Big Data Analytics Exam Answer Key

Question bank Q1 to Q126 (CO1 to CO6) with the questions, short answers and diagrams

Module	Contents
CO1 (Q1 to Q21)	Introduction to Big Data: 5 Vs, types, life cycle, case studies
CO2 (Q22 to Q42)	Hadoop (HDFS, YARN), Pig, HBase, Spark, NoSQL
CO3 (Q43 to Q63)	MapReduce: execution, combiners, relational algebra, matrix operations
CO4 (Q64 to Q84)	Mining data streams: sampling, Bloom filter, FM, DGIM, DSMS
CO5 (Q85 to Q105)	Frequent patterns (PCY, SON), clustering, PCA, regression
CO6 (Q106 to Q126)	PageRank, HITS, social networks, recommendation engines

Contents

Contents 2

CO1: Introduction to Big Data (Q1 to Q21) 3

CO2: Big Data Frameworks, Hadoop and NoSQL (Q22 to Q42) 10

CO3: MapReduce Paradigm (Q43 to Q63) 21

CO4: Mining Big Data Streams (Q64 to Q84) 28

CO5: Big Data Mining Algorithms (Q85 to Q105) 36

CO6: Big Data Analytics Applications (Q106 to Q126) 44

CO1: Introduction to Big Data (Q1 to Q21)

Q1 **How does Big Data create value for businesses?** **[2]**

- **Better decisions** from data, not guesswork.
- **Customer insight and personalisation** (recommendations, targeted offers).
- **Efficiency:** lower cost, predictive maintenance, optimised supply chains.
- **Risk and fraud detection**, and new products and revenue streams.

Q2 **What is the significance of velocity in Big Data?** **[2]**

Velocity is the **speed at which data is generated and must be processed**. It matters because the value of data falls with time: real-time processing allows instant fraud alerts, live recommendations and stock trading. It needs stream tools such as Kafka, Storm, Spark Streaming.

Q3 **List the tools used in big data analytics.** **[2]**

Storage/processing: Hadoop (HDFS, MapReduce, YARN), Spark. **Query/scripting:** Hive, Pig. **NoSQL:** HBase, Cassandra, MongoDB. **Ingestion:** Sqoop, Flume, Kafka. **Streaming:** Storm, Flink. **Visualisation:** Tableau, Power BI.

Q4 **What are the challenges of storing Big Data?** **[2]**

- Volume is too big for one machine, so **distributed storage** is needed.
- **Cost** and **scalability** of hardware.
- **Variety** of formats (structured, semi, unstructured).
- **Fault tolerance** (disks fail), **security** and fast retrieval.

Q5 **Explain the concept of Big Data and identify various sources from which Big Data is generated.** **[2]**

Big Data = datasets so **large, fast or varied** that traditional tools (single-server RDBMS) cannot store or process them. **Sources:** social media, sensors/IoT, online and bank transactions, web logs and clickstreams, mobile apps, healthcare records, satellite images, emails, audio and video.

Q6 **Explain Unstructured Data with example.** **[2]**

Data with **no predefined model or schema**; it cannot be put in neat rows and columns. Examples: emails, images, videos, audio, social media posts, PDFs. It is about 80% of all data, stored in HDFS or NoSQL and analysed with text mining, NLP or computer vision.

Q7 **Explain the “Volume” characteristic in Big Data.** **[2]**

Volume is the **sheer amount of data** (terabytes, petabytes, zettabytes) produced by users, machines and sensors every second. It cannot fit on one machine, so it needs distributed storage and parallel processing (HDFS, MapReduce).

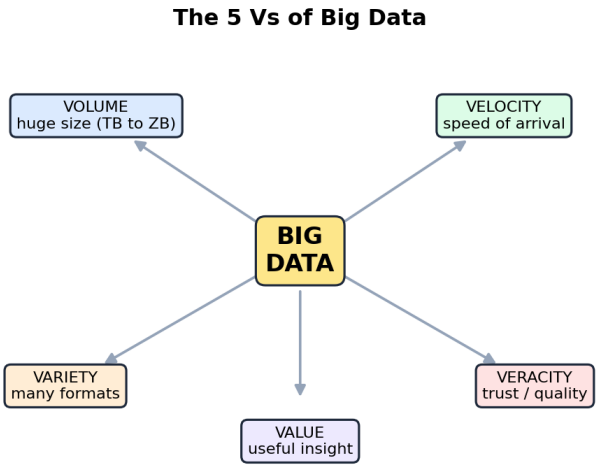


Figure: The 5 Vs of Big Data (Volume, Velocity, Variety, Veracity, Value)

Q8 Describe the different types of Big Data. [5]

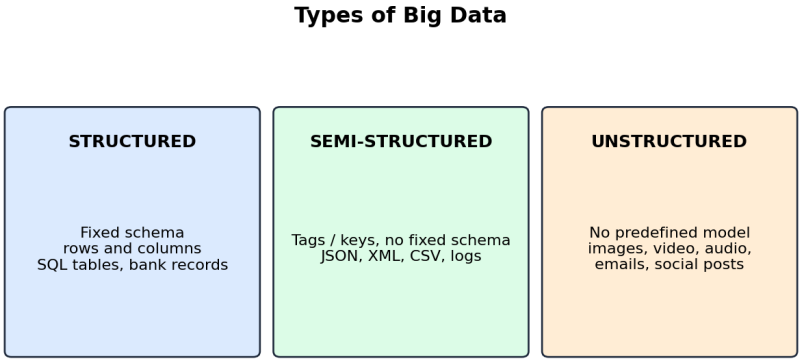


Figure: Structured, semi-structured and unstructured data

Type	Meaning	Examples	Handled by
Structured	Fixed schema, rows and columns	Bank records, SQL tables, Excel	RDBMS, Hive
Semi-structured	Self-describing tags, no rigid schema	JSON, XML, CSV, server logs	MongoDB, HBase
Unstructured	No predefined model	Images, video, audio, emails, tweets	HDFS, data lakes, NoSQL

Q9 Distinguish between Traditional Data Processing and Big Data Processing with example. [5]

Point	Traditional	Big Data
Data size	GB to TB	TB to PB and more
Data type	Mostly structured	Structured, semi and unstructured
Architecture	Centralised, single server	Distributed cluster
Scaling	Vertical (bigger machine)	Horizontal (add nodes)
Schema	Schema-on-write	Schema-on-read
Hardware / cost	Costly high-end servers	Cheap commodity hardware
Tools	RDBMS, SQL	Hadoop, Spark, NoSQL

Example: a bank keeps account balances in Oracle (traditional) but analyses millions of clickstream and transaction logs on Hadoop/Spark for fraud detection (Big Data).

Q10 Examine the influence of the 'Volume' characteristic of Big Data on the effectiveness and scalability of Big Data analytics systems. [5]

- **Effectiveness:** more data gives more accurate models and reveals rare patterns (e.g. fraud), but also more noise to clean.
- **Scalability pressure:** storage, memory, network and processing time grow with volume. Systems must **scale out** (add nodes) rather than up.
- **Design responses:** distributed file systems (HDFS), data locality (move code to data), parallel frameworks (MapReduce, Spark), sampling and compression.
- **Trade-off:** higher volume means more cost and slower queries unless the architecture scales linearly.

Q11 Discuss the new trends in Big Data analytics and their impact on various industries. [5]

Trend	Impact
AI / ML and Generative AI on big data	Predictive and automated decisions in every industry
Real-time / stream analytics	Fraud detection in banking, live traffic in smart cities
Edge computing and IoT	Predictive maintenance in manufacturing, smart homes
Cloud data lakes and lakehouses	Cheaper, elastic storage for retail and telecom
Data governance and privacy	Compliance in healthcare and finance
Augmented analytics, data fabric / mesh	Self-service insight for business users

Q12 Outline the life cycle of big data analytics, detailing each phase and its significance. [5]

Big Data analytics life cycle

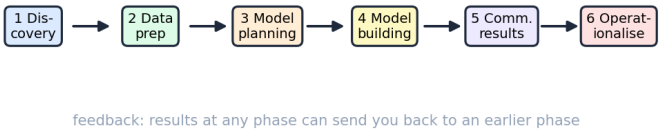


Figure: Big Data analytics life cycle

Phase	What happens	Significance
1 Discovery	Understand the business problem, data sources, goals	Sets direction
2 Data preparation	Collect, clean, transform, load (ETL)	Quality of data decides quality of results
3 Model planning	Select variables and techniques	Right method for the problem
4 Model building	Train and test models	Creates the insight
5 Communicate results	Share findings with visuals	Turns insight into decisions
6 Operationalise	Deploy and monitor	Delivers continuous value

Q13 Describe how social media collects and uses big data to make decisions. [5]

- **Collects:** posts, likes, shares, comments, clicks, searches, location, device data and ad interactions, through apps and APIs, stored in Hadoop/NoSQL.
- **Uses:** targeted advertising, content and friend recommendation, sentiment analysis of brands, trend detection, fake-news and spam detection, A/B testing of features.
- **Decisions:** what to show in your feed, which ads to run, which features to build or remove.

Q14 How does the diversity of data types and formats within a dataset show the 'Variety' characteristic of big data? [5]

Variety means data arrives in **many forms and formats**. Example, an e-commerce dataset holds a product table (structured), customer reviews in JSON (semi-structured), product photos and videos (unstructured) and server logs (semi-structured). Each needs a different storage and processing method, which is why one RDBMS is not enough and Big Data systems (HDFS, NoSQL, Spark) are used.

