Big Data Analytics - Viva

1. What is BDA? Characteristics of BDA

Definition: Big Data Analytics refers to the process of examining, cleaning, transforming, and modeling large volumes of data with the goal of discovering insights, patterns, and trends that can help organizations make informed decisions.

There are multiple **tools for processing Big Data such as Hadoop, Pig, Hive, Cassandra, Spark, Kafka, etc.** depending upon the requirement of the organization.

Characteristics of Big Data Analytics:

Volume: BDA deals with large and often massive datasets - terabytes to petabytes in size.

Variety: Data comes in various forms, including structured (e.g., databases), semi-structured (e.g., XML, JSON), and unstructured (e.g., text, images, videos). BDA processes and analyzes data in all these formats.

Velocity: Data streams into systems at high speeds, often in real-time or near-real-time. BDA needs to process data quickly to extract insights and make timely decisions.

Veracity: BDA must handle noisy, incomplete, and inconsistent data. Data cleansing and preprocessing are critical steps to ensure data consistency, accuracy, quality, and trustworthiness.

Value: The ultimate goal of BDA is to extract value from data. It involves various analytics techniques, including descriptive, diagnostic, predictive, and prescriptive analytics

Complexity: BDA often involves complex analysis, machine learning, and statistical modeling.

Scalability: BDA systems must be able to scale horizontally to handle the vast amounts of data. Distributed computing and parallel processing are common strategies.

Real-Time and Batch Processing: BDA can be used for both real-time analytics and batch processing. It's adaptable to various business needs.

2. Four Vs

It is typically characterized by the four V's:
Volume (large amounts of data),
Velocity (high data processing speed),
Variety (different data types and sources), and
Veracity (uncertainty or data quality).

3. **DGIM ALGO** - https://www.youtube.com/watch?v=i9MCGnA-ZPQ&ab_channel=Gyanpur

- DGIM (Datar-Gionis-Indyk-Motwani) is an algorithm used for approximate counting of ones in a sliding window over a stream of binary data. Error rate not more than 50%.
- Main components are:
  a)Timestamp
  b)Bucket
- Bucket size is always in the size of 2^n.
- Same size of buckets can't be repeated more than 2 times.

The DGIM algorithm provides an efficient way to estimate the number of ones in a binary data stream within a sliding window of fixed size, without storing all the data. It uses a compact data structure to achieve this, making it suitable for real-time processing and analysis of large data streams.

Social Media Analytics: It can be applied to estimate the number of likes, shares, retweets, or other interactions with social media posts, helping social media platforms understand user engagement and trending topics.

Network Traffic Monitoring: In network monitoring, DGIM can help estimate the volume of specific types of network traffic (e.g., DDoS attacks, video streaming, or email traffic) to detect unusual patterns or anomalies.

DGIM uses a structure called a "bucket array," which is a collection of buckets. Each bucket is associated with a specific time window and can contain up to two pieces of information:

- The **count of ones** within its time window
- The **timestamp of the most recent one** within the time window.

4. **Big Data vs Traditional data approach**

Traditional Data	Big Data
Traditional data is generated at the enterprise level.	Big data is generated outside the enterprise level.
Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
Traditional database system deals with structured data.	Big data systems deal with structured, semi-structured, database, and unstructured data

Traditional data is generated per hour or per day or more.	But big data is generated more frequently, mainly per seconds.
Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.
Data integration is very easy.	Data integration is very difficult.
Normal system configuration is capable of processing traditional data.	High system configuration is required to process big data.
Traditional data is typically stored in relational databases and processed using SQL-based technologies.	Big Data systems use distributed and scalable storage and processing frameworks like Hadoop, Spark, and NoSQL databases.
Traditional database tools are required to perform any database operation.	Special kinds of database tools are required to perform any database schema-based operation.
Normal functions can manipulate data.	Special kinds of functions can manipulate data.
Its data model is strict schema based and it is static.	Its data model is a flat schema based and it is dynamic.
Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.
Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.
It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
Its data sources includes ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

5. Word count algorithm

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

6. Sliding Window

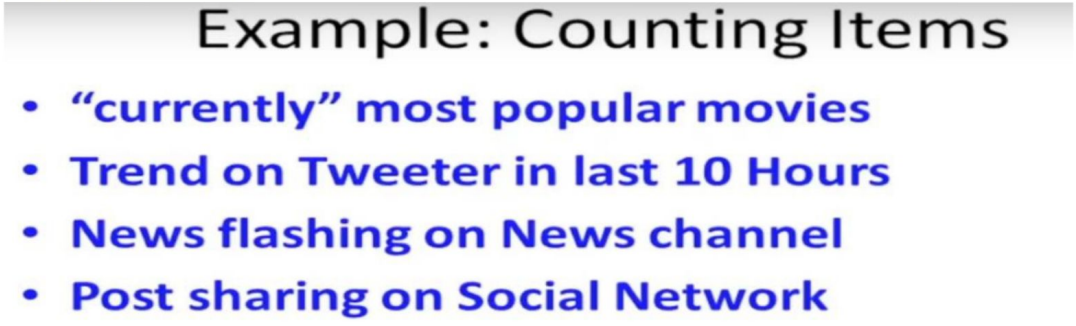
In the context of Big Data and data processing, a sliding window is a technique used for analyzing data over a continuous or rolling time or data range. It involves a fixed-size "window" that moves or "slides" over a larger dataset, allowing you to perform operations or calculations on the data within the window as it advances through the dataset.

1. Real-Time Data Analysis: Sliding windows are often used in real-time or stream processing to analyze data as it arrives, making it suitable for applications that require up-to-date insights.

