Model Question Paper-1 with effect from 2021 **(CBCS Scheme)**

USN

7 th Semester B.E. Degree Examination

BIG DATA ANALYTICS

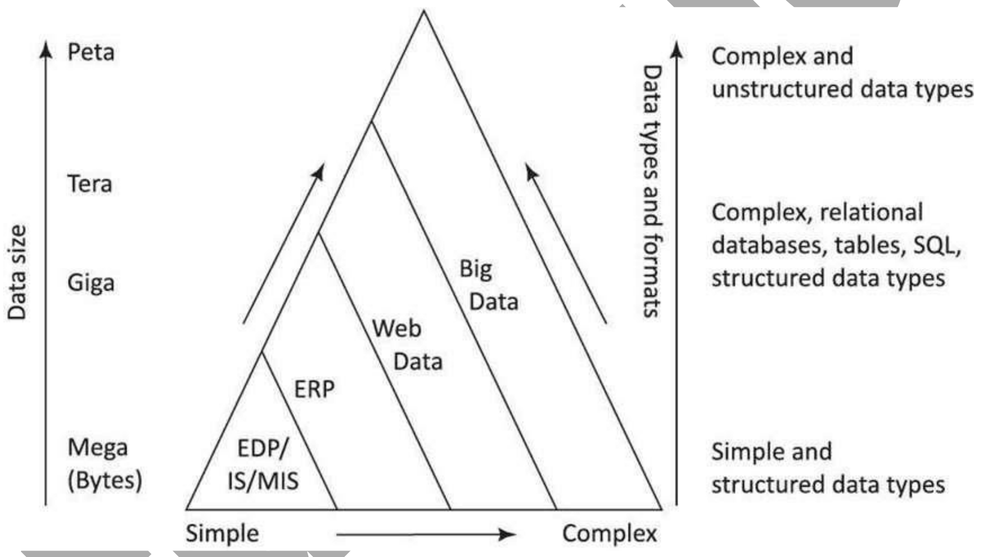
TIME: 03 Hours **Max. Marks: 100**

Note: 01. Answer any **FIVE** full questions choosing at least **ONE** question from each **MODULE**

THESE ANSWERS FROM NOTES

		Module -1 **DOWNLOAD**	**COs**	**Marks**
Q.01	a	What is Big Data? Explain evolution of big data & characteristics.	CO1	10

Big Data refers to high-volume, high-velocity, and high-variety information assets that necessitate new processing methods for improved decision-making and insight discovery. The term "Big Data" describes datasets that are so large or complex that traditional data processing applications are inadequate to handle them. The evolution of Big Data has been significantly influenced by advancements in technology, leading to the generation and storage of massive amounts of data, transitioning from megabytes to petabytes.


Data size
Peta
Tera
Giga
Mega (Bytes)
EDP/ IS/MIS
ERP
Web Data
Big Data
Simple
Complex
Data types and formats
Complex and unstructured data types
Complex, relational databases, tables, SQL, structured data types
Simple and structured data types

The evolution of Big Data can be understood through its key characteristics, often summarized as the "4Vs":

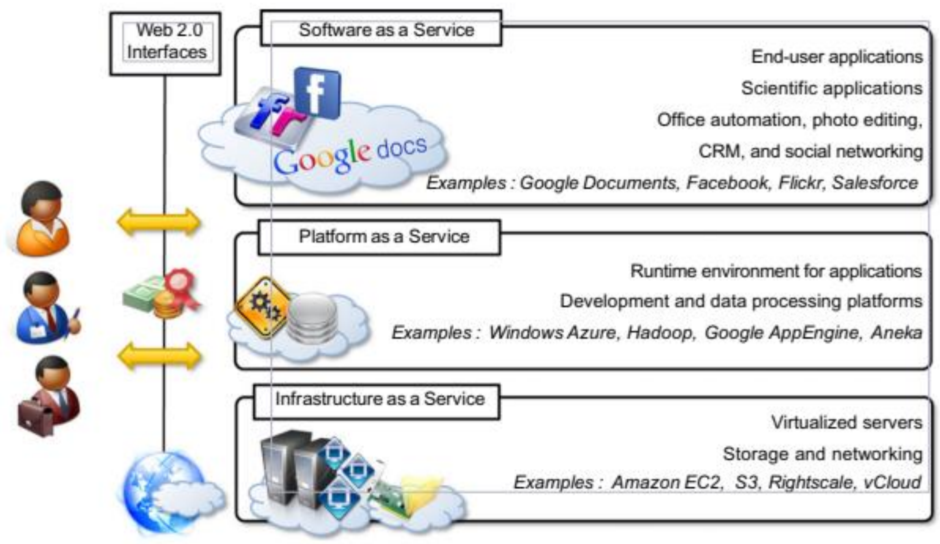
1. Volume : This characteristic highlights the enormous size of data generated from various applications. As organizations collect vast amounts of information, managing and storing this data becomes a significant challenge.

2. Velocity : This refers to the speed at which data is generated and processed. In today's fast-paced digital landscape, the ability to quickly analyze and act on data is crucial for businesses to maintain a competitive edge.

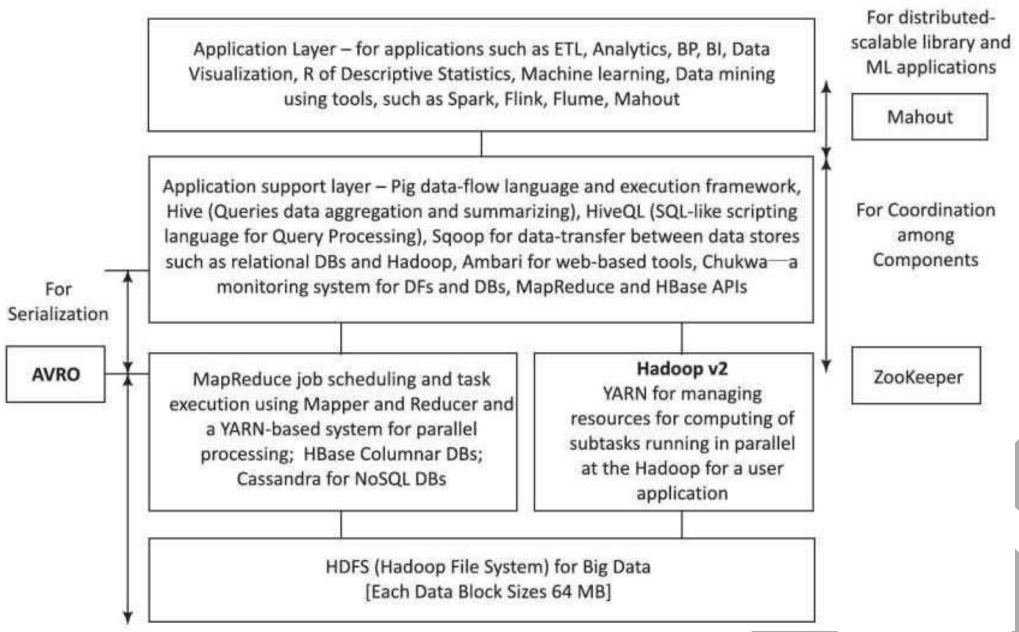
3. Variety : Big Data encompasses a wide range of data types and formats, including structured, semi-structured, and unstructured data. This diversity arises from multiple sources, such as social media, sensors, and transaction records, adding complexity to data management and analysis.

4. Veracity : This characteristic addresses the quality and accuracy of the data. With the vast amounts of data being generated, ensuring that the data is reliable and valid is essential for accurate analysis and informed decision-

		making.		
	b	Explain the following terms. i. Scalability & Parallel Processing ii. Grid & Cluster Computing. i. Scalability & Parallel Processing Scalability refers to the capability of a system to handle an increasing workload by adding resources. There are two main types of scalability: - Vertical Scalability (Scaling Up) : This involves adding more power (CPU, RAM) to an existing machine. For example, if you have a server that is running out of resources, you can upgrade it to a more powerful server to improve its performance. - Horizontal Scalability (Scaling Out) : This means adding more machines to your pool of resources. Instead of upgrading a single machine, you distribute the workload across multiple machines. This is particularly useful for handling large datasets, as it allows for parallel processing. Parallel Processing is a method where multiple computations or processes are carried out simultaneously. In this approach, a computational problem is divided into smaller sub-tasks that can be processed at the same time across multiple CPUs or computers. This significantly reduces the total time required for processing compared to using a single compute resource. ii. Grid & Cluster Computing Grid Computing is a distributed computing model where a network of computers from different locations collaborates to achieve a common goal. These computers, which can be spread across various geographical locations, work together to process large datasets. Grid computing is particularly effective for data - intensive tasks, as it allows for flexible and secure resource sharing among users. However, it can face challenges, such as a single point of failure if one of the participating nodes underperforms. Cluster Computing , on the other hand, involves a group of computers connected by a network that work together to perform a specific task. Clusters are primarily used for load balancing, where processes are shifted between nodes to maintain an even load across the group. This setup is often used in environments like Hadoop, which utilizes similar methods for processing large datasets efficiently.	CO1	10
		OR		
Q.02	a	What is Cloud Computing? Explain different services of Cloud. Cloud Computing is a type of Internet-based computing that provides shared processing resources and data to computers and other devices on demand. It allows users to access and store data and applications over the Internet instead of on local servers or personal computers. This model offers flexibility, scalability, and cost-effectiveness, making it a popular choice for businesses and individuals alike. Cloud services can be classified into three fundamental types:	CO1	10

