USN | | | | | | | | | | |

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 1 – OCT. 2024

Sub:	BIG DATA ANALYTICS			Sub Code:	21CS71	Branch:	CSE
Date:	15/10/2024	Duration:	90 mins	Max Marks: 50	Sem / Sec:	7/A,B,C	OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a) Define Big Data. Consider the usage examples of Big data for a Car company. Assume that company manufactures five models of cars, and each model is available in five colours and five shades.The Company collects inputs from customers and sales centres and inputs of component malfunctions from service centers for different models.The company also uses social media inputs. Explain 5Vs characteristics in this company's data.	5	CO1	L2

Solution :

Big Data refers to extremely large and complex datasets generated from various sources that cannot be easily processed using traditional data processing tools. These datasets often require advanced technologies, analytics, and techniques to capture, store, manage, and analyze for valuable insights.

For a car company that manufactures multiple models in various colors and shades and collects customer feedback, sales data, component malfunction data from service centers, and social media inputs, the data it handles exemplifies the **5Vs** characteristics of Big Data:

1. Volume:

- **Definition**: Volume refers to the massive amount of data generated and stored by an organization.
- **Application in the Company**: The car company generates a large volume of data from numerous sources, including customer feedback, sales center inputs, service center records, and social media. This high volume of data comes from the diversity of car models, colors, and shades offered, as well as from various locations, creating a huge dataset that requires significant storage and processing capacity.

2. Velocity:

- **Definition**: Velocity is the speed at which data is generated, collected, and processed.
- **Application in the Company**: Data is continuously generated from multiple streams, such as real-time sales data, live social media inputs, and instant service center reports on malfunctions. For example, customer reviews on social media may be posted anytime, requiring the company to capture and analyze feedback quickly to respond to potential issues or trends in customer preferences.

	3. Variety: • **Definition**: Variety refers to the diversity in data types and sources. • **Application in the Company**: The company’s data comes in various formats, such as structured data from sales centers (transactional sales records), unstructured data from social media comments (text), and semi-structured data from service reports (logs of malfunctions). This mix of data types from different sources, including text, images, and numerical data, adds complexity to data analysis. **4. Veracity:** • **Definition**: Veracity pertains to the accuracy and reliability of data. • **Application in the Company**: Data veracity is crucial for the car company because insights based on inaccurate data can lead to poor decisions. For example, false customer complaints or incorrect malfunction reports may mislead the company, so it must filter and validate data from multiple sources to ensure it reflects accurate and actionable insights. **5. Value:** • **Definition**: Value represents the actionable insights and benefits gained from analyzing Big Data. • **Application in the Company**: The primary purpose of collecting and analyzing this extensive data is to derive value, such as improving product quality, enhancing customer satisfaction, and boosting sales. Insights from customer feedback can inform future designs, while malfunction reports help improve service and reduce warranty costs, ultimately adding value to the company’s offerings and customer experience.			
	b) How do you classify data as structured, semi-structured and unstructured? Solution: **1. Structured Data** • **Definition**: Structured data is highly organized and follows a specific format or schema. It is often stored in databases where it can be easily accessed, queried, and analyzed. • **Characteristics**: o Stored in rows and columns (e.g., in relational databases like SQL). o Follows a defined schema, making it predictable and easy to process. o Usually quantitative data, like numbers, dates, and strings. • **Examples**: o Customer information in a CRM (Customer Relationship Management) system. o Sales transaction records with fields like customer ID,	5	CO1	L2

  - transaction date, and amount.
  - Inventory databases with structured attributes like product ID, name, and quantity.

2. Semi-Structured Data

- **Definition**: Semi-structured data has some organizational properties but lacks a strict schema. It may have tags or markers to separate elements and provide some structure, but it does not fit neatly into rows and columns.
- **Characteristics**:
  - Contains elements of both structured and unstructured data.
  - Can be stored in non-relational or NoSQL databases (e.g., JSON, XML).
  - Flexible in format, often with metadata or tags to provide structure.
- **Examples**:
  - JSON and XML files with nested elements and tags.
  - Email messages, where each message has a structured header (sender, recipient, timestamp) and an unstructured body.
  - Sensor data from IoT devices, which may have some consistent attributes but vary in their details and formats.

