Times asked: 6 times
5 times
4 times
3 times
2 times
1 time

# indicates 5-mark question

Big Data Analytics Question bank

1. Introduction to Big Data and Hadoop

1. Explain four characteristics of Big Data. #
2. Describe the components of Hadoop ecosystem with the help of a diagram.

2. Hadoop HDFS and MapReduce

3. Write a map reduce pseudo code for word count problem. Apply map reduce working on the following document: (PYQs)
   i. "This is NoSQL. NoSQL handles complex data."
   ii. "Big data is powerful. Big data drives decisions."
4. Explain Map Reduce execution pipeline with suitable example.
5. Write a map reduce pseudo code to multiply two matrices. Apply map reduce working to perform following matrix multiplication.

$$M = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \times V = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$$

6. Explain natural join and grouping and aggregation relational algebraic operation using MapReduce.

3. NoSQL

7. Explain various architectural patterns in NoSQL databases.
8. List and explain the core business drivers behind the NoSQL movement. #

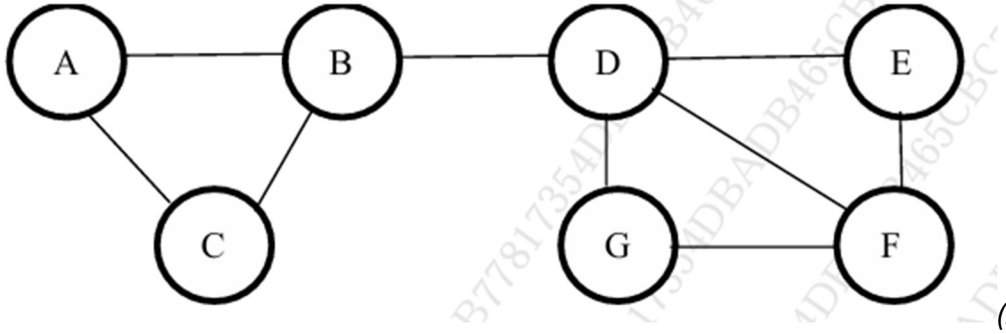
4. Mining Data Streams

9. Show how the Flajolet- Martin algorithm will estimate the number of distinct elements in a stream. PYQs:
   i. Suppose the stream is S = {10, 12, 8, 15, 6, 9, 14, 7}.
   Let hash functions h(x) = 5x + 11 mod 32 and treat the result as a 5-bit binary integer.
   ii. For the stream of integers: 9, 8, 7, 6, 5, 4, 3, 2
   Use the hash function, h(x) = (2x + 1) mod 32 and treat the result as a 5-bit binary integer.
10. Explain the architecture of the data-stream management system with a neat diagram.

11. List down all six constraints that must be satisfied for representing a stream by buckets using DGIM algorithm with examples. #
12. Explain DGIM algorithm for counting ones in a stream with example.
13. Explain the concept of bloom filter with an example. #

5. Real-time Big Data Models

14. Determine communities for the given social network graph using Girvan-Newman algorithm.



(2 times)

15. How is recommendation done based on properties of product (Content-based recommendation). Elaborate with a suitable example.
16. Explain Collaborative filtering in recommendation systems with an example.

6. Data Analytics with R

17. Create data frame in R and perform operations.

I] The project manager at XYZ Ltd., Ms. Meera, is responsible for main details of all active projects. She has organized the project information in the following table:

Project Id	Project Name	Budget	Status
1	CRM Implementation	120000	In Progress
2	Cloud Infrastructure	180000	Completed
3	Network Upgrade	60000	Not Started
4	E-Commerce Platform	220000	Completed
5	Data Analytics	90000	In Progress

i) Create a Data frame in R for the above project data and display the output.

ii) Ms. Meera has recently approved 2 new projects and wants to find their information. The new projects are as follows:

Project Id	Project Name	Budget	Status
6	UX Research	160000	Not Started
7	Cloud Integration	190000	Not Started

iii) Update the Data frame to include the new projects and demonstrate the final output.

II] 1. Create a data frame from the following 4 vectors and demonstrate the output:

emp_id = c(1:5)

emp_name = c("Rick", "Dan", "Michelle", "Ryan", "Gary")

start_date = c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")

salary = c(60000, 45000, 75000, 84000, 20000)

2. Display structure and summary of the above data frame.

3. Extract the emp_name and salary columns from the above data frame.

4. Extract the employee details whose salary is less than or equal to 60000.

18. Write script in R for creating subsets:

Consider the following data frame:

course	id	class	marks
1	11	1	56
2	12	2	75
3	13	1	48
4	14	2	69
5	15	1	84
6	16	2	53

i) Create a subset of course less than 5 by using [ ] brackets and demonstrate the output.

ii) Create a subset where the course column is less than 4 or the class equals to 1 by using subset() function and demonstrate the output.

19. Write the script to sort the values contained in the following vector in ascending order and descending order: (46, 23, 15, 38, 98, 56, 28, 78). Demonstrate the output. #

20. List and explain various types of data structures in R.

21. Name and explain the operators used to form data subsets in R. #

22. Explain the various functions provided by R to combine different sets of data. #

	1	2	3	4	5	6
2025 May	15	25	15	25	20	20
2024 Dec	15	20	20	25	20	20
2024 May	10	20	15	25	20	30
2023 Dec	10	25	10	25	20	30
2023 May	10	20	15	25	20	30
2022 Dec	10	20	15	25	20	30
Estimate	10	20	15	25	20	30
Total	80	130	80	150	120	160

Asked once:

# indicates 5-mark question

1. Introduction to Big Data and Hadoop

1. Secondary Name is a backup of Name node. Is this statement True or False? Justify your answer. #
2. Explain how big data problems are handled by Hadoop system. #
3. What is the basic difference between traditional RDBMS and Hadoop? #
4. Distinguish between Name node and Data node. #
5. What are the Core Hadoop components? Explain in detail.

2. Hadoop HDFS and MapReduce

6. Explain selection and projection relational algebraic operation using MapReduce.
7. Explain how node failure is handled in Hadoop. #
8. What is function of Map Tasks in the Map Reduce framework? Explain with the help of an example. #
9. Why is HDFS more suited for applications having large datasets and not when there are small files? Elaborate. #
10. Name the three ways that resources can be shared between computer systems. Name the architecture used in big data solutions.

3. NoSQL

11. Demonstrate how business problems have been successfully solved faster, cheaper and more effectively considering NoSQL Google's Big case study. Also illustrate the business drivers and the findings in it. #
12. Demonstrate how business problems have been successfully solved faster, cheaper and more effectively considering NoSQL Google's MapReduce case study. Also illustrate the business drivers and the findings in it. #
13. Explain CAP. How is CAP different from ACID property in databases? #
14. Describe the four ways by which big data problems are handled by NoSQL.

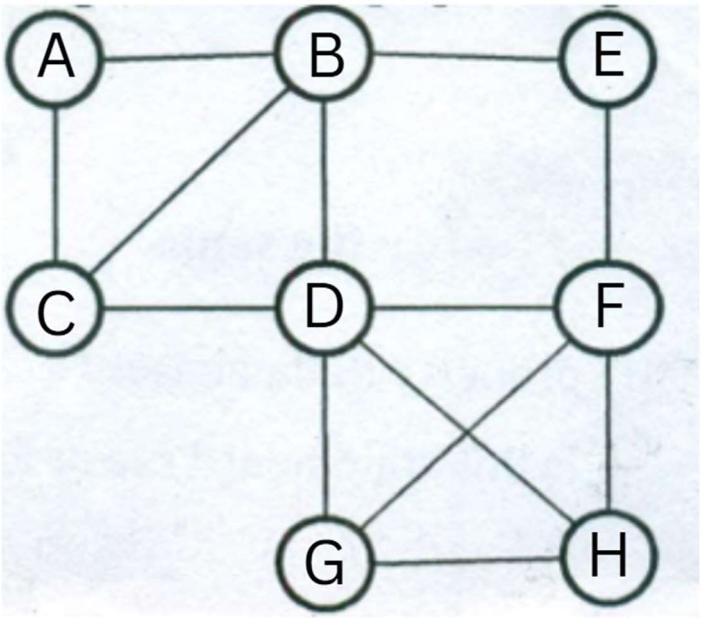
4. Mining Data Streams

15. Create a Bloom filter with the following parameters:
    Size of the bit array m = 8
    Hash functions:
    a. $h_1(x) = x \bmod m$
       $h_2(x) = (2x + 1) \bmod m$
       $h_3(x) = (3x + 2) \bmod m$
       (i) Insert the following elements into the Bloom filter: 12, 25, 30, 5
       (ii) Check if the following elements are present in the Bloom filter: 6, 55
       (iii) Discuss the results of your checks, identifying which elements are true positive and which is true negative.

16. List and explain the different issues and challenges in data stream query processing. #

5. Real-time Big Data Models

17. What is a recommendation system? How is classification algorithm used in a recommendation system?
18. Write an algorithm for the Clique Percolation Method and discover the communities in the given below graph using Clique Percolation Method with clique k=3.



6. Data Analytics with R

19. Describe applications of data visualization.
20. List and explain various functions that allow users to handle data in R workspace with appropriate examples.
21. What are the advantages of using functions over scripts? #

22. Write a script to create a dataset named data1 in R containing the following text:
    Text: 2, 3, 4, 5, 6.7, 7, 8.1, 9 #
23. Suppose you have two datasets A and B.
    Dataset A has the following data: 6 7 8 9
    Dataset B has the following data: 1 2 4 5
    Which function is used to combine the data from both datasets into dataset C?
    Demonstrate the function with the input values and write the output. #

24. The data analyst of Argon technology Mr. John needs to enter the salaries of 10 employees in R. The salaries of the employees are given in the following table:

Sr. No.	Name of employees	Salaries
1	Vivek	21000
2	Karan	55000
3	James	67000
4	Soham	50000
5	Renu	54000
6	Farah	40000
7	Hetal	30000
8	Mary	70000
9	Ganesh	20000
10	Krish	15000

i. Which R command will Mr. John use to enter these values demonstrate the output.
ii. Now Mr. John wants to add the salaries of 5 new employees in the existing table, which command he will use to join datasets with new values in R. Demonstrate the output.

25. The following table shows the number of units of different products sold on different days:

Product	Monday	Tuesday	Wednesday	Thursday	Friday
Bread	12	3	5	11	9
Milk	21	27	18	20	15
Cola Cans	10	1	33	6	12
Chocolate bars	6	7	4	13	12
Detergent	5	8	12	20	23

Create five sample numeric vectors from this data. #

Big Data Analytics Answer bank

# indicates 5-mark question

1. Introduction to Big Data and Hadoop

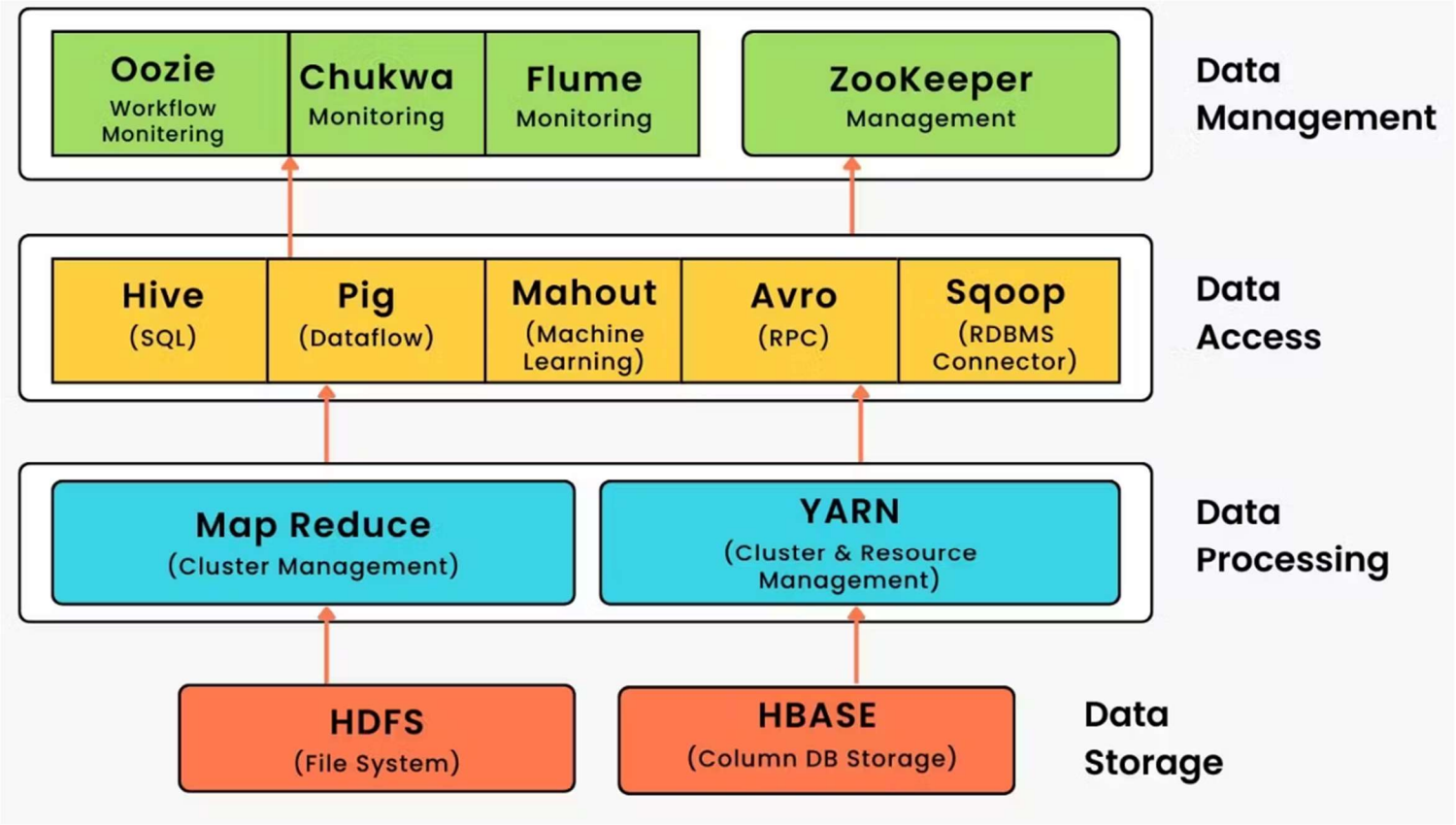
1. Explain four characteristics of Big Data. #

Big Data refers to extremely large and complex data sets that cannot be efficiently stored, processed, or analyzed using traditional database management tools. It is defined by certain key characteristics that highlight its scale and complexity, commonly known as the Four V's: Volume, Velocity, Variety, and Veracity.