		 Web 2.0 Interfaces Software as a Service End-user applications Scientific applications Office automation, photo editing, CRM, and social networking Examples : Google Documents, Facebook, Flickr, Salesforce Google docs Platform as a Service Runtime environment for applications Development and data processing platforms Examples : Windows Azure, Hadoop, Google AppEngine, Aneka Infrastructure as a Service Virtualized servers Storage and networking Examples : Amazon EC2, S3, Rightscale, vCloud 1. Infrastructure as a Service (IaaS) : This service provides access to essential resources such as virtual servers, storage, and networking. Users can rent IT infrastructure on a pay-as-you-go basis. Examples include Amazon Web Services (AWS) Elastic Compute Cloud (EC2) and Tata CloudStack. 2. Platform as a Service (PaaS) : PaaS offers a runtime environment for developers to build, test, and deploy applications without worrying about the underlying infrastructure. It supports various development tools and services. Examples include IBM BigInsight and Microsoft Azure HD Insights. 3. Software as a Service (SaaS) : This service delivers software applications over the Internet, allowing users to access them via a web browser without needing to install or maintain them locally. Examples include GoogleSQL, IBM BigSQL, and Oracle Big Data SQL.		
	b	Explain any two Big Data different Applications. Big Data has a wide range of applications across various industries, but let me highlight two significant ones: Marketing and Sales and Fraud Detection . 1. Marketing and Sales : Big Data plays a crucial role in enhancing marketing strategies and sales processes. Companies leverage Big Data analytics to gain insights into customer behavior, preferences, and trends. For instance, businesses can analyze data to determine the most effective content at each stage of the sales cycle, which helps in tailoring marketing campaigns to specific customer needs. Additionally, Big Data enables companies to invest in improving their Customer Relationship Management (CRM) systems, which can lead to increased Customer Lifetime Value (CLTV) and reduced Customer Acquisition Cost (CAC). By utilizing contextual marketing, businesses can send targeted advertisements based on users' recent browsing patterns, thereby increasing the chances of conversion. 2. Fraud Detection : In the realm of financial services and e-commerce, Big Data analytics is instrumental in detecting and preventing fraud. By integrating multiple data sources and analyzing them, companies can gain greater insights into transaction patterns and identify anomalies that may indicate fraudulent activity. For example, advanced analytics can help in generating structured reports and visualizations that highlight unusual behaviors. Moreover, the high volume of data allows for faster detection of threats and the ability to predict potential frauds by utilizing publicly available information. This proactive approach not only helps in safeguarding assets but also enhances overall business intelligence.	CO1	05

		3. Healthcare : In the healthcare industry, Big Data analytics is used to improve patient outcomes and streamline operations. By analyzing vast amounts of patient data, including electronic health records, wearable device data, and genomic information, healthcare providers can identify trends, predict disease outbreaks, and personalize treatment plans. For instance, wearable devices can track patient health metrics in real-time, allowing for better risk profiling and proactive healthcare management. 4. Telecommunications : Telecom companies use Big Data to improve customer service and network management. By analyzing call data records, customer complaints, and usage patterns, they can identify areas for service improvement and optimize network performance. Predictive analytics can also help in reducing churn by identifying at-risk customers and implementing retention strategies.		
	c	How does Berkeley data analytics stack helps in analytics take? Big Data refers to high-volume, high-velocity, and high-variety information assets that necessitate new processing methods for improved decision-making and insight discovery. The term "Big Data" describes datasets that are so large or complex that traditional data processing applications are inadequate to handle them. The evolution of Big Data has been significantly influenced by advancements in technology, leading to the generation and storage of massive amounts of data, transitioning from megabytes to petabytes. The evolution of Big Data can be understood through its key characteristics, often summarized as the "4Vs": 1. Volume : This characteristic highlights the enormous size of data generated from various applications. As organizations collect vast amounts of information, managing and storing this data becomes a significant challenge. 2. Velocity : This refers to the speed at which data is generated and processed. In today's fast-paced digital landscape, the ability to quickly analyze and act on data is crucial for businesses to maintain a competitive edge. 3. Variety : Big Data encompasses a wide range of data types and formats, including structured, semi-structured, and unstructured data. This diversity arises from multiple sources, such as social media, sensors, and transaction records, adding complexity to data management and analysis. 4. Veracity : This characteristic addresses the quality and accuracy of the data. With the vast amounts of data being generated, ensuring that the data is reliable and valid is essential for accurate analysis and informed decision- making.	CO1	05
		Module-2 **DOWNLOAD**		
Q. 03	a	What is Hadoop? Explain Hadoop eco-system with neat diagram Hadoop is an open-source framework developed by the Apache Software Foundation that enables the distributed processing of large datasets across clusters of computers. It is designed to scale from a single server to thousands of machines, each providing local computation and storage. The core components of Hadoop include:	CO2	08


Application Layer – for applications such as ETL, Analytics, BP, BI, Data Visualization, R of Descriptive Statistics, Machine learning, Data mining using tools, such as Spark, Flink, Flume, Mahout
For distributed-scalable library and ML applications
Mahout
Application support layer – Pig data-flow language and execution framework, Hive (Queries data aggregation and summarizing), HiveQL (SQL-like scripting language for Query Processing), Sqoop for data-transfer between data stores such as relational DBs and Hadoop, Ambari for web-based tools, Chukwa—a monitoring system for DFs and DBs, MapReduce and HBase APIs
For Coordination among Components
For Serialization
AVRO
MapReduce job scheduling and task execution using Mapper and Reducer and a YARN-based system for parallel processing; HBase Columnar DBs; Cassandra for NoSQL DBs
Hadoop v2
YARN for managing resources for computing of subtasks running in parallel at the Hadoop for a user application
ZooKeeper
HDFS (Hadoop File System) for Big Data
[Each Data Block Sizes 64 MB]

1. Hadoop Common : This module contains the libraries and utilities required by other Hadoop modules, including components for the distributed file system and general input/output operations.

2. Hadoop Distributed File System (HDFS) : A Java-based distributed file system that stores all types of data across multiple disks in a cluster. It is designed to handle large files and provides high throughput access to application data.

3. MapReduce : A programming model for processing large datasets in parallel across a Hadoop cluster. It consists of two main functions: the Mapper, which processes input data and produces intermediate key-value pairs, and the Reducer, which takes these intermediate pairs and aggregates them into a final output.

4. YARN (Yet Another Resource Negotiator) : This is the resource management layer of Hadoop that manages and schedules resources across the cluster, allowing multiple data processing engines to run and share resources efficiently.

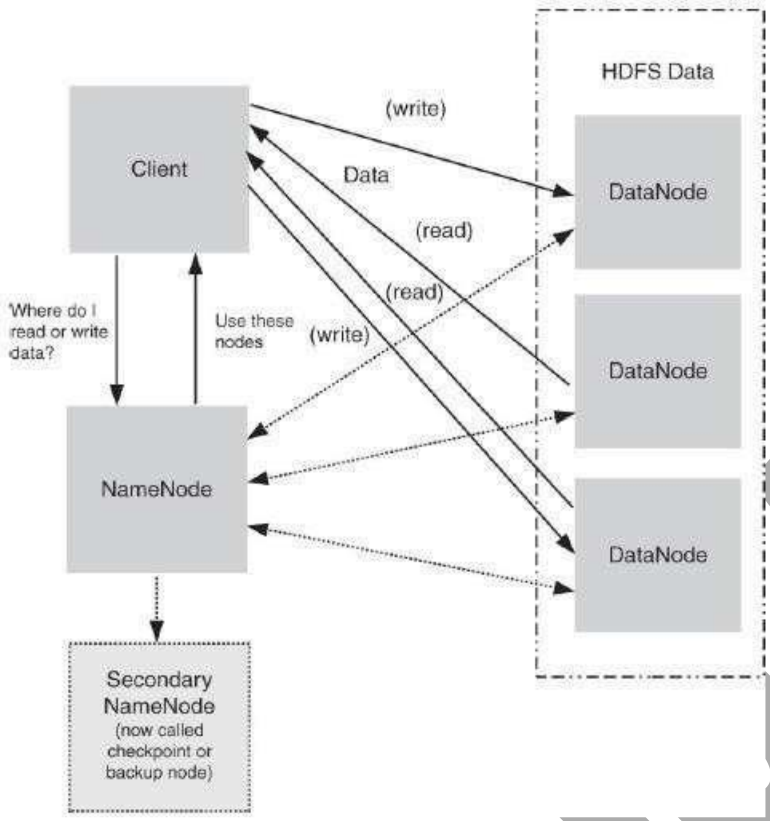
In addition to these core components, the Hadoop ecosystem includes various tools that enhance its functionality:

- Apache Pig : A high-level platform for creating programs that run on Hadoop, using a language called Pig Latin for data transformation.
- Apache Hive : A data warehouse infrastructure that provides data summarization and ad-hoc querying using a SQL-like language called HiveQL.
- Apache Oozie : A workflow scheduler system that manages Hadoop jobs and allows users to define complex workflows.
- Apache HBase : A distributed, scalable, NoSQL database that runs on top of HDFS and provides real-time read/write access to large datasets.

.

	b	Explain with neat diagram HDFS Components.	CO2	08

HDFS, or Hadoop Distributed File System, is a crucial component of the Hadoop framework designed for storing and managing large datasets across distributed clusters. Here are the key components of HDFS:



1. NameNode : This is the master server that manages the metadata of the file system. It keeps track of the file system namespace, which includes operations like opening, closing, and renaming files and directories. The NameNode also determines how data blocks are mapped to DataNodes and handles any DataNode failures. Importantly, the NameNode does not store any actual data; it only maintains the metadata.

2. DataNodes : These are the slave nodes in the HDFS architecture. DataNodes are responsible for storing the actual data blocks and serving read and write requests from clients. Each DataNode manages the storage of data blocks and periodically sends heartbeat signals to the NameNode to confirm its status.

3. SecondaryNameNode : This component is not a failover node but performs periodic checkpoints of the NameNode's metadata. It helps in reducing the load on the NameNode by merging the namespace image with the edit log, thus ensuring that the NameNode can recover quickly in case of a failure.

4. HDFS Blocks : HDFS divides files into large blocks, typically 64MB or 128 MB in size. This design is optimized for high-throughput access to large datasets, making it efficient for streaming data rather than random access.

5. Replication : HDFS ensures data reliability through replication. By default, each data block is stored on at least three different DataNodes. This redundancy means that even if one or more DataNodes fail, the data remains accessible.

6. Write-Once, Read-Many Model : HDFS is designed for a write-once, read-many access pattern. This means that once a file is written, it cannot be modified, but it can be read multiple times. New data can only be appended to the end of the file.

7. Data Locality : HDFS is designed to move computation closer to where the data is stored, rather than moving large amounts of data across the network. This approach enhances performance and reduces network congestion.

	c	Write short note on Apache hive. Apache Hive is a powerful data warehouse infrastructure built on top of the	CO2	04

Hadoop framework, designed to facilitate data summarization, ad hoc queries, and the analysis of large datasets using a SQL-like language known as HiveQL. It has become the de facto standard for executing interactive SQL queries over massive amounts of data stored in Hadoop.
Some of the essential features of Hive include:

1. ETL Tools : Hive provides tools that simplify the extraction, transformation , and loading (ETL) of data.
2. Data Structuring : It imposes structure on various data formats, making it easier to manage and query.
3. Data Access : Users can access files stored directly in HDFS or in other data storage systems like HBase.
4. Query Execution : Hive executes queries using MapReduce or Tez, which is an optimized version of MapReduce.