2. Fixed Size: The window has a fixed size, which can be defined in terms of time (e.g., the last 5 minutes of data) or data points (e.g., the last 1,000 records). The size is typically chosen based on the specific use case and the desired level of analysis.

3. Overlapping Windows: In some cases, windows may overlap to ensure continuity in the analysis. For example, you might have a new window every minute with a 30-second overlap for smooth transitions.

7. Decaying Window



refer to a technique for giving more weight to recent data while gradually reducing the importance of older data as time progresses. commonly used in scenarios where you want to perform calculations or analysis on a continuous data stream and need to focus on the most recent information while still considering historical data.
Applications- financial markets, network monitoring, or social media analytics.

The decay in the importance of older data is typically implemented using an exponential decay function. The decay function assigns a weight to each data point, and the weight decreases exponentially as the data point ages. Exponential decay is chosen because it allows for a straightforward mathematical representation of how the data's importance diminishes over time.

Advantage - we don't need to worry about older elements going out of the window on arrival of new elements.

8. Difference between Sliding window and Decaying window

Sliding window and decaying window are two techniques used in data analysis, particularly in the context of time-series data and streaming data processing.

Sliding Window:

Fixed Window Size: Sliding windows have a fixed size or duration, which remains constant as it moves through the data stream. For example, a sliding window of size 5 minutes will always cover a 5-minute time interval, and it advances over the data with each new interval.

Equal Weight for Data: In a sliding window, all data points within the window have equal weight or significance in the analysis. Older data points are not given less importance compared to more recent data points.

Commonly Used for: Sliding windows are commonly used for tasks such as monitoring, aggregation, and analysis of recent data, anomaly detection, and trend analysis. They are suited for real-time analytics when it is important to have consistent and equally weighted data across the window.

Decaying Window:

Variable Weight for Data: In a decaying window, data points have varying weights based on their recency. Recent data points are given more weight, while older data points are gradually assigned less importance or decay in significance over time.

Adaptive to Change: Decaying windows adapt to changing data patterns and prioritize recent data, which can be particularly useful when dealing with evolving data distributions.

Commonly Used for: Decaying windows are often used for tracking trends and identifying short-term patterns. They are valuable for applications where older data becomes less relevant over time, such as social media trends, stock price volatility, and clickstream analysis.

9. Cost of Exact Count

In Big Data Analytics (BDA), when counting distinct items, such as unique elements in a dataset, you have the option to use exact counting or approximate counting methods. Each approach comes with its own costs and benefits. Let's discuss the cost of exact counting in comparison to both exact count and approximate count methods:

Exact Count:

Cost: The cost of exact counting in BDA is often high because it requires processing and maintaining a massive amount of data. For example, if you're counting distinct words in a very large text corpus, you need to keep track of every unique word, which can be extremely resource-intensive.
Benefits: Exact counting provides highly accurate results, ensuring that you get the precise count of distinct items in your data. This is important for applications where precision is critical.
Approximate Count (Probabilistic Counting):

Cost: Approximate counting methods are generally more resource-efficient than exact counting. They use probabilistic data structures and algorithms that require less memory and computation. Examples of approximate counting techniques include HyperLogLog and Count-Min Sketch.
Benefits: Approximate counting is suitable for scenarios where an exact count is not necessary, or where you are willing to tolerate a small amount of error in the results. These methods can provide a good estimate of the distinct count with significantly reduced computational requirements.

10. Difference between DSMS & DBMS

DBMS	DSMS
DBMS refers to Data Base Management System.	DSMS refers to Data Stream Management System.
Data Base Management System deals with persistent data.	Data Stream Management System deals with stream data.
In DBMS random data access takes place.	In DSMS sequential data access takes place.
It is based on Query Driven processing model i.e called pull based model.	It is based on Data Driven processing model i.e called push based model.
In DBMS query plan is optimized at beginning/fixed.	DSMS is based on adaptive query plans.
The data update rates in DBMS is relatively low.	The data update rates in DSMS is relatively high.
In DBMS the queries are one time queries.	But in DSMS the queries are continuous.
In DBMS the query gives the exact answer.	In DSMS the query gives the exact/approximate answer.
DBMS provides no real time service.	DSMS provides real time service.
DBMS uses unbounded disk store means unlimited secondary storage.	DSMS uses bounded main memory means limited main memory.

11. What is DSMS? Tools used in DSMS

A Data Stream Management System (DSMS) is a specialized software framework for processing and analyzing data streams in real-time.