Four Characteristics of Big Data

1. **Volume:** Indicates the massive scale of data generated from sources like social media, sensors and transactions. These data can range from terabytes to zettabytes, far beyond the limits of traditional databases.
2. **Velocity:** Refers to the high speed at which data is produced, collected, and processed—often in real time or near real time. Examples include live video streams or sensor data.
3. **Variety:** Represents the diversity of data formats and structures, including structured (tables), semi-structured (XML, JSON), and unstructured (text, images, audio, video) data.
4. **Veracity:** Describes the reliability and trustworthiness of data. Big Data often includes noise, missing values, or inconsistencies, making data quality an important concern.

2. Describe the components of Hadoop ecosystem with the help of a diagram.



The Hadoop Ecosystem is a framework of open-source tools built around Hadoop to store, process, manage, and analyze large-scale data efficiently.
It is organized into four layers: Data Storage, Data Processing, Data Access, and Data Management.

1. Data Storage Layer

- **HDFS (Hadoop Distributed File System)**
  - Distributed storage system that stores data across multiple nodes.
  - Provides fault tolerance by replicating data blocks.
  - Stores very large files by splitting them into blocks.
- **HBase**
  - A NoSQL, column-oriented database that runs on top of HDFS.
  - Supports real-time read/write access to large datasets.

2. Data Processing Layer

- **MapReduce**
  - Programming model for parallel data processing.
  - Splits jobs into Map tasks (processing) and Reduce tasks (aggregation).
- **YARN (Yet Another Resource Negotiator)**
  - Manages cluster resources and schedules jobs.
  - Allows multiple data processing engines (MapReduce, Spark) to run on Hadoop.

3. Data Access Layer

- **Hive** – Data warehouse tool with SQL-like query language (HiveQL).
- **Pig** – High-level scripting language (Pig Latin) for data transformation.
- **Mahout** – Machine learning library for building scalable algorithms.
- **Avro** – Data serialization system for data exchange between programs.
- **Sqoop** – Transfers bulk data between Hadoop and relational databases.

4. Data Management Layer

- **Oozie** – Workflow scheduler for managing Hadoop jobs.
- **Chukwa** – Tool for collecting and monitoring large amounts of log data.
- **Flume** – Distributed service for ingesting streaming data into HDFS.
- **ZooKeeper** – Coordination and configuration management service for distributed systems.

2. Hadoop HDFS and MapReduce

3. Write a map reduce pseudo code for word count problem. Apply map reduce working on the following document: (PYQs)
    i. “This is NoSQL. NoSQL handles complex data.”
    ii. “Big data is powerful. Big data drives decisions.”

MapReduce Pseudocode for Word Count

Map Function:

```
Map(key, value):
  for each word w in value:
    Emit(w, 1)
```

Reduce Function:

```
Reduce(word, counts):
  total = sum(counts)
  Emit(word, total)
```

Input Document:

i. “This is NoSQL. NoSQL handles complex data.”

1. Map Phase

Each word is emitted as (word, 1):

Mapper Output
(This, 1)
(is, 1)
(NoSQL, 1)
(NoSQL, 1)
(handles, 1)
(complex, 1)
(data, 1)

2. Shuffle and Sort Phase

Group all identical words together:

Word	List of Values
This	[1]
is	[1]
NoSQL	[1, 1]
handles	[1]
complex	[1]
data	[1]

3. Reduce Phase

Sum all values for each key:

Word	Count
This	1
is	1
NoSQL	2
handles	1
complex	1
data	1

Final Output:

(This, 1)

(is, 1)

(NoSQL, 2)

(handles, 1)

(complex, 1)

(data, 1)

4. Explain Map Reduce execution pipeline with suitable example.

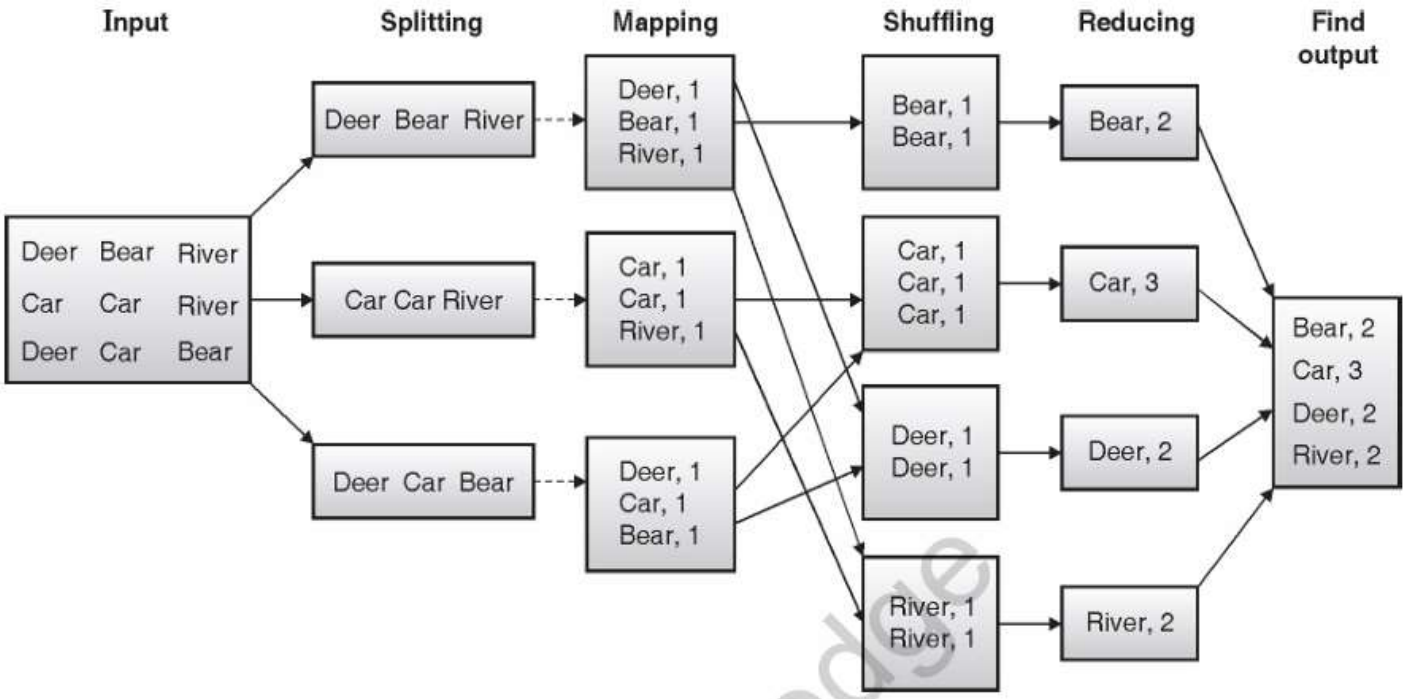


Fig. 3.2.1 : MapReduce process for word frequency count

1. Input Phase

- The input dataset is stored in the HDFS (Hadoop Distributed File System).
- The large file is divided into blocks which are processed independently.

Example Input:

Deer Bear River

Car Car River

Deer Car Bear

2. Splitting Phase

- The input data is divided into smaller logical chunks called input splits.
- Each split typically corresponds to a line, paragraph, or block of data.

Example:

- Split 1: Deer Bear River
- Split 2: Car Car River
- Split 3: Deer Car Bear

Each split is sent to a different mapper.

3. Mapping Phase

- Each Mapper processes one split of input and produces key–value pairs (K, V).
- In the word count problem, each word is the key, and the value is 1 (indicating one occurrence).

Output of Mapping:

(Deer, 1), (Bear, 1), (River, 1)

(Car, 1), (Car, 1), (River, 1)

(Deer, 1), (Car, 1), (Bear, 1)

4. Shuffling and Sorting Phase

- The framework groups together all pairs with the same key across all mappers.
- All values belonging to the same key are collected together.

Shuffled Output:

Bear → [1, 1]

Car → [1, 1, 1]

Deer → [1, 1]

River→ [1, 1]

This step ensures that all data for one key goes to the same reducer.

5. Reducing Phase

- Each Reducer processes one key and its list of values.
- The reducer aggregates values to produce the final count.

Reduce Function Example:

sum(values)

Reduced Output:

Bear, 2

Car, 3
Deer, 2
River, 2

6. Output Phase

- The final (key, value) pairs are written back to HDFS as the output file.

Final Output:

Bear 2

Car 3

Deer 2

River 2

5. Write a map reduce pseudo code to multiply two matrices. Apply map reduce working to perform following matrix multiplication.

i. $\begin{matrix}1 & 2\\ 3 & 4\end{matrix} \times \begin{matrix}6 & 7\\ 8 & 9\end{matrix}$

ii. $M = \begin{matrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{matrix} \times V = \begin{matrix}1\\ 2\\ 3\end{matrix}$

Map pseudo code:

for each element A[i][k] in matrix A:

  emit (k, ('A', i, A[i][k]))

for each element B[k][j] in matrix B:

  emit (k, ('B', j, B[k][j]))

Mapper for Matrix A → (k,v) = ((i,k), (A,j,Aij)) for all k

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

k = 1

- i=1 j=1 → ((1,1), (A,1,1))
  j=2 → ((1,1), (A,2,2))
- i=2 j=1 → ((2,1), (A,1,3))
  j=2 → ((2,1), (A,2,4))

k = 2

- i=1 j=1 → ((1,2), (A,1,1))
  j=2 → ((1,2), (A,2,2))
- i=2 j=1 → ((2,2), (A,1,3))
  j=2 → ((2,2), (A,2,4))

Mapper for Matrix B → (k,v) = ((i,k), (B,j,Bjk)) for all i

$$\begin{bmatrix}6 & 7\\ 8 & 9\end{bmatrix}$$

i = 1

- j = 1 k = 1 → ((1, 1), (B, 1, 6))
  k = 2 → ((1, 2), (B, 1, 7))
- j = 2 k = 1 → ((1, 1), (B, 2, 8))
  k = 2 → ((1, 2), (B, 2, 9))

i = 2

- j = 1 k = 1 → ((2, 1), (B, 1, 6))
  k = 2 → ((2, 2), (B, 1, 7))
- j = 2 k = 1 → ((2, 1), (B, 2, 8))
  k = 2 → ((2, 2), (B, 2, 9))

Reduce pseudo code:

for each key k:

listA = all ('A', i, A[i][k]) values for this k

listB = all ('B', j, B[k][j]) values for this k

for each (i, Aik) in listA:

for each (j, Bkj) in listB:

emit ((i, j), Aik * Bkj)

Reducer(k, v) formula:

For (i, k), Make sorted A list and B list

Sum($A_{ij} \times B_{jk}$) for all j

Output ⇒((i, k), sum)

From Mapper results, we have 4 common keys: (1,1),(1,2),(2,1),(2,2)

1. **(1,1):**
   A list = {(A, 1, 1), (A, 2, 2)}
   B list = {(B, 1, 6), (B, 2, 8)}

$A_{ij} \times B_{jk} = (1 \times 6) + (2 \times 8) = 6 + 16 = 22$

2. **(1,2):**
   A list = {(A, 1, 1), (A, 2, 2)}
   B list = {(B, 1, 7), (B, 2, 9)}

$A_{ij} \times B_{jk} = (1 \times 7) + (2 \times 9) = 7 + 18 = 25$

3. **(2,1):**
   A list = {(A, 1, 3), (A, 2, 4)}
   B list = {(B, 1, 6), (B, 2, 8)}

$A_{ij} \times B_{jk} = (3 \times 6) + (4 \times 8) = 18 + 32 = 50$

4. **(2,2):**
   A list = {(A, 1, 3), (A, 2, 4)}
   B list = {(B, 1, 7), (B, 2, 9)}

$A_{ij} \times B_{jk} = (3 \times 7) + (4 \times 9) = 21 + 36 = 57$

Final Output:

((1,1), 22), ((1,2), 25), ((2,1), 50), ((2,2), 57)

$$= \begin{bmatrix} 22 & 25 \\ 50 & 57 \end{bmatrix}$$

6. Explain natural join and grouping and aggregation relational algebraic operation using MapReduce.

1. Natural Join using MapReduce

A Natural Join combines two relations (tables) based on their common attribute(s) and merges tuples having the same value for those attributes.