To use Hive, a user with access to HDFS can run Hive queries by simply entering the `hive` command in the command line. If Hive starts correctly, the user will see a `hive>` prompt, indicating that they can begin executing queries.

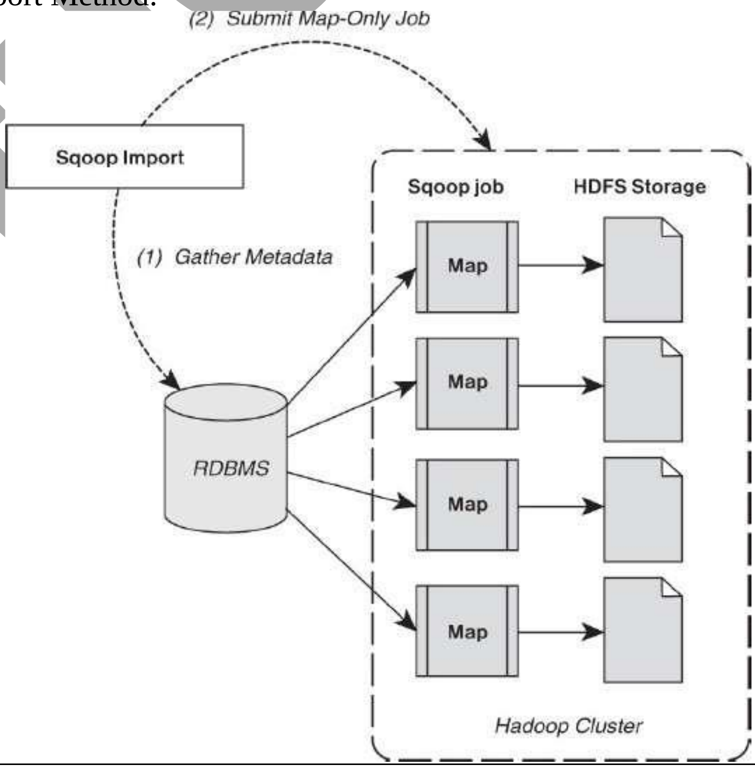
For example, to create a table in Hive, a user would enter:
```sql
hive> CREATE TABLE pokes (foo INT, bar STRING);
```
To verify the table creation, they can run:
```sql
hive> SHOW TABLES;
```
And to drop the table, the command would be:
```sql
hive> DROP TABLE pokes;
```

OR

Q.04	a	Explain Apache Sqoop Import and Export methods.	CO2	08

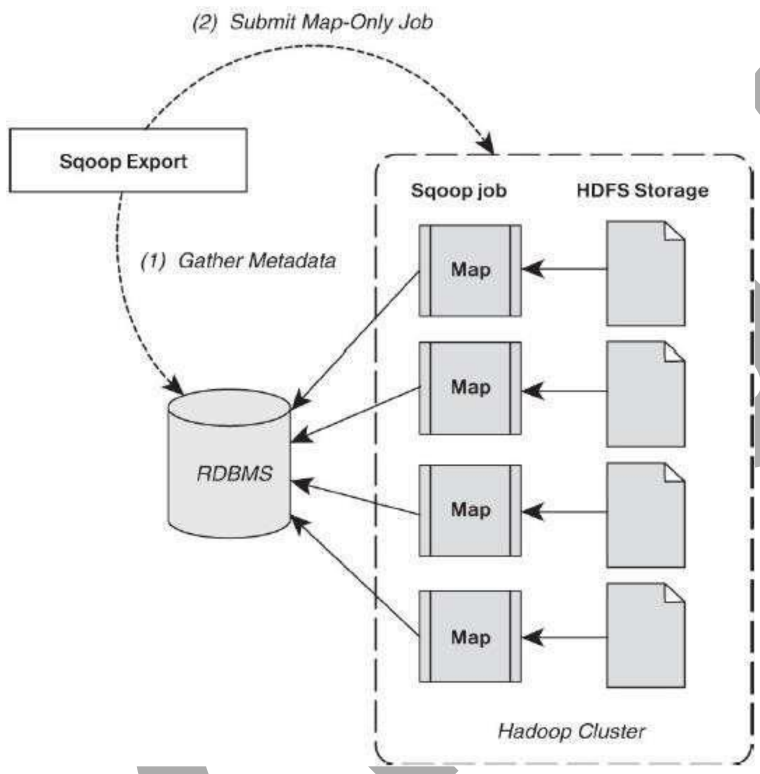
Apache Sqoop is a powerful tool designed for transferring data between Hadoop and relational databases. It facilitates both the import of data from relational database management systems (RDBMS) into the Hadoop Distributed File System (HDFS) and the export of data from HDFS back into RDBMS. Let's break down the import and export methods in detail:

Sqoop Import Method:



The import process in Sqoop is executed in two main steps:
1. Metadata Examination : Sqoop first examines the database to gather the necessary metadata about the data that needs to be imported. This includes information about the structure of the tables, data types, and other relevant details .
2. Data Transfer : After gathering the metadata, Sqoop initiates a Map-only Hadoop job. This job is responsible for transferring the actual data from the RDBMS to HDFS using the metadata collected in the first step. The data is typically imported in a format where fields are comma-delimited, and records are separated by new lines.

Sqoop Export Method:



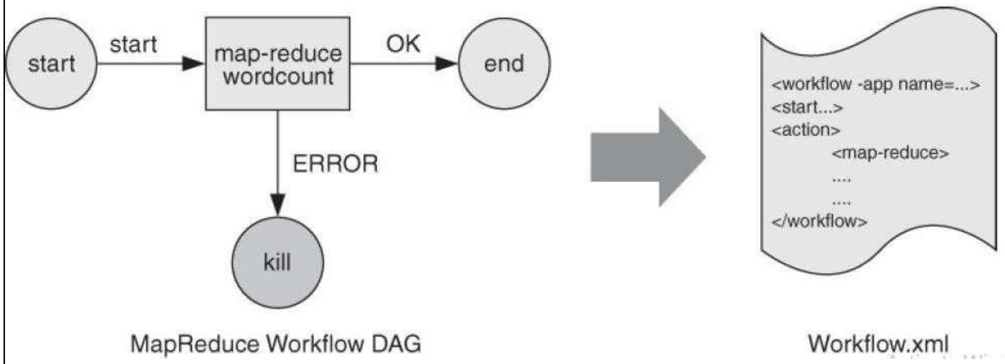
The export process is quite similar to the import method and also consists of two steps:

1. Metadata Examination : Just like in the import process, Sqoop examines the database for metadata before exporting data. This ensures that the data being exported matches the structure of the target database.

2. Data Writing : Sqoop then executes a Map-only Hadoop job to write the data back to the RDBMS. During this process, Sqoop divides the input data set into splits and uses individual map tasks to push these splits to the database. This parallel processing allows for efficient and fast data transfer.

Key Features:
- Parallel Processing : Sqoop exploits the MapReduce framework to perform both import and export operations, allowing for parallel processing of sub-tasks, which significantly speeds up the data transfer.
- Fault Tolerance : Sqoop provisions for fault tolerance, ensuring that data transfer can recover from failures without losing data.
- Command Line Interface : Users can interact with Sqoop through a command line interface, and it can also be accessed using Java APIs, providing flexibility in how it is used.

	b	Explain Apache Oozie with neat diagram.	CO2	07



Apache Oozie is an open-source workflow scheduler system specifically designed to manage and coordinate multiple related Apache Hadoop jobs. It plays a crucial role in the Hadoop ecosystem by allowing users to define complex workflows where the output of one job can serve as the input for another. This is particularly useful in big data analysis, where tasks often need to be executed in a specific sequence.

Oozie represents workflows as Directed Acyclic Graphs (DAGs), which are essentially graphs that do not contain any directed loops. This structure allows for clear representation of job dependencies and execution order. There are three main types of jobs that Oozie supports:

1. Workflow Jobs : These define a sequence of actions that must be executed in a specific order, with control dependencies ensuring that one action cannot start until the previous one has completed.

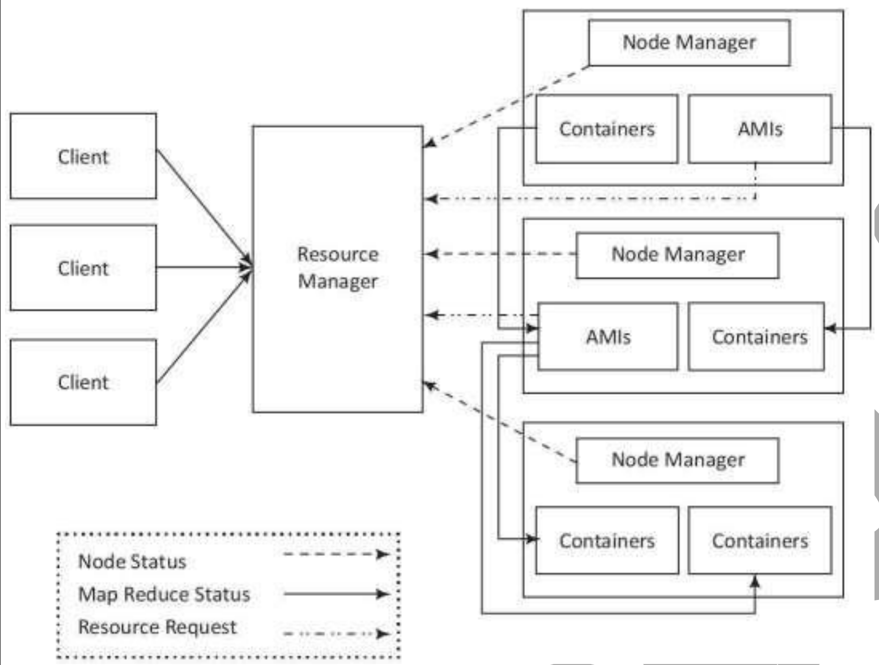
2. Coordinator Jobs : These are scheduled jobs that can run at specified time intervals or trigger based on the availability of data. This feature is particularly useful for recurring tasks.

