Big Data refers to vast volumes of data—both structured and unstructured—that are so large and complex that traditional data processing applications cannot handle them efficiently. Big data is not just about the size; it also involves a variety of data sources and high velocity, which presents unique challenges for storage, processing, and analysis. **Characteristics of Big Data. 1. Volume :** This refers to the sheer amount of data being generated every second. The amount of data has been growing exponentially, with billions of data points collected daily from social media, sensors, transactions, and more. **2. Velocity :** The speed at which data is generated, processed, and analyzed. In many cases, data is produced in real-time (such as stock market data, social media feeds, etc.), and businesses must quickly process it to gain insights before it loses value. **3. Variety :** The diversity of data types, including structured (e.g., databases), semi-structured (e.g., JSON, XML), and unstructured data (e.g., text, images, videos). Modern big data solutions need to handle all these types and provide useful insights from them . **Digital data** refers to data that is represented in a format that computers can process and store. In the digital world, data can take various forms depending on its structure, origin, and how it is used. Below are the main types of digital data, categorized based on structure, format, and how it is represented .

The Hadoop Ecosystem is a collection of tools and frameworks designed to process, store, and analyze big data efficiently. It is built around the Hadoop framework and provides various components for different tasks like storage, computation, data integration, and analysis. **Components of 1.HDFS** • Distributed storage system for storing large datasets. • Provides fault tolerance and high throughput. **2. MapReduce:** • Programming model for processing large datasets in parallel. • Distributes the processing across multiple nodes. **3. YARN** (Yet Another Resource Negotiator): • Resource management layer. • Manages and schedules resources for Hadoop applications. **4. Hive** • Hive is a data warehouse infrastructure built on top of Hadoop. • It provides an SQL-like language called HiveQL for querying data. • Best suited for users who are familiar with SQL. **5. ZooKeeper** • ZooKeeper is a centralized coordination service for distributed systems. • It helps manage configuration, naming, and synchronization. **6. HBase** • HBase is a NoSQL database built on top of HDFS. • It allows real-time read/write access to big data. **7. Pig.** • Pig is a high-level platform for processing large datasets. • It uses a scripting language called Pig Latin.

(HDFS) is designed to provide a scalable, fault-tolerant, and high-performance distributed storage system that can handle large datasets. HDFS is a foundational component of the Apache Hadoop framework and is optimized for batch processing and high throughput rather than low latency. **Key Features of HDFS: 1. Distributed Storage:** • Splits large files into blocks (usually 128MB or 256MB). • Distributes these blocks across multiple nodes in a Hadoop cluster. **2. Fault Tolerance:** • Each block is replicated (default 3 copies) on different nodes. • If one node fails, data is still available from another node. **3. High Throughput:** • Optimized for batch processing and large files. • Suitable for applications needing high read/write speeds **4. Write Once, Read Many:** • Files are written once and then read many times (ideal for big data analysis). **5. Master-Slave Architecture:** • NameNode : Manages metadata (file names, block locations). • **DataNodes :** Store actual data blocks. **HDFS Architecture 1. NameNode (Master).** • Manages metadata (file names, block locations). • Does not store actual data. **2. DataNodes (Slaves)** • Store actual data blocks. • Regularly send heartbeats to NameNode. **3. Secondary NameNode** • Assists the NameNode by merging metadata and logs. • Not a real-time backup.

MapReduce is an algorithm developed by Google to address the limitations of traditional enterprise systems, which rely on centralized servers that bottleneck when processing large, scalable datasets. It divides a task into smaller parts, distributes them across multiple computers, and then collects and integrates the results into a final dataset. **Phases of MapReduce 1. Input Splitting** • The input data is split into blocks (InputSplits). • Each split is assigned to a Map task. **2.Mapping** • Each Mapper processes a split and generates intermediate key-value pairs. **3.Shuffling** • Transfers the intermediate key-value pairs from Mappers to appropriate Reducers. • Ensures that all values for the same key go to the same Reducer. **4. Sorting** • Keys are automatically sorted before being passed to Reducers. • Happens on the Reducer side. **5. Reducing** • Reducers aggregate the values for each key. •Produces the final output key-value pairs. **6.Output** • The final output is written to HDFS using the OutputFormat. • Stored in files like part-00000 . **Failures 1. Task Failure** • A Map or Reduce task crashes or hangs. • Causes: Code bugs, memory errors, data corruption. **2. TaskTracker Failure** • The worker node (where tasks run) fails. • All running tasks on that node are lost. **3. JobTracker Failure** • Master node responsible for managing jobs crashes. • Impact: Entire job might need to be restarted. **4. Network Failure** • Loss of communication between nodes (Map to Reduce, or task to master). **5. Disk Failure.** Loss of blocks or intermediate task data.