Q15 Outline the five major challenges in managing Big Data and explore their effects on decision-making processes in organizations. [10]

Challenge	Effect on decision-making
1 Storage and scalability of huge volumes	Data is left unused, so decisions miss information
2 Data quality (veracity): noise, duplicates, errors	Wrong insights lead to wrong decisions
3 Integration of varied data (variety, silos)	Incomplete or inconsistent view of the business
4 Security, privacy and compliance	Breaches, legal penalties, loss of customer trust
5 Real-time processing, cost and skills shortage	Late insights and high cost, so slow decisions

Remedies: distributed storage (HDFS, cloud), data cleaning and governance, ETL and data lakes, encryption and access control, stream tools and trained data teams.

Q16 Discuss the distinguishing characteristics of Big Data compared to traditional data, using examples to illustrate these differences. [10]

Characteristic	Big Data	Traditional data	Example
Volume	Petabytes and beyond	Gigabytes	Facebook logs vs a college database
Velocity	Real-time streams	Batch, periodic	Stock ticks vs monthly reports
Variety	All formats	Mostly structured	Videos and tweets vs sales table
Veracity	Uncertain, noisy	Cleaner, controlled	Social posts vs verified records
Value	Hidden, needs analytics	Direct reporting	Recommendations vs balance sheet
Processing	Distributed, parallel	Single server, SQL	Hadoop vs Oracle

Q17 Define big data analytics and explain the types of big data analytics with suitable example. [10]

Big data analytics is the process of examining large and varied datasets to uncover hidden patterns, correlations, trends and insights that support decisions.

Type	Question answered	Example (online retailer)
Descriptive	What happened?	Sales report: 20% rise in December
Diagnostic	Why did it happen?	Sales rose because of a festival discount and ads
Predictive	What will happen?	Forecast next month's demand with ML
Prescriptive	What should we do?	Recommend stock levels and best price

Q18 a. List and explain examples of big data. b. Discuss the challenges related to data privacy and security in Big Data, proposing viable solutions. [10]

(a) Examples:

- **Social media:** posts, likes, videos (Facebook, X).
- **E-commerce:** clicks, orders, reviews (Amazon).
- **Healthcare:** electronic records, scans, wearable data.
- **Banking:** transactions, card swipes. **Telecom:** call detail records.
- **IoT / transport:** sensor and GPS data.

(b) Privacy and security challenges and solutions:

Challenge	Solution
Unauthorised access, insider misuse	Authentication (Kerberos), role-based access control, audit logs
Data breaches in storage and transit	Encryption at rest and in transit (TLS)
Exposure of personal data	Anonymisation, masking, tokenisation, differential privacy
Regulations (GDPR, India DPDP Act)	Data governance, consent management, retention policies
Many nodes mean a large attack surface	Secure Hadoop (Ranger, Knox), network segmentation, monitoring

Q19 Construct a basic architecture for a Big Data solution focused on real-time analytics, using a Walmart case study as the foundation for your design. [10]

Walmart handles huge streams of sales, inventory and online-order data and wants **instant** decisions (restocking, pricing, fraud). A **Lambda-style** architecture fits:

Real-time analytics architecture (Walmart case)

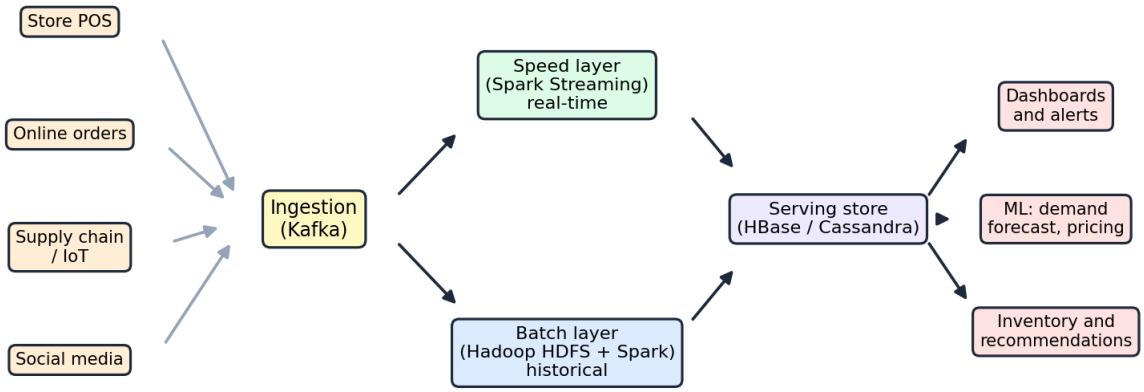


Figure: Real-time analytics architecture for a retail case like Walmart

- **Sources:** store POS, online orders, supply-chain/IoT sensors, social media.
- **Ingestion:** Kafka buffers events. **Speed layer:** Spark Streaming computes live results. **Batch layer:** HDFS + Spark computes accurate historical views.
- **Serving store:** HBase/Cassandra for fast reads. **Outputs:** dashboards and alerts, ML demand forecasts, inventory and recommendations.
- **Benefit:** shelves are restocked before they empty and prices react to demand within seconds.

Q20 **Examine how Big Data is applied in healthcare, detailing its impact on patient outcomes with relevant examples or case studies.** **[10]**

Data sources: electronic health records (EHR), medical images, genomics, wearables, insurance claims, lab results.

Application	Impact on patients	Example
Early diagnosis	Detect disease sooner	ML on scans detects eye disease or cancer
Predictive analytics	Predict readmission and risk	Hospitals flag high-risk patients
Personalised medicine	Treatment tailored to genes	Genomic data guides cancer therapy
Remote monitoring	Continuous care at home	Smartwatch detects irregular heart rhythm
Epidemic tracking	Faster public-health response	COVID-19 dashboards and contact data
Operations and cost	Shorter waits, lower cost	Bed and staff planning

Q21	**Develop a Netflix case study in Big Data analytics, explaining the role of machine learning with specific examples.**	[10]

Netflix big data and ML pipeline

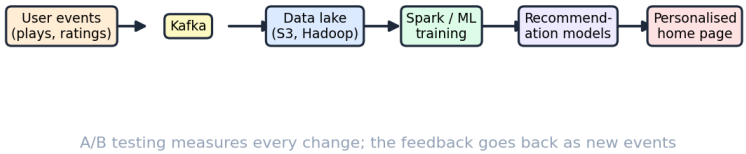


Figure: Netflix big data and ML pipeline

- **Data collected:** what you watch, when, on which device, pauses, searches, ratings, even the thumbnails you click.
- **Storage and processing:** events flow through Kafka into a cloud data lake (AWS S3, Hadoop) and are processed with Spark; Cassandra serves fast lookups.
- **ML role:** (1) **recommendation engine** (collaborative filtering, matrix factorisation, deep learning) that drives most of what people watch; (2) **personalised artwork** per user; (3) **content decisions** (which shows to produce); (4) **adaptive streaming quality** prediction; (5) **A/B testing** of every change.
- **Outcome:** higher engagement and lower churn.

CO2: Big Data Frameworks, Hadoop and NoSQL (Q22 to Q42)

Q22	**What is Hadoop?**	[2]

Hadoop is an **open-source Apache framework** for **distributed storage (HDFS) and distributed processing (MapReduce)** of very large datasets on clusters of commodity hardware. It is scalable, fault tolerant and cost effective.

Q23	**List the core components of Hadoop.**	[2]

1. **HDFS** (storage)
2. **YARN** (resource management and job scheduling)
3. **MapReduce** (parallel processing)
4. **Hadoop Common** (shared libraries and utilities)

Q24	**Explain the concept of fault tolerance in Hadoop.**	[2]

Fault tolerance = the system keeps working when nodes fail. HDFS **replicates each block (default 3 copies)** on different nodes and racks. DataNodes send **heartbeats**; if one stops, the NameNode re-replicates its blocks, and failed tasks are re-run on other nodes. NameNode HA uses a standby.

Q25	**Describe the role and significance of blocks in HDFS for managing large datasets.**	[2]

HDFS splits every file into **fixed-size blocks (128 MB by default)** stored on different DataNodes. This lets files bigger than any one disk be stored, allows **parallel processing**, makes replication simple (per block) and keeps NameNode metadata small.

Q26 Illustrate what is a Secondary Name Node? Does it substitute the Name Node? [2]

The Secondary NameNode **periodically merges the FsImage with the edit log** to create a fresh checkpoint, so the edit log does not grow forever. It is **not a backup or hot standby** and does **not** replace the NameNode (a Standby NameNode in HA does).

Q27 Classify why HDFS is more suited for applications having large datasets and not when there are small files? [2]

- HDFS is built for **large files, streaming reads, write-once-read-many** and high throughput.
- Every file, directory and block takes about 150 bytes of **NameNode RAM**; millions of small files exhaust memory.
- Many small files also mean many small map tasks and slow seeks, so low-latency access is poor.