3. Bundle Jobs : This is a higher-level abstraction that allows users to group multiple coordinator jobs together, making it easier to manage complex workflows.

Oozie integrates seamlessly with other Hadoop ecosystem tools, such as MapReduce, Apache Hive, Apache Pig, and Apache Sqoop, enabling users to orchestrate a variety of data processing tasks. It also provides a command-line interface (CLI) and a web user interface (UI) for monitoring and managing jobs.

	c	Explain YARN application framework.	CO2	05

YARN, which stands for Yet Another Resource Negotiator, is a key component of the Hadoop ecosystem that serves as a resource management platform. It plays a crucial role in managing and scheduling resources for various applications running on a Hadoop cluster. Here’s a detailed breakdown of the YARN application framework:



1. Architecture : YARN separates the resource management and job scheduling functionalities from the data processing components of Hadoop. This architecture allows for better resource utilization and scalability.

2. Core Components :
- Resource Manager (RM) : The RM is the master daemon that manages the allocation of resources across all applications in the system. It keeps track of the available resources and the status of all Node Managers (NMs) in the cluster.
- Node Manager (NM) : Each cluster node runs an NM, which is responsible for managing the resources on that node. It monitors resource usage (CPU, memory) and reports this information back to the RM.
- Application Master (AM) : For each application submitted to YARN, an AM is instantiated. The AM is responsible for negotiating resources from the RM and working with the NMs to execute and monitor the application’s tasks.
- Containers : These are the basic units of resource allocation in YARN. A container encapsulates the resources (CPU, memory) required to run a specific task of an application.

3. Execution Model : When a client submits an application, the following steps occur:
- The client sends a request to the RM to start the application.
- The RM allocates a container for the AM, which then registers itself with the RM.
- The AM requests containers from the RM for the application’s tasks, which are then allocated on the NMs.
- The AM manages the execution of tasks within these containers, handling failures and resource requests dynamically.

4. Resource Allocation and Scheduling : YARN employs various scheduling algorithms to allocate resources efficiently. It can handle multiple applications simultaneously, ensuring that resources are distributed based on priority and resource requirements.

5. Scalability and Flexibility : One of the significant advantages of YARN is

its ability to scale from a single server to thousands of machines. This modular design allows for the addition or replacement of components without disrupting the entire system.

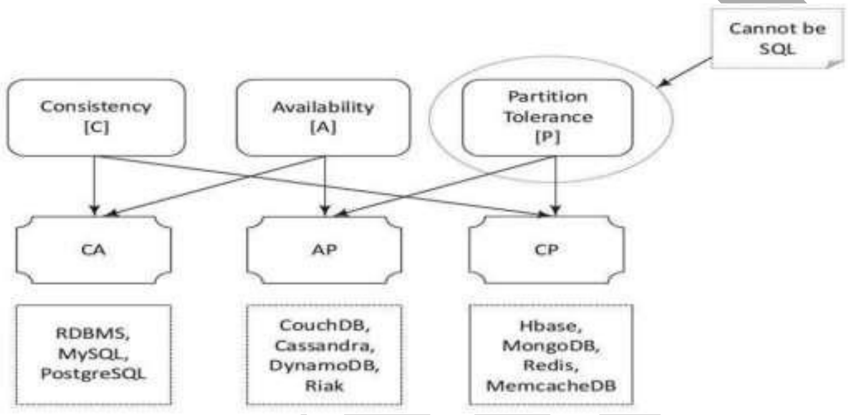
Module-3

DOWNLOAD

Q. 05	a	What is NOSQL? Explain CAP Theorem.	CO3	10

NoSQL, which stands for "Not Only SQL," is a category of non-relational data storage systems designed to handle large volumes of data with flexible data models. Unlike traditional SQL databases, NoSQL databases do not require a fixed schema, allowing for dynamic and schema-less data storage. This makes them ideal for managing big data, accommodating various data types like key-value pairs, document stores (e.g., MongoDB), column-family stores (e.g., Cassandra), and graph databases.

The CAP theorem, formulated by computer scientist Eric Brewer, is a fundamental principle that applies to distributed data systems. It states that in the presence of a network partition, a distributed system can only guarantee two out of the following three properties:



1. Consistency (C) : This means that every read operation receives the most recent write or an error. In other words, all nodes in the system see the same data at the same time. If one node updates data, all other nodes must reflect that change immediately.

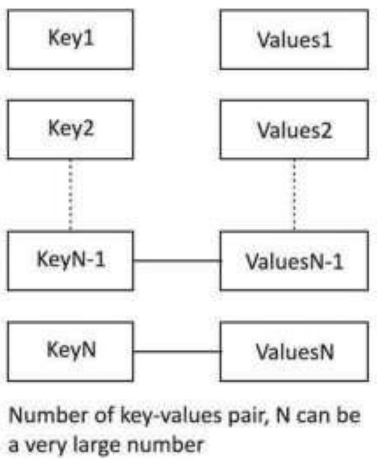
2. Availability (A) : This property ensures that every request (whether read or write) receives a response, regardless of whether it contains the most recent data . This means that the system remains operational and responsive even if some nodes are down or unreachable.

3. Partition Tolerance (P) : This refers to the system's ability to continue operating despite network partitions that prevent some nodes from communicating with others. In a partitioned network, the system must still function, even if some nodes cannot reach others.

In essence, the CAP theorem implies that if a network partition occurs, a system must choose between consistency and availability. For example, if a system prioritizes consistency, it may become unavailable during a partition. Conversely, if it prioritizes availability, it may return stale or outdated data. This trade-off is crucial for designing distributed systems, especially in the context of big data applications.

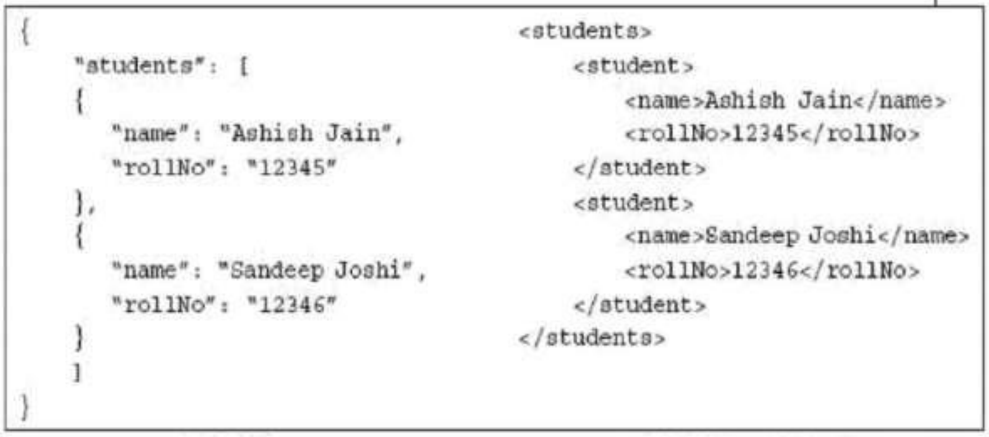
	b	Explain NOSQL Data Architecture Patterns.	CO3	10

NoSQL Data Architecture Patterns refer to the various ways in which NoSQL databases are structured to handle data storage and retrieval efficiently. Here are some key patterns:



Key	Value
"Ashish"	"Category: Student; Class: B.Tech.; Semester: VII; Branch: Engineering; Mobile:9999912345"
"Mayuri"	"Category: student; class: M.Tech.; Mobile:8888823456"

1. Key-Value Pairs : This is the simplest NoSQL architecture, where data is stored as a collection of key-value pairs. Each key is a unique identifier that maps to a large data string or BLOB (Binary Large Object). This model is known for its high performance, scalability, and flexibility. Data retrieval is fast, as a query simply requests the value associated with a specific key. Key-value stores are eventually consistent, meaning that while data may not be immediately consistent across all nodes, it will become consistent over time.

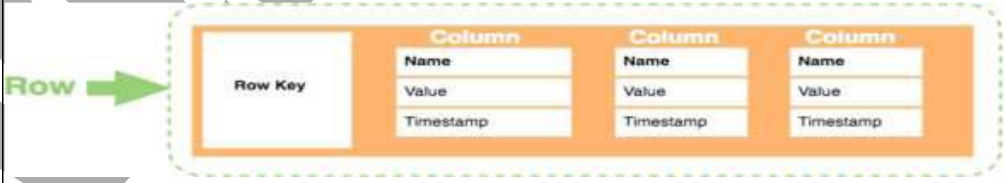


```
{
    "students": [
    {
        "name": "Ashish Jain",
        "rollNo": "12345"
    },
    {
        "name": "Sandeep Joshi",
        "rollNo": "12346"
    }
    ]
}
```

```
<students>
    <student>
        <name>Ashish Jain</name>
        <rollNo>12345</rollNo>
    </student>
    <student>
        <name>Sandeep Joshi</name>
        <rollNo>12346</rollNo>
    </student>
</students>
```

(a) JSON (b) XML equivalent

2. Document-Based Stores : In this pattern, data is stored in documents, typically in formats like JSON or BSON. Each document can have a different structure, allowing for a schema-less design. MongoDB is a prominent example of a document-based store, which provides flexibility in data representation and is well-suited for applications that require rapid development and iteration.