Example:

Let
R(A, B) = {(1, x), (2, y)}
S(B, C) = {(x, 100), (y, 200)}

Expected Result:

R ⋈ S → {(1, x, 100), (2, y, 200)}

MapReduce Implementation:

Map Phase:

From relation R, emit (key = B, value = (“R”, other attributes))
From relation S, emit (key = B, value = (“S”, other attributes))

Map Output:

(x, ("R",1)) , (y,("R",2))

(x, ("S",100)) , (y,("S",200))

Shuffle Phase:

Group values by key (common attribute).

Reduce Phase:

For each key, find matching tuples from R and S, and combine them.

Reducer Output:

(x, (1,100)), (y, (2,200))

2. Grouping and Aggregation using MapReduce

These operations group tuples based on an attribute and perform aggregate functions like COUNT, SUM, AVG, MIN, or MAX.

Example:

Relation Sales(Product, Amount)

(Pen, 10), (Book, 20), (Pen, 15), (Book, 25)

Goal: Calculate the total amount of sales for each product.

MapReduce Implementation:

Map Phase:

Emit key = Product, value = Amount

Map Output: (Pen,10), (Book,20), (Pen,15), (Book,25)

Shuffle Phase:

Group by key

(Pen,[10,15]), (Book,[20,25])

Reduce Phase:

Apply aggregation function (SUM in this case).

Reducer Output: (Pen,25), (Book,45)

3. NoSQL

7. Explain various architectural patterns in NoSQL databases.

Key-Value store pattern:

Data is stored as a collection of key–value pairs, where each key acts as a unique identifier to retrieve its associated value. The system functions like a dictionary or hash map, making data retrieval extremely fast.

Example:

Key	Value
Name	John
Age	21
Hobbies	Reading, Football

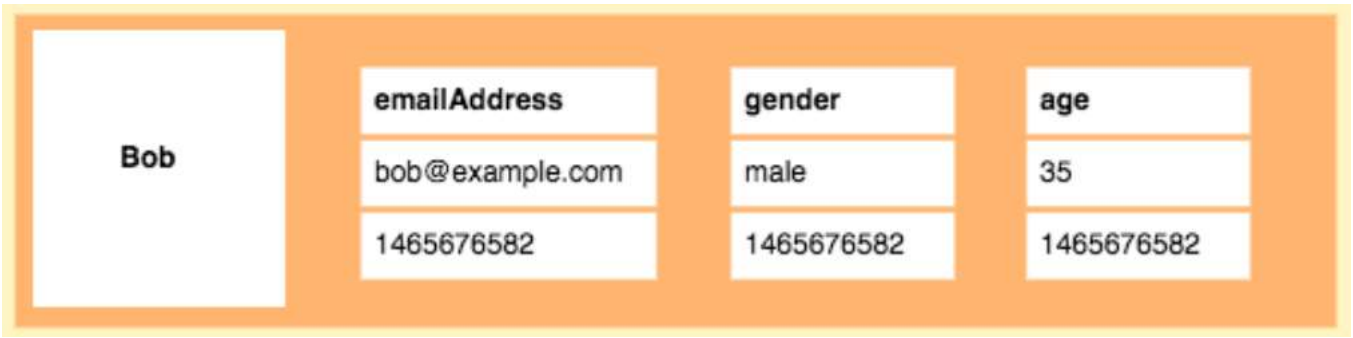
Each key uniquely identifies its corresponding value. The database can quickly fetch or update data by querying the key. For example, by querying the key “Name”, it retrieves the value “John.”

Example databases: Redis, Riak.

Column-Family Store Pattern

Column-family databases store data by columns instead of rows and are based on Google’s BigTable. Values of a single column are stored together on disk, which allows fast retrieval. The data is still organized in a table-like structure but optimized for large-scale read and write operations.

Example



Each row key (like “Bob”) uniquely identifies one record, and all details related to that person are stored under it as columns such as emailAddress, gender, and age. Each column stores a value along with a timestamp showing when it was last updated.

Example databases: Apache Cassandra, HBase.

Document Store Pattern:

A document store stores data in the form of documents, usually using formats like JSON, BSON, or XML. Each document holds key–value pairs and supports a flexible, schema-less structure. It is ideal for managing semi-structured data like user profiles or product catalogues.

Example:


```
{
  "Name": "Rohan",
  "Age": 22,
  "City": "Mumbai",
  "Skills": ["Football", "Coding"]
}
```

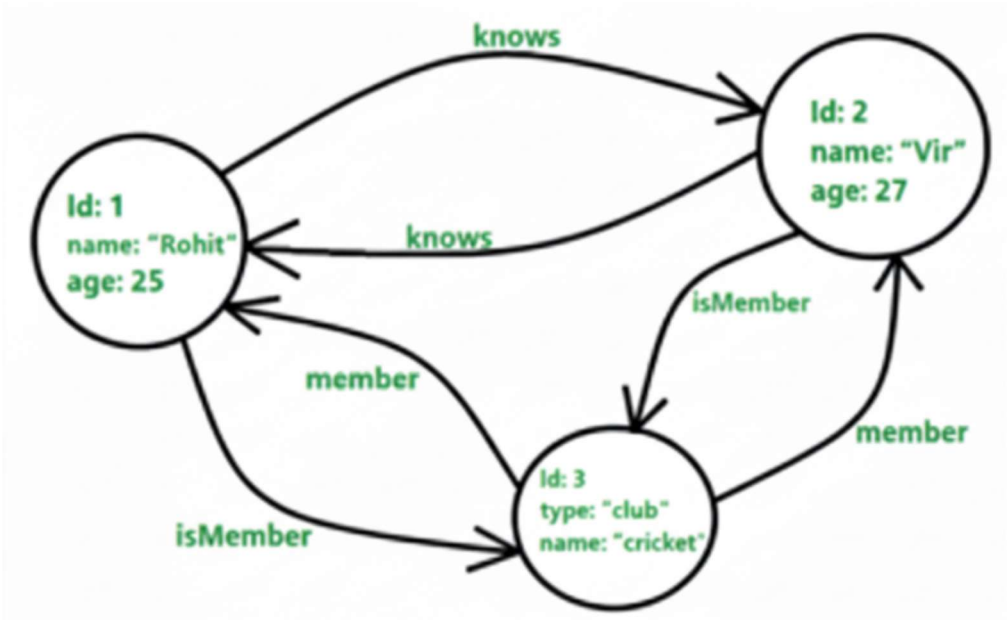

In a document store, data is stored as documents. Each document contains key–value pairs and can have a flexible structure, meaning new fields can be added anytime. For example, this document stores details of Rohan with fields like Name, Age, and Skills.

Example databases: MongoDB, CouchDB.

Graph Data-store:

Graph databases store data as nodes, edges, and properties to represent relationships. Nodes represent entities, edges represent relationships between nodes and properties store details about nodes or edges. This structure allows fast traversal and querying of complex relationships.

Example:



In this graph data store, nodes represent entities like people (e.g., Rohit), and edges represent relationships between them, such as “knows” or “isMemberOf.” Each node and edge can have properties defining their details, for example, Rohit has properties like name: “Rohit” and age: 25

Example databases: Example: Neo4j, OrientDB.

8. List and explain the core business drivers behind the NoSQL movement. #

Traditional relational databases (RDBMS) with a single CPU often cannot meet modern business needs due to challenges in Volume, Velocity, Variability, and Agility. NoSQL databases address these limitations.

1. Volume

- Organizations need to handle huge datasets by using groups of normal, low-cost computers instead of depending on a single faster CPU.
- The “power wall” problem (heat dissipation limits in CPUs) forced a shift from scaling up to scaling out via parallel processing.
- NoSQL supports horizontal scaling — splitting the work into smaller parts and processing them in parallel.

2. Velocity

- Applications like e-commerce and social media need real-time data inserts and queries.
- Traditional RDBMS can become bottlenecks due to heavy indexing or sudden traffic spikes.
- NoSQL supports high read/write throughput and handles unpredictable bursts efficiently.

3. Variability

- Modern data is diverse — structured, semi-structured, and unstructured — which does not fit well into rigid RDBMS schemas.
- In RDBMS, changing the structure (schema) is slow and may require downtime.
- NoSQL is schema-free, so it stores different types of data easily without stopping the system.

4. Agility

- Agility means the ability to adapt quickly to change — whether in data models, operational requirements, or application features.
- RDBMS often need extra programming layers to handle complex data, which slows down development.
- NoSQL’s flexible design makes it easy to update features, scale up, and adapt without big delays.

4. Mining Data Streams

9. Show how the Flajolet- Martin algorithm will estimate the number of distinct elements in a stream. PYQs:

i. Suppose the stream is S = {10, 12, 8, 15, 6, 9, 14, 7}.
Let hash functions h(x) = 5x + 11 mod 32 and treat the result as a 5-bit binary integer.

Given:

Stream $S = \{10,12,8,15,6,9,14,7\}$, hash $h(x) = 5x + 11 mod 32$.

Step 1: Apply the hash function to each value, then convert each hashed result to a 5-bit binary and count the trailing(ending) zeros.

Element (x)	h(x)=5x+11 mod 32)	5-bit binary	trailing zeros
10	29	11101	0
12	7	00111	0
8	19	10011	0
15	22	10110	1
6	9	01001	0
9	24	11000	3
14	17	10001	0
7	14	01110	1

Step 2: Identify the maximum number of trailing zeros

The maximum trailing-zeros observed $R_{\max} = 3$ (from hashed value 24).

Step 3: Apply the Flajolet-Martin estimation formula:

The estimate for the number of distinct elements is $2^{Rmax} = 2^3 = 8$.

Result: FM estimates **8 distinct elements** in the stream (which matches the actual distinct count 8).

ii. For the stream of integers: 9, 8, 7, 6, 5, 4, 3, 2
Use the hash function, h(x) = (2x + 1) mod 32 and treat the result as a 5-bit binary integer.

Given:

Stream $S = \{9,8,7,6,5,4,3,2\}$, hash $h(x) = (2x + 1) mod 32$

Step 1: Apply hash, convert to 5-bit binary, count trailing (ending) zeros

Element (x)	h(x) = (2x+1) mod 32	5-bit binary	Trailing zeros
9	19	10011	0
8	17	10001	0
7	15	01111	0
6	13	01101	0
5	11	01011	0
4	9	01001	0
3	7	00111	0
2	5	00101	0

Step 2: Identify the maximum number of trailing zeros

The maximum trailing-zeros observed $R_{\max} = 0$(all hashes end in 1 so zero trailing zeros).

Step 3: Apply the Flajolet–Martin estimation formula

The estimate for the number of distinct elements is $2^{Rmax} = 2^0 = 1$.

Result: FM estimates **1 distinct element** in the stream.

Other problems:

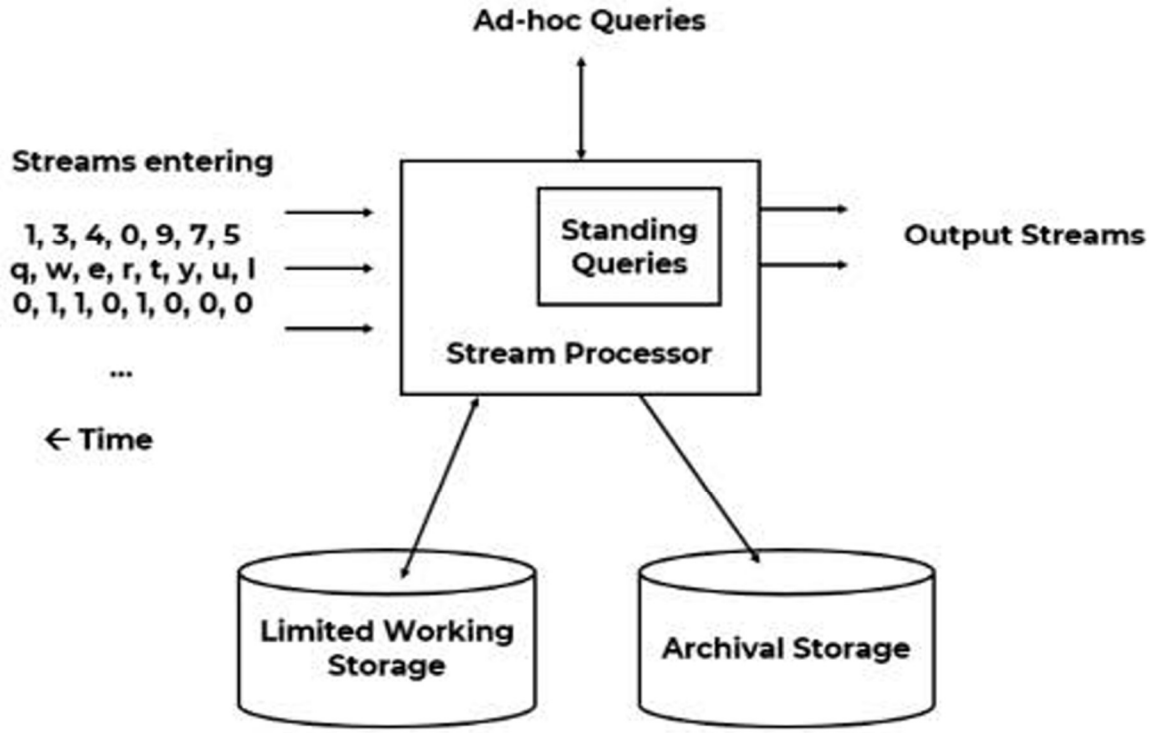
iii. Suppose the stream is S = {4, 2, 5 ,9, 1, 6, 3, 7}.
Let hash functions h(x) = 3x + 7 mod 32 for some a and b, treat result as a 5-bit binary integer. Show how the Flajolet- Martin algorithm will estimate the number of distinct elements in this stream.

iv. Suppose the stream is S = {2, 1, 6, 1, 5, 9, 2, 3, 5}.
Let hash functions h(x) = ax + b mod 16 for some a and b, treat result as a 4-bit binary integer. Show how the Flajolet- Martin algorithm will estimate the number of distinct elements, h(x) = 4x + 1 mod 16.

v. Suppose the stream is 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1.
Let h(x) = 6x + 1 mod 5. Show how the Flajolet-Martin algorithm will estimate the number of distinct elements in this stream.