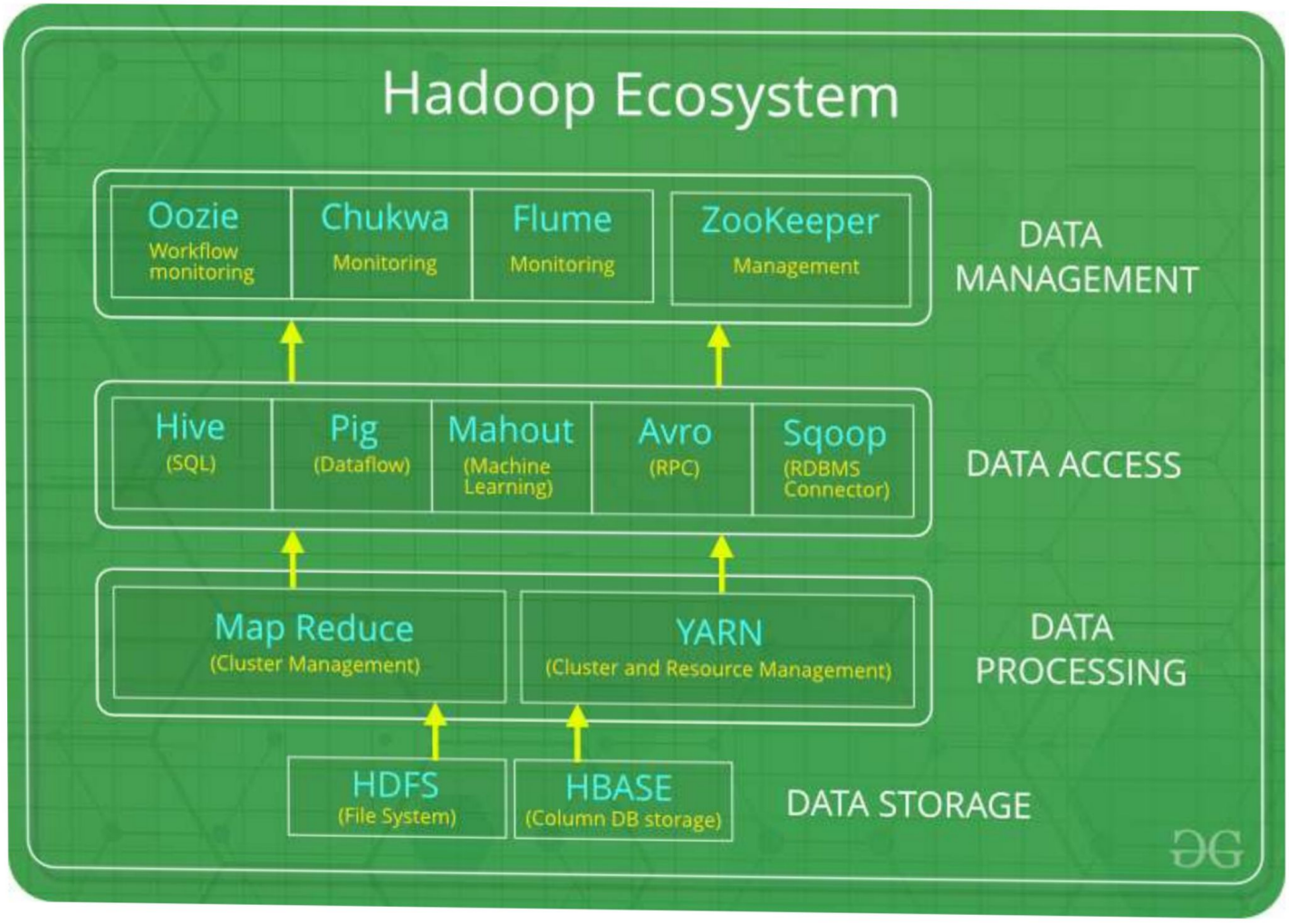
Popular DSMS frameworks and platforms include Apache Kafka Streams, Apache Flink, Apache Storm, Confluent Platform, and commercial solutions like TIBCO StreamBase and IBM Streams.

12. Hadoop and mapreduce

Based on	Hadoop	MapReduce
Definition	The Apache Hadoop is a software that allows all the distributed processing of large data sets across clusters of computers using simple programming	MapReduce is a programming model which is an implementation for processing and generating big data sets with distributed algorithm on a cluster.
Meaning	The name “Hadoop” was named after Doug cutting’s son’s toy elephant. He named this project as “Hadoop” as it was easy to pronounce.	The “MapReduce” name came into existence as per the functionality itself of mapping and reducing in key-value pairs.
Framework	Hadoop not only has storage framework which stores the data but creating name node’s and data node’s it also has other frameworks which include MapReduce itself.	MapReduce is a programming framework which uses a key, value mappings to sort/process the data
Invention	Hadoop was created by Doug Cutting and Mike Cafarella.	Mapreduce was invented by Google.
Features	Hadoop is Open Source Hadoop cluster is Highly Scalable	Mapreduce provides Fault Tolerance Mapreduce provides High Availability
Concept	The Apache Hadoop is an eco-system which provides an environment which is reliable, scalable and ready for distributed computing.	MapReduce is a submodule of this project which is a programming model and is used to process huge datasets which sit on HDFS (Hadoop distributed file system)

Language	Hadoop is a collection of all modules and hence may include other programming/scripting languages too	MapReduce is basically written in Java programming language
Pre-requisites	Hadoop runs on HDFS (Hadoop Distributed File System)	MapReduce can run on HDFS/GFS/NDFS or any other distributed system for example MapR-FS

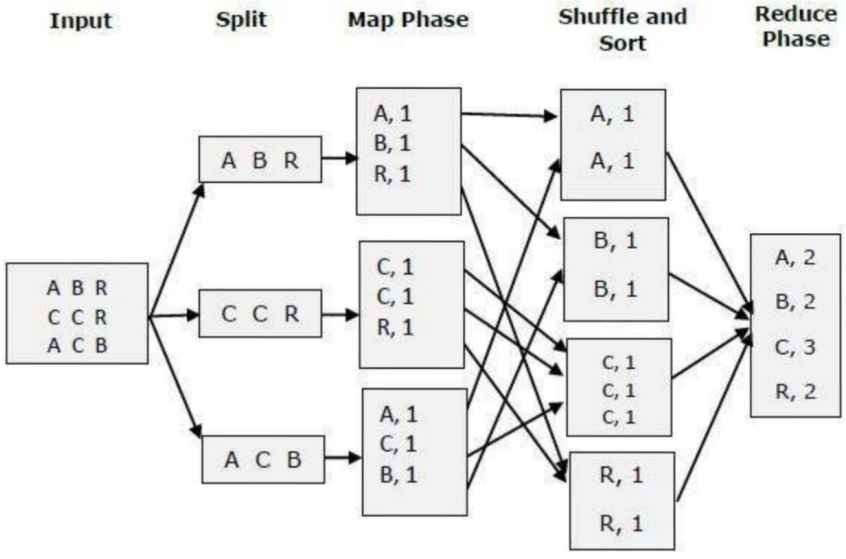
13. HDFS - types and components
14. Hadoop ecosystem