3. Unstructured Data

- **Definition**: Unstructured data lacks a predefined format or organization and does not fit neatly into database tables. It is often qualitative and requires advanced processing techniques, such as Natural Language Processing (NLP), to analyze.
- **Characteristics**:
  - No consistent format or structure.
  - Often text-heavy, multimedia, or qualitative in nature.
  - Requires specialized tools and techniques for processing and analysis (e.g., AI and machine learning algorithms).
- **Examples**:
  - Text documents (e.g., Word files, PDFs).
  - Images, audio files, and videos.
  - Social media posts, customer reviews, and emails where data is free-form and varies in content.

2	a) Explain uses of massive parallel processing and cluster computing in Big data Scenario. Solution: **Massive Parallel Processing (MPP)** and **Cluster Computing** are essential technologies for handling Big Data due to their ability to process large volumes of data quickly and efficiently. Here's how each technology is	5	CO1	L2

used in a Big Data scenario:

1. Massive Parallel Processing (MPP)

- **Definition**: MPP refers to the use of multiple processors to execute different parts of a data processing task simultaneously. Each processor operates independently, handling a segment of the data in parallel with other processors.
- **Uses in Big Data**:
    - **Data Analysis and Computation**: MPP allows for large datasets to be divided and processed concurrently, drastically reducing the time required for data analysis. For example, an MPP database could process billions of rows of data in a fraction of the time compared to a traditional system.
    - **High-Speed Query Processing**: MPP enables faster execution of complex queries in Big Data environments, such as business intelligence queries for real-time insights or customer behavior analysis.
    - **Scalability for Large Data Volumes**: MPP systems can easily scale to handle larger data volumes by adding more processors or nodes. This scalability is crucial in Big Data scenarios, where data volume continually grows.
    - **Examples**: Technologies like Amazon Redshift, Google BigQuery, and Teradata are MPP databases that can handle large-scale data analysis by distributing queries across multiple processors.

2. Cluster Computing

- **Definition**: Cluster computing involves connecting multiple computers (nodes) to work together as a single system. These nodes share tasks and resources, creating a powerful computing cluster.
- **Uses in Big Data**:
    - **Distributed Data Storage and Processing**: Cluster computing enables the storage and processing of large datasets across multiple machines, essential for Big Data. Technologies like Apache Hadoop and Apache Spark split data across nodes, which allows for distributed data storage and processing.
    - **Fault Tolerance and Reliability**: In cluster computing, if one node fails, the data and tasks are automatically redirected to other nodes, ensuring that operations can continue without interruption. This reliability is critical in Big Data environments where continuous data processing is required.
    - **Cost-Effective Scalability**: Cluster computing allows organizations to add inexpensive commodity hardware to expand the system as data grows, making it a cost-effective solution for Big Data processing.
    - **Examples**: Apache Hadoop’s HDFS (Hadoop Distributed File System) and MapReduce leverage cluster computing to process and store large data volumes, while Apache Spark enables high-speed data processing across clusters for tasks

	like data transformation and machine learning.			
	b) What is analytics scalability to Big Data? Also explain horizontal scalability and vertical scalability. Solution: **1. Horizontal Scalability (Scale-Out)** • **Definition**: Horizontal scalability, or scale-out, refers to expanding a system's capacity by adding more nodes (machines or servers) to the network. • **How it Works**: In a horizontally scalable system, data and tasks are distributed across multiple machines. When demand increases, additional machines are added to share the workload, allowing the system to handle more data and more users. • **Benefits**: o **Cost-Effectiveness**: It can be less expensive to add commodity servers than to invest in a high-powered single machine. o **Fault Tolerance**: Horizontal scalability allows systems to continue working even if one node fails, as tasks can be distributed to other nodes. o **Flexibility**: More machines can be added incrementally as needed, making it easier to adjust capacity based on demand. • **Examples**: Big Data frameworks like **Apache Hadoop** and **Apache Spark** are designed for horizontal scalability, enabling distributed data storage and processing across many nodes in a cluster. **2. Vertical Scalability (Scale-Up)** • **Definition**: Vertical scalability, or scale-up, involves increasing the capacity of a single machine by adding more resources such as CPU, RAM, and storage. • **How it Works**: In a vertically scalable system, performance improves by upgrading or adding more powerful components to a single machine rather than distributing tasks across multiple machines. • **Benefits**: o **Simplicity**: Vertical scalability requires less configuration and management than a distributed system. o **High Performance for Specific Tasks**: For certain types of tasks that are not easily distributed, a high-powered single machine can be more efficient. • **Limitations**: o **Cost**: Upgrading individual machines can become expensive, particularly as higher performance hardware becomes costly. o **Hardware Limitations**: There is a practical limit to how much you can upgrade a single machine, which can constrain scalability. • **Examples**: Traditional databases like **MySQL** or **Oracle** can be vertically scaled by adding more resources to a single server to	5	CO1	L2