10. Explain the architecture of the data-stream management system with a neat diagram.

A Data Stream Management System (DSMS) is a system designed to process continuous, high-speed data streams in real-time. Unlike traditional DBMS, which stores data first and then queries it, DSMS processes data instantly as it flows through the system, providing immediate results.



Key Components:

1. **Streams Entering**
   - Continuous flows of data, such as numbers, sensor readings, logs, or user clicks.
   - Examples:
     - Numerical: 1, 3, 4, 0, 9, ...
     - Textual: q, w, e, r, ...
   - These streams arrive continuously over time.
2. **Stream Processor**
   - Core of the DSMS, continuously processes incoming data streams.
   - Contains Standing Queries: long-running queries that produce results as new data arrives.
   - Example queries:
     - Average temperature in last 10 minutes
     - Number of transactions per second
3. **Output Streams**
   - After processing, results are generated as continuous output streams.
   - These results can update dashboards, generate alerts, or be used for further processing.

4. **Ad-hoc Queries**
   - Users can send on-demand queries to extract insights from live streams or historical data.
   - Example: “Show me total sales in the last 5 minutes.”
5. **Limited Working Storage**
   - DSMS cannot store infinite data because streams are unbounded.
   - It uses a sliding window to keep a recent subset of data (e.g., last 1 minute). This ensures efficient processing and memory usage.
6. **Archival Storage**
   - Older stream data that is not needed for immediate queries is moved to archival storage.
   - Useful for long-term analytics, audits, or machine learning training later.

11. List down all six constraints that must be satisfied for representing a stream by buckets using DGIM algorithm with examples. #

1. The right end of every bucket must always correspond to a 1
Example: If a bucket covers bits from timestamp 3 to 5, then timestamp 5 (the right end) must be a 1.

2. Every 1 in the stream must belong to exactly one bucket
Example: If the stream is 1 0 1 1 0 1, each 1 at positions 1, 3, 4, and 6 will be part of some bucket.

3. No bit position can belong to more than one bucket
Example: If the 1 at position 4 is already part of a size-2 bucket, it cannot appear again in any other bucket.

4. There can be at most two buckets of the same size
Example: If three buckets of size 1 exist, the two oldest are merged into a single bucket of size 2.

5. All bucket sizes must be powers of 2 (1, 2, 4, 8, ...)
Example: Buckets can represent 1, 2, or 4 ones — not 3 or 5 ones.

6. Bucket sizes must not decrease as we move left (back in time)
Example: If the newest bucket is of size 2, older buckets can only be of size 2, 4, 8, 16, ... as you move left.

12. Explain DGIM algorithm for counting ones in a stream with example.

The DGIM (Datar–Gionis–Indyk–Motwani) algorithm is used to approximate the number of 1's in the last k bits of a binary data stream using limited memory. It is especially useful when the stream is continuous and too large to store completely.

Algorithm Steps:

1. **When a new bit arrives:**
   - If it is 0 → ignore it.
   - If it is 1 → create a new bucket of size 1 with the current timestamp.
2. **Merging:**
   - If more than two buckets of the same size exist then merge the two oldest into one bucket of double size.
3. **Deleting old buckets:**
   - Buckets older than the window size k are deleted.
4. **Counting 1's:**
   - Add the sizes of all buckets completely inside the window.
   - Add half of the last (oldest) bucket since it may only be partially inside the window.

Example:

Given Stream (left → right):
1 0 0 1 0 1

Goal: To estimate the number of 1's in the last 6 bits.

Step 1: Process Stream (Right → Left)

Bit	Action	Buckets
1	Create size-1 bucket	[1]
0	Ignore	[1]
1	New size-1 bucket [1, 1]; two size-1 buckets present, which is allowed.	[1, 1]
0	Ignore	[1, 1]
0	Ignore	[1, 1]
1	New size-1 bucket [1, 1, 1]; three size-1 buckets found, so merge the two oldest size-1 buckets to form one size-2 bucket.	[1, 2]

Step 2: Final Buckets
Final buckets after processing = [1, 2]

Step 3: Estimate Number of 1's

Use DGIM rule:
Estimate = (sum of sizes of all buckets except the oldest) + (½ × oldest bucket)

$$\text{Estimate} = 1 + (2/2) = 1 + 1 = 2$$

Result

DGIM Estimate = 2

Actual Count = 3

The estimate (2) is an approximation and less than the true count (3); this is expected because DGIM trades exactness for very low memory usage.

13. Explain the concept of bloom filter with an example. #

A Bloom Filter is a probabilistic data structure used to check whether an element is in a set. It is fast and space-efficient, but may return false positives — it can say an element is present even when it is not.

Example:

Let bit array size be m = 6, and two hash functions:

$h_1(x) = x \bmod 6$

$h_2(x) = (2x + 1) \bmod 6$

Insert elements: 5 and 10

Element	**$h_1(x) = x \bmod 6$**	**$h_2(x) = (2x + 1) \bmod 6$**	**Bits Set (positions)**
5	5	$(2\times5 + 1) \bmod 6 = 11 \bmod 6 = 5$	5
10	$10 \bmod 6 = 4$	$(2\times10 + 1) \bmod 6 = 21 \bmod 6 = 3$	4, 3

Bit array after insertion [Bits 0-5]:

[0, 0, 0, 1, 1, 1]

Membership Test for Element 4:

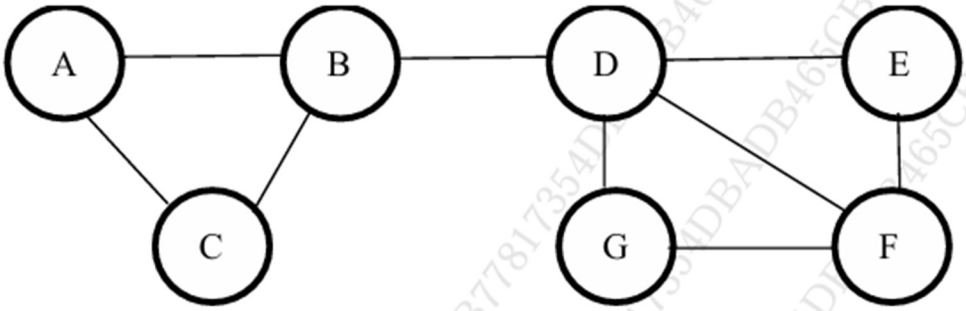
$h_1(4) = 4 \bmod 6 = 4$
$h_2(4) = (2\times4 + 1) \bmod 6 = 9 \bmod 6 = 3$

Positions 3 and 4 are both 1, so the Bloom filter would indicate that 4 is probably present.

However, since 4 was not actually inserted, this shows a false positive — the Bloom filter wrongly says 4 is present because its hash positions overlap with other elements.

5. Real-time Big Data Models

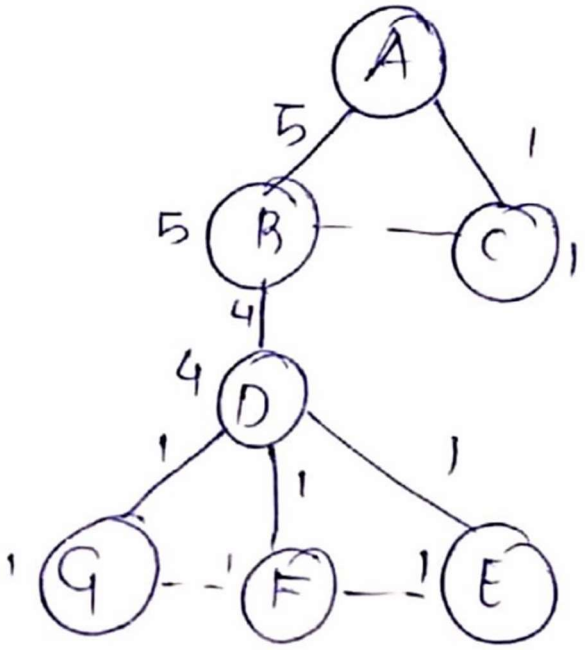
14. Determine communities for the given social network graph using Girvan-Newman algorithm.



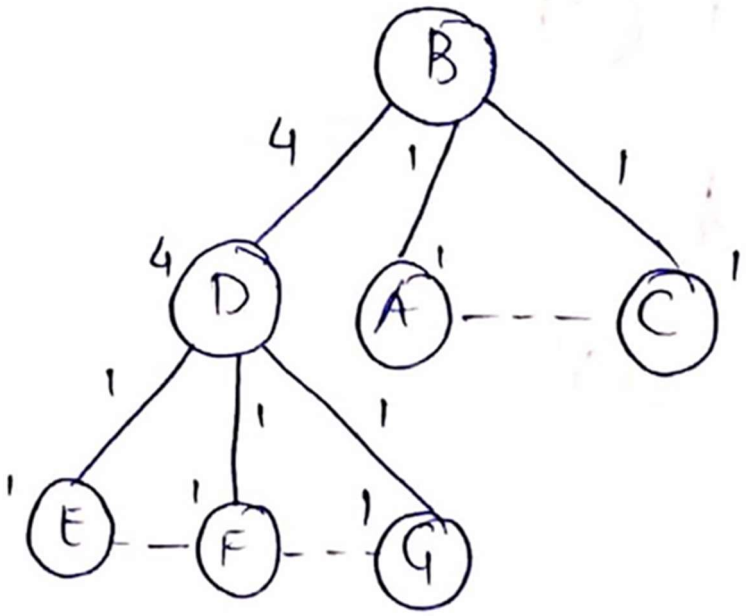
(2 times)

https://www.youtube.com/watch?v=dmMKJ1YUl-M

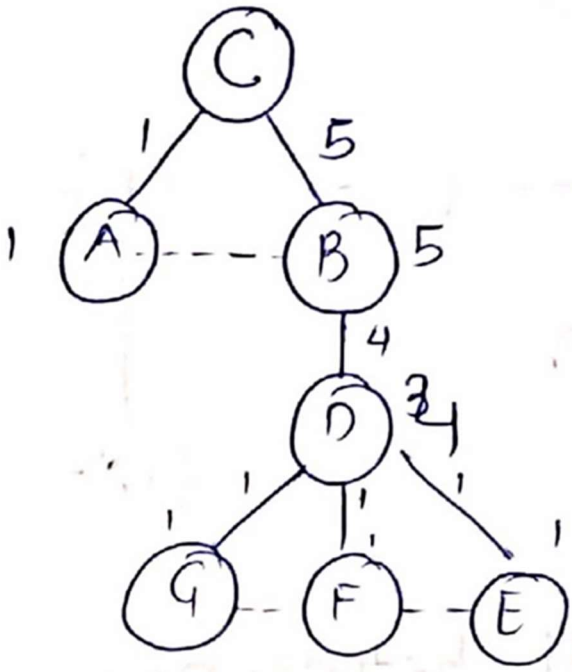
For A:



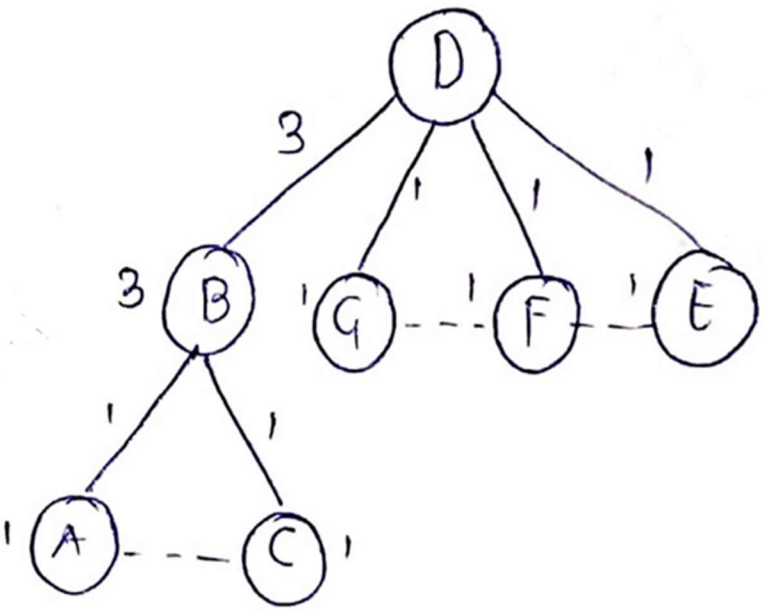
For B:



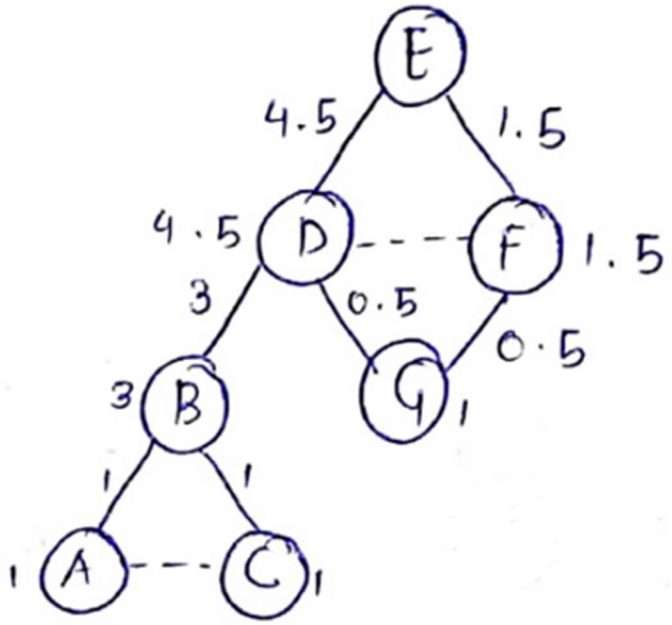
For C:



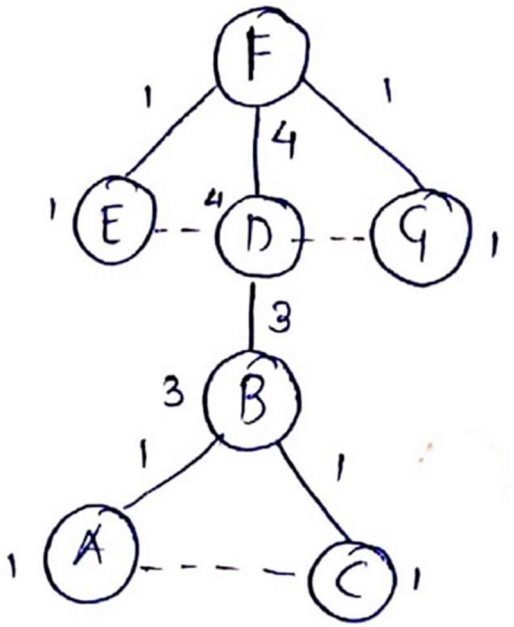
For D:



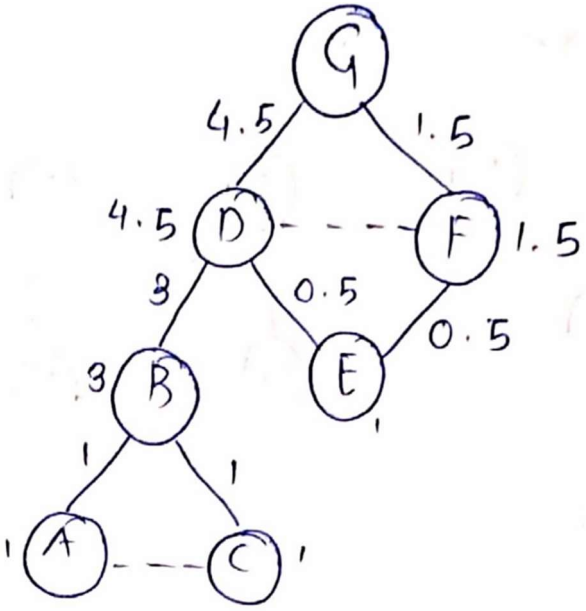
For E:



For F:

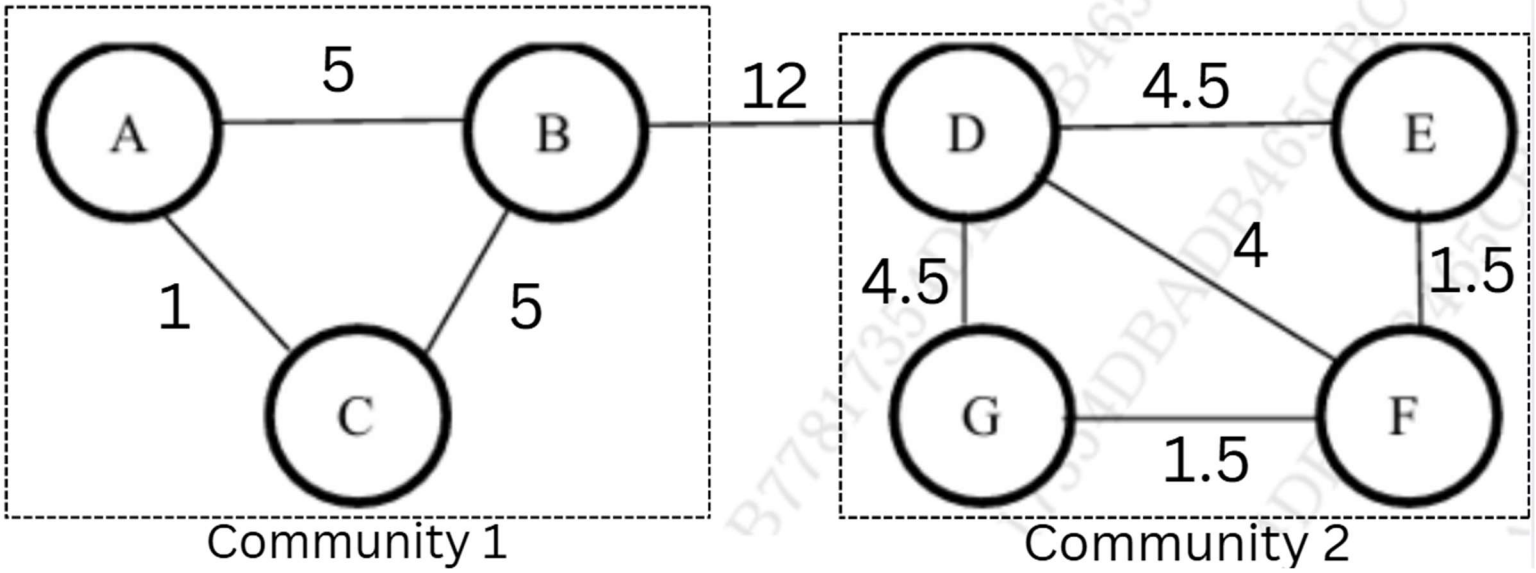


For G:



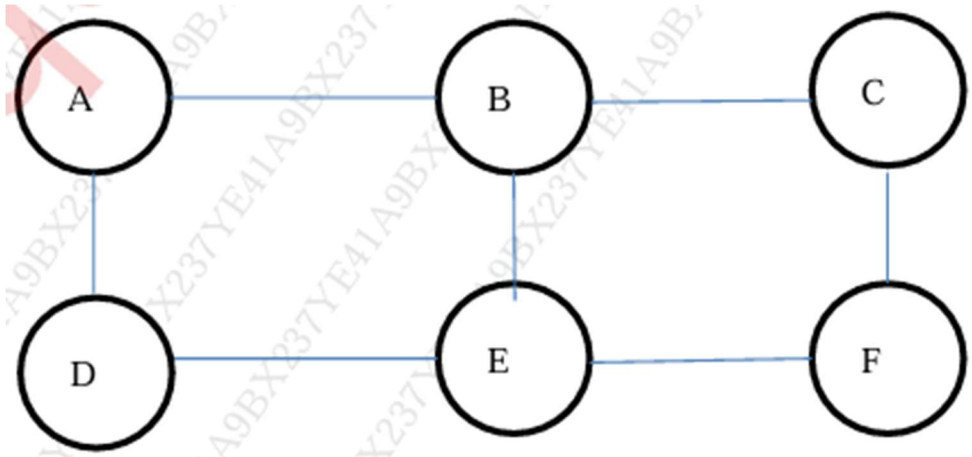
Edge	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Total**	**Dividing centrality by 2 as edges are bidirectional.**
AB	5	1	0	1	1	1	1	10	10/2 = 5
AC	1	0	1	0	0	0	0	2	2/2 = 1
BC	0	1	5	1	1	1	1	10	10/2 = 5
BD	4	4	4	3	3	3	3	24	24/2 = 12
DG	1	1	1	1	0.5	0	4.5	9	9/2 = 4.5
DF	1	1	1	1	0	4	0	8	8/2 = 4
DE	1	1	1	1	4.5	0	0.5	9	9/2 = 4.5
EF	0	0	0	0	1.5	1	0.5	3	3/2 = 1.5
FG	0	0	0	0	0.5	1	1.5	3	3/2 = 1.5

Divide the network into communities by removing the edge with the highest betweenness centrality.

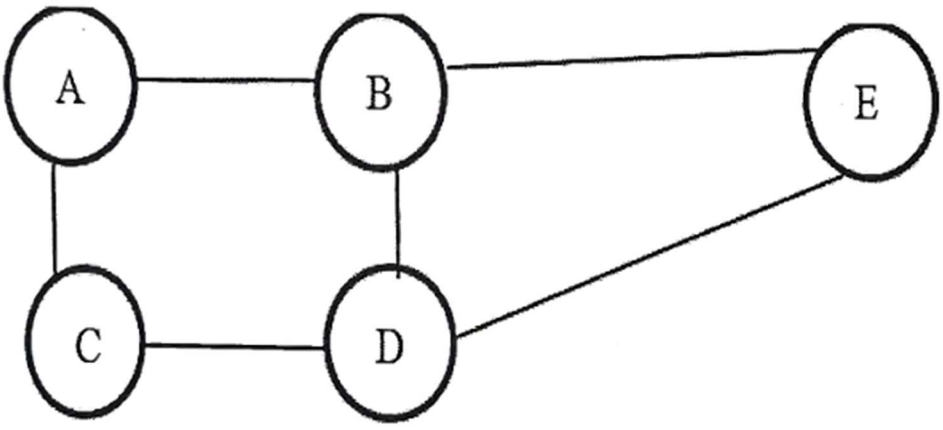


Other graphs asked:

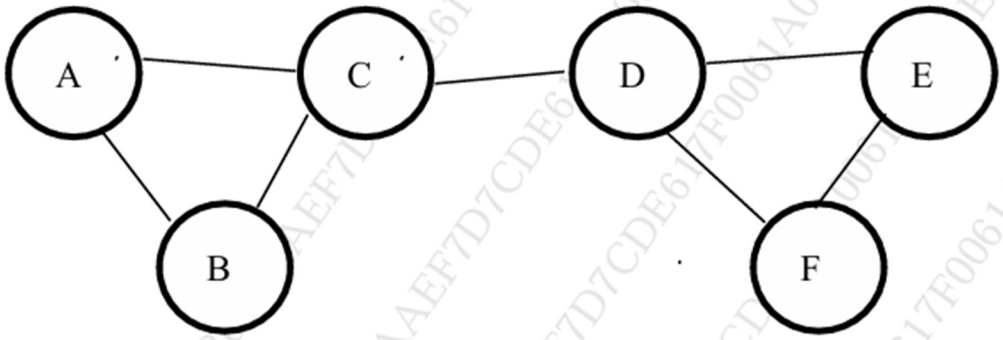
(i)



(ii)



(iii)



15. How is recommendation done based on properties of product (Content-based recommendation). Elaborate with a suitable example.

A content-based recommendation system suggests items to a user by analyzing the properties or features of the items and matching them with the user's preferences. It focuses on the content (attributes or characteristics) of items rather than other users' behavior.

Working Principle

Each item (like a movie or product) is represented by its features such as category, brand, or keywords. The system builds a user profile from items the user has previously liked and compares it with other items using similarity measures (e.g., cosine similarity). Items most similar to the user's preferences are then recommended.

User Profile:
Represents the interests, preferences, and behavior of a user, created based on features of items the user has interacted with or rated positively.

Item Profile:
Describes an item using its key attributes or features that help in comparing it with user preferences during recommendation.

Steps Involved

1. **Feature Extraction:** Identify important features from each product (e.g., category, price range, specifications).
2. **User Profile Creation:** Build a preference profile using features from products the user has liked.
3. **Similarity Calculation:** Compute similarity between user profile and other products.
4. **Recommendation Generation:** Suggest top products most similar to the user's preferences.

Example (E-commerce Platform like Amazon/Flipkart)

Suppose a user views and purchases a Samsung Galaxy S24 with the following features:

- Category: Electronics
- Brand: Samsung
- Features: 5G, 256GB storage, AMOLED display

The system analyzes these preferences and identifies other smartphones with similar specifications such as the OnePlus 12 (5G, 256GB, AMOLED) or Google Pixel 8 (5G, 256GB, OLED) and recommends them to the user.

16. Explain Collaborative filtering in recommendation systems with an example.

Collaborative filtering is a widely-used technique in recommendation systems that makes predictions for a user by considering the preferences and behaviours of other users with similar tastes. Unlike content-based filtering, which relies on product attributes, collaborative filtering uses user-item interactions such as ratings, clicks, and purchase history to generate recommendations.