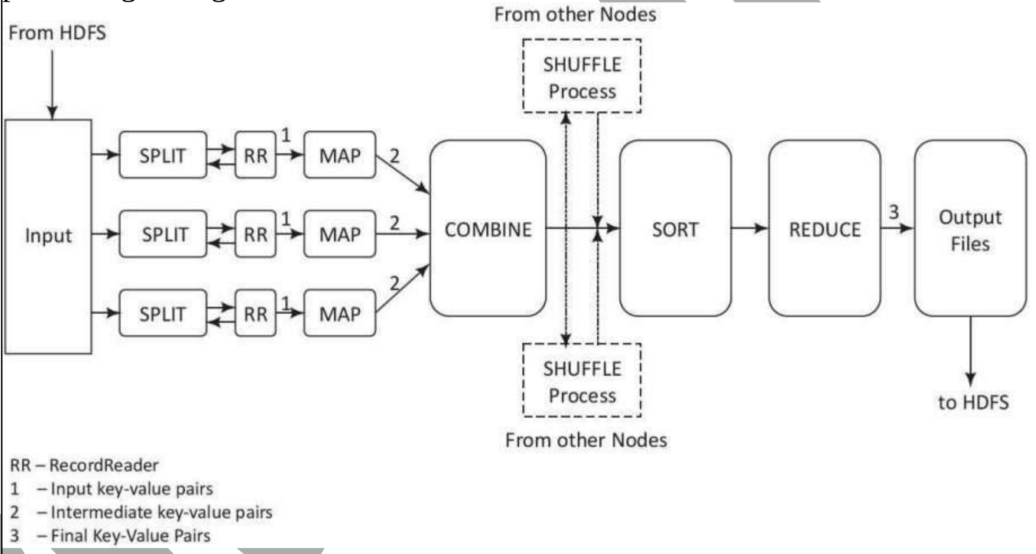


3. Column-Family Stores : This architecture organizes data into columns rather than rows, which is particularly useful for handling large datasets. Each column family can store a different set of data, and this model is optimized for read and write operations. Cassandra is a well-known example of a column-family store, designed for high availability and scalability.

4. Graph Databases : These databases are designed to handle complex relationships between data points. They use nodes, edges, and properties to represent and store data, making them ideal for applications that require link analysis and relationship modeling. Examples include Neo4J and AllegroGraph.

5. Object Stores : This pattern allows for the storage of data as objects, which

		can include both the data itself and metadata. Object stores are often used for unstructured data and are designed to handle large amounts of data efficiently. 6. Wide-Column Stores : Similar to column-family stores, wide-column stores allow for a flexible schema where each row can have a different number of columns. This is useful for applications that require dynamic data structures.		
OR				
Q. 06	a	Explain Shared Nothing Architecture for Big Data tasks. Shared Nothing Architecture (SNA) is a distributed computing model that plays a crucial role in managing Big Data tasks. In this architecture, each node in the system operates independently and does not share its memory or storage with any other node. This independence allows for several key advantages: 1. Independence : Each node is self-sufficient, meaning it can process data and execute tasks without relying on other nodes. This leads to increased reliability since the failure of one node does not affect the others. 2. Parallel Processing : In a Shared Nothing Architecture, data is partitioned into shards, and each node processes its own shard independently. This allows for parallel execution of queries, significantly improving performance and throughput. 3. Self-Healing : If a link between nodes fails, the architecture can create new links to maintain connectivity, ensuring that the system remains operational even in the face of hardware failures. 4. No Network Contention : Since each node operates independently, there is no competition for network resources, which can lead to more efficient data processing. 5. Scalability : SNA supports horizontal scaling, meaning that as data volumes grow, additional nodes can be added to the system without significant reconfiguration. This is particularly beneficial for handling large datasets typical in Big Data applications. 6. Data Replication : To enhance fault tolerance and availability, data can be replicated across multiple nodes. This ensures that even if one node fails, the data remains accessible from other nodes. Examples of technologies that utilize Shared Nothing Architecture include Hadoop, Flink, and Spark, which are designed to efficiently manage and process large volumes of data across distributed systems.	CO3	10
	b	Explain MONGO DATABASE. MongoDB is a powerful, open-source NoSQL database designed to handle large volumes of data with flexibility and scalability. It was developed by a New York-based organization originally named 10gen, now known as MongoDB Inc. Here are some key features and characteristics of MongoDB: 1. Document-Based Storage : MongoDB stores data in the form of JSON-like documents, which allows for a flexible schema. This means that documents within a collection can have different fields, making it easy to adapt to changing data requirements. 2. Collections and Databases : In MongoDB, a collection is analogous to a table in traditional relational databases (RDBMS), and it can hold multiple documents. Each database can contain multiple collections, and each collection can store documents that do not necessarily share the same structure. 3. Schema-less Design : The schema-less nature of MongoDB allows for dynamic data storage, meaning you can easily add or remove fields from documents without needing to alter the entire database structure.	CO3	10

		4. High Availability and Fault Tolerance : MongoDB uses a feature called replica sets, which are groups of MongoDB server processes that maintain the same dataset. This setup ensures that if one server fails, others can take over, providing high availability. 5. Powerful Querying : MongoDB supports a rich query language that allows for deep querying capabilities, including dynamic queries on documents. This is nearly as powerful as SQL, enabling complex data retrieval. 6. Scalability : MongoDB is designed to scale out easily by adding more servers to handle increased loads, making it suitable for big data applications. 7. Data Types : MongoDB supports various data types, including strings, numbers, arrays, and embedded documents, allowing for versatile data representation. 8. Commands and Operations : Common operations in MongoDB include creating databases and collections, inserting, updating, and deleting documents, as well as querying data using commands like `db.collection.find()`.		
		Module-4 **DOWNLOAD**		
Q. 07	a	Explain Map Reduce Execution steps with neat diagram. The execution of a MapReduce job involves several key steps that facilitate the processing of large datasets in a distributed environment.  From HDFS Input SPLIT RR MAP 1 2 COMBINE SHUFFLE Process From other Nodes SORT REDUCE 3 Output Files to HDFS RR – RecordReader 1 – Input key-value pairs 2 – Intermediate key-value pairs 3 – Final Key-Value Pairs MapReduce execution steps: 1. Input Submission : The application submits the input data, which is typically stored in the Hadoop Distributed File System (HDFS). This data is divided into smaller chunks called InputSplits, which are processed in parallel. 2. Mapping : The Map phase begins where the input data is processed by the Mapper. Each Mapper takes a key-value pair as input and applies the `map()` function to generate intermediate key-value pairs. The Mapper operates independently on each piece of data, allowing for parallel processing across multiple nodes. 3. Combining : After mapping, a Combiner may be used to perform a local aggregation of the intermediate key-value pairs produced by the Mapper. This step is optional but can reduce the amount of data transferred to the Reducer. 4. Shuffling and Sorting : Once the mapping is complete, the Shuffle phase begins. This involves redistributing the data based on the intermediate keys generated by the Mappers. The system groups all the intermediate key-value pairs	CO4	10

by their keys, which prepares them for the Reduce phase. During this phase, the data is also sorted, ensuring that all values associated with the same key are brought together.

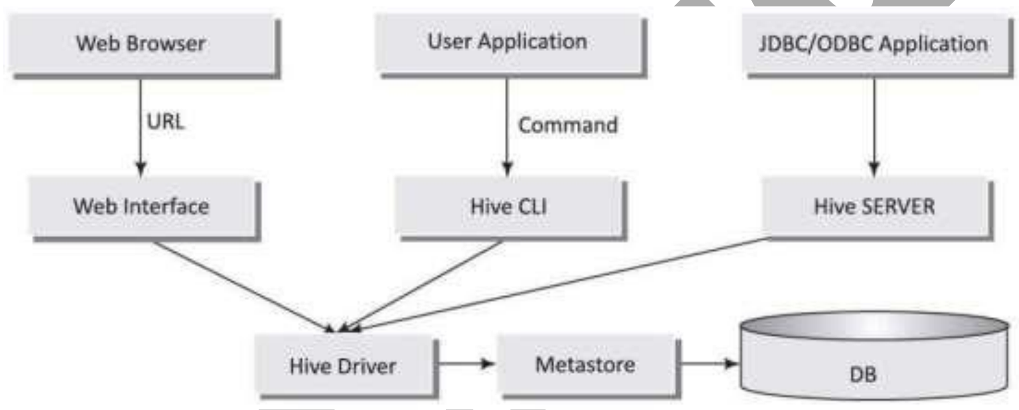
5. Reducing : The Reduce phase takes the grouped data and processes it using the `reduce()` function. Each Reducer receives a key and a list of values associated with that key. The Reducer combines these values to produce a smaller set of output data, which is the final result of the MapReduce job.

6. Output Storage : Finally, the output from the Reduce tasks is written back to HDFS. This output can then be used for further analysis or processing.

	b	What is HIVE? Explain HIVE Architecture.	CO4	10

Hive is a data warehousing tool that was created by Facebook and is built on top of Hadoop. It allows users to manage and analyze large datasets stored in Hadoop's HDFS (Hadoop Distributed File System) using a SQL-like query language called HiveQL (or HQL). Hive is particularly suited for processing structured data and can integrate data from various heterogeneous sources, making it a powerful tool for enterprises that need to track, manage, and analyze large volumes of data.

Hive Architecture



The architecture of Hive consists of several key components that work together to facilitate data processing and querying:

1. Hive Server (Thrift) : This is an optional service that allows remote clients to submit requests to Hive and retrieve results. It exposes a simple client API for executing HiveQL statements, enabling interaction with Hive using various programming languages.

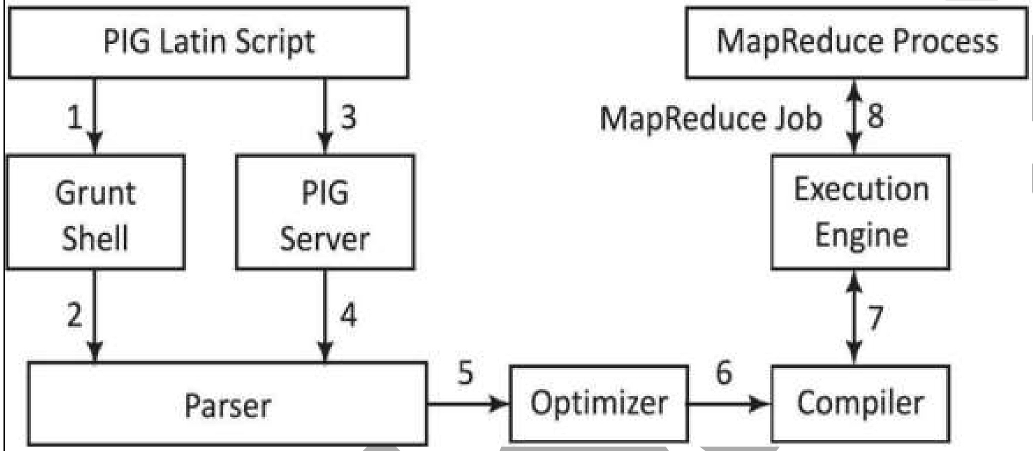
2. Hive CLI (Command Line Interface) : This is a popular interface for interacting with Hive. When running the CLI on a Hadoop cluster, it operates in local mode, utilizing local storage instead of HDFS.