Q28 List Advantages and Disadvantages of NoSQL. [2]

Advantages	Disadvantages
Horizontal scaling, flexible schema, high availability, fast for big data, handles unstructured data	Weaker consistency (eventual), no standard query language, limited joins and transactions, less mature tools

Q29 Explain how big data problems are handled by Hadoop system. [5]

Big Data problem	How Hadoop handles it
Huge volume	**HDFS** splits and spreads data across many nodes
Slow processing	**MapReduce / Spark** run in parallel, moving code to the data (data locality)
Hardware failure	Block **replication** and task re-execution
Cost	Runs on cheap **commodity hardware**
Scalability	Just **add nodes** (horizontal scaling)
Variety	**Schema-on-read** stores any format

Q30 Compare and contrast RDBMS with NoSQL databases, emphasizing key differences and scenarios where each is most effective. [5]

Point	RDBMS	NoSQL
Schema	Fixed, predefined	Flexible / dynamic
Data model	Tables	Key-value, column, document, graph
Scaling	Vertical	Horizontal
Consistency	ACID	BASE (eventual consistency)
Query	SQL, joins	API / custom query
Best for	Banking, ERP, structured transactions	Social media, IoT, real-time big data, catalogues

Q31 **Examine the architecture of Pig, explaining how it processes data and integrates with the Hadoop ecosystem.** **[5]**

Pig architecture



Grunt shell / script / embedded (UDF) are the ways to submit Pig Latin

Figure: Apache Pig architecture

- **Pig Latin** scripts are submitted via Grunt shell, script file or embedded code.
- **Parser** checks syntax and builds a logical plan (DAG); **Optimizer** improves it; **Compiler** converts it into MapReduce jobs; **Execution engine** runs them on Hadoop.
- **Integration:** reads from and writes to HDFS, runs on YARN, works with HBase and Hive data. Saves writing long Java code.

```
A = LOAD 'input.txt' AS (line:chararray);
B = FOREACH A GENERATE FLATTEN(TOKENIZE(line)) AS word;
C = GROUP B BY word;
D = FOREACH C GENERATE group, COUNT(B);
STORE D INTO 'wordcount';
```

Q32 **Analyze the architecture of HBase, focusing on its design principles and how it supports large-scale data storage and retrieval.** **[5]**

HBase architecture

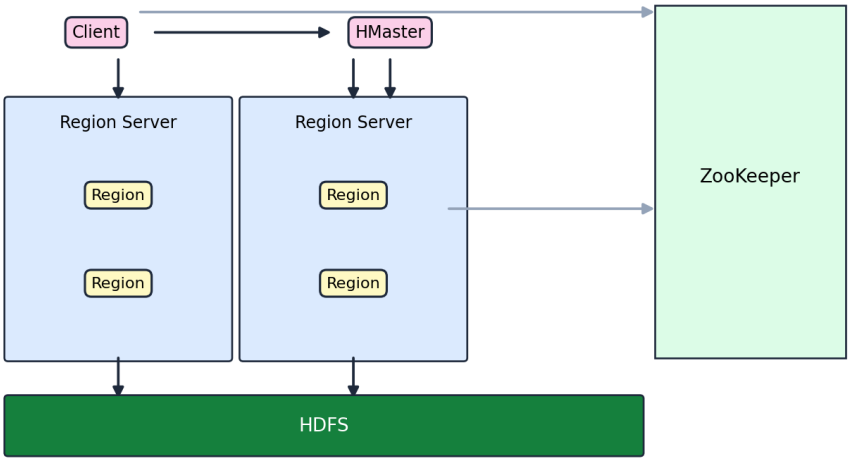


Figure: HBase architecture (Client, HMaster, Region Servers, ZooKeeper, HDFS)

- **HMaster:** assigns regions, handles schema changes, balances load.
- **Region Server:** serves reads and writes for many **Regions** (horizontal slices of a table sorted by row key). It has a **WAL**, a **MemStore** (memory) and **HFiles** (disk).
- **ZooKeeper:** tracks live servers and elects the master. **HDFS:** stores the data durably.
- **Design principles:** column-oriented, sorted row keys, automatic sharding (regions split as data grows), strong consistency per row, random real-time read/write on top of HDFS.

Q33 / Q41 **Q33: Examine the role of Apache Spark in the Hadoop ecosystem, discussing its architecture and analyzing different use cases. Q41: Analyze the role of Apache Spark in the Hadoop ecosystem, discussing its architecture and analyzing different use cases.** Q33 (5 marks) and Q41 (10 marks) ask the same thing; answer once, add depth for 10 marks **[5]**

Role: Spark is a fast, **in-memory** processing engine that sits on top of Hadoop: it uses **HDFS** for storage and **YARN** for resource management, and replaces slow disk-based MapReduce for many jobs (up to about 100 times faster in memory).

Apache Spark architecture

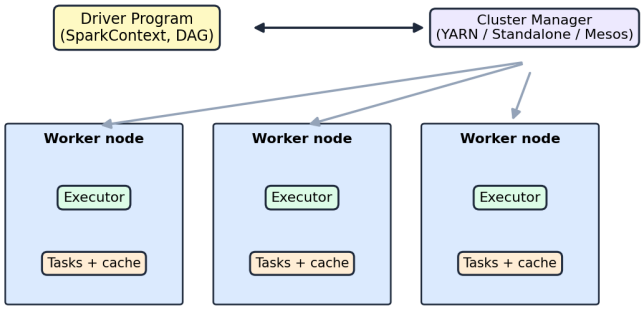


Figure: Spark architecture (Driver, Cluster Manager, Executors)

- **Driver** runs the main program and builds the DAG of tasks; **Cluster Manager** (YARN, Standalone, Mesos) allocates resources; **Executors** on worker nodes run tasks and cache data.
- **Components:** Spark Core (RDDs), Spark SQL, Spark Streaming, MLlib (machine learning), GraphX (graphs).

Use case	Why Spark
ETL and batch analytics	Fast transformations on HDFS data

Use case	Why Spark
Iterative machine learning	Keeps data in memory between iterations
Real-time stream processing	Micro-batch Spark Streaming
Interactive queries	Spark SQL on large tables
Graph processing	GraphX

Q34 **Analyze basic Hadoop commands, explaining their functions and how they are used to manage Hadoop operations.** **[5]**

Command	Function
hdfs dfs -ls /path	List files in HDFS
hdfs dfs -mkdir /dir	Create a directory
hdfs dfs -put local.txt /dir	Copy file from local to HDFS (also -copyFromLocal)
hdfs dfs -get /dir/f.txt .	Copy from HDFS to local
hdfs dfs -cat /dir/f.txt	Show file contents
hdfs dfs -rm -r /dir	Delete file or directory
hdfs dfs -du -h /dir	Show size used
hadoop jar app.jar Main in out	Run a MapReduce job
hdfs dfsadmin -report	Cluster and DataNode status
start-dfs.sh / start-yarn.sh / jps	Start daemons / list running daemons

Q35 **Evaluate the role of “FsImg” and “Edit logs” in HDFS, discussing their significance in ensuring data consistency and integrity.** **[5]**

- **FsImage:** a snapshot of the whole file-system namespace (directories, files, block mapping) saved on the NameNode disk.
- **Edit log:** a journal that records every change made after the last snapshot (create, delete, rename).
- **On start-up** the NameNode loads FsImage and **replays** the edit log to rebuild the current state; it then stays in safe mode until blocks are reported.
- **Checkpoint:** the Secondary/Standby NameNode merges the two into a new FsImage so recovery is fast.
- **Significance:** ensures no metadata is lost after a crash, so the namespace stays consistent. Keeping copies on multiple disks protects integrity.

Q36 **Examine the architecture of YARN, describing its components and how it manages resources in a Hadoop cluster.** **[10]**

YARN architecture (master-slave)

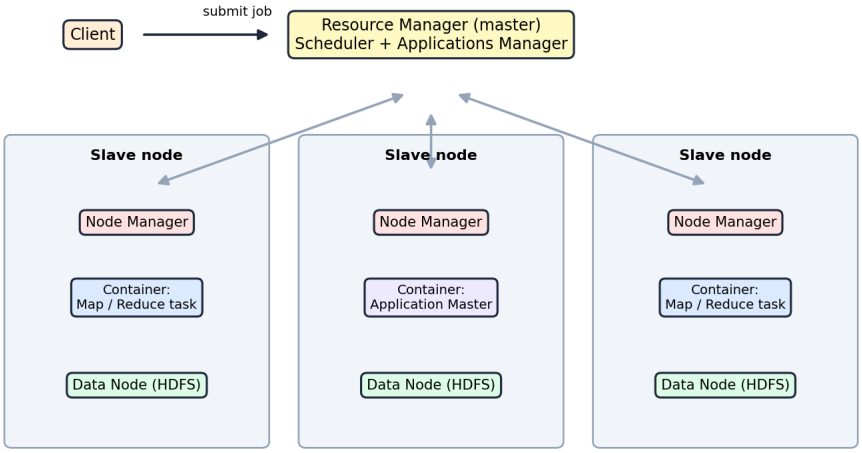


Figure: YARN architecture (master-slave)

- **Resource Manager (master):** has a **Scheduler** (allocates resources by capacity/fair policy, does no monitoring) and an **Applications Manager** (accepts jobs, starts the ApplicationMaster).
- **Node Manager (per slave node):** manages containers, monitors CPU/memory and reports to the Resource Manager.
- **Application Master (per job):** negotiates resources and tracks the job's tasks.
- **Container:** a bundle of CPU and memory in which a task runs.

1. Client submits a job to the Resource Manager.
2. Resource Manager starts a container for the Application Master.
3. Application Master requests containers for map and reduce tasks.
4. Node Managers launch the containers; tasks run near the data.
5. Application Master reports completion; resources are released.

Benefit: YARN separates resource management from processing, so MapReduce, Spark and others share one cluster with better utilisation and scalability.