The Hadoop ecosystem is a collection of open-source software tools, frameworks, and libraries that work together with the Hadoop platform to store, process, and analyze big data. It's designed to handle the various aspects of big data management and analytics. Here are some key components of the Hadoop ecosystem:

Hadoop Distributed File System (HDFS): HDFS is the primary storage component of Hadoop. It's a distributed file system designed to store large volumes of data across a cluster of machines. HDFS automatically replicates data for fault tolerance.

MapReduce: MapReduce is a programming model and processing framework for distributed data processing. It's used to break down data processing tasks into smaller, parallelizable steps. While MapReduce was one of the earliest components of Hadoop, it's been largely superseded by more modern data processing frameworks like Apache Spark.



YARN: Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
Consists of three major components i.e.

Resource Manager
Nodes Manager
Application Manager

Apache Spark: Spark is a fast and general-purpose cluster computing framework. It provides in-memory processing, making it much faster than traditional MapReduce. It includes libraries for data processing, machine learning, and graph analysis.

Apache Hive: Hive is a data warehousing and SQL-like query language system for Hadoop. It allows users to query and analyze data stored in Hadoop using SQL-like syntax.

Apache Pig: Pig is a high-level platform for creating MapReduce programs used for data processing. It provides a scripting language called Pig Latin for expressing data transformations.

Apache HBase: HBase is a NoSQL database that runs on top of HDFS. It is suitable for real-time, random read/write access to large data sets.

Apache ZooKeeper: ZooKeeper is a distributed coordination service. It helps manage distributed systems by providing distributed synchronization and configuration maintenance.

Apache Mahout: Mahout is a machine learning library for Hadoop. It offers scalable implementations of various machine learning algorithms.

Apache Flume: Flume is used for collecting, aggregating, and moving large amounts of log data into Hadoop for storage and analysis.

Apache Sqoop: Sqoop is a tool for efficiently transferring bulk data between Hadoop and structured data stores like relational databases.

Oozie: Oozie is a workflow scheduler system for managing Hadoop jobs. It allows you to define and coordinate various data processing tasks.

Apache Kafka: While not part of the Hadoop project, Kafka is often used with Hadoop for real-time data streaming and event processing.

15. How is Hadoop and Big Data related?

Hadoop is a technology designed to handle very large and complex datasets, which is what we call "big data." It does this by allowing data to be stored and processed across multiple computers in a network, making it easier to manage and analyze huge amounts of information. Hadoop is closely related to big data because it provides the tools and infrastructure needed to work with this kind of data effectively. It's like a giant toolbox for managing and making sense of big data.

Features	Big Data	Hadoop
Definition	Big Data refers to a large volume of both structured and unstructured data.	Hadoop is a framework to handle and process this large volume of Big data
Significance	Big Data has no significance until it is processed and utilized to generate revenue.	It is a tool that makes big data more meaningful by processing the data.
Storage	It is very difficult to store big data because it comes in structured and unstructured form.	Apache Hadoop HDFS is capable of storing big data.
Accessibility	When it comes to accessing the big data, it is very difficult.	Hadoop framework lets you access and process the data very fast when compared to other tools.

16. Hadoop core components

Hadoop is an open-source framework designed for distributed storage and processing of big data.
Data node, Name node, job tracker, task tracker

Features of Hadoop
1. Open Source: Hadoop is open-source, which means it is free to use.
2. Highly Scalable Cluster: highly scalable model. A large amount of data is divided into multiple inexpensive machines in a cluster which is processed parallelly.
3. Fault Tolerance is Available: Hadoop uses commodity hardware(inexpensive systems) which can be crashed at any moment. In Hadoop data is replicated on various DataNodes in a Hadoop cluster which ensures the availability of data if somehow any of your systems got crashed.
4. Provides Faster Data Processing: Hadoop uses a HDFS(Hadoop Distributed File System). In DFS(Distributed File System) a large size file is broken into small size file blocks then distributed among the Nodes available in a Hadoop cluster, as this massive number of file blocks are processed parallely.