3. Web Interface : Hive can also be accessed through a web browser, provided that a Hive Web Interface (HWI) server is running. Users can access Hive by navigating to a specific URL format: `http://hadoop:<port no>/hwi`.

4. Metastore : This is the system catalog of Hive, where all other components interact. The Metastore stores metadata about tables, databases, and columns, including their data types and HDFS mappings. It is crucial for managing the schema of the data being processed.

5. Hive Driver : This component manages the lifecycle of a HiveQL statement, overseeing its compilation, optimization, and execution.

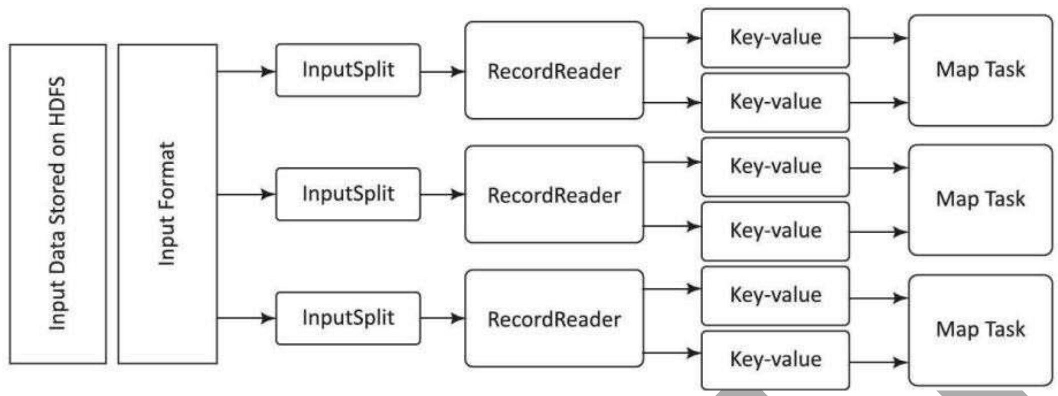
Workflow Steps in Hive

		The workflow for executing a query in Hive involves several steps: 1. Execute Query : The Hive interface (CLI or Web Interface) sends a query to the Database Driver. 2. Get Plan : The Driver sends the query to the query compiler, which checks the syntax and prepares a query plan. 3. Get Metadata : The compiler requests metadata from the Metastore. 4. Send Metadata : The Metastore responds with the necessary metadata.		
OR				
Q. 08	a	Explain Pig architecture for scripts dataflow and processing Apache Pig architecture is designed to facilitate the execution of Pig Latin scripts in a Hadoop environment, specifically within the Hadoop Distributed File System (HDFS). Here's a detailed breakdown of how the architecture works for scripts dataflow and processing:  PIG Latin Script 1 3 Grunt Shell PIG Server 2 4 Parser 5 Optimizer 6 Compiler 7 Execution Engine MapReduce Job 8 MapReduce Process 1. Pig Latin Scripts Submission : The process begins when a Pig Latin script is submitted to the Apache Pig Execution Engine. This engine is responsible for interpreting and executing the commands written in Pig Latin. 2. Parser : Once the script is submitted, it goes through a parser. The parser performs several critical functions: - Type Checking : It checks the types of the data being processed to ensure they are compatible with the operations specified in the script. - Syntax Checking : It verifies that the script adheres to the correct syntax of Pig Latin. - Directed Acyclic Graph (DAG) Generation : The output of the parser is a Directed Acyclic Graph (DAG). In this graph, nodes represent logical operators (like join, filter, etc.), and edges represent the data flows between these operations. The acyclic nature ensures that there are no cycles in the data flow, meaning that data flows in one direction without looping back. 3. Optimizer : After parsing, the DAG is passed to an optimizer. The optimizer enhances the execution plan by applying various optimization techniques to improve performance before the actual execution begins. 4. Execution Engine : The optimized DAG is then executed by the Pig Execution Engine. This engine translates the high-level Pig Latin commands into low-level MapReduce jobs that can be processed by the Hadoop framework. 5. Data Processing : The execution engine processes the data according to the operations defined in the Pig Latin script. It reads input data from HDFS, performs the specified transformations, and writes the output back to HDFS. 6. Execution Modes : Pig can operate in different execution modes: - Local Mode : For testing purposes, where all data files are processed on a local host.	CO4	10

		- MapReduce Mode : Where data is processed in a distributed manner using the MapReduce framework, leveraging the power of Hadoop. 7. User Defined Functions (UDFs) : If there are specific functions that are not available in the built-in Pig operators, users can create UDFs in other programming languages (like Java) and embed them in their Pig Latin scripts.		

b Explain Key Value pairing in Map Reduce. CO4 10

In MapReduce, key-value pairing is a fundamental concept that underpins the entire processing model. Each phase of the MapReduce process—both the Map phase and the Reduce phase—utilizes key-value pairs as input and output. Here's a detailed breakdown of how key-value pairs are generated and used:



1. Input Preparation : Before data can be processed by the mapper, it must be converted into key-value pairs. This is crucial because the mapper only understands data in this format. The transformation into key-value pairs is typically handled by a component called the RecordReader.

2. InputSplit : This defines a logical representation of the data and breaks it into smaller, manageable pieces for processing. Each piece is then passed to the mapper.

3. RecordReader : This component interacts with the InputSplit to convert the split data into records formatted as key-value pairs. By default, it uses `TextInputFormat` to read the data, ensuring that it is in a suitable format for the mapper.

4. Map Phase : During the Map phase, the mapper processes each key-value pair (k1, v 1). The key (k1) represents a unique identifier, while the value (v1) is the associated data. The output of the map function can either be zero (if no relevant values are found) or a set of intermediate key-value pairs (k2, v2). Here, k2 is a new key generated based on the processing logic, and v2 contains the information needed for the subsequent Reduce phase.

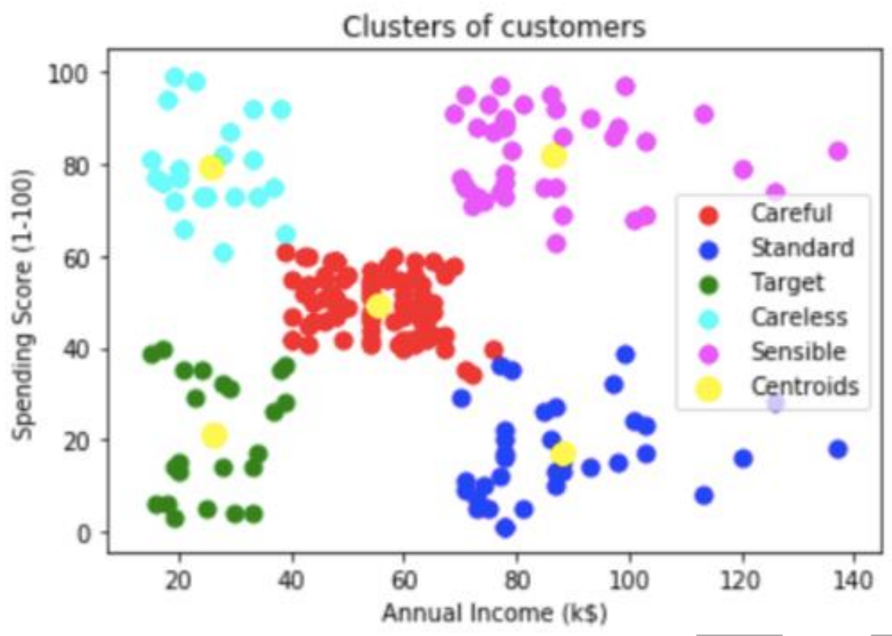
5. Reduce Phase : After the Map phase, the output key-value pairs are shuffled and sorted. The Reduce task takes these intermediate key-value pairs (k2, v2) as input. It groups the values associated with each key and applies a reducing function to produce a smaller set of output key-value pairs (k3, v3). This output is then written to the final output file.

6. Grouping and Aggregation : The grouping operation is performed during the shuffle phase, where all pairs with the same key are collected together. This allows the reducer to apply aggregate functions like count, sum, average, min, and max on the grouped data.

In summary, key-value pairing in MapReduce is essential for organizing and processing large datasets efficiently. It allows for parallel processing and facilitates the transformation of data through the Map and Reduce phases, ultimately leading to meaningful insights from big data.

Module-5 DOWNLOAD				
Q. 09	a	What is Machine Learning? Explain different types of Regression Analysis. Machine Learning (ML) is a subset of Artificial Intelligence (AI) that focuses on the development of algorithms that allow computers to learn from and make predictions based on data. It involves using statistical techniques to enable machines to improve their performance on a specific task over time without being explicitly programmed for each scenario. ML can be broadly categorized into supervised learning, unsupervised learning, and reinforcement learning. When it comes to regression analysis, it is a statistical method used to estimate the relationships among variables. Here are the different types of regression analysis: 1. Simple Linear Regression : This is the most basic form of regression analysis. It models the relationship between a single independent variable (predictor) and a dependent variable (outcome) using a linear equation. The goal is to find the best-fitting line through a scatter plot of data points, minimizing the deviation (error) $y = f(a_0, a_1) = a_0 + a_1 x,$ 2. Multiple Linear Regression : This extends simple linear regression by using multiple independent variables to predict a single dependent variable. It helps in understanding how the dependent variable changes when any one of the independent variables is varied while keeping the others constant. 3. Non-linear Regression : Unlike linear regression, non-linear regression models the relationship between variables using a non-linear equation. This is useful when the data shows a curvilinear relationship. $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3,$ 4. Polynomial Regression : A form of non-linear regression where the relationship between the independent variable and the dependent variable is modeled as an nth degree polynomial. This allows for more flexibility in fitting the data. 5. Logistic Regression : Although it has "regression" in its name, logistic regression is used for binary classification problems. It predicts the probability that a given input point belongs to a certain category, using a logistic function to model the relationship. 6. Ridge and Lasso Regression : These are techniques used to prevent overfitting in regression models. Ridge regression adds a penalty equal to the square of the magnitude of coefficients, while Lasso regression adds a penalty equal to the absolute value of the magnitude of coefficients. 7. Support Vector Regression (SVR) : This is an extension of Support Vector Machines (SVM) for regression problems. It aims to find a function that deviates from the actual observed values by a value no greater than a specified margin.	CO5	10

	b	Explain with neat diagram K-means clustering.	CO5	05



K-means clustering is a popular and straightforward algorithm used in data mining and machine learning for partitioning a dataset into distinct groups, or clusters. The main goal of K-means is to divide the data into K clusters, where each data point belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

Here's how the K-means algorithm works in detail:

1. Initialization : First, you need to choose the number of clusters, K. Then, K initial centroids (the center points of the clusters) are randomly selected from the dataset.