Q37 **Design a data pipeline using Hadoop, where data is ingested using Sqoop, processed using Apache Spark, and stored in HBase. Include a diagram to illustrate the architecture.** **[10]**

Data pipeline: Sqoop → Spark → HBase

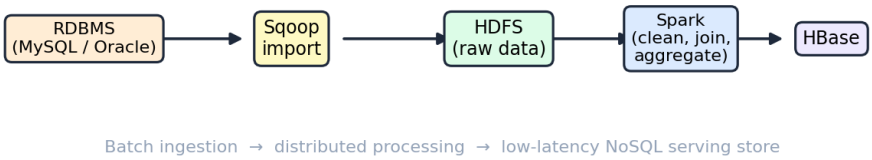


Figure: Sqoop to Spark to HBase pipeline

1. **Ingest (Sqoop):** import tables from the RDBMS into HDFS in parallel.
2. **Process (Spark):** read from HDFS, clean, join and aggregate.
3. **Store (HBase):** write results to HBase for fast random reads by the application.
4. **Schedule** the steps with Oozie or Airflow; monitor with YARN UI.

```
sqoop import --connect jdbc:mysql://host/shop --table sales \
      --username u -P --target-dir /data/sales -m 4

# PySpark
df = spark.read.csv('/data/sales', header=True)
summary = df.groupBy('product').sum('amount')
# write 'summary' to HBase table 'sales_summary' (column family 'cf')
```

Q38 Analyze the components of the Hadoop Ecosystem, summarizing their roles and how they work together to handle Big Data. [10]

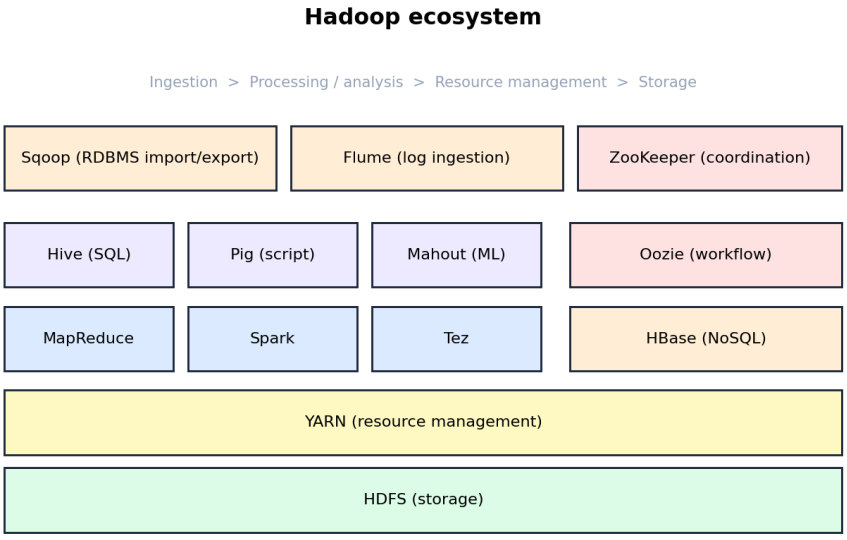


Figure: Hadoop ecosystem layers

Component	Role
HDFS	Distributed storage
YARN	Resource management and scheduling
MapReduce / Spark / Tez	Parallel processing engines
Hive	SQL-like queries (HiveQL) on HDFS
Pig	Dataflow scripting (Pig Latin)
HBase	Real-time NoSQL column store
Sqoop	Transfer between RDBMS and Hadoop
Flume / Kafka	Ingest logs and streams
Oozie	Workflow scheduling
ZooKeeper	Coordination and configuration
Mahout / MLlib	Machine learning

How they work together: data is ingested (Sqoop, Flume, Kafka), stored in HDFS or HBase, scheduled by Oozie, processed by MapReduce/Spark/Hive/Pig under YARN, with ZooKeeper coordinating the services.

Q39 Explore the process of reading and writing data in HDFS, detailing the steps involved and their significance in Hadoop's operation. [10]

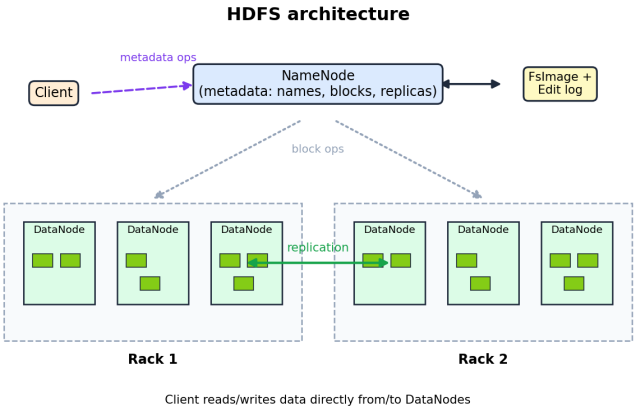


Figure: HDFS architecture

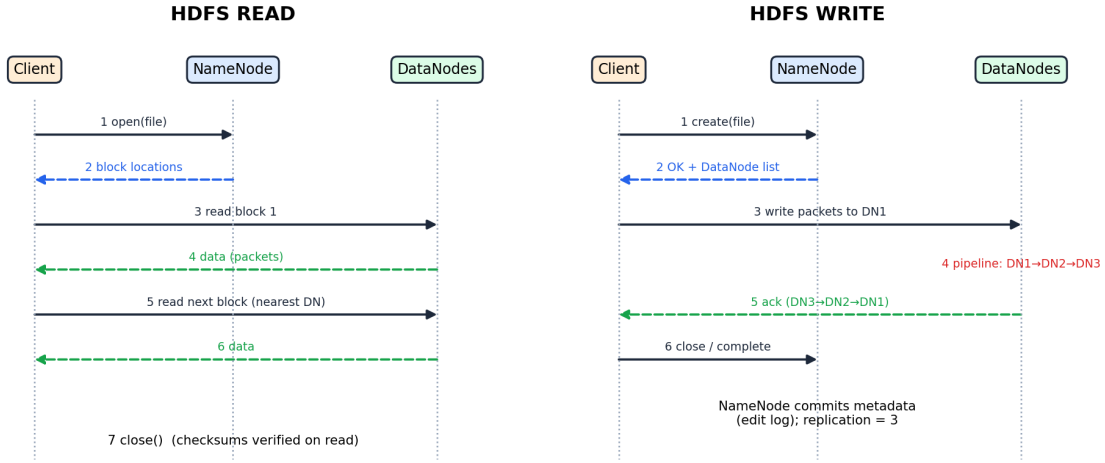


Figure: HDFS read and write flow

Read: (1) client asks the NameNode to open the file; (2) NameNode returns block locations (nearest DataNodes first); (3) client reads blocks **directly from DataNodes** in order; (4) checksums are verified; (5) client closes the file.

Write: (1) client asks the NameNode to create the file; (2) NameNode returns a list of DataNodes for each block; (3) client sends packets to DataNode 1, which forwards to 2 and 3 as a **replication pipeline**; (4) acknowledgements flow back; (5) client closes the file and the NameNode commits the metadata.

Significance: the NameNode only handles metadata and data flows directly to DataNodes, so there is no bottleneck; replication gives fault tolerance and rack awareness gives speed and safety.

Q40 Examine RDDs in Spark, explaining their types and providing detailed examples of their usage in real-world applications. [10]

An **RDD (Resilient Distributed Dataset)** is Spark's core data structure: an **immutable, partitioned collection** spread over the cluster. It is **resilient** (lost partitions are rebuilt from its **lineage**), **distributed** and **lazily evaluated**.

- **Creation:** from a collection (parallelize) or an external file (textFile from HDFS).
- **Transformations** (lazy, make a new RDD): map, filter, flatMap, reduceByKey, join. *Narrow* (map, filter) need no shuffle; *wide* (reduceByKey, join) shuffle data.
- **Actions** (trigger work): count, collect, reduce, take, saveAsTextFile.
- **Types / kinds:** basic RDD, **Pair RDD** (key-value, for reduceByKey and join), RDD from files (HadoopRDD), and RDDs produced by transformations (MapPartitionsRDD, ShuffledRDD).

```
lines = sc.textFile('hdfs:///logs/app.log')
errors = lines.filter(lambda l: 'ERROR' in l)      # transformation
counts = errors.map(lambda l: (l.split()[2], 1)) \
```

```
                .reduceByKey(lambda a, b: a + b)      # transformation
print(counts.collect())                               # action
```

Real-world uses: log analysis (count errors per service), fraud detection (iterative scoring on card data), recommendation (iterative ALS in MLlib), clickstream aggregation, word count on text corpora.

Q42	**a. Analyze the key features of different types of NoSQL databases and differentiate their capabilities and use cases. b. Analyze the benefits and limitations of using NoSQL databases like Cassandra in comparison to traditional RDBMS.**	**[10]**

Types of NoSQL databases

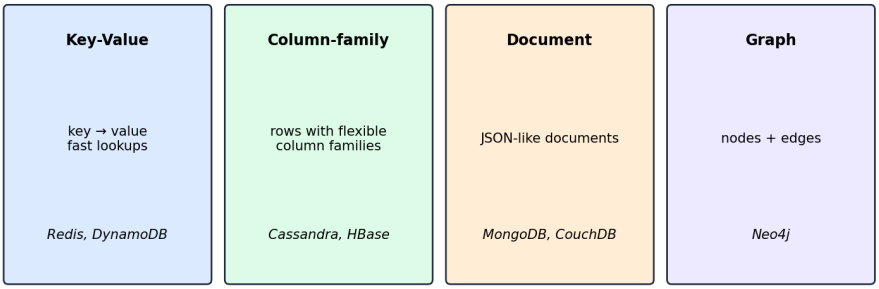


Figure: Four types of NoSQL databases

Type	Key features	Examples	Use cases
Key-value	Simple key to value, very fast	Redis, DynamoDB	Caching, sessions, shopping carts
Column-family	Wide rows, column families, high write throughput	Cassandra, HBase	Time series, logs, IoT
Document	JSON documents, flexible schema, rich queries	MongoDB, CouchDB	Content management, catalogues, user profiles
Graph	Nodes and edges, relationship queries	Neo4j	Social networks, fraud rings, recommendations

(b) Cassandra vs RDBMS:

Benefits of Cassandra	Limitations
Linear horizontal scaling, no single point of failure (peer-to-peer), always available, very fast writes, flexible schema, multi-datacenter replication	Eventual consistency, no joins or complex transactions, query-driven data modelling, limited ad-hoc queries, less mature tooling and SQL features

CO3: MapReduce Paradigm (Q43 to Q63)

Q43 **Describe the function of the Map task within the MapReduce framework.** **[2]**

The Map task reads one **input split**, applies the user's map function to every record and emits intermediate **(key, value)** pairs. Map tasks run in parallel on the nodes that hold the data (data locality).