17. Yarn, resource manager, node manager.

18. **Phases of Mapreduce**
19. **Word Count using MapReduce**

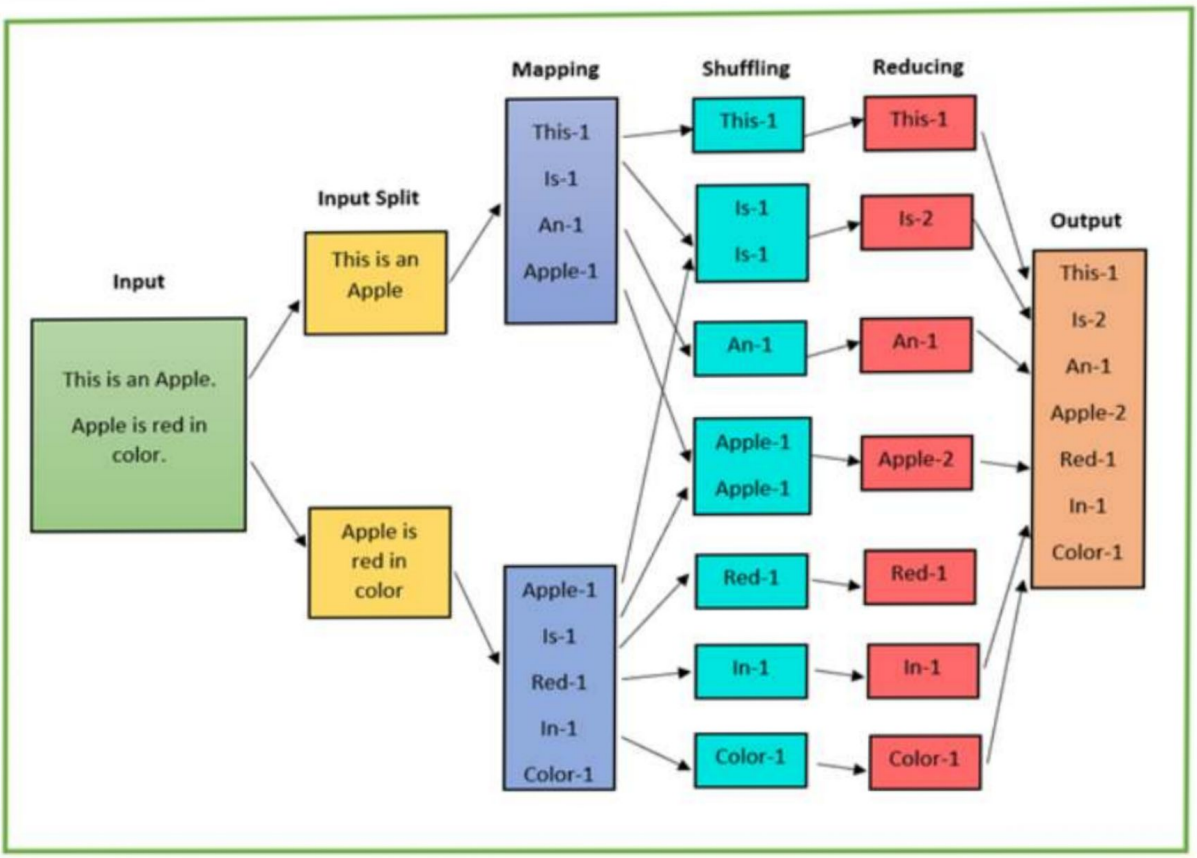
MapReduce is a programming model for processing and generating large datasets that can be executed on Hadoop. It divides tasks into two stages - the Map stage for data filtering and sorting, and the Reduce stage for summarizing the results.

Splitting- the input data is divided into smaller chunks, and a map function is applied to each chunk independently
Mapping- The map function takes the input data and produces a set of key-value pairs
Shuffling- the framework performs shuffling and sorting, where it groups together key-value pairs based on their keys.
Sorting-
Reducing- the reduce function is applied to each group of key-value pairs with the same key, allowing you to aggregate or summarize the data.

Mapper: In the Mapper phase, you read input text data, tokenize it into words, and emit key-value pairs, where the key is the word and the value is 1 (to indicate a count of 1 for each word occurrence).

Shuffle and Sort: The MapReduce framework automatically groups and sorts the key-value pairs generated by the mappers, so that all occurrences of the same word are together.

Reducer: In the Reducer phase, you sum up the values (which are 1s) for each key (word). This effectively counts the occurrences of each word.



- ➢ Spam Detection: In email and content filtering systems, analyzing the frequency of words in messages helps identify spammy or suspicious content.
- ➢ Social Media Analytics: Analyzing word frequencies in social media posts can provide insights into trending topics, sentiment analysis, and identifying popular hashtags.
- ➢ Plagiarism Detection: By comparing the word frequencies and patterns in documents, it's possible to identify potential cases of plagiarism.

Sum of squares using MapReduce

Map Phase:

1. Input Data: Your input data consists of a list of numbers (e.g., [1, 2, 3, 4, 5]).
2. Map Function: In the Map phase, each number is mapped to its square. For each input number `x`, the map function emits a key-value pair, where the key is typically constant (to ensure that all results are sent to the same reducer) and the value is the square of the number. For example, you emit (1, 1), (1, 4), (1, 9), (1, 16), and (1, 25) for the input [1, 2, 3, 4, 5].

Shuffling and Sorting: The framework performs shuffling and sorting, grouping the key-value pairs by key (in this case, the constant key).

Reduce Phase:

3. Reduce Function: In the Reduce phase, each group of values with the same key is processed by the reducer. The reducer's task is to calculate the sum of the squares for each group. For example, the reducer would calculate the sum of (1, 1, 4, 9, 16, 25) for the constant key 1.

4. Output: The output of the MapReduce job is a single key-value pair, where the key is a constant (e.g., 1), and the value is the sum of squares.

20. Difference between MapReduce and Big Data

21. Column Database
22. Graph Database

NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data. Unlike traditional relational databases that use tables with predefined schemas to store data

Key-value store

This is typically considered the simplest form of NoSQL databases. This schema-less data model is organized into a dictionary of key-value pairs, where each item has a key and a value. The key could be like something similar found in a SQL database, like a shopping cart ID, while the value is an array of data, like each individual item in that user's shopping cart.

Document store

The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, or XML documents.