improve performance.

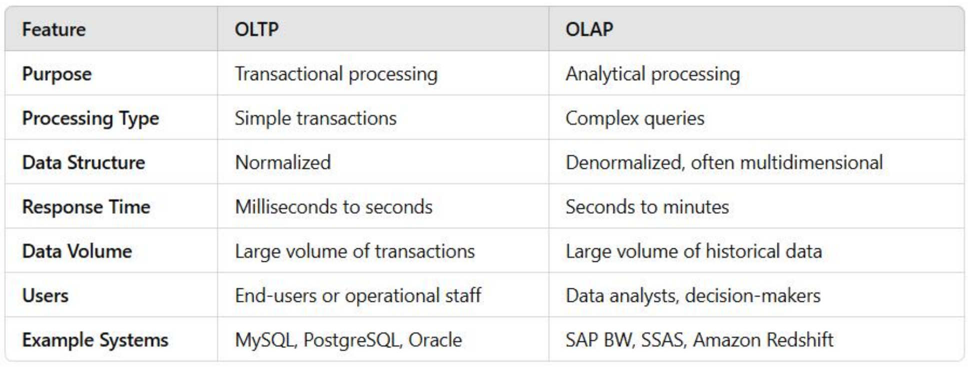
Summary

- **Horizontal scalability** is ideal for distributed computing environments, where data processing can be split across multiple nodes.
- **Vertical scalability** is often used in environments that benefit from single-machine processing, where adding resources to one server may yield sufficient performance improvement.

3 a) Differentiate i) OLTP Vs OLAP ii) SQL Vs NoSQL — 4 — CO1 — L2

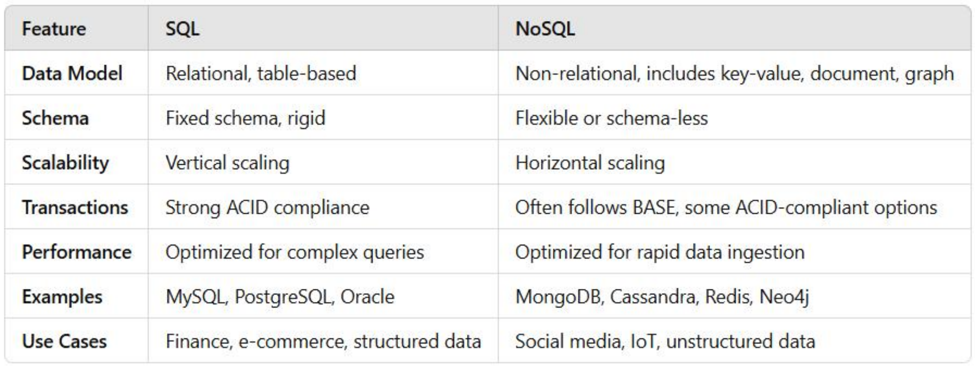
Solution : (any 2 points- 2 marks for each)

OLTP (Online Transaction Processing) and **OLAP (Online Analytical Processing)** are two types of data processing systems, each designed for distinct purposes in the data management ecosystem.