Q44 **Define the term “Reducer” in the context of MapReduce.** **[2]**

A Reducer receives each key with the **list of all its values** (after shuffle and sort) and combines them (sum, count, join, max) to produce the final output pairs, which are written to HDFS.

Q45 **Explain the concept of “Grouping by Key” in MapReduce.** **[2]**

After the map phase, the framework collects all values that share the **same key** into one list, giving (key, [v1, v2, ...]). The **partitioner** (hash(key) mod number of reducers) sends one key to exactly one reducer. This is the shuffle and sort step.

Q46 **Explain the concepts of union and intersection in the context of relational algebra operations.** **[2]**

- **Union R ∪ S:** all tuples in R or S, with duplicates removed.
- **Intersection R ∩ S:** only the tuples that appear in both R and S.

Q47 **What is Grouping & Aggregation in Relational Algebra Operations** **[2]**

$\gamma_{A,\,\theta(B)}(R)$: **group** the tuples of R by attribute(s) A and apply an **aggregate** θ (SUM, COUNT, AVG, MAX, MIN) to attribute B inside each group. Same as SQL *SELECT A, SUM(B) ... GROUP BY A*.

Q48 **Identify and describe a real-world example where MapReduce is effectively applied** **[2]**

Web indexing / word count at Google: map emits (word, document) for every word of every page; reduce collects the list of pages per word, building the inverted index used by search. Other uses: log analysis, counting URL access frequency.

Q49 **Analyze and list the benefits associated with using the MapReduce process.** **[2]**

- **Scalability** to thousands of nodes and petabytes.
- **Fault tolerance** (failed tasks are re-run).
- **Parallelism and data locality** (code goes to the data).
- **Simplicity** (just map and reduce), runs on **cheap commodity hardware**, handles any data type.

Q50 **Describe the role of combiners in optimizing MapReduce jobs and provide an example.** **[5]**

A **combiner** is a mini-reducer run on the map side. It aggregates a mapper's output **before the shuffle**, so less data crosses the network. It must be **commutative and associative** (sum, max, count).

Example (word count): mapper output (car,1)(car,1)(river,1) becomes (car,2)(river,1) after the combiner: 3 pairs reduced to 2.



Figure: Without and with a combiner

Q51 Develop a pseudo code for implementing the Matrix-Vector Multiplication algorithm using MapReduce. [5]

Matrix M (n × n) is stored as triples (i, j, m_{ij}); vector v is available to every mapper. Result $x_i = \Sigma_j m_{ij} \cdot v_j$.

```
map(key, value):            # value = (i, j, m_ij)
    emit(i, m_ij * v[j])

reduce(i, list_of_values):
    emit(i, sum(list_of_values))
```

Q52 Compare and analyze traditional approaches to parallel and distributed processing, highlighting their methods and limitations. [5]

Approach	Method	Limitations
Multi-threading / shared memory	Threads on one machine	Limited by one machine's cores and memory
MPI / message passing	Programmer sends messages between processes	Complex, manual partitioning, no built-in fault tolerance
Grid / HPC clusters	Central storage, compute nodes	Network bottleneck moving data to compute, costly
Parallel RDBMS	Partitioned SQL databases	Expensive, mostly structured data, hard to scale out
MapReduce	Simple map and reduce, framework manages the rest	Batch only, disk heavy, not for iterative jobs

Q53 Describe the detailed execution process of a MapReduce job, from the submission to the output. [5]

MapReduce execution flow



Map output is partitioned by key; all values of a key reach the same reducer

Figure: MapReduce execution flow

1. **Submit:** client sends the job to the Resource Manager; code and config are copied to HDFS.
2. **Split:** input is divided into splits (one map task per split).
3. **Map:** tasks run near the data and emit (key, value) pairs, buffered in memory then spilled to disk, partitioned and sorted.
4. **Combine** (optional) on the map side.
5. **Shuffle and sort:** reducers copy their partitions over the network and merge them by key.
6. **Reduce:** reduce function runs once per key.
7. **Output:** results are written to HDFS. Failed tasks are re-executed on other nodes.

Q54 Analyze how MapReduce can be used to perform matrix-vector multiplication. [5]

- The product is $x_i = \Sigma_j m_{ij} \cdot v_j$.
- **Map:** for each matrix element (i, j, m_{ij}) emit the key i with the value $m_{ij} \cdot v_j$.
- **Reduce:** for each row i, add all the values to get x_i.
- If the vector is too big for memory, split matrix into vertical **stripes** and the vector into matching pieces so each mapper holds one piece. A worked example is in Q59.

Q55 Illustrate with an example how MapReduce can be used to compute the projection operation in relational algebra. [5]

$\pi_{A,B}(R)$ keeps only the chosen attributes and removes duplicates. **Map:** for each tuple t emit (t[A,B], t[A,B]). **Reduce:** receives (key, list of copies) and emits the key **once**, eliminating duplicates.

R(A, B, C)	Map output (key = (A,B))	Reduce output
(1, 2, 3)	((1,2), (1,2))	(1, 2)
(1, 2, 4)	((1,2), (1,2))	(1, 2) emitted once
(2, 3, 5)	((2,3), (2,3))	(2, 3)

Q56 Write a map reduce pseudo code for word count problem. Illustrate with an example showing all the steps. [5]

```
map(key, line):
    for each word w in line:
        emit(w, 1)

reduce(word, counts):
    emit(word, sum(counts))
```

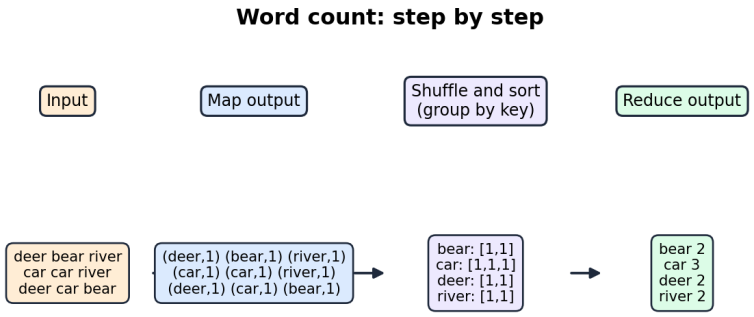


Figure: Word count, input to output

Q57 **Analyze how relational algebra operations are computed and executed using MapReduce, including the role of Map and Reduce functions** **[10]**

Operation	Map function	Reduce function
Selection $\sigma_C(R)$	For each tuple t, if t satisfies C emit (t, t)	Identity (just output)
Projection $\pi_A(R)$	Emit (t[A], t[A])	Emit each key once (removes duplicates)
Union	Emit (t, t) for tuples from both R and S	Emit key once
Intersection	Emit (t, t) from R and S	Emit key only if it has 2 values (in both)
Difference R − S	Emit (t, 'R') or (t, 'S')	Emit t only if its list is just ['R']
Natural join R(A,B) ⋈ S(B,C)	Emit (b, ('R', a)) and (b, ('S', c))	For each b, pair every R value with every S value
Grouping / aggregation	Emit (group key, value)	Apply SUM/COUNT/AVG/MAX on the list

The key is chosen so that tuples that must meet (same join value, same duplicate tuple, same group) get the **same key** and arrive at the same reducer through shuffle.

Q58 **i. Describe Map Reduce algorithm for Natural Join of two relations with example. ii. Describe Map Reduce algorithm to perform Intersection of two sets with example.** **[10]**

(i) Natural join R(A, B) ⋈ S(B, C):

- **Map:** for each tuple (a, b) of R emit key b with value ('R', a); for each tuple (b, c) of S emit key b with value ('S', c).
- **Reduce:** for each b, take the R values and S values and emit (a, b, c) for every pair.

R(A,B)	S(B,C)	Reducer for key b1	Join output
(a1,b1) (a2,b1) (a3,b2)	(b1,c1) (b2,c2) (b2,c3)	R: a1, a2 ; S: c1	(a1,b1,c1) (a2,b1,c1)
		Reducer for key b2	R: a3 ; S: c2, c3 gives (a3,b2,c2) (a3,b2,c3)

(ii) Intersection R ∩ S: Map: for every element t of R or S emit (t, t). **Reduce:** if the key has **two** values (it came from both sets) emit it; if one, drop it. Example: R = {1, 2, 3, 4}, S = {3, 4, 5}: keys 3 and 4 get two values, so the result is **{3, 4}**.

Q59 Evaluate matrix vector multiplication by map reduce. Illustrate with an example, consider the following matrix: M = [[1, 2], [2, 1], [3, 4]] and [[1, 2], [1, 3]]. [10]

Your photo shows a 3×2 matrix M and a 2×2 matrix. I treat each column of the second matrix as a vector, so the answer is M·v for v_1 = (1, 1) and v_2 = (2, 3), which together give the matrix product.

Map emits $(i, m_{ij} \cdot v_j)$; **Reduce** sums per row i.

Row i	Map output for v_1 = (1,1)	Sum	Map output for v_2 = (2,3)	Sum
1	(1, 1×1) (1, 2×1)	**3**	(1, 1×2) (1, 2×3)	**8**
2	(2, 2×1) (2, 1×1)	**3**	(2, 2×2) (2, 1×3)	**7**
3	(3, 3×1) (3, 4×1)	**7**	(3, 3×2) (3, 4×3)	**18**

Result: $M \cdot v_1$ = (3, 3, 7) and $M \cdot v_2$ = (8, 7, 18). As a matrix: [[3, 8], [3, 7], [7, 18]].

Q60 Explain the operation of grouping and aggregation using Map Reduce with the given example (two map workers, each with two files of columns A, B, C, D). [10]

The question does not state which columns to group and aggregate. I used SELECT A, SUM(B) ... GROUP BY A. The same steps work for any other columns. Values are as read from your photo.

Map: each worker reads its rows and emits (A, B). **Combiner (optional)** adds up per worker. **Reduce:** adds the partial sums for each key.