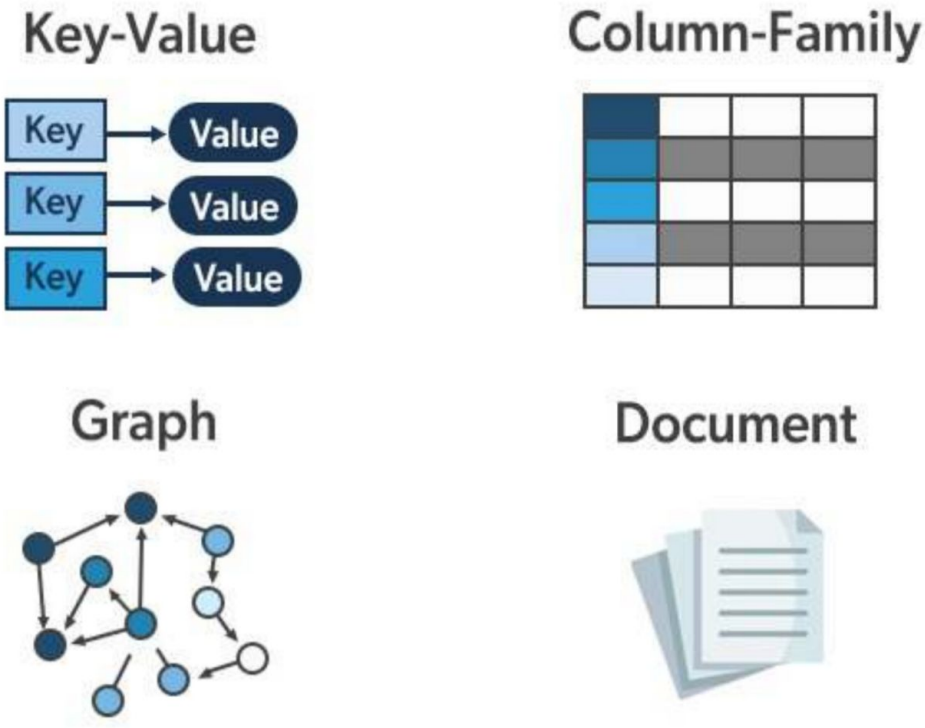
column store

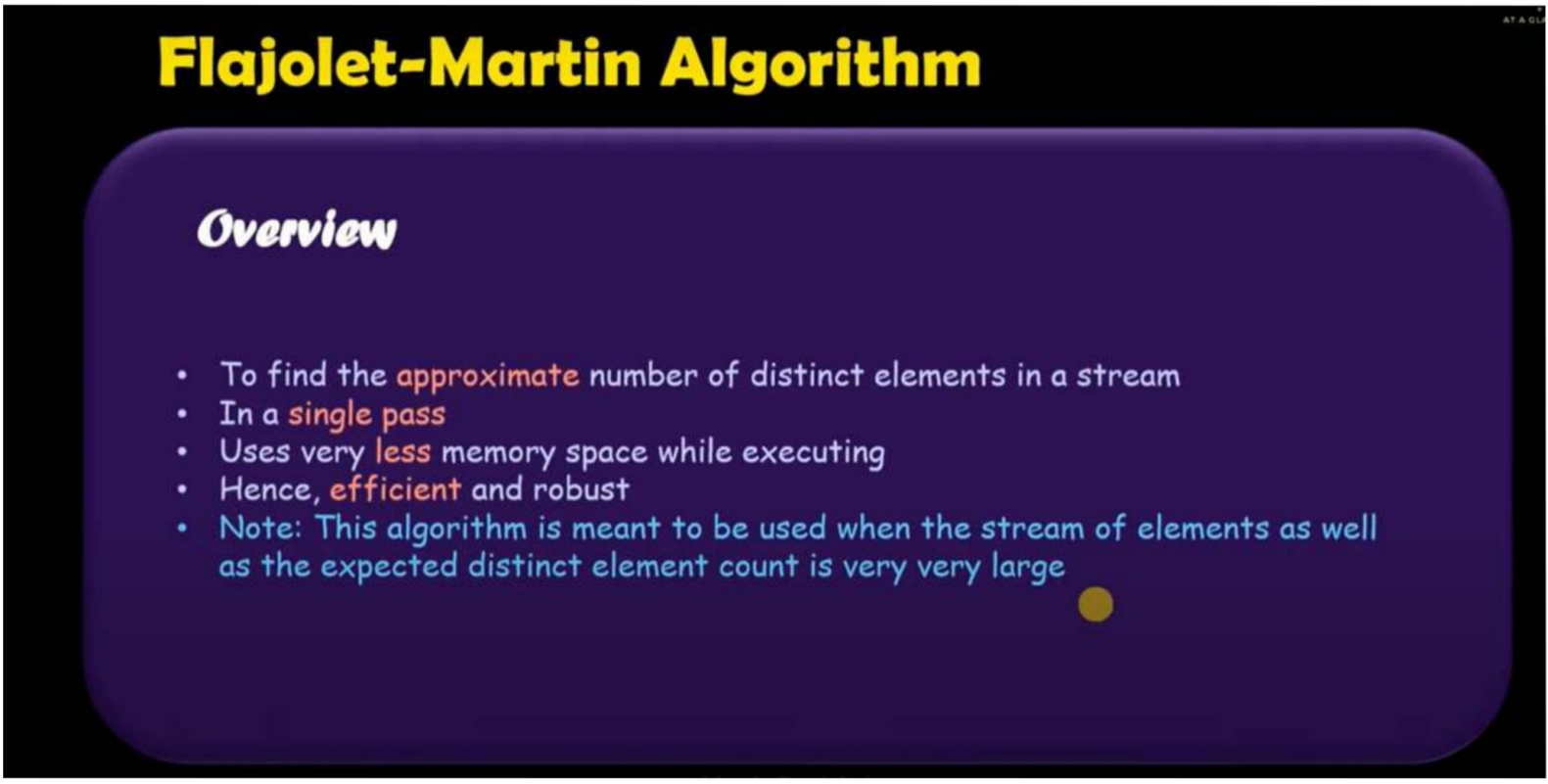
These databases store information in columns, enabling users to access only the specific columns they need without allocating additional memory on irrelevant data. This database tries to solve for the shortcomings of key-value and document stores, but since it can be a more complex system to manage, it is not recommended for use for newer teams and projects.