Feature	OLTP	OLAP
Purpose	Transactional processing	Analytical processing
Processing Type	Simple transactions	Complex queries
Data Structure	Normalized	Denormalized, often multidimensional
Response Time	Milliseconds to seconds	Seconds to minutes
Data Volume	Large volume of transactions	Large volume of historical data
Users	End-users or operational staff	Data analysts, decision-makers
Example Systems	MySQL, PostgreSQL, Oracle	SAP BW, SSAS, Amazon Redshift

ii)



Feature	SQL	NoSQL
Data Model	Relational, table-based	Non-relational, includes key-value, document, graph
Schema	Fixed schema, rigid	Flexible or schema-less
Scalability	Vertical scaling	Horizontal scaling
Transactions	Strong ACID compliance	Often follows BASE, some ACID-compliant options
Performance	Optimized for complex queries	Optimized for rapid data ingestion
Examples	MySQL, PostgreSQL, Oracle	MongoDB, Cassandra, Redis, Neo4j
Use Cases	Finance, e-commerce, structured data	Social media, IoT, unstructured data

b) Describe ways of usages of Big Data analytics in health care systems and medicine — 6 — CO1 — L2

Solution:

Big Data analytics is transforming healthcare and medicine by leveraging vast amounts of data to improve patient care, streamline operations, and advance medical research. Here are some key applications of Big Data analytics in healthcare:

1. Predictive Analytics for Patient Care

- **Usage**: Predictive models use historical patient data to forecast potential health risks and outcomes. For example, predictive analytics can help identify patients at high risk of chronic diseases like diabetes or cardiovascular diseases.

- **Impact**: Early intervention and preventive care can be applied to high-risk patients, reducing hospitalizations, improving outcomes, and lowering healthcare costs.

2. Personalized Medicine and Treatment Plans

- **Usage**: Big Data enables the analysis of genetic, environmental, and lifestyle data to customize treatment plans. For instance, genomic data can be analyzed to determine a patient's response to certain medications.
- **Impact**: Personalized treatments are more effective, reducing side effects and improving patient adherence and recovery times.

3. Improving Diagnosis Accuracy

- **Usage**: Machine learning models trained on large datasets of medical images and patient records assist in diagnosing conditions such as cancer, cardiovascular disease, and neurological disorders.
- **Impact**: Early and accurate diagnoses can lead to timely treatment and better patient outcomes, with reduced chances of misdiagnosis.

4. Operational Efficiency in Healthcare Facilities

- **Usage**: Big Data analytics helps optimize resource allocation in hospitals, such as managing staff schedules, predicting patient admission rates, and reducing wait times.
- **Impact**: Improved operational efficiency enhances patient experience, reduces healthcare costs, and ensures better use of medical resources.

5. Remote Monitoring and Telemedicine

- **Usage**: Wearable devices and remote monitoring tools generate massive amounts of real-time patient data. Big Data analytics processes this data to monitor vital signs, detect anomalies, and provide insights to physicians.
- **Impact**: Remote monitoring improves chronic disease management, enables timely interventions, and makes healthcare accessible to people in remote or underserved areas.

6. Drug Discovery and Development

- **Usage**: Big Data is used to analyze biological data, clinical trial results, and existing pharmaceutical data to identify potential drug candidates. It also aids in repurposing existing drugs for new treatments.
- **Impact**: Reduces the time and cost involved in drug development, accelerating the discovery of new treatments for diseases and improving patient outcomes.

4	Define Big Data architecture. Draw five layers in architecture design and explain functions in each layer.	10	CO1	L1

Solution :

Big Data Architecture is a blueprint for managing and processing vast amounts of data by integrating various components and tools into a streamlined pipeline. It defines the framework for collecting, storing, processing, analyzing, and visualizing data in a scalable and efficient manner, often employing distributed computing technologies.

The typical Big Data architecture consists of **five layers**:

1. Data Ingestion Layer

- **Function**: This layer is responsible for collecting and importing data from multiple data sources into the system. It gathers structured, semi-structured, and unstructured data from diverse sources like sensors, social media feeds, databases, and transactional systems.
- **Technologies**: Kafka, Apache Flume, Apache NiFi, Sqoop, and real-time streaming tools.
- **Purpose**: Ensures that data flows efficiently into the architecture from various sources, often at high velocity.

2. Data Storage Layer

- **Function**: This layer stores the raw ingested data in a scalable and reliable format. The data may be stored as-is (raw data) or undergo minor processing to prepare it for the next layers. Data storage solutions are chosen based on factors like volume, variety, and velocity of data.
- **Technologies**: HDFS, Amazon S3, NoSQL databases (e.g., Cassandra, MongoDB), or cloud-based storage solutions.
- **Purpose**: Provides storage that can scale horizontally and handle large amounts of diverse data while ensuring data integrity and reliability.