A	Worker 1 emits (A,B)	Worker 2 emits (A,B)	Reduce: SUM(B)
1	(1,2) (1,2)	(1,2) (1,3)	2+2+2+3 = **9**
2	(2,2)	(2,3) (2,3)	2+3+3 = **8**
3	(3,2)	(3,2) (3,4)	2+2+4 = **8**
4	(4,2)	–	**2**
6	(6,8)	–	**8**

Output: (1, 9), (2, 8), (3, 8), (4, 2), (6, 8). Check: total of B = 35 = 9 + 8 + 8 + 2 + 8.

Q61 Demonstrate the Map Reduce program without combiner and with Combiner using suitable example [10]

Input: Mapper 1 reads “car car river”; Mapper 2 reads “car deer car”.

	Without combiner	With combiner
Mapper 1 output	(car,1) (car,1) (river,1)	(car,2) (river,1)
Mapper 2 output	(car,1) (deer,1) (car,1)	(car,2) (deer,1)
Pairs sent over the network	6	4
Reducer input	car: [1,1,1,1]; river: [1]; deer: [1]	car: [2,2]; river: [1]; deer: [1]
Final output	car 4, river 1, deer 1	car 4, river 1, deer 1 (same)



Figure: The combiner reduces shuffled data

```
// Java: enable a combiner (here the reducer class works as combiner)
```

```
job.setMapperClass(TokenMapper.class);
job.setCombinerClass(SumReducer.class);
job.setReducerClass(SumReducer.class);
```

Result is the same, but the combiner cuts network traffic and speeds up the job.

Q62 Create a detailed diagram that summarizes the MapReduce process in nutshell [10]

MapReduce execution flow

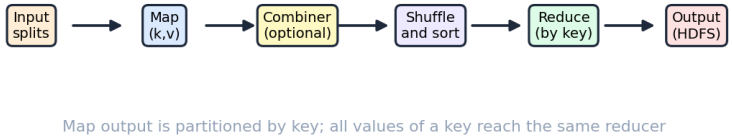


Figure: MapReduce in a nutshell

Word count: step by step

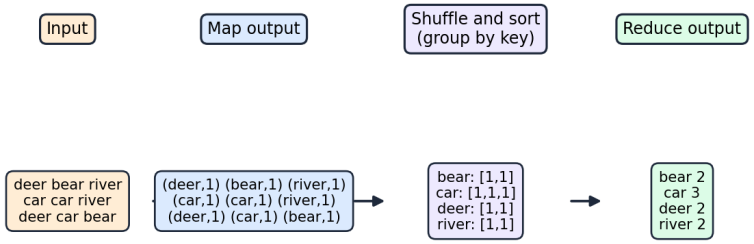


Figure: The same process with word-count data

- **Input** is split and read from HDFS.
- **Map** turns records into (key, value) pairs in parallel.
- **Combiner** (optional) pre-aggregates on each mapper.
- **Shuffle and sort** groups all values by key and sends each key to one reducer.
- **Reduce** aggregates each group and writes the result to HDFS.

Q63 Explain the operation of union using Map Reduce with the given example (two map workers, each with Table 1 and Table 2 of columns A, B). [10]

Rule: Union R ∪ S = every distinct tuple that is in R or in S. **Map:** for each tuple t (from Table 1 or Table 2) emit (t, t). **Reduce:** receives (t, list of copies); emit t **once**.

	Table 1 (R)	Table 2 (S)
Map Worker 1	(1,2) (2,3) (5,6)	(2,3) (4,4) (6,1)
Map Worker 2	(6,1) (6,3) (7,6)	(9,8) (3,3) (0,1)

(2,3) and (6,1) appear in both tables, so their reducer lists have two copies but each is output once.

Union output (10 tuples): (1,2), (2,3), (5,6), (6,1), (6,3), (7,6), (4,4), (9,8), (3,3), (0,1).

CO4: Mining Big Data Streams (Q64 to Q84)

Q64 What is Decaying Windows? Explain in detail? [2]

A decaying window gives **more weight to recent stream elements** and less to old ones, instead of a hard sliding window. With decay constant c (very small, e.g. 10^{-6}), the sum is $S = \Sigma\, a_{t-i}(1-c)^i$. **Update:** $S \leftarrow S\cdot(1-c) + a_{t+1}$, so no old data is stored. Use: finding the "most popular / trending" items by giving each item a score and dropping scores below ½.

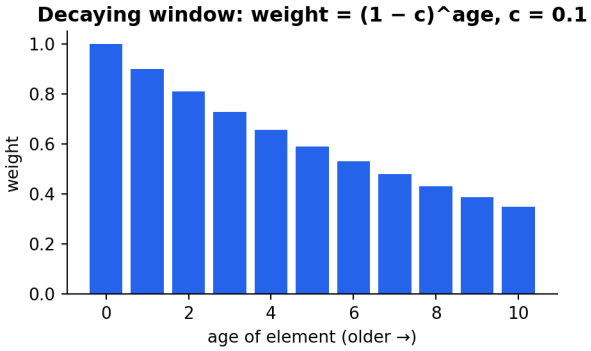


Figure: Weight falls exponentially with age

Q65 Discuss different issues in data stream processing [2]

- Data is **unbounded** and arrives fast; only **one pass** is possible.
- **Limited memory**, so only summaries or approximate answers can be kept.
- Need **real-time** (low latency) answers; **bursty** input.
- Out-of-order or late data, **concept drift**, fault tolerance and data quality.

Q66 List and explain various data stream sources [2]

- **Sensor networks / IoT:** temperature, GPS, smart meters.
- **Web:** click streams and search queries.
- **Social media:** tweets, posts, likes.
- **Finance:** stock tickers, card transactions.
- **Network and telecom:** packet logs, call records. **Images and satellite** feeds.

Q67 How do sensor networks and social media platforms contribute as sources of data streams? [2]

Sensor networks produce continuous time-stamped readings (weather, traffic, patient vitals) at high rates. **Social media** produces a never-ending flow of posts, likes and shares per second. Both are unbounded, fast and time-ordered, so they are analysed in real time for alerts, traffic control and trend or sentiment detection.

Q68 How does a stream query differ from a traditional SQL query? [2]

	Stream query	Traditional SQL query
Data	Unbounded, arriving stream	Finite stored table
Execution	**Continuous** (standing), runs forever	**One-time**, runs and ends
Answer	Incremental, often approximate, over windows	Exact, complete
Example	Alert if temperature above 40 in the last 5 minutes	SELECT AVG(temp) FROM readings

Q69	**What is data stream? Explain data stream operations in the context of big data?**	**[2]**

A **data stream** is a continuous, unbounded, ordered sequence of data items arriving at high speed.
Operations: filtering, sampling, windowing (sliding and tumbling), aggregation (count, sum), joins, counting distinct elements, frequent-item detection and pattern detection.

Q70 Discuss the main components of the DGIM algorithm? [2]

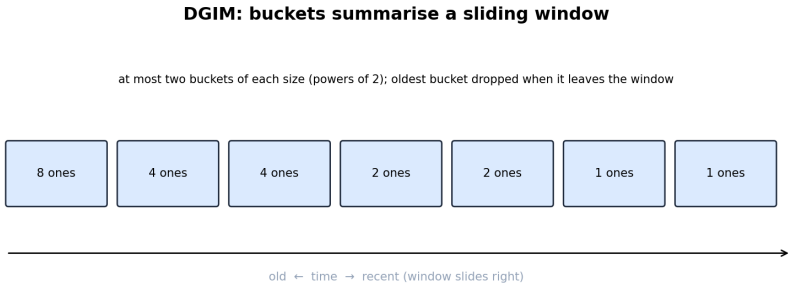


Figure: DGIM buckets over a sliding window

- **Window of size N** over a binary stream; the aim is to count the 1s in the last k bits.
- **Buckets:** each stores the timestamp of its most recent 1 and a **size** (number of 1s) that is a power of 2.
- **Rules:** at most two buckets of each size; sizes do not decrease towards the past; buckets do not overlap; a bucket is dropped when its timestamp leaves the window.
- **Query estimate:** add the sizes of all buckets except the oldest, plus **half** of the oldest. Error is at most 50% using only $O(\log^2 N)$ bits.

Q71 Explain count-distinct problem in stream processing with suitable example? [5]

Problem: count the number of **different** elements in a stream when there is not enough memory to remember all of them. Example: unique visitors to a website, or distinct IP addresses.

- **Exact:** keep a hash set of every element, which needs memory proportional to the number of distinct items (too much for billions).
- **Approximate (Flajolet-Martin):** hash each element, find the longest run of trailing zeros R, estimate distinct count = 2^R. Memory is only O(log m).
- Example: stream 1, 3, 2, 1, 2, 3, 4 has **4 distinct** elements (1, 2, 3, 4).

Q72 Explain the difference between batch processing and stream processing, highlighting the advantages of stream processing. [5]

	Batch	Stream
Data	Large stored dataset	Continuous, unbounded
Latency	Minutes to hours	Milliseconds to seconds
Run	Scheduled	Always on
Tools	MapReduce, Hive	Kafka, Storm, Flink, Spark Streaming
Example	Monthly sales report	Credit-card fraud alert

Advantages of streaming: real-time insight and quick reaction, no need to store everything, continuous monitoring, fresher decisions.

Q73 Investigate and discuss scenarios where a Bloom filter might produce false positives and false negatives. How can these issues be mitigated? [5]

- **False positive:** bits set by **other** items make all k bits of a new item 1, so the filter wrongly says "probably present". It rises as the filter fills up (n large, m small).
- **False negative:** a standard Bloom filter **never** gives one. It can happen only if bits are cleared to support deletion, or if the bit array is corrupted.
- **Mitigation:** use a bigger m, the optimal $k = (m/n) \cdot \ln 2$, good independent hash functions; use a **counting Bloom filter** for deletions; use a **scalable Bloom filter**; confirm "present" answers against the real store when exactness matters.