2. Assignment Step : Each data point is assigned to the nearest centroid based on a distance metric, typically Euclidean distance. This means that for each data point, the algorithm calculates the distance to each centroid and assigns the point to the cluster represented by the closest centroid.

3. Update Step : After all points have been assigned to clusters, the centroids are recalculated. This is done by taking the mean of all the data points that belong to each cluster. The new centroid is the average position of all the points in that cluster.

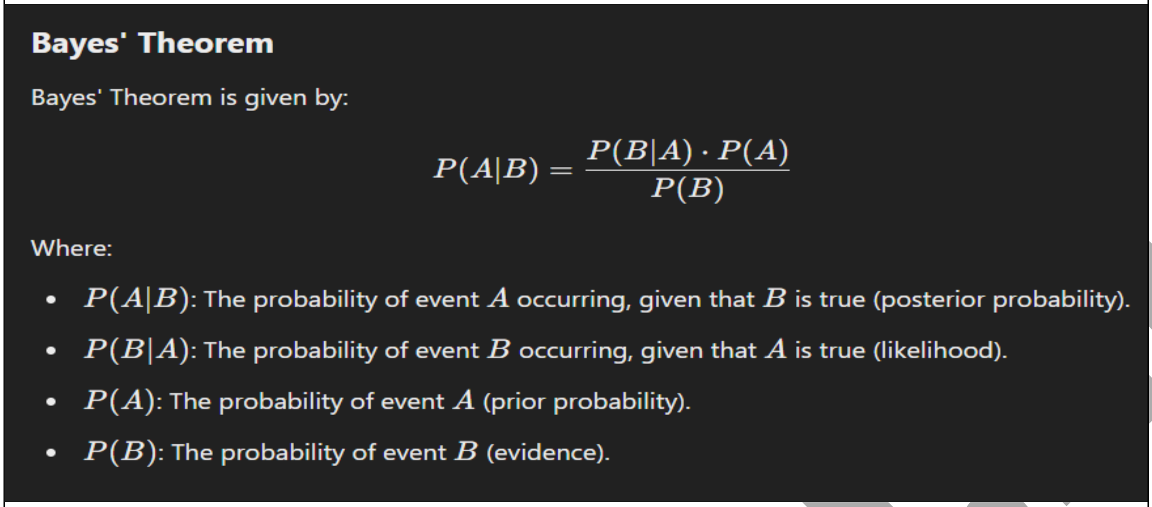
4. Repeat : Steps 2 and 3 are repeated until the centroids no longer change significantly, indicating that the algorithm has converged, or until a predetermined number of iterations is reached.

5. Output : The final output of the K-means algorithm is the K clusters of data points, along with their corresponding centroids.

K-means clustering is widely used due to its simplicity and efficiency, especially for large datasets. However, it does have some limitations, such as sensitivity to the initial placement of centroids and the requirement to specify the number of clusters in advance.

	c	Explain Naïve Bayes Theorem with example.	CO5	05

The **Naïve Bayes Theorem** is a supervised machine learning algorithm based on **Bayes' Theorem** and is primarily used for classification tasks. It assumes that features are **independent** of each other, which is why it's called *naïve*. Despite this simplifying assumption, it often works surprisingly well in many practical applications



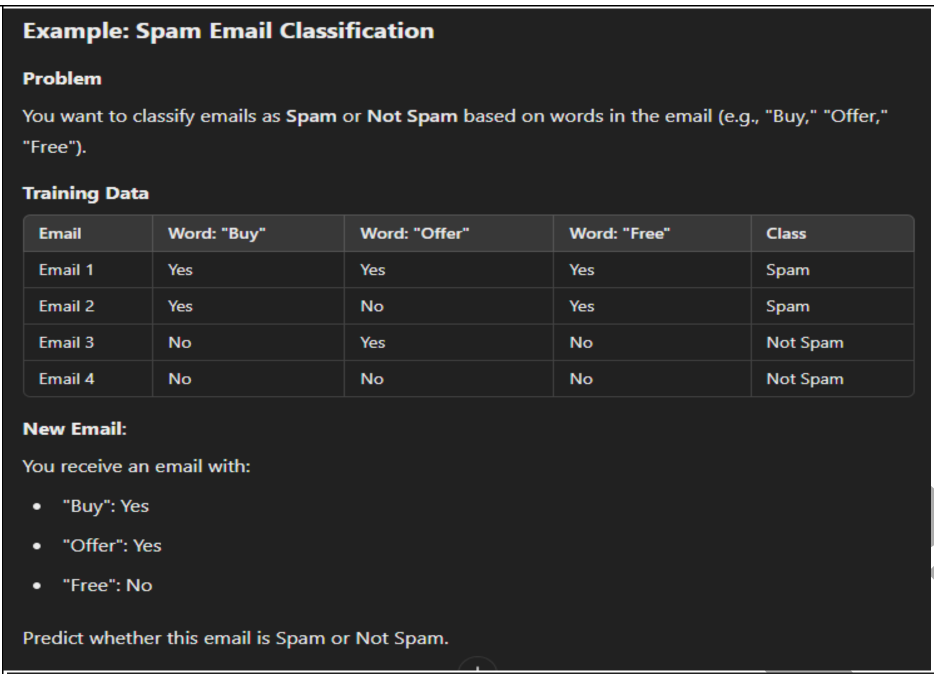
Bayes' Theorem

Bayes' Theorem is given by:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

- $P(A|B)$: The probability of event A occurring, given that B is true (posterior probability).
- $P(B|A)$: The probability of event B occurring, given that A is true (likelihood).
- $P(A)$: The probability of event A (prior probability).
- $P(B)$: The probability of event B (evidence).



Example: Spam Email Classification

Problem

You want to classify emails as **Spam** or **Not Spam** based on words in the email (e.g., "Buy," "Offer," "Free").

Training Data

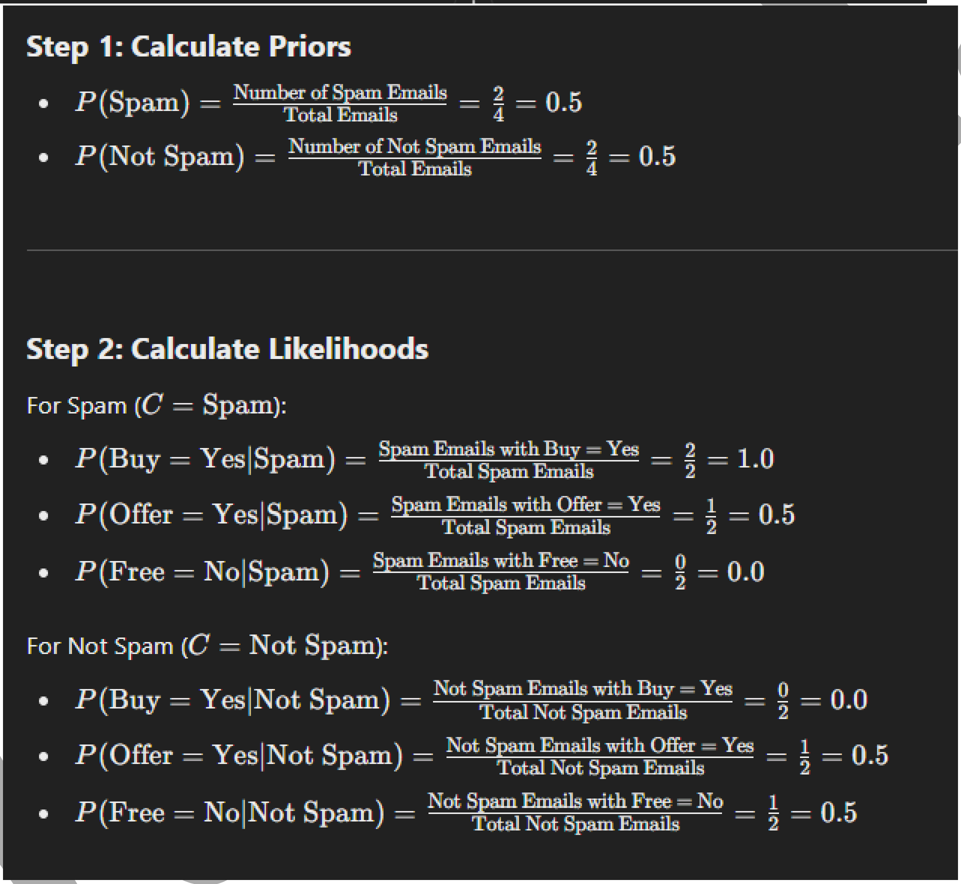
Email	Word: "Buy"	Word: "Offer"	Word: "Free"	Class
Email 1	Yes	Yes	Yes	Spam
Email 2	Yes	No	Yes	Spam
Email 3	No	Yes	No	Not Spam
Email 4	No	No	No	Not Spam

New Email:

You receive an email with:

- "Buy": Yes
- "Offer": Yes
- "Free": No

Predict whether this email is Spam or Not Spam.