3. Data Processing Layer

- **Function**: This is the core of Big Data architecture, where raw data is transformed, cleansed, enriched, and prepared for analysis. It handles batch processing (for large datasets) and real-time processing (for immediate insights). It prepares data for analytics, aggregating, and summarizing it to make it meaningful.
- **Technologies**: Apache Spark, Hadoop MapReduce, Apache Storm, Flink.
- **Purpose**: Enables efficient processing of data at scale, ensuring the data is in a usable format for analytical purposes.

4. Data Analytics Layer

- **Function**: This layer is responsible for analyzing the processed data. It uses advanced analytics, including machine learning, predictive

analytics, and data mining, to extract actionable insights and identify trends.
- **Technologies**: R, Python, TensorFlow, machine learning frameworks, and specialized analytics platforms.
- **Purpose**: Provides meaningful insights, helps in making data-driven decisions, and enables predictive analysis.

5. Data Visualization Layer

- **Function**: The final layer presents the analyzed data to end-users in an understandable format. It allows users to interact with data through visualizations, dashboards, and reports, often in real time.
- **Technologies**: Tableau, Power BI, QlikView, D3.js, Grafana.
- **Purpose**: Helps stakeholders and decision-makers understand the data insights through interactive and user-friendly visual representations.

Diagram

The architecture can be visualized with the five layers stacked from bottom to top:

```
|----------------------------|
|      Data Visualization       | <-- Visualize insights,
dashboards, reports
|----------------------------|
|         Data Analytics        | <-- Analyze data, machine
learning, predictive models
|----------------------------|
|         Data Processing       | <-- Transform, clean, and
prepare data
|----------------------------|
|        Data Storage           | <-- Store raw and processed
data at scale
|----------------------------|
|      Data Ingestion          | <-- Collect data from various
sources
|----------------------------|
```

Each layer works together to enable efficient handling, processing, and presentation of Big Data, allowing organizations to derive valuable insights and make informed decisions.

5	List Hadoop Core Components and explain with appropriate diagram.	10	CO2	L1

Solution:

Hadoop is a powerful framework for processing and storing large datasets using distributed computing. The Hadoop ecosystem is built on several core components that work together to provide scalable, fault-tolerant data processing and storage solutions. The **four main core components** of Hadoop are:

1. **Hadoop Distributed File System (HDFS)**
2. **MapReduce**
3. **YARN (Yet Another Resource Negotiator)**
4. **Hadoop Common**

Below is an overview of each component with an appropriate diagram to illustrate their interactions.

1. Hadoop Distributed File System (HDFS)

- **Function**: HDFS is Hadoop's primary storage system, designed to store large volumes of data across multiple machines in a cluster. It splits files into large blocks (usually 128 MB or 256 MB) and distributes them across the cluster nodes, with each block replicated on multiple nodes to ensure fault tolerance.
- **Key Components**:
  - **NameNode**: Manages the metadata and directory structure of the file system, overseeing which nodes hold data blocks.
  - **DataNode**: Stores the actual data blocks and periodically reports back to the NameNode.
- **Purpose**: HDFS provides reliable, scalable, and distributed storage, making it ideal for handling big data.

2. MapReduce

- **Function**: MapReduce is Hadoop's processing engine that allows parallel processing of large data sets across the Hadoop cluster. It processes data in two phases:
  - **Map Phase**: Divides tasks and processes each data block to generate intermediate key-value pairs.
  - **Reduce Phase**: Aggregates and summarizes the results from the map phase to produce the final output.
- **Purpose**: MapReduce enables distributed data processing, ensuring efficient and fault-tolerant computation over massive datasets.

3. YARN (Yet Another Resource Negotiator)

- **Function**: YARN is Hadoop's resource management layer, which allocates system resources like CPU and memory to various applications running in the Hadoop cluster. It manages and schedules tasks for MapReduce and other frameworks (like Spark) that run on Hadoop.
- **Key Components**:
  - **ResourceManager**: Manages the allocation of resources across applications.
  - **NodeManager**: Manages resources at the node level, reporting back to the ResourceManager.