Q74 Explain what a Data Stream Management System (DSMS) is, and illustrate its architecture with a block diagram. [5]

A DSMS is software that accepts **continuous input streams**, runs **standing queries** over them and also answers **ad-hoc queries**, while keeping only limited summaries.

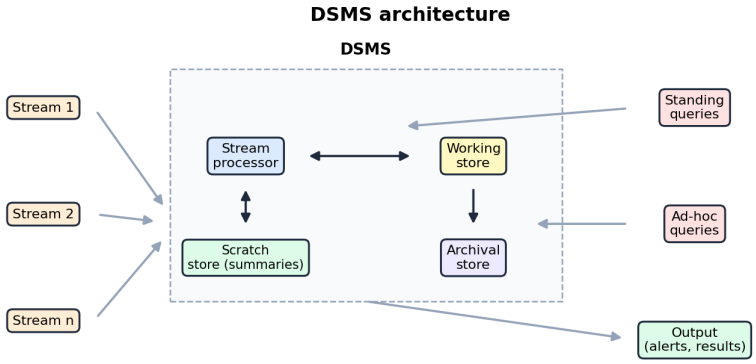


Figure: DSMS architecture

- **Stream processor** runs queries; **working store** (memory) holds recent window data; **scratch/summary store** holds sketches; **archival store** (disk) keeps old data for slow ad-hoc queries.
- Inputs are many streams; outputs are results and alerts.

Q75 Explain what stream queries are and describe the different categories of stream queries with examples. [5]

Stream queries are questions asked about the data in a stream. Two main categories:

Category	Meaning	Example
Standing (continuous) queries	Registered once, run permanently, output as data arrives	Alert whenever temperature exceeds 40 °C
Ad-hoc queries	One-time question asked at any moment, answered from stored summaries or window	What was the maximum temperature in the last 24 hours?

Queries are often evaluated over **windows** (sliding or tumbling), for example “average stock price over the last 10 minutes”.

Q76 Imagine you're processing a binary stream to estimate the count of ones in a sliding window. Apply the concept of the DGIM algorithm to achieve this. [5]

Take window N = 16 and the current time 16. Suppose DGIM keeps these buckets (oldest to newest): size 4 (ts 5), size 4 (ts 9), size 2 (ts 12), size 2 (ts 14), size 1 (ts 15), size 1 (ts 16).

- **When a new bit arrives:** if it is 0, do nothing except drop any bucket whose timestamp is outside the window. If it is 1, create a size-1 bucket.
- **Merge:** if there are now three buckets of the same size, merge the two oldest into one of double size (this can cascade).
- **Estimate** = sum of all bucket sizes except the oldest + half of the oldest = (4 + 2 + 2 + 1 + 1) + 4/2 = 10 + 2 = **12** ones in the window.
- Error is at most 50% and memory is $O(\log^2 N)$.

Q77 Compare the efficiency of reservoir sampling with other sampling techniques in stream processing. [5]

Reservoir sampling keeps a uniform random sample of k items from a stream of unknown length: store the first k; for the i-th item (i > k) keep it with probability k/i, replacing a random slot.

Technique	Memory	Notes
Reservoir sampling	O(k), one pass	Uniform sample, no need to know stream length
Fixed-rate (every n-th / hash on key)	O(1) to small	Sample size grows with the stream; hashing keeps all tuples of one user together
Sliding-window sampling	O(window)	Only recent data; needs window bookkeeping
Systematic / periodic	O(1)	Can be biased by periodic patterns

Q78 **Given a scenario where you need to check the existence of an item in a large dataset, explain how you would use a Bloom filter to perform this check?** **[10]**

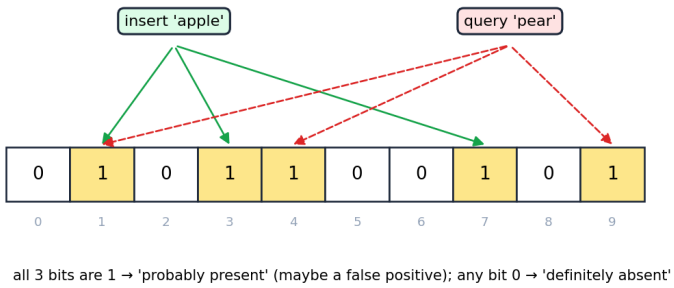


Figure: Bloom filter insert and query

Scenario: a web crawler must check whether a URL has already been crawled, among billions of URLs.

1. Choose the bit-array size m and number of hash functions k from the expected items n and the allowed false-positive rate.
2. Start with all m bits = 0.
3. **Insert** each crawled URL: compute k hashes and set those bits to 1.
4. **Query** a new URL: compute its k hashes. If **any bit is 0**, it is **definitely not crawled**; if **all are 1**, it is **probably crawled** (small false-positive chance).

Why use it: very small memory, O(k) time, no false negatives. **False-positive rate** $\approx (1 - e^{-kn/m})^k$.

Q79 **Determine the distinct element in the stream using FM algorithm. Input stream of Integers = 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1 with h(x) = (6x + 1) mod 5** **[10]**

Hash each distinct value, write it in binary and count the **trailing zeros**.

x	h(x) = (6x+1) mod 5	Binary	Trailing zeros
1	7 mod 5 = 2	010	1
3	19 mod 5 = 4	100	2
2	13 mod 5 = 3	011	0
4	25 mod 5 = 0	000	0 (hash value 0 is counted as 0 zeros)

Maximum trailing zeros R = **2**. **Estimate = $2^R = 2^2 = 4$ distinct elements.** This matches the true number of distinct values {1, 2, 3, 4}.

Q80 **Determine the distinct element in the stream using FM algorithm. Input stream of Integers = 4, 2, 5, 9, 1, 6, 3, 7. a) h(x) = 3x + 1 mod 32 b) h(x) = x + 6 mod 32** **[10]**

x	a) 3x+1 mod 32	Binary	Zeros	b) x+6 mod 32	Binary	Zeros
4	13	01101	0	10	01010	1
2	7	00111	0	8	01000	3
5	16	10000	4	11	01011	0
9	28	11100	2	15	01111	0
1	4	00100	2	7	00111	0
6	19	10011	0	12	01100	2
3	10	01010	1	9	01001	0
7	22	10110	1	13	01101	0

- **(a)** R = 4, estimate = 2^4 = **16**.
- **(b)** R = 3, estimate = 2^3 = **8**.
- The true number of distinct elements is 8, so hash (b) happens to be exact; FM is an estimate and one hash can be off by a power of 2 (use many hashes and average or take medians).

Q81 **Analyze the operation of a Bloom filter in detail. Provide a thorough explanation of how it works and illustrate its functionality with a practical example.** **[10]**

A Bloom filter is a bit array of m bits (all 0) plus k hash functions. **Insert:** set the k hashed bits to 1. **Query:** answer "possibly present" only if all k bits are 1.

Example: m = 10, k = 3, h1(x) = x mod 10, h2(x) = (2x + 3) mod 10, h3(x) = (3x + 1) mod 10.

Action	h1, h2, h3	Bits set / checked
Insert 11	1, 5, 4	bits 1, 4, 5 set to 1
Insert 19	9, 1, 8	bits 1, 8, 9 set to 1. Array now has 1s at 1, 4, 5, 8, 9
Query 15	5, 3, 6	bit 3 is 0, so **definitely absent**
Query 29	9, 1, 8	all three bits are 1, so **probably present**, but 29 was never inserted: a **false positive**

- **Properties:** no false negatives; false positives possible; no deletion; very small memory.
- **Tuning:** false-positive rate ≈ $(1 - e^{-kn/m})^k$; best k = (m/n)·ln 2.
- **Uses:** spam filters, web caches, crawler URL checks, database lookups.

Q82 **Explain how the Flajolet-Martin algorithm can be used to estimate the number of distinct elements in a stream. Analyse the trade-offs between the Flajolet-Martin algorithm and the DGIM algorithm in terms of accuracy and memory space requirements.** **[10]**

FM: pick a hash function mapping elements to long bit strings; for each element compute the hash and the number of trailing zeros; keep the maximum R seen; estimate the number of distinct elements as 2^R. A rare long run of zeros implies many different elements were hashed. To improve accuracy use many hash functions, group them, take the average within groups and the **median** of the averages.

	Flajolet-Martin	DGIM
Solves	Count of distinct elements	Count of 1s in a sliding window
Accuracy	Estimate within a factor of about 2 for one hash; better with many hashes	Error at most 50% (can be reduced by keeping more buckets per size)
Memory	O(log m) bits per hash (just R)	O($\log^2$ N) bits (buckets)

	Flajolet-Martin	DGIM
Window	Whole stream	Last N elements only
Trade-off	More hashes means more memory but higher accuracy	More buckets per size means more memory but lower error

Q83 **Compare and contrast the challenges of estimating the count of distinct elements in a stream with the challenges of counting ones in a sliding window. How do the algorithms address these challenges?** **[10]**

	Distinct elements	Ones in a sliding window
Core challenge	Cannot remember all elements seen (memory)	Cannot store the whole window, and old bits must **expire**
Needs	Detect duplicates without storing items	Track which bits are inside the window
Algorithm	**FM:** hash and track the longest run of trailing zeros	**DGIM:** buckets of power-of-2 sizes with timestamps
Result	Approximate 2^R	Approximate count with at most 50% error
Common idea	Both give **approximate answers in tiny memory** using summaries (sketches) in one pass	

Q84 **A) Apply the concept of Bloom filters to a situation where you're processing incoming emails. Explain how a Bloom filter could be used to identify potential spam emails. B) A Bloom filter with m = 1000 cells is used to store information about n = 100 items, using k = 4 hash functions. Calculate the false positive probability.** **[10]**

(A) Spam detection:

1. Build a Bloom filter containing the **known spam sender addresses / domains / message signatures** (set k bits for each).
2. For every incoming email hash the sender (or signature) with the k functions.
3. If any bit is 0: the sender is **not known spam**, so deliver it.
4. If all bits are 1: the email is **probably spam**, so send it to a stricter check or the spam folder. Because false positives are possible, a second check protects genuine mail.