Types of Collaborative Filtering:

1. **User-Based Collaborative Filtering**
   - Finds users who have similar taste and recommends items liked by those users.
   - Example:
     If User A and User B both liked similar movies before, and User A liked Inception, then Inception is recommended to User B.
2. **Item-Based Collaborative Filtering**
   - Finds similar items based on user ratings.
   - Recommends items that are similar to those the user has already liked.
   - Example:
     If a user liked Interstellar, and Inception is often liked by users who liked Interstellar, then recommend Inception.

Collaborative Filtering in E-Commerce (Example: Amazon/Flipkart)

Collaborative filtering recommends products based on similarities between users or items.
On Amazon or Flipkart, it analyzes purchase history, ratings, or browsing behavior of many users.

Example:
If many users who bought a Smartphone also bought a Phone Case, then when a new user buys a Smartphone, the system recommends a Phone Case.
Similarly, if two users have bought similar electronics before, and one buys a Smartwatch, the system suggests that Smartwatch to the other.

Advantages

- Works well without needing detailed content about items.
- Learns from real user behavior.
- Can capture complex, non-obvious patterns of taste similarity.

Limitations

- **Cold Start Problem:** Cannot recommend for new users/items with no history.
- **Data Sparsity:** Many users rate only a few items, reducing accuracy.

6. Data Analytics with R

17. Create data frame in R and perform operations.

I] The project manager at XYZ Ltd., Ms. Meera, is responsible for main details of all active projects. She has organized the project information in the following table:

Project Id	Project Name	Budget	Status
1	CRM Implementation	120000	In Progress
2	Cloud Infrastructure	180000	Completed
3	Network Upgrade	60000	Not Started
4	E-Commerce Platform	220000	Completed
5	Data Analytics	90000	In Progress

i) Create a Data frame in R for the above project data and display the output.

ii) Ms. Meera has recently approved 2 new projects and wants to find their information. The new projects are as follows:

Project Id	Project Name	Budget	Status
6	UX Research	160000	Not Started
7	Cloud Integration	190000	Not Started

iii) Update the Data frame to include the new projects and demonstrate the final output.

1. Create Data Frame in R with Given Projects

```
project_id <- c(1, 2, 3, 4, 5)
project_name <- c("CRM Implementation", "Cloud Infrastructure", "Network Upgrade",
        "E-Commerce Platform", "Data Analytics")
budget <- c(120000, 180000, 60000, 220000, 90000)
status <- c("In Progress", "Completed", "Not Started", "Completed", "In Progress")

df <- data.frame(Project_ID = project_id, Project_Name = project_name,
        Budget = budget, Status = status)
# Display output
print(df)
```

2. Add New Projects

```
new_project_id <- c(6, 7)

new_project_name <- c("UX Research", "Cloud Integration")

new_budget <- c(160000, 190000)

new_status <- c("Not Started", "Not Started")


new_projects <- data.frame(Project_ID = new_project_id, Project_Name = new_project_name,
                           Budget = new_budget, Status = new_status)
```

3. Update Data Frame with New Projects

```
df_updated <- rbind(df, new_projects)


# Display final data frame

print(df_updated)
```

Output:

Project_ID	Project_Name	Budget	Status
1	CRM Implementation	120000	In Progress
2	Cloud Infrastructure	180000	Completed
3	Network Upgrade	60000	Not Started
4	E-Commerce Platform	220000	Completed
5	Data Analytics	90000	In Progress
6	UX Research	160000	Not Started
7	Cloud Integration	190000	Not Started

II] 1. Create a data frame from the following 4 vectors and demonstrate the output:

emp_id = c(1:5)

emp_name = c("Rick", "Dan", "Michelle", "Ryan", "Gary")

start_date = c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")

salary = c(60000, 45000, 75000, 84000, 20000)

2. Display structure and summary of the above data frame.

3. Extract the emp_name and salary columns from the above data frame.

4. Extract the employee details whose salary is less than or equal to 60000.

1. Create a Data Frame

```
emp_id <- c(1:5)

emp_name <- c("Rick", "Dan", "Michelle", "Ryan", "Gary")

start_date <- c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")

salary <- c(60000, 45000, 75000, 84000, 20000)


employee <- data.frame(emp_id, emp_name, start_date, salary)

print(employee)
```

Output:

emp_id	emp_name	start_date	salary
1	Rick	2012-01-01	60000
2	Dan	2013-09-23	45000
3	Michelle	2014-11-15	75000
4	Ryan	2014-05-11	84000
5	Gary	2015-03-27	20000

2. Display Structure and Summary

```
str(employee)

summary(employee)
```

Structure Output:

```
'data.frame': 5 obs. of  4 variables:

 $ emp_id   : int  1 2 3 4 5
```

$ emp_name : chr "Rick" "Dan" "Michelle" "Ryan" ...

$ start_date: chr "2012-01-01" "2013-09-23" "2014-11-15" "2014-05-11" ...

$ salary   : num 60000 45000 75000 84000 20000

Summary Output:

emp_id	emp_name	start_date	salary
Min. :1	Length: 5	Length: 5	Min. :20000
1st Qu.:2	Class :character	Class :character	1st Qu.:45000
Median :3	Mode :character	Mode :character	Median :60000
Mean :3		Mean :56800	
3rd Qu.:4		3rd Qu.:75000	
Max. :5		Max. :84000	

3. Extract emp_name and salary columns

```
employee_subset <- employee[, c("emp_name", "salary")]
print(employee_subset)
```

Output:

emp_name	salary
Rick	60000
Dan	45000
Michelle	75000
Ryan	84000
Gary	20000

4. Extract employee details with salary ≤ 60000

```
employee _salary <- subset(employee, salary <= 60000)
print(employee _salary)
```

Output:

emp_id	emp_name	start_date	salary
1	Rick	2012-01-01	60000
2	Dan	2013-09-23	45000
5	Gary	2015-03-27	20000

18. Write script in R for creating subsets:

Consider the following data frame:

course	id	class	marks
1	11	1	56
2	12	2	75
3	13	1	48
4	14	2	69
5	15	1	84
6	16	2	53

i) Create a subset of course less than 5 by using [ ] brackets and demonstrate the output.

ii) Create a subset where the course column is less than 4 or the class equals to 1 by using subset() function and demonstrate the output.

(i) Subset of course less than 5 using [ ] brackets

```
subset1 <- df[df$course < 5, ]

print(subset1)
```

Output:

course	id	class	marks
1	11	1	56
2	12	2	75
3	13	1	48
4	14	2	69

(ii) Subset where course < 4 or class == 1 using subset() function

```
subset2 <- subset(df, course < 4 | class == 1)

print(subset2)
```

Output:

course	id	class	marks
1	11	1	56
2	12	2	75
3	13	1	48
5	15	1	84

19. Write the script to sort the values contained in the following vector in ascending order and descending order: (46, 23, 15, 38, 98, 56, 28, 78). Demonstrate the output. #

We are given the vector:

vec <- c(46, 23, 15, 38, 98, 56, 28, 78)

1. Sorting in Ascending Order

Use the sort() function:

sort(vec)

Output: 15 23 28 38 46 56 78 98

sort() by default arranges elements in ascending order.

2. Sorting in Descending Order

Use sort() with the decreasing=TRUE argument:

sort(vec, decreasing=TRUE)

Output: 98 78 56 46 38 28 23 15

decreasing=TRUE tells R to sort in descending order.

20. List and explain various types of data structures in R.

R provides various data structures to efficiently store, organize, and manipulate data. These include basic and advanced structures depending on data type and dimensionality.

1. Vector

A vector is the simplest and most common data structure in R. It is a sequence of elements that are all of the same data type — numeric, character, or logical.

Example:

```
v <- c(10, 20, 30, 40)
```

2. List

A list is a collection of elements that can be of different data types, such as numbers, strings, vectors, matrices, or even other lists. It is often used to store complex or hierarchical data.

Example:

```
lst <- list(name="John", age=25, scores=c(80, 85, 90))
```

3. Matrix

A matrix is a two-dimensional data structure containing elements of the same type, arranged in rows and columns. It is mainly used for mathematical and statistical operations.

Example:

```
m <- matrix(1:9, nrow=3, ncol=3)
```

4. Array

An array is a multi-dimensional extension of a matrix, capable of storing data in more than two dimensions. All elements must be of the same data type.

Example:

```
arr <- array(1:12, dim=c(3,2,2))
```

5. Data Frame

A data frame is a tabular data structure similar to a spreadsheet or database table. Each column can hold different data types, such as numeric, character, or logical values.

Example:

```
df <- data.frame(ID = c(1, 2, 3), Name = c("A", "B", "C"),  Marks = c(90, 85, 88))
```

6. Factor

A factor is used to represent categorical data, such as gender or grade levels. It stores unique category labels as levels and is useful for statistical modelling.

Example:

```
gender <- factor(c("Male", "Female", "Male"))
```

21. Name and explain the operators used to form data subsets in R. #

In R, subsetting is the process of extracting parts of a data frame. R provides several operators and functions for this purpose.

Using following data frame for all examples:

```
df <- data.frame(name=c("Mark", "Fred", "Cyrus"),
                 age=c(20,21,22),
                 marks=c(85,90,88))
```

1. Square Brackets [ ]

Extract elements by position or condition.

Examples:

```
df[1, ]      # Output: Row 1 → name="Mark", age=20, marks=85
df[, 2]      # Output: Column 2 → 20 21 22
```

2. Double Square Brackets [[ ]]

Extract a single column as a vector.

Examples:

```
df[[1]]          # Output: "Mark" "Fred" "Cyrus" (first column)
df[["marks"]]    # Output: 85 90 88 (marks column)
```

3. Dollar Sign $

Access named columns directly.

Examples:

```
df$name      # Output: "Mark" "Fred" "Cyrus"
df$marks     # Output: 85 90 88
```

4. subset() Function

Extract rows or columns based on a condition.

Examples:

```
subset(df, age > 20)                  # Output: name="Fred", "Cyrus"; age=21,22; marks=90,88
subset(df, age > 20, select=name)  # Output: (only 'name' column) "Fred" "Cyrus"
```

22. Explain the various functions provided by R to combine different sets of data. #

1. **rbind() – Row Binding**
   - Combines two or more data frames or vectors by adding rows.
   - Example: rbind(df1, df2) joins df2 below df1 (same column names required).
2. **cbind() – Column Binding**
   - Combines data frames, matrices, or vectors by adding columns.
   - Example: cbind(df1, df2) adds columns of df2 beside df1.
3. **merge() – Database-style Join**
   - Merges two data frames based on common columns or row names (like SQL join).
   - Example: merge(df1, df2, by = "id").
4. **append() – Add Elements to a Vector**
   - Adds new elements to an existing vector.
   - Example: append(x, y) adds vector y to x.
5. **bind_rows() and bind_cols()**
   - Enhanced versions of rbind() and cbind() that handle mismatched column names easily.
   - Example: bind_rows(df1, df2).

Asked once:

# indicates 5-mark question

1. Introduction to Big Data and Hadoop

1. **Secondary Name is a backup of Name node. Is this statement True or False? Justify your answer. #**

Answer: False

Justification:
The Secondary NameNode is not a backup of the NameNode. It only helps by periodically merging the NameNode's edit logs and FsImage to create a new checkpoint. This process prevents the logs from growing too large and improves system performance.
If the NameNode fails, the Secondary NameNode cannot take its place since it doesn't store the most recent metadata. It only maintains a checkpoint copy, which can be used later to recover the system manually.

2. **Explain how big data problems are handled by Hadoop system. #**

Hadoop is specifically designed to solve big data challenges by providing:

Distributed Storage:
Hadoop uses HDFS (Hadoop Distributed File System) to split large datasets into smaller blocks and store them across multiple nodes, allowing scalable and efficient storage.

Parallel Processing:
Hadoop's MapReduce framework processes data in parallel across cluster nodes, significantly speeding up computation.

Fault Tolerance:
Hadoop automatically replicates data across nodes. If one node fails, another copy is used, preventing data loss and ensuring continuous processing.

Scalability:
Hadoop can easily scale from a few machines to thousands; adding more nodes increases its capacity to handle more data.

Cost-Effectiveness:
Hadoop runs on low-cost commodity hardware, making it an affordable solution for storing and processing massive datasets.

3. What is the difference between traditional RDBMS and Hadoop?

Parameter	Traditional RDBMS	Hadoop
Data Type	Handles structured data only.	Handles structured, semi-structured, and unstructured data.
Storage	Stores data on a single centralized server.	Stores data in a distributed manner using HDFS across multiple nodes.
Processing	Uses sequential processing (one task at a time).	Uses parallel processing.
Scalability	Vertically scalable – limited by server capacity.	Horizontally scalable – add more nodes to increase capacity.
Cost	Requires expensive high-end hardware.	Runs on low-cost commodity hardware, making it economical.
Fault Tolerance	No built-in fault tolerance – data loss if server fails.	Highly fault-tolerant – data replicated across nodes.
Data Volume Handling	Best for small to medium datasets.	Designed to handle massive datasets (Big Data) efficiently.