- **Purpose**: YARN improves Hadoop’s scalability by decoupling resource management and job scheduling, enabling more efficient use of cluster resources.

4. Hadoop Common

- **Function**: Hadoop Common includes shared libraries and utilities needed by other Hadoop components. It provides essential services and configuration parameters for other modules in the Hadoop ecosystem.
- **Purpose**: Ensures that all other Hadoop components can work seamlessly together, offering a foundation for other tools in the Hadoop ecosystem.

Diagram

Below is a simplified diagram representing the interaction among Hadoop core components:

```
  +--------------------------+
  |       Hadoop Common      |    <--- Core utilities and
           libraries
  +--------------------------+
             |
+------------+------------+
|                         |
|                         |
v                         v
+----------+       +-------------------+
|  HDFS    |       |       YARN        |  <--- Distributed
storage and resource management
+----------+       +-------------------+
| NameNode|        | ResourceManager   |
| DataNode|        | NodeManager       |
+----------+       +-------------------+
             |
      +------+------+
      |  MapReduce  |          <--- Data processing engine
      +-------------+
      |  Map Phase  |
      | Reduce Phase|
      +-------------+
```

Summary of Component Interactions

- **HDFS** provides distributed storage where data is stored as blocks across nodes.
- **MapReduce** leverages the data stored in HDFS, using the distributed nature of HDFS to perform parallel processing.
- **YARN** allocates and manages resources within the Hadoop cluster

	to run processing jobs like MapReduce. • **Hadoop Common** supports the entire ecosystem, offering essential libraries and services for effective operation across all components. Together, these components form a comprehensive framework for Big Data storage, processing, and management, making Hadoop a powerful tool for large-scale data handling.			
6	a) How does MapReduce function as a programming model for distributed computing? Solution: MapReduce is a programming model and processing framework designed to efficiently process and generate large datasets across distributed clusters of computers. It simplifies the complexities of parallel computing by providing a high-level abstraction for developers. Here’s how MapReduce functions as a programming model for distributed computing: **1. Basic Concept of MapReduce** The MapReduce model is based on two primary functions: **Map** and **Reduce**. It operates in a distributed computing environment, breaking down tasks into smaller sub-tasks that can be processed in parallel. **Map Function** • **Input**: The Map function takes a set of input key-value pairs. • **Processing**: It processes each input pair to produce a set of intermediate key-value pairs. • **Output**: The output is a list of key-value pairs, which are often emitted to a temporary storage area. **Reduce Function** • **Input**: The Reduce function receives the intermediate key-value pairs generated by the Map function. • **Processing**: It aggregates and processes these pairs based on their keys. • **Output**: The output is a smaller set of key-value pairs that represent the final result of the computation. **2. Workflow of MapReduce** The MapReduce framework follows a systematic workflow, which includes the following steps: **1. Input Splitting** • The input data is divided into smaller chunks called splits. Each split is processed independently, enabling parallelism. **2. Mapping**	5	CO2	L2

- Each split is processed by the Map function, which outputs intermediate key-value pairs. This function is executed in parallel across multiple nodes in the cluster.

3. Shuffling and Sorting

- The intermediate key-value pairs produced by the Map function are shuffled and sorted by key. This process groups all values associated with the same key together, preparing them for the Reduce phase.

4. Reducing

- The Reduce function processes the grouped intermediate key-value pairs. It performs aggregation or other operations to produce the final output for each key.

5. Output

- The final results from the Reduce phase are written to an output file or database for further analysis or storage.

3. Key Features of MapReduce in Distributed Computing

MapReduce provides several features that facilitate distributed computing:

Scalability: The model can efficiently handle large datasets by distributing the workload across many nodes in a cluster. As the amount of data grows, additional nodes can be added to the cluster to accommodate increased processing demands.

Fault Tolerance: If a node fails during processing, the MapReduce framework can automatically reassign the failed task to another node. This ensures that the overall computation continues without significant interruption.

Data Locality: MapReduce optimizes performance by moving computation closer to where the data is stored. This reduces data transfer times across the network and enhances overall efficiency.