Graph store

Graph-based databases focus on the relationship between the elements. It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships.

NoSQL





23. Content based and collaborative based recommendation

24. Bloom Filtering

Bloom filtering is a data structure and probabilistic algorithm used in Big Data and distributed computing, extremely space efficient and to test whether a given element is a member of a set by using one or more hashing functions and an array of bits. It is called a filter, as it acts as a preliminary test for data entry.

This type of algorithm is widely used in scenarios where false positives are acceptable but false negatives are not. For instance, it is okay (if cumbersome) to raise a false alarm for a security event that has not happened but unacceptable to overlook a real event (i.e., false negative).

Example:
Suppose you have a large dataset of email addresses and want to check if a new email address is already in the dataset. Using a Bloom filter:

Initialization: Create a Bloom filter with a bit array and multiple hash functions, all initially set to 0.

Insertion: To add an email address to the filter, pass it through the hash functions, which generate multiple hash values. Set the corresponding bits in the bit array to 1.

Membership Testing: To check if a new email address is in the dataset, pass it through the same hash functions and check if all corresponding bits in the bit array are set to 1. If any of them are 0, the email address is definitely not in the dataset. If all are 1, the address might be in the dataset (but there's a small probability of a false positive).

Applications:

- **Networking and Security:** Intrusion detection systems use Bloom filters to maintain a list of known malicious IP addresses and quickly identify potentially harmful traffic.
- **Recommendation Systems:** Bloom filters can be used to maintain a list of items a user has already interacted with to prevent showing them again in recommendation systems.
- **Duplicate Detection:** Bloom filters help identify duplicate records in large datasets, such as when processing logs or event streams.

25. NoSQL

Schema Flexibility: schema-less or schema-agnostic. This means that you can store data without the need for a predefined schema, and the data within a NoSQL database can have varying structures.

Data Models: NoSQL databases support various data models, including document-oriented, key-value, column-family, and graph models.

Scalability: are designed for horizontal scalability, allowing organizations to add more servers or nodes to the database cluster

High Performance: NoSQL databases often optimize for high performance, offering low-latency data access. They are well-suited for applications that require real-time or near-real-time data processing.

CAP Theorem: The CAP theorem (Consistency, Availability, Partition Tolerance) is a fundamental concept in NoSQL databases. It states that a distributed system can achieve at most two out of the three CAP properties. NoSQL databases are often designed with a trade-off between these properties in mind, depending on the specific use case.

Use Cases: NoSQL databases are commonly used in applications involving Big Data, content management, real-time analytics, Internet of Things (IoT), mobile apps, social media, e-commerce, and more. They are ideal for scenarios where data is diverse, rapidly changing, and difficult to structure in a traditional relational database.

26. Nosql database architecture

NoSQL databases use document-oriented, key-value, graph, or column-family formats to store data.

- Key-value store pattern is simple and efficient, suitable for fast read and write operations, and highly scalable and fault-tolerant. Ex. DynamoDB, Berkeley DB
  Use Cases: Caching, session management, user profiles, and scenarios that require quick access to frequently accessed data.

- Document store pattern is flexible and suitable for indexing and querying. Ex. MongoDB, CouchDB
  Use Cases: Content management systems, catalogs, blogging platforms, and applications where data structures may evolve over time.

- Column-family store pattern is ideal for big data applications and efficient storage of large, sparse data sets.
  Ex. HBase, Bigtable by Google, Cassandra
  Use Cases: Time series data, event logging, sensor data, and applications requiring high availability and scalability.

- Graph database pattern is suitable for complex, interconnected data sets.
  Ex. Neo4J, FlockDB(Used by Twitter)
  Use Cases: Social networks, recommendation engines, fraud detection, and applications that rely on understanding and analyzing relationships in data.

27. Advantages and disadvantages of Big Data and NoSQL.

Advantages of Big Data:

- Good for handling massive amounts of different types of data.
- Can process data in real-time.
- Can be cost-effective.

- Useful for advanced analytics and predictions.

Disadvantages of Big Data:

- Complex to set up and manage.
- Data security can be challenging.
- Data quality and integration issues.
- Requires specific skills.

Advantages of NoSQL:

- Flexible with changing data structures.
- Scales easily.
- High performance for specific tasks.
- Different types to choose from.

Disadvantages of NoSQL:

- May have consistency issues.
- Learning curve for new data models and languages.
- Some lack transaction support.
- Less mature than traditional databases.

28. NoSQL vs SQL

SQL	NoSQL
RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)	Non-relational or distributed database system.
These databases have fixed or static or predefined schema	They have a dynamic schema
These databases are not suited for hierarchical data storage.	These databases are best suited for hierarchical data storage.
These databases are best suited for complex queries	These databases are not so good for complex queries
Vertically Scalable	Horizontally scalable
Follows ACID property	Follows CAP(consistency, availability, partition tolerance)

Examples: MySQL, PostgreSQL, Oracle, MS-SQL Server, etc	Examples: MongoDB, HBase, Neo4j, Cassandra, etc

29. Case Study: NoSQL Database for a Social Media Platform**

Background:
A growing social media platform with millions of users needs a robust database system to handle user data, posts, comments, and interactions in a scalable and flexible manner.