Step 1: Calculate Priors

- $P(\text{Spam}) = \frac{\text{Number of Spam Emails}}{\text{Total Emails}} = \frac{2}{4} = 0.5$
- $P(\text{Not Spam}) = \frac{\text{Number of Not Spam Emails}}{\text{Total Emails}} = \frac{2}{4} = 0.5$

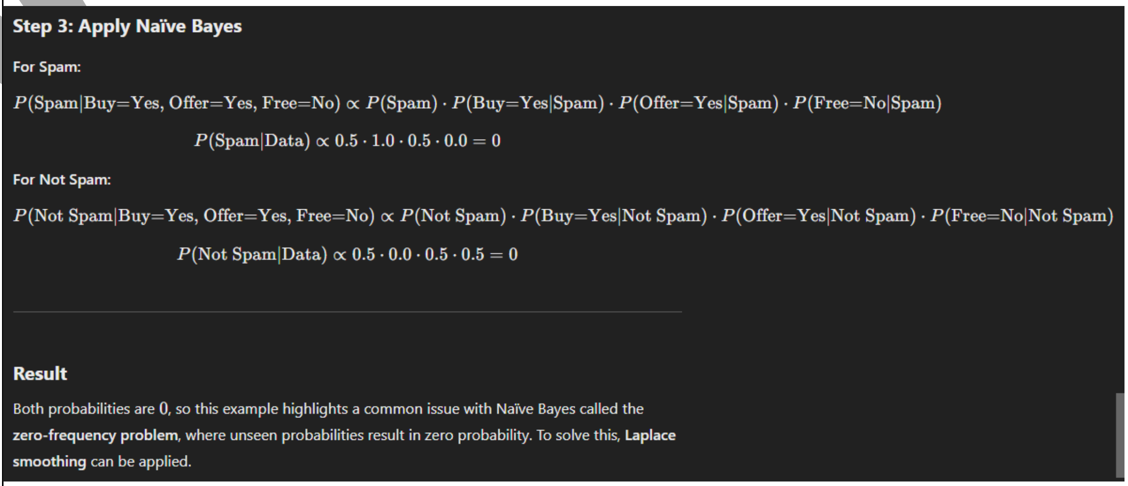
Step 2: Calculate Likelihoods

For Spam ($C = \text{Spam}$):

- $P(\text{Buy} = \text{Yes}|\text{Spam}) = \frac{\text{Spam Emails with Buy} = \text{Yes}}{\text{Total Spam Emails}} = \frac{2}{2} = 1.0$
- $P(\text{Offer} = \text{Yes}|\text{Spam}) = \frac{\text{Spam Emails with Offer} = \text{Yes}}{\text{Total Spam Emails}} = \frac{1}{2} = 0.5$
- $P(\text{Free} = \text{No}|\text{Spam}) = \frac{\text{Spam Emails with Free} = \text{No}}{\text{Total Spam Emails}} = \frac{0}{2} = 0.0$

For Not Spam ($C = \text{Not Spam}$):

- $P(\text{Buy} = \text{Yes}|\text{Not Spam}) = \frac{\text{Not Spam Emails with Buy} = \text{Yes}}{\text{Total Not Spam Emails}} = \frac{0}{2} = 0.0$
- $P(\text{Offer} = \text{Yes}|\text{Not Spam}) = \frac{\text{Not Spam Emails with Offer} = \text{Yes}}{\text{Total Not Spam Emails}} = \frac{1}{2} = 0.5$
- $P(\text{Free} = \text{No}|\text{Not Spam}) = \frac{\text{Not Spam Emails with Free} = \text{No}}{\text{Total Not Spam Emails}} = \frac{1}{2} = 0.5$



Step 3: Apply Naïve Bayes

For Spam:

$$P(\text{Spam}|\text{Buy=Yes, Offer=Yes, Free=No}) \propto P(\text{Spam}) \cdot P(\text{Buy=Yes}|\text{Spam}) \cdot P(\text{Offer=Yes}|\text{Spam}) \cdot P(\text{Free=No}|\text{Spam})$$

$$P(\text{Spam}|\text{Data}) \propto 0.5 \cdot 1.0 \cdot 0.5 \cdot 0.0 = 0$$

For Not Spam:

$$P(\text{Not Spam}|\text{Buy=Yes, Offer=Yes, Free=No}) \propto P(\text{Not Spam}) \cdot P(\text{Buy=Yes}|\text{Not Spam}) \cdot P(\text{Offer=Yes}|\text{Not Spam}) \cdot P(\text{Free=No}|\text{Not Spam})$$

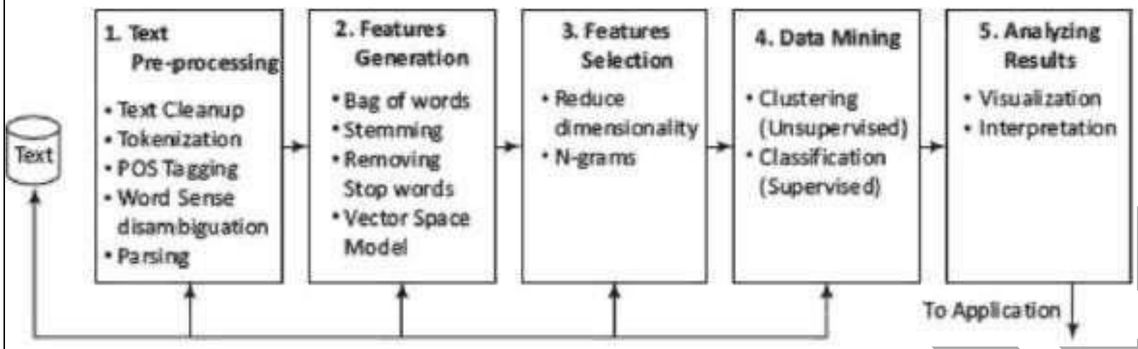
$$P(\text{Not Spam}|\text{Data}) \propto 0.5 \cdot 0.0 \cdot 0.5 \cdot 0.5 = 0$$

Result

Both probabilities are 0, so this example highlights a common issue with Naïve Bayes called the **zero-frequency problem**, where unseen probabilities result in zero probability. To solve this, **Laplace smoothing** can be applied.

		OR		
Q. 10	a	Explain five phases in a process pipeline text mining.	CO5	10

The five phases in a process pipeline for text mining are designed to efficiently analyze and extract valuable information from unstructured text. Here's a detailed breakdown of each phase:



1. Text Pre-processing : This initial phase involves preparing the text for analysis. It includes several steps such as:
   - Tokenization : Breaking down the text into individual words or tokens.
   - Normalization : Converting all text to a standard format, such as lowercasing.
   - Removing Stop Words : Filtering out common words that may not add significant meaning (e.g., "and," "the").
   - Stemming and Lemmatization : Reducing words to their base or root form to treat different forms of a word as the same (e.g., "running" to "run").

2. Feature Extraction : In this phase, relevant features are extracted from the pre-processed text. This can involve:
   - Vectorization : Converting text into numerical format using techniques like Term Frequency-Inverse Document Frequency (TF-IDF) or word embeddings.
   - Identifying Key Phrases : Extracting important phrases or terms that represent the content of the text.

3. Modeling : This phase involves applying various analytical techniques to the extracted features. Common methods include:
   - Classification : Assigning categories to text documents (e.g., spam detection).
   - Clustering : Grouping similar documents together without predefined categories. - Sentiment Analysis : Determining the sentiment expressed in the text (positive, negative, neutral).

4. Evaluation : After modeling, the results need to be evaluated to assess their accuracy and effectiveness. This can involve:
   - Performance Metrics : Using metrics such as precision, recall, and F1-score to measure the model's performance.
   - Cross-Validation : Testing the model on different subsets of data to ensure its robustness.

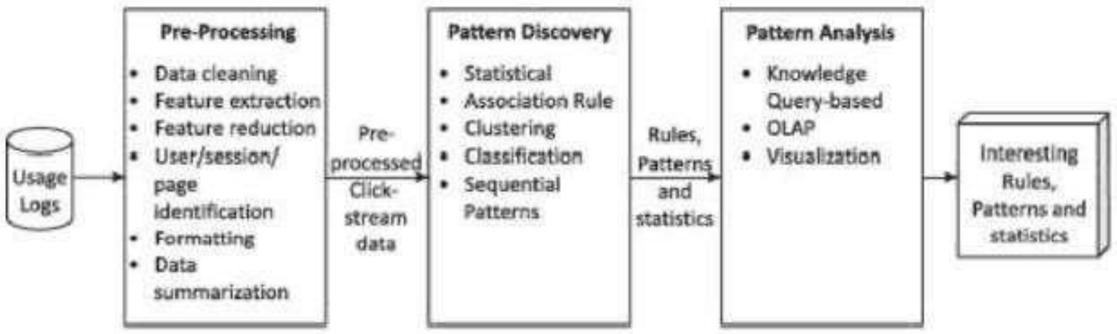
5. Analysis of Results : The final phase focuses on interpreting the outcomes of the text mining process. This includes:
   - Visualizing Data : Creating visual representations of the results to identify patterns and insights.
   - Using Results for Decision Making : Applying the insights gained to improve business processes, enhance marketing strategies, or inform future actions.

These phases work iteratively and interactively, allowing for continuous refinement and improvement of the text mining process. Each phase is crucial for transforming raw text data into actionable knowledge.

	b	Explain Web Usage Mining.	CO5	10

Web Usage Mining is a fascinating area of data mining that focuses on discovering and analyzing patterns from web usage data. This type of mining is particularly concerned with understanding how users interact with web resources, which can provide valuable insights for improving website design, enhancing user experience, and optimizing marketing strategies.

The process of Web Usage Mining can be broken down into three main phases:



1. Pre-processing : This initial phase involves converting the raw usage data collected from various sources into a format suitable for analysis. This data often comes from web server logs, which typically include information such as the IP address of the user, the pages they accessed, and the time of access. The goal here is to clean and organize the data to facilitate effective pattern discovery.

2. Pattern Discovery : In this phase, various algorithms and methods are applied to the pre-processed data to uncover interesting patterns. Techniques from fields such as machine learning, statistics, and information retrieval are utilized. For example, methods like clustering, classification, and association rule mining can help identify common user behaviors, such as frequently accessed pages or typical navigation paths.

3. Pattern Analysis : After patterns have been discovered, they are analyzed to extract meaningful insights. This analysis can reveal trends in user behavior, such as peak usage times, popular content, and user preferences. The insights gained can be used to inform decisions about website design, content placement, and targeted marketing efforts.

Web Usage Mining has several practical applications, including:

- Website Optimization : By understanding how users navigate a site, webmasters can improve the layout and structure to enhance user experience.
- Targeted Marketing : Insights from user behavior can inform marketing strategies, allowing businesses to tailor their advertising efforts based on user interests and past behavior.
- Fraud Detection : Analyzing usage patterns can help identify unusual activities that may indicate fraudulent behavior, such as unauthorized access or unusual transaction patterns.

Bloom's Taxonomy Level: Indicate as L1, L2, L3, L4, etc. It is also desirable to indicate the COs and POs to be attained by every bit of questions.