4. Distinguish between Name node and Data node.

Aspect	NameNode	DataNode
Role	Acts as the master in HDFS architecture.	Acts as the slave that stores actual data.
Function	Manages metadata — file names, block locations, and directory structure.	Stores and retrieves data blocks when requested by clients or NameNode.
Storage	Does not store actual data, only metadata.	Stores the actual file data in the form of blocks.
Failure Impact	Failure may cause the whole HDFS to stop working.	Failure affects only the data stored on that specific node.
Communication	Coordinates with clients and DataNodes.	Sends heartbeats and block reports to NameNode.
Data Handling	Does not perform any data processing.	Performs read/write operations on data blocks.
Quantity in Cluster	Usually one active NameNode.	Multiple DataNodes across the cluster.

5. What are the Core Hadoop components? Explain in detail.

Hadoop mainly consists of three core components: HDFS, YARN, and MapReduce which work together to store and process large-scale data efficiently.

1. Hadoop Distributed File System (HDFS)

HDFS is the storage layer of Hadoop that stores huge data files across multiple machines in a distributed manner. It provides fault tolerance and high throughput access to data.
It consists of two main components:

- **NameNode:** The master node that stores metadata such as file names, locations, and block information.
- **DataNode:** The worker nodes that actually store the data blocks and respond to read/write requests from clients.

2. YARN (Yet Another Resource Negotiator)

YARN acts as the resource management layer of Hadoop. It manages and allocates system resources among different applications and schedules their execution.
It consists of:

- **ResourceManager:** Manages global resource allocation.
- **NodeManager:** Runs on each node and reports resource usage to the ResourceManager.
- **ApplicationMaster:** Handles the execution and coordination of each application running on the cluster.

3. MapReduce

MapReduce is the data processing framework of Hadoop that allows parallel processing of large datasets across clusters.
It works in two phases:

- **Map Phase:** Divides the input into key-value pairs for processing.
- **Reduce Phase:** Aggregates the intermediate results to produce final output.

Example:
In a word count program, the Map phase emits (“word”, 1) pairs for each occurrence, and the Reduce phase sums these to get the total count for each word.

2. Hadoop HDFS and MapReduce

6. Explain selection and projection relational algebraic operation using MapReduce.

1. Selection Operation (σ)

- **Definition:**
  The selection operation retrieves rows from a relation that satisfy a given condition. It is represented as σ condition (R).
  Example: σ age > 25 (Employee) → selects records where age > 25.

Implementation using MapReduce:

- **Map Phase:** Each mapper checks if the record satisfies the condition (e.g., age > 25). If true, it emits the record.
- **Reduce Phase:** The reducer collects and outputs all filtered records.

Example Input:

(John, 24), (Ravi, 30), (Asha, 28)

Output (σ age > 25):

(Ravi, 30), (Asha, 28)

2. Projection Operation (π)

- **Definition:**
  The projection operation extracts specific columns (attributes) from a relation. It is represented as π column_list (R).
  Example: π name, age (Employee) → selects only 'name' and 'age' columns.

Implementation using MapReduce:

- **Map Phase:** Each mapper reads the record and outputs only the required columns.
- **Reduce Phase:** The reducer removes duplicate records (if any).

Example Input:

(ID, Name, Age, Salary)

(1, John, 24, 40000)

(2, Ravi, 30, 55000)

Output (π name, age):

(John, 24)

(Ravi, 30)

7. Explain how node failure is handled in Hadoop.

1. **Heartbeat Mechanism:**
   Each DataNode sends regular heartbeats to the NameNode to indicate it is active and functioning.
2. **Failure Detection:**
   If the NameNode does not receive a heartbeat from a DataNode within a specific time, it marks that node as dead or failed.
3. **Data Replication:**
   Since HDFS stores multiple replicas of each data block, the NameNode ensures data remains available even if one node fails.
4. **Re-replication:**
   The NameNode automatically creates new replicas of the lost data blocks on healthy DataNodes to maintain the replication factor.
5. **Automatic Recovery:**
   Tasks running on the failed node are reassigned to other active nodes, ensuring continuous processing and fault tolerance.

8. What is function of Map Tasks in the Map Reduce framework? Explain with the help of an example.

Function: The Map task processes large datasets in parallel by splitting them into smaller chunks. Each map function takes input as key–value pairs, performs filtering, sorting, or transformation, and outputs intermediate key–value pairs for the Reduce phase.

Example:
Suppose we want to count word occurrences in a text file:

- **Input:** "Hadoop is fast, Hadoop is reliable"
- **Map Output:**
  - (Hadoop, 1)
  - (is, 1)
  - (fast, 1)
  - (Hadoop, 1)
  - (is, 1)
  - (reliable, 1)
    Each word is treated as a key, and the value 1 represents one occurrence, which is later aggregated by the Reduce task.

9. Why is HDFS more suited for applications having large datasets and not when there are small files? Elaborate. #

1. **Designed for Large Files:** HDFS is optimized to store and process very large files (GBs/TBs) efficiently.
2. **NameNode Memory Limitation:** Each file and block is tracked by the NameNode, consuming memory. Too many small files can overload it.
3. **High Metadata Overhead:** Managing thousands of small files increases metadata overhead, slowing down the system.
4. **Poor Block Utilization:** Small files don't fully use HDFS block space, wasting storage capacity.
5. **Reduced Processing Efficiency:** MapReduce performs better with large, continuous data blocks, not scattered small files.

10. Name the three ways that resources can be shared between computer systems. Name the architecture used in big data solutions.

Three Ways Resources Can Be Shared Between Computer Systems

1. Time Sharing:

- Multiple users or programs use the same system resources (CPU, memory, I/O) one after another for short time slots.
- The switching happens so quickly that it feels like all are running at the same time.
- Example: Many users working on a mainframe through different terminals.

2. Space Sharing:

- The system divides physical resources (processors, memory, or storage) among users or tasks at the same time.
- Each task gets its own dedicated portion, allowing true parallel work.
- Example: Different nodes in a cluster working on separate parts of a job.

3. Resource Pooling / Data Sharing:

- Resources from multiple computers are shared over a network to perform data storage or computation.
- Common in distributed, grid, or cloud systems.
- Example: Hadoop HDFS, cloud platforms, or grid computing.

Architecture Used in Big Data Solutions

- Big Data systems typically use a Distributed Computing Architecture, following a Shared-Nothing Architecture model.
- In this design, each node in the cluster operates independently, with its own memory and disk, minimizing contention for shared resources.
- Examples: Hadoop, Spark.

3. NoSQL

11. Demonstrate how business problems have been successfully solved faster, cheaper and more effectively considering NoSQL Google's Big case study. Also illustrate the business drivers and the findings in it. #

Google developed Bigtable, a NoSQL column-family database, to manage and process massive amounts of web data efficiently across thousands of servers.

How it solved business problems:

- **Faster:** Provided quick access to structured and semi-structured data for services like Gmail, YouTube, and Google Maps.
- **Cheaper:** Used clusters of commodity hardware instead of expensive mainframes.
- **More Effective:** Scalable and fault-tolerant storage enabled handling of petabytes of data with low latency.

Business Drivers:

- Need for real-time access to web-scale data.
- Demand for scalable, distributed storage.

Findings:

- Bigtable became the foundation for many Google services.
- Inspired open-source systems like HBase and Cassandra.

12. Demonstrate how business problems have been successfully solved faster, cheaper and more effectively considering NoSQL Google's MapReduce case study. Also illustrate the business drivers and the findings in it. #

Google introduced MapReduce, a programming model for processing and generating large datasets in parallel using distributed clusters.

How it solved business problems:

- **Faster:** Enabled parallel data processing on thousands of machines.
- **Cheaper:** Reduced infrastructure cost by using commodity hardware.
- **More Effective:** Simplified large-scale computations like indexing web pages and data mining.

Business Drivers:

- Need to process huge web data efficiently.
- Desire to automate parallel programming and fault recovery.

Findings:

- MapReduce drastically reduced data processing time.
- Formed the core of Hadoop's architecture, influencing global Big Data processing frameworks.

13. Explain CAP. How is CAP different from ACID property in databases?

The CAP Theorem, proposed by Eric Brewer states that a distributed system can provide only two of the following three guarantees at the same time:

1. **Consistency (C):** Every node sees the same data at the same time.
2. **Availability (A):** Every request receives a response, even if some nodes fail.
3. **Partition Tolerance (P):** The system continues to function despite network failures or communication breaks.

Aspect	CAP Theorem	ACID Properties
Applies to	Distributed systems like NoSQL.	Traditional relational databases.
Goal	Balances consistency, availability, and partition tolerance.	Ensures reliable and error-free transactions.
Components	Consistency, Availability, Partition Tolerance.	Atomicity, Consistency, Isolation, Durability.
Property Handling	Cannot achieve all three properties at once.	Tries to achieve all four properties together.
Examples	NoSQL systems (e.g., Cassandra, MongoDB).	RDBMS systems (e.g., MySQL, Oracle).

14. Describe the four ways by which big data problems are handled by NoSQL.

NoSQL systems are built to efficiently handle massive, distributed, and fast-growing datasets. They achieve scalability, reliability, and high performance through the following four key techniques:

1. **Moving Queries to the Data (Not Data to Queries):**
   Instead of transferring large datasets over the network, NoSQL systems send queries directly to the data nodes where the data resides. This reduces network load, speeds up processing, and returns only the final results to the user.
2. **Even Data Distribution using Hash Rings:**
   NoSQL databases use hash rings to evenly distribute data across all nodes in the cluster. Each piece of data is assigned a node based on its hash key, ensuring balanced workload and seamless scalability when new nodes are added.
3. **Replication for Faster Reads and High Availability:**
   Data is replicated across multiple nodes so that reads can be served from any replica. This increases read performance and ensures data availability even if a node fails, maintaining fault tolerance and reliability.
4. **Distributing Queries Evenly Among Data Nodes:**
   Incoming queries are distributed evenly across all available data nodes, which process them in parallel and send results back to a central query analyzer. This parallelism improves query speed, scalability, and overall system throughput.

4. Mining Data Streams

15. Create a Bloom filter with the following parameters:

Size of the bit array m = 8

Hash functions:

$h_1(x) = x \bmod m$

$h_2(x) = (2x + 1) \bmod m$

$h_3(x) = (3x + 2) \bmod m$

(i) Insert the following elements into the Bloom filter: 12, 25, 30, 5

(ii) Check if the following elements are present in the Bloom filter: 6, 55

(iii) Discuss the results of your checks, identifying which elements are true positive and which is true negative.

Given:

- Size of bit array $m = 8$
- Hash functions:

$$h_1(x) = x \bmod 8$$
$$h_2(x) = (2x + 1) \bmod 8$$
$$h_3(x) = (3x + 2) \bmod 8$$

(i) Insert the elements: 12, 25, 30, 5

We start with an 8-bit array initialized to 0:

$$[0,0,0,0,0,0,0,0]$$

For 12:

- $h_1(12) = 12 \bmod 8 = 4$
- $h_2(12) = (2 \times 12 + 1) \bmod 8 = 25 \bmod 8 = 1$
- $h_3(12) = (3 \times 12 + 2) \bmod 8 = 38 \bmod 8 = 6$

→ Set bits at positions 1, 4, 6:

$$[0,1,0,0,1,0,1,0]$$

For 25:

- $h_1(25) = 25 \bmod 8 = 1$
- $h_2(25) = (2 \times 25 + 1) \bmod 8 = 51 \bmod 8 = 3$
- $h_3(25) = (3 \times 25 + 2) \bmod 8 = 77 \bmod 8 = 5$

→ Set bits at positions 1, 3, 5:

$$[0,1,0,1,1,1,1,0]$$

For 30:

- $h_1(30) = 30 \bmod 8 = 6$
- $h_2(30) = (2 \times 30 + 1) \bmod 8 = 61 \bmod 8 = 5$
- $h_3(30) = (3 \times 30 + 2) \bmod 8 = 92 \bmod 8 = 4$

→ Set bits at positions 4, 5, 6 (already 1s):

$$[0,1,0,1,1,1,1,0]$$

For 5:

- $h_1(5) = 5 \bmod 8 = 5$
- $h_2(5) = (2 \times 5 + 1) \bmod 8 = 11 \bmod 8 = 3$
- $h_3(5) = (3 \times 5 + 2) \bmod 8 = 17 \bmod 8 = 1$

→ Set bits at positions 1, 3, 5 (already 1s):

Final Bloom Filter after insertion:

$$\boxed{[0,1,0,1,1,1,1,0]}$$

(ii) Check elements: 6, 55

For 6:

- $h_1(6) = 6 \bmod 8 = 6$
- $h_2(6) = (2 \times 6 + 1) \bmod 8 = 13 \bmod 8 = 5$
- $h_3(6) = (3 \times 6 + 2) \bmod 8 = 20 \bmod 8 = 4$

→ Bits 4, 5, 6 = (1, 1, 1)

All bits = 1 → **Element 6 is possibly present**

For 55:

- $h_1(55) = 55 \bmod 8 = 7$
- $h_2(55) = (2 \times 55 + 1) \bmod 8 = 111 \bmod 8 = 7$
- $h_3(55) = (3 \times 55 + 2) \bmod 8 = 167 \bmod 8 = 7$

→ Only bit 7 checked (bit 7 = 0)

Bit 7 = 0 → **Element 55 is definitely not present**

(iii) Discussion of Results

Element 6 is a **false positive** since all its bits (4, 5, 6) were 1 due to previous insertions, though it wasn't inserted.
Element 55 is a **true negative** as one of its bits (7) was 0, correctly indicating it wasn't inserted; Bloom filters never give false negatives.