Business Problem:
The platform wants to:
1. Handle a massive amount of unstructured user-generated content.
2. Ensure high availability and fast response times for its users.

Solution Using NoSQL:
1. Handling Unstructured Content:
- Use a NoSQL document database to store user profiles, posts, and comments. NoSQL's flexible schema allows easy adaptation to evolving data structures.

2. Ensuring High Availability:
- Implement a distributed NoSQL database cluster to ensure the platform remains available even during high traffic periods.

Benefits of NoSQL in this Case:
- NoSQL databases handle unstructured data efficiently.
- They provide high availability and scalability to accommodate a growing user base.

Conclusion:
NoSQL databases are ideal for a social media platform dealing with vast amounts of user-generated content, enabling scalability and high availability to ensure a seamless user experience.

R Data structures

The five basic data structures in R are: Vectors, Lists, Dataframes,Array, Matrices, Factors.
Data structures are used to organize and store data. Some structures require that all members be of the same data type, while others permit multiple data types. For example,
vectors and matrices require all members to be of the same data type,
while lists and data frames permit multiple data types.

R's base data structure can be organized based on their dimensionality (1d, 2d, 3d, Nd) and if they are homogenous or not.

Vectors

A vector is an ordered collection of basic data types of a given length. The only key thing here is all the elements of a vector must be of the identical data type e.g homogeneous data structures. Vectors are one-dimensional data structures.

```
X = c(1, 3, 5, 7, 8)

# Printing those elements in console
print(X)
```

Output:

```
[1] 1 3 5 7 8
```

Lists

A list is a generic object consisting of an ordered collection of objects. Lists are heterogeneous data structures. These are also one-dimensional data structures. A list can be a list of vectors, list of matrices, a list of characters and a list of functions and so on.

```
empId = c(1, 2, 3, 4)
empName = c("Debi", "Sandeep", "Subham", "Shiba")
numberOfEmp = 4
empList = list(empId, empName, numberOfEmp)
print(empList)
```

Output:

```
[[1]]
[1] 1 2 3 4

[[2]]
[1] "Debi"    "Sandeep" "Subham"  "Shiba"

[[3]]
[1] 4
```

Dataframes

Dataframes are generic data objects of R which are used to store the **tabular data.** Dataframes are the foremost popular data objects in R programming because of ease in seeing the data within the tabular form. They are two-dimensional, heterogeneous data structures. These are lists of vectors of equal lengths.

Data frames have the following constraints placed upon them:

- A data-frame must have column names and every row should have a unique name.
- Each column must have the identical number of items.
- Each item in a single column must be of the same data type.
- Different columns may have different data types.
- To create a data frame we use the data.frame() function.

```
Name = c("Amiya", "Raj", "Asish")
Language = c("R", "Python", "Java")
Age = c(22, 25, 45)
df = data.frame(Name, Language, Age)

print(df)
```

Output:

```
  Name Language Age
1 Amiya        R  22
2   Raj   Python  25
3 Asish     Java  45
```

Matrices

A matrix is a rectangular arrangement of numbers in rows and columns. Matrices are two-dimensional, homogeneous data structures.

To create a matrix in R you need to use the function called matrix. The arguments to this matrix() are the set of elements in the vector.

You have to pass how many numbers of rows and how many numbers of columns you want to have in your matrix. By default, matrices are in column-wise order.

```
A = matrix(
    c(1, 2, 3, 4, 5, 6, 7, 8, 9),
    nrow = 3, ncol = 3,
    byrow = TRUE
)
print(A)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Arrays

Arrays are the R data objects which store the data in more than two dimensions.

Arrays are n-dimensional data structures. For example, if we create an array of dimensions (2, 3, 3) then it creates 3 rectangular matrices each with 2 rows and 3 columns.

They are homogeneous data structures.

To create an array in R you need to use the function called array(). The arguments to this array() are the set of elements in vectors and you have to pass a vector containing the dimensions of the array.

```
A = array(
    c(1, 2, 3, 4, 5, 6, 7, 8),
    dim = c(2, 2, 2)
)

print(A)
```

Output:

```
, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Factors

Factors are the data objects which are used to categorize the data and store it as levels. They are useful for storing categorical data.

They can store both strings and integers. They are useful to categorize unique values in columns like “TRUE” or “FALSE”, or “MALE” or “FEMALE”, etc..

They are useful in data analysis for statistical modeling.

To create a factor in R you need to use the function called factor(). The argument to this factor() is the vector.

```
fac = factor(c("Male", "Female", "Male",
               "Male", "Female", "Male", "Female"))

print(fac)
```

Output:

```
[1] Male   Female   Male   Male   Female    Male   Female
Levels: Female Male
```

What does the c () function do in R?

The c function in R programming usually stands for 'combine. ' This function is used to get the output by giving parameters inside the function.

Girvan-Newman algorithm

The Girvan-Newman algorithm is a method used for detecting community structures or clusters within a complex network, such as a social network, transportation network, or communication network.

It aims to identify groups of nodes within the network that are more densely connected to each other than to nodes in other groups.

The algorithm does this by iteratively removing "bridge" edges in the network to reveal its underlying community structure.

Applications:

Social Networks: Identifying communities of users with common interests, friendships, or interactions in social media networks.
Recommendation Systems: Clustering users or items in recommendation systems to provide more personalized recommendations based on community preferences.
Transportation Networks: Analyzing traffic flow and identifying regions with similar commuting patterns in urban transportation networks.