Simplified Programming Model: Developers can focus on writing the Map and Reduce functions without needing to manage the underlying complexity of distributed systems, such as load balancing, resource management, and fault tolerance.

4. Example Use Case

An example use case for MapReduce is counting the occurrences of words in a large collection of documents.

-

Map Function: The Map function reads each document and emits a

	key-value pair for each word (key) along with the count (value) of 1.
	•
	•
	Intermediate Output: For the input "Hello World," the output would be:
	`(Hello, 1)`
	`(World, 1)`
	Reduce Function: The Reduce function takes these key-value pairs and aggregates the counts for each word.
	Final Output: The final output would provide the total count of occurrences for each word across all documents.			

	b) List the functions of YARN Solution : (any 5 functions- 5 marks) YARN (Yet Another Resource Negotiator) is a key component of the Hadoop ecosystem that acts as the resource management layer for Hadoop. It provides a framework for job scheduling and cluster resource management, enabling multiple data processing engines to run and share the same resources in a Hadoop cluster. Here are the main functions of YARN: **1. Resource Management** • **Resource Allocation**: YARN allocates cluster resources (CPU, memory, disk) to various applications running on the Hadoop cluster. It ensures that resources are utilized effectively and fairly among different jobs. • **Resource Monitoring**: YARN continuously monitors the resources used by different applications and nodes in the cluster, maintaining an overview of resource utilization. **2. Job Scheduling** • **Application Scheduling**: YARN schedules and manages jobs submitted to the cluster. It determines when and how applications should be run based on resource availability. • **Fair Scheduling**: YARN can implement scheduling policies such as fair scheduling, which ensures that all applications get an equitable share of resources based on their requirements. **3. Fault Tolerance** • **Node Failure Handling**: YARN is designed to handle node failures gracefully. If a node fails, YARN can restart the failed application on another node, ensuring job completion without data loss. • **Rescheduling**: In the event of a task failure, YARN reschedules tasks on available nodes, maintaining the overall reliability of the system.	5	CO2	L1

4. Multi-tenancy

- **Support for Multiple Frameworks**: YARN allows various processing frameworks (such as MapReduce, Spark, and Tez) to run on the same Hadoop cluster simultaneously. This multi-tenancy enables efficient use of cluster resources and facilitates different data processing workloads.

5. Dynamic Resource Allocation

- **Resource Adjustment**: YARN supports dynamic allocation of resources during runtime, allowing applications to request additional resources as needed. This adaptability helps optimize performance based on changing workload demands.

6. Application Lifecycles Management

- **Application Submission**: YARN handles the submission and tracking of applications in the cluster. It provides APIs for clients to submit jobs, check status, and retrieve outputs.
- **Application Coordination**: YARN manages the execution of applications by coordinating between different components and handling the lifecycle of tasks within an application.

7. Security and Access Control

- **Authentication and Authorization**: YARN provides security features to authenticate users and authorize access to resources. This ensures that only authorized users can submit applications or access certain resources.

8. Metrics and Monitoring

- **Performance Metrics**: YARN collects and exposes metrics related to resource usage, application performance, and system health. These metrics can be monitored for performance tuning and troubleshooting.

All the best

CI **CCI** **HOD**

CO PO Mapping

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Understand fundamentals of Big Data analytics.	**1**	3	0	2	2	3	0	0	0	0	0	0	0	0	0	0	3
CO2	Investigate Hadoop framework and Hadoop Distributed File system.	**1,2**	2	3	2	3	3	0	0	0	0	0	0	0	0	2	0	3
CO3	Illustrate the concepts of No SQL using MongoDB and Cassandra for Big Data.	**3**	2	2	3	3	3	0	0	0	0	0	0	0	0	2	0	3
CO4	Demonstrate the Map Reduce programming model to process the big data along with Hadoop tools.	**2,3,4**	3	3	3	2	3	0	0	0	0	0	0	0	0	2	0	3
CO5	Use Machine Learning algorithms for real world big data.	**5**	3	3	3	3	3	0	0	0	0	0	0	0	0	2	0	3
CO6	Analyze web contents and Social Networks to provide analytics with relevant visualization tools.	**5**	3	3	2	2	3	0	0	0	0	0	0	0	0	2	0	3

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.

L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				