16. List and explain the different issues and challenges in data stream query processing. #

The main issues and challenges in data stream query processing are:

1. **Unbounded Data Streams:** Streams are continuous and infinite, making it impossible to store all data; systems must process data on-the-fly.
2. **High Data Arrival Rate:** Data arrives rapidly, requiring real-time or near-real-time processing with minimal delay.
3. **Memory Constraints:** Limited memory forces use of approximations, sampling, or summarization techniques.
4. **Out-of-Order and Late Data:** Data may arrive late or out of sequence, complicating accurate query results.
5. **Dynamic Query Requirements:** Continuous queries need to adapt to changing data patterns, requiring flexible and efficient query optimization.

5. Real-time Big Data Models

17. What is a recommendation system? How is classification algorithm used in a recommendation system?

A recommendation system is a tool that suggests relevant items (like movies, books, or products) to users by analyzing their preferences, behavior, and interaction history. It predicts what users might like and helps improve user experience and engagement.

Types of Recommendation Systems:

1. Content-Based Filtering: Recommends items similar to those the user has liked before.

2. Collaborative Filtering: Suggests items based on the preferences of users with similar tastes.

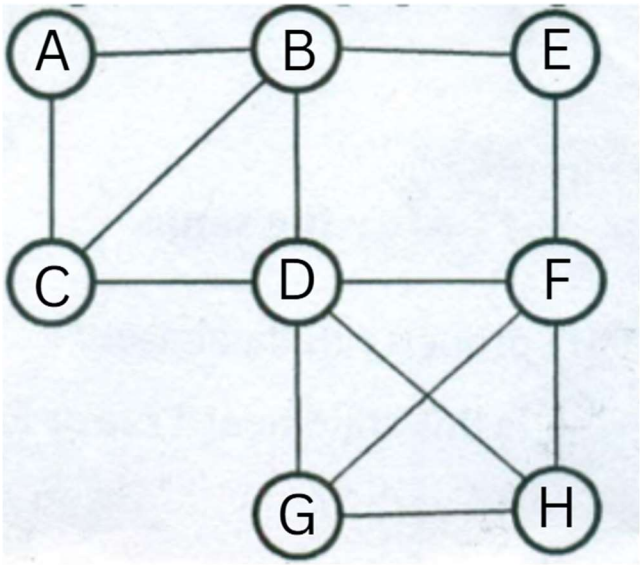
3. Hybrid Approach: Combines both content-based and collaborative methods.

Use of Classification algorithm in recommendation systems:

- A classification algorithm predicts whether a user will like or interact with an item, based on features of the user and the item.
- Recommendation systems use classifiers (such as decision trees, SVM, logistic regression) to assign items to categories like "Will Like" or "Will Not Like" for individual users.
- The system is trained on historical interaction data (clicks, ratings, purchases) to learn patterns.
- For new items or users, the classifier infers preferences and recommends items predicted as "liked" or relevant.

Example: In an online store, the classifier studies what the user has viewed or bought before and recommends products the user is likely to like.

18. Write an algorithm for the Clique Percolation Method and discover the communities in the given below graph using Clique Percolation Method with clique k=3.



Clique Percolation Method Algorithm (k=3)

1. **Find all 3-cliques (triangles) in the graph:** A 3-clique is a set of 3 nodes where every node is connected to each other.
2. **Build a clique adjacency graph:** Two 3-cliques are adjacent if they share 2 nodes.
3. **Identify communities:** A community is a maximal set of adjacent k-cliques (connected components in the clique graph).

Step 1: Find all 3-cliques:

- Clique 1: A, B, C
- Clique 2: B, C, D
- Clique 3: D, G, H
- Clique 4: F, G, H
- Clique 5: D, F, H
- Clique 6: D, F, G

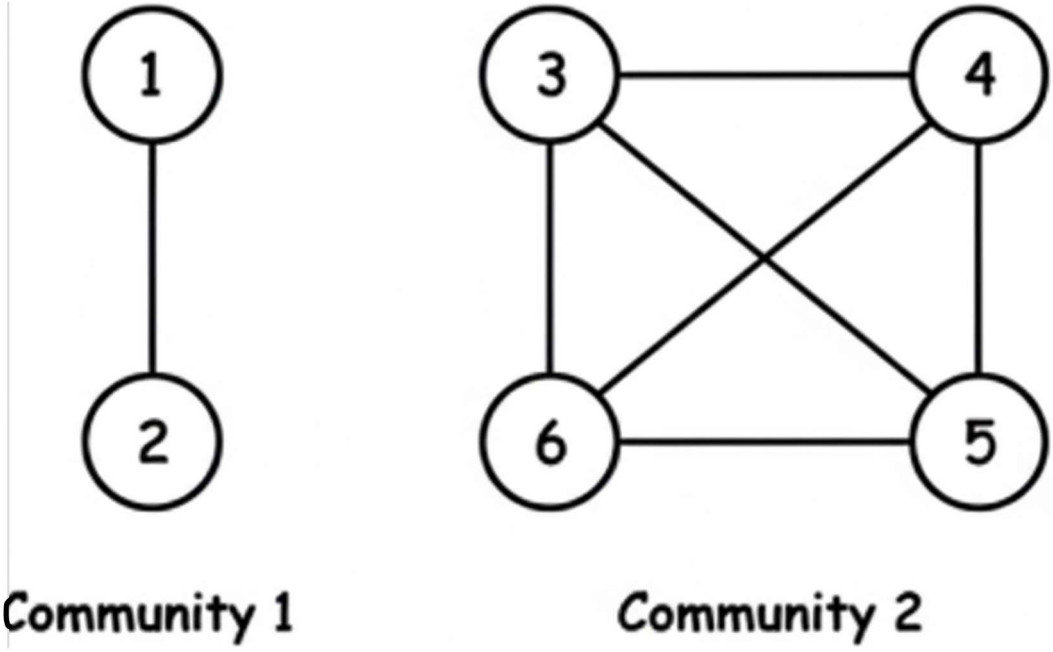
Step 2: Find Clique Adjacencies

Cliques that share 2 nodes (overlap):

- (A, B, C) <–> (B, C, D) (share B, C)
- (D, G, H) <–> (D, F, H) (share D, H)
- (D, G, H) <–> (F, G, H) (share G, H)
- (D, F, H) <–> (D, F, G) (share D, F)
- (D, F, H) <–> (F, G, H) (share F, H)
- (D, F, G) <–> (F, G, H) (share F, G)

Step 3: Group cliques into communities:

- Community 1:
  Clique 1 and Clique 2
  → Nodes: {A, B, C, D}
- Community 2:
  Cliques 3, 4, 5, 6
  (since they all overlap via pairs of nodes)
  → Nodes: {D, F, G, H}



6. Data Analytics with R

19. Describe applications of data visualization.

Data visualization helps convert complex data into easy-to-understand visual formats like charts, graphs, and dashboards. Its key applications include:

1. **Business Decision-Making:**
   Helps managers track sales trends, customer behaviour, and KPIs to make data-driven decisions.
2. **Scientific Research:**
   Enables researchers to analyze experimental results, correlations, and patterns quickly.
3. **Healthcare Analytics:**
   Used for monitoring patient health data, predicting disease outbreaks, and visualizing medical reports.
4. **Financial Analysis:**
   Helps investors and analysts understand market trends, stock performance, and risk factors.
5. **Social Media and Marketing:**
   Visualizes engagement metrics, campaign performance, and audience demographics to improve marketing strategies.
6. **Government and Public Policy:**
   Used to present census data, crime rates, or pollution levels for better planning and policy-making.
7. **Education and Learning:**
   Simplifies complex topics using interactive charts and dashboards to enhance understanding.
8. **Big Data Analytics:**
   Helps detect hidden patterns, outliers, and relationships within large datasets efficiently.

20. List and explain various functions that allow users to handle data in R workspace with appropriate examples.

1. getwd() – Get Working Directory

Shows the current working directory (where R reads/writes files).

```
getwd()                # Output: "C:/Data/R "
```

2. setwd() – Set Working Directory

Changes the working directory to a specified folder.

```
setwd("C:/Data/R2")    # Changes directory to R2
```

3. ls() – List Objects

Displays all objects currently in the R workspace.

```
x <- 10
y <- c(1, 2, 3)
ls()                # Output: "x" "y"
```

4. rm() – Remove Objects

Deletes one or more objects from the workspace.

```
rm(x)              # removes object x
rm(list = ls())  # removes all objects
```

5. save() – Save Objects to a File

Saves selected objects in a .RData file for later use.

```
a <- 5; b <- 10
save(a, b, file = "mydata.RData")    # Saves objects 'a' and 'b'
```

6. load() – Load Saved Objects

Restores saved objects from an .RData file into the current session.

```
load("mydata.RData")                # Loads objects 'a' and 'b'
```

7. history() – View Command History

Displays a list of previously executed R commands.

```
history()
```

8. summary() – Summarize Data

Gives statistical summary (min, max, mean, etc.) of a dataset.

```
summary(df)        # Output: Min, Median, Max
```

9. quit() or q() – Exit R (Optionally Save Workspace)

Ends the current R session and optionally saves the workspace image.

```
q()      # or quit()
```

21. What are the advantages of using functions over scripts? #

Advantages of Using Functions over Scripts in R:

1. **Code Reusability:** Once a function is defined in R, it can be reused across multiple scripts or projects.
2. **Simplifies Debugging:** Functions can be tested independently, making it easier to identify and fix issues.
3. **Improves Readability:** Functions make R scripts cleaner and easier to understand by grouping related operations.
4. **Enhances Modularity:** Large R programs can be broken into smaller functional parts for better organization.
5. **Supports Parameterization:** Functions can take arguments, allowing flexible and dynamic data analysis in R.

22. Write a script to create a dataset named data1 in R containing the following text:
Text: 2, 3, 4, 5, 6.7, 7, 8.1, 9 #

```
# Create a dataset named data1
data1 <- c(2, 3, 4, 5, 6.7, 7, 8.1, 9)


# Display the dataset
print(data1)   # Output: 2 3 4 5 6.7 7 8.1 9
```

23. Suppose you have two datasets A and B.
Dataset A has the following data: 6 7 8 9
Dataset B has the following data: 1 2 4 5
Which function is used to combine the data from both datasets into dataset C?
Demonstrate the function with the input values and write the output. #

In R, the c() function is used to combine two (or more) datasets (vectors) into one.
Demonstration with input values:

```
A <- c(6, 7, 8, 9)
B <- c(1, 2, 4, 5)

C <- c(A, B)    # Combine A and B into C
print(C)
```

Output:
6 7 8 9 1 2 4 5

24. The data analyst of Argon technology Mr. John needs to enter the salaries of 10 employees in R. The salaries of the employees are given in the following table:

Sr. No.	Name of employees	Salaries
1	Vivek	21000
2	Karan	55000
3	James	67000
4	Soham	50000
5	Renu	54000
6	Farah	40000
7	Hetal	30000
8	Mary	70000
9	Ganesh	20000
10	Krish	15000

i. Which R command will Mr. John use to enter these values demonstrate the output.
ii. Now Mr. John wants to add the salaries of 5 new employees in the existing table, which command he will use to join datasets with new values in R. Demonstrate the output.

i. Enter salaries of 10 employees in R

To enter data, Mr. John uses the data.frame() function.

```
# Create the vectors
names <- c("Vivek", "Karan", "James", "Soham", "Renu", "Farah", "Hetal", "Mary", "Ganesh", "Krish")
salaries <- c(21000, 55000, 67000, 50000, 54000, 40000, 30000, 70000, 20000, 15000)


# Create the data frame
employees <- data.frame(Name = names, Salary = salaries)
print(employees)
```

Output:

Name	Salary
Vivek	21000
Karan	55000

James	67000
Soham	50000
Renu	54000
Farah	40000
Hetal	30000
Mary	70000
Ganesh	20000
Krish	15000

ii. Add 5 new employees and join with existing table

To add new rows (new employees), he uses the rbind() command to combine both datasets.

```
# New employees data
new_names <- c("Rohit", "Sana", "Sameer", "Ahmed", "Wilson")
new_salaries <- c(72000, 46000, 38000, 58000, 53000)
new_employees <- data.frame(Name = new_names, Salary = new_salaries)

# Combine datasets
all_employees <- rbind(employees, new_employees)
print(all_employees)
```

Output:

Name	**Salary**
Vivek	21000
Karan	55000
James	67000
Soham	50000
Renu	54000
Farah	40000
Hetal	30000

Mary	70000
Ganesh	20000
Krish	15000
Rohit	72000
Sana	46000
Sameer	38000
Ahmed	58000
Wilson	53000

25. The following table shows the number of units of different products sold on different days:

Product	Monday	Tuesday	Wednesday	Thursday	Friday
Bread	12	3	5	11	9
Milk	21	27	18	20	15
Cola Cans	10	1	33	6	12
Chocolate bars	6	7	4	13	12
Detergent	5	8	12	20	23

Create five sample numeric vectors from this data. #

 # Vector for Bread sales

bread <- c(12, 3, 5, 11, 9)

 # Vector for Milk sales

milk <- c(21, 27, 18, 20, 15)

 # Vector for Cola Cans sales

cola_cans <- c(10, 1, 33, 6, 12)

 # Vector for Chocolate bars sales

choc-bars <- c(6, 7, 4, 13, 12)

 # Vector for Detergent sales

detergent <- c(5, 8, 12, 20, 23)