Pig is a scripting platform that runs on hadoop clusters designed to process and analyze large datasets. Pig is extensible, self-optimizing, and easily programmed. Programmers can use Pig to write data transformations without knowing Java. Pig uses both structured and unstructured data as input to perform analytics and uses hdfs to store the results **Importance of Pig in had.eco.** • Pig scripts are easier to write than complex MapReduce programs. • Reduces development time with fewer lines of code. • Ideal for semi-structured and unstructured data like logs, JSON, etc. • You can create UDFs (User Defined Functions) in multiple languages. • Can read from and write to HDFS, HBase, Hive, etc. • Runs Pig scripts in MapReduce, Tez, or Spark mode.

Hive • Data warehousing for big data on Hadoop. • Query Language HiveQL • Handles very large datasets • Uses HDFS Storage • Higher latency (not real-time) • Schema-on-read. • Limited transaction support. • Batch processing, data analysis. • Limited indexing support. **Traditional RDBMS.** •OLTP (Online Transaction Processing) systems • Query Language SQL • Optimized for smaller, structured data • Uses local disk storage • Low latency (fast query response) • Schema-on-write • Full ACID transaction support • Real-time operations, CRUD operations • Advanced indexing support

Apache Hive is a data warehousing tool built on top of Hadoop that allows users to query and manage large datasets stored in HDFS using a SQL-like language called HiveQL. 💡 It is mainly used for batch processing and data analysis. **Key Components. 1. User Interface (UI)** • Allows users to submit HiveQL queries. • Can be CLI (Command Line), Web UI, or JDBC/ODBC. **2.Driver** • Acts as the controller. • Manages the session, compiles queries, and monitors execution. **3.Compiler** • Converts HiveQL queries into MapReduce jobs or Tez/Spark tasks. • Performs parsing, semantic analysis, and optimization. **4. Execution Engine** • Executes the query using MapReduce, Tez, or Spark. • Works closely with the JobTracker/ResourceManager. **5.Metastore** • Stores metadata (tables, columns, data types, partitions). • Usually backed by RDBMS like MySQL or Derby. **6.HDFS** • Stores the actual data processed by Hive. • Hive only manages schema and logic, data lives in HDFS.

Machine learning allows computers to learn and make decisions without being explicitly programmed. It involves feeding data into algorithms to identify patterns and make predictions on new data. used in various applications, including image and speech recognition, natural language processing. **Types**
Supervised Learning: • Uses labeled data (input with correct output). • Goal is to learn from past data to make predictions. • Example tasks: predicting house prices, classifying emails (spam or not). • Output is known (like categories or numbers). • Human help is required for labeling the data.• Real-life example: Like a teacher gives questions with answers to help students learn.
Unsupervised Learning: • Uses unlabeled data (only input, no output). • Goal is to find hidden patterns or groupings in the data. • Example tasks: customer segmentation, grouping similar products. • Output is unknown groups or structures. • No human help needed for labeling. • Real-life example: Like students group topics without knowing the correct answers.

Big Data Analytics with BigR refers to the process of analyzing massive datasets using the R programming language integrated with the Hadoop ecosystem. BigR allows data scientists to use familiar R functions and statistical techniques to analyze big data stored in Hadoop, enabling complex analysis, data modeling, and machine learning—even on datasets that are too large for a single machine.
Integration with Hadoop: • BigR connects with HDFS (Hadoop Distributed File System) to access data stored in a distributed environment. • Uses MapReduce or Apache Spark in the background to process data in parallel. • Allows R scripts to be run on large datasets without moving them out of Hadoop. • Often works with tools like Hive and HBase to query and manage structured data within Hadoop.
Role of BigR: • Enables data analysts to use familiar R syntax to analyze big data. • Helps in statistical computing and creating data models over large datasets. • Supports exploratory data analysis (EDA) and predictive analytics. • Extends R's capabilities to work beyond memory limits of a single machine.