023-01-05	Product B	3000	30	100

Using Power Pivot in Excel, you can handle large datasets, perform detailed analytics, and create insightful visualizations.

WEEK-7

Q7. Use R-Project to carry out statistical analysis of big data?

Using R for big data analysis involves leveraging R's extensive libraries and packages for statistical computing and visualization. R is particularly powerful for statistical analysis, data manipulation, and generating detailed graphs and charts. Let's go through an example of how to perform statistical analysis of big data using R.

Step 1: Install Required Packages

First, you need to install and load the necessary packages. data.table and dplyr are popular for data manipulation, while ggplot2 is great for visualization. Additionally, ff and bigmemory packages can help manage large datasets efficiently.

R Program

```
install.packages(c("data.table", "dplyr", "ggplot2", "ff", "bigmemory"))
library(data.table)
library(dplyr)
library(ggplot2)
library(ff)
library(bigmemory)
```

Step 2: Load and Preprocess Data

For this example, let's assume we have a large CSV file named bigdata.csv with the following structure:

Example Input (bigdata.csv):

Date	Product	Sales	Quantity
2023-01-01	Product A	1000	10
2023-01-02	Product B	1500	15
2023-01-03	Product A	2000	20
2023-01-04	Product C	2500	25
2023-01-05	Product B	3000	30

Load the data into R and convert it into a data.table for efficient processing.

R Program

```
# Load data
data <- fread("bigdata.csv")
```

```
# Convert to data.table
setDT(data)
```

Step 3: Perform Statistical Analysis

Perform some basic statistical analyses on the data.
Summary Statistics: Calculate summary statistics for the Sales and Quantity columns.

R Program

```
summary_stats <- data[, .(
  mean_sales = mean(Sales),
  median_sales = median(Sales),
  sd_sales = sd(Sales),
  mean_quantity = mean(Quantity),
  median_quantity = median(Quantity),
  sd_quantity = sd(Quantity)
)]
print(summary_stats)
```

Correlation Analysis: Calculate the correlation between Sales and Quantity.

R program

```
correlation <- cor(data$Sales, data$Quantity)
print(correlation)
```

Step 4: Visualize Data
Create visualizations to better understand the data.
Scatter Plot: Create a scatter plot to visualize the relationship between Sales and Quantity.

R Program

```
ggplot(data, aes(x = Quantity, y = Sales)) +
  geom_point() +
  labs(title = "Scatter Plot of Sales vs Quantity", x = "Quantity", y = "Sales")
```

Time Series Plot: Create a time series plot to visualize sales over time.

R Program

```
ggplot(data, aes(x = as.Date(Date), y = Sales, group = Product, color = Product)) +
  geom_line() +
  labs(title = "Time Series Plot of Sales Over Time", x = "Date", y = "Sales")
```

Example Output

Summary Statistics:

	mean_sales	median_sales	sd_sales	mean_quantity	median_quantity	sd_quantity
1:	2000	2000	790.5694	20.0	20	7.905694

Correlation:

[1] 1

Scatter Plot:

Time Series Plot:

Using R and its powerful libraries, you can perform extensive statistical analysis and create detailed visualizations for big data. These steps provide a foundation for analyzing and visualizing large datasets in R.

WEEK-8

Q8. Use R-Project for data visualization of social media data?

Let's walk through an example of how to use R for data visualization of social media data. We'll use a dataset that includes information about social media posts such as post content, user interactions (likes, comments, shares), and timestamps.

Step 1: Install Required Packages
First, ensure you have the necessary packages installed. We'll use ggplot2 for visualization and dplyr for data manipulation.

R Program
```
install.packages(c("ggplot2", "dplyr"))
library(ggplot2)
library(dplyr)
```

Step 2: Load and Prepare Data
Assume we have a dataset in a CSV file named social_media_data.csv with the following structure:

Example Input (social_media_data.csv):

PostID	UserID	Content	Likes	Comments	Shares	Timestamp
1	u1	Loving R for data viz!	120	15	30	2023-01-01 12:00:00
2	u2	Just had a great lunch!	85	12	25	2023-01-01 13:00:00
3	u1	Exploring R's ggplot2	90	18	20	2023-01-01 14:00:00
4	u3	Happy New Year everyone!	200	50	60	2023-01-01 15:00:00
5	u2	Back to work.	70	10	15	2023-01-01 16:00:00

Load the data into R and convert the Timestamp to a date-time format.

R Program
```
# Load data
data <- read.csv("social_media_data.csv")
```

```
# Convert Timestamp to date-time format
data$Timestamp <- as.POSIXct(data$Timestamp, format="%Y-%m-%d %H:%M:%S")
```

Step 3: Visualize Data
Create visualizations to analyze social media data.
Bar Plot of Likes per Post: Create a bar plot to visualize the number of likes each post received.

R Program

```
ggplot(data, aes(x = PostID, y = Likes)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Number of Likes per Post", x = "Post ID", y = "Likes")
```

Time Series Plot of Interactions Over Time: Create a time series plot to visualize the total number of interactions (likes, comments, shares) over time.

R Program

```
# Calculate total interactions
data <- data %>%mutate(TotalInteractions = Likes + Comments + Shares)
ggplot(data, aes(x = Timestamp, y = TotalInteractions, color = UserID)) +
  geom_line() +
  labs(title = "Total Interactions Over Time", x = "Timestamp", y = "Total Interactions")
```

Scatter Plot of Likes vs. Comments: Create a scatter plot to visualize the relationship between likes and comments.

R Program

```
ggplot(data, aes(x = Likes, y = Comments, color = UserID)) +
  geom_point() +
  labs(title = "Scatter Plot of Likes vs. Comments", x = "Likes", y = "Comments")
```

Output

Bar Plot of Likes per Post:
Time Series Plot of Interactions over Time:
Scatter Plot of Likes vs. Comments:
Using R and its visualization packages, you can gain valuable insights from social media data and present it in an easily understandable way.