(B) False positive probability: $P = (1 - e^{-kn/m})^k$ with k = 4, n = 100, m = 1000.

```
kn/m = 4 x 100 / 1000 = 0.4
e^-0.4 = 0.6703
1 - 0.6703 = 0.3297
P = 0.3297^4 = 0.0118
```

False positive probability ≈ 0.0118, about 1.2%.

CO5: Big Data Mining Algorithms (Q85 to Q105)

Q85	What is the main focus of frequent pattern mining, and why is it important in data analysis?	[2]

It finds **itemsets or patterns that occur frequently** (support ≥ a minimum threshold) in a dataset, e.g. items bought together. Important for **market-basket analysis, association rules, recommendations, cross-selling and fraud detection**.

Q86	Describe the fundamental principle behind clustering algorithms and their purpose in data analysis.	[2]

Group objects so that those in the **same cluster are similar** (high intra-cluster similarity) and those in **different clusters are dissimilar**. It is unsupervised and is used for segmentation, discovering structure, summarisation and outlier detection.

Q87	Discuss the applications of Canopy Algorithm	[2]

Canopy is a cheap **pre-clustering** step that speeds up k-means or hierarchical clustering on huge, high-dimensional data. Uses: document and web-page clustering, customer segmentation, duplicate (record linkage) detection and clustering with MapReduce.

Q88	Describe the main goals of visual data analysis techniques and how they facilitate better understanding of complex datasets.	[2]

Goals: **see patterns, trends, outliers and relationships** quickly, explore data interactively and **communicate insights**. Charts, heat maps and dashboards turn big tables into pictures that people grasp faster.

Q89	Discuss the advantages of PCY algorithm?	[2]

- Uses the idle memory of pass 1 to **hash pairs into buckets**, so far **fewer candidate pairs** in pass 2 than A-Priori.
- Needs only **two passes**, with lower memory use, so it suits large datasets.

Q90	How large data is handled in main memory? Enlist the algorithms used to handle large datasets in main memory?	[2]

In frequent-itemset mining, **counting pairs** needs the most memory. Memory is saved by counting only promising candidates, hashing and sampling instead of counting everything. **Algorithms:** A-Priori, PCY, Multistage, Multihash, Random Sampling / Toivonen, and SON (partition based).

Q91	Explain all three pass of multistage algorithm	[2]

1. **Pass 1:** count the items and hash every pair into buckets with **hash function h1**.
2. **Pass 2:** between passes make a bitmap of frequent buckets. Then hash only the pairs whose items are both frequent **and** whose h1 bucket is frequent, with a second independent **hash function h2**, into new buckets.
3. **Pass 3:** count a pair only if both items are frequent **and** its h1 bucket bit **and** its h2 bucket bit are both 1. These are the final candidates.

Q92 Given a large dataset and memory constraints, outline the steps of the basic Park, Chen, and Yu (PCY) algorithm for frequent pattern mining. [5]

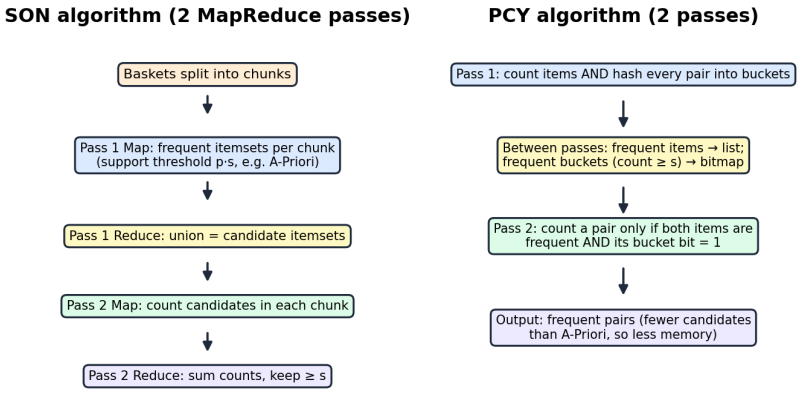


Figure: SON (left) and PCY (right)

1. **Pass 1:** read the baskets; count each item and also hash every pair of items in the basket to a bucket, incrementing the bucket count.
2. **Between passes:** keep items with count ≥ s (frequent items). Turn bucket counts into a **bitmap**: 1 if the bucket count ≥ s.
3. **Pass 2:** count a pair (i, j) only if i and j are both frequent **and** the pair hashes to a bucket whose bit is 1.
4. **Output:** pairs with count ≥ s are the frequent pairs.

Q93 List down two algorithms used for frequent pattern mining? Describe their key characteristics in detail. [5]

	A-Priori	FP-Growth
Idea	Level-wise: frequent k-itemsets build candidate (k+1)-itemsets	Compress the data into an **FP-tree** and mine it recursively
Key property	Anti-monotone: every subset of a frequent itemset is frequent	Shares common prefixes, so no candidate generation
Passes	One pass per level (many)	Two scans of the data
Cost	Many candidates, heavy I/O	Needs the tree in memory; faster on dense data

(PCY and SON are further options built on A-Priori for large data.)

Q94 Analyse the strengths and limitations of the CURE algorithm for clustering in comparison to traditional centroid-based clustering algorithms. [5]

CURE (Clustering Using Representatives) is a hierarchical method: each cluster is represented by several **well-scattered points**, which are **shrunk towards the centroid** by a factor α. It works on a random sample and partitions to scale.

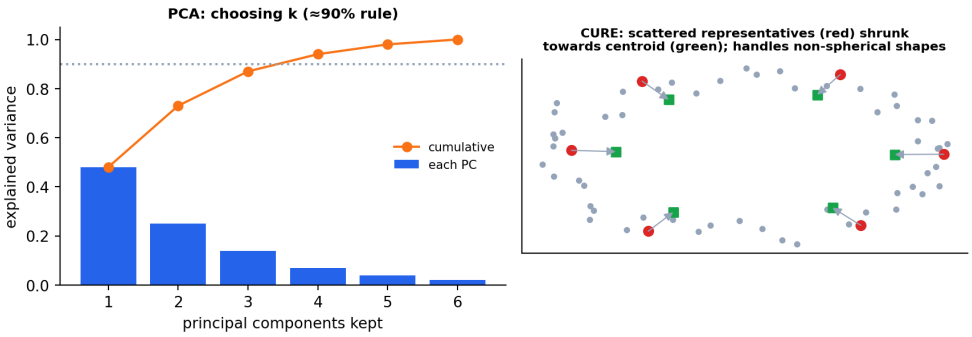


Figure: PCA variance chart (left) and CURE representatives (right)

	CURE	Centroid-based (k-means)
Cluster shape	Handles **non-spherical** and different sizes	Assumes round clusters of similar size
Outliers	Robust (shrinking dampens them)	Sensitive
Scalability	Sampling + partitioning	Fast and simple
Limitations	More parameters (number of representatives, α), sample quality, higher cost, complex	Needs k in advance, local optima

Q95 **Compare the Canopy clustering approach to hierarchical agglomerative clustering. What are the computational and memory trade-offs between these methods?** **[5]**

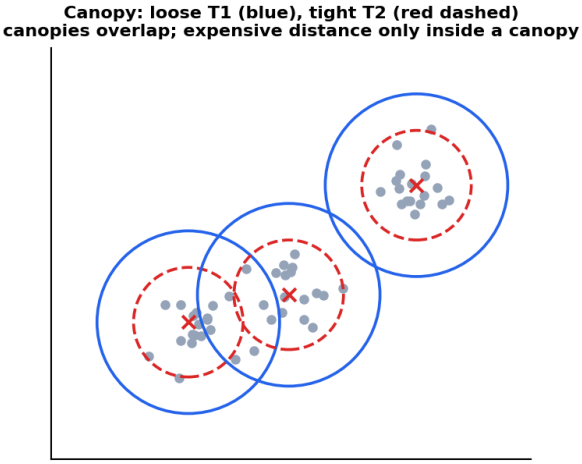


Figure: Canopy clustering

	Canopy	Hierarchical agglomerative
Idea	Cheap rough grouping into **overlapping** canopies with loose (T1) and tight (T2) thresholds, then precise clustering inside each	Start with each point alone and repeatedly merge the closest clusters into a dendrogram
Time	About O(n), as expensive distances are only computed inside canopies	$O(n^2)$ to $O(n^3)$
Memory	Low	$O(n^2)$ distance matrix
Output	Rough pre-clusters (usually refined by k-means)	Complete hierarchy
Scale	Very large data, MapReduce friendly	Small to medium data

Q96 **Illustrate how the SON partition-based algorithm helps to perform frequent item set mining for large datasets. How does this algorithm avoid False negatives?** **[5]**

1. **Split** the baskets into chunks that each fit in memory.
2. **Pass 1:** in each chunk find the frequent itemsets using a lowered threshold p·s (p = chunk fraction, s = support) with any in-memory method (A-Priori, PCY).
3. The **union** of these is the candidate set.
4. **Pass 2:** count all candidates over the whole data and keep those with count ≥ s.

No false negatives: an itemset that is frequent in the whole dataset must be frequent (≥ p·s) in **at least one chunk**, so it is certain to be in the candidate set. Pass 2 only removes false positives.

Q97 **What is the primary function of visualization techniques in data analysis, and how do they aid in extracting insights from data?** **[5]**

- **Function:** turn data into graphics (bar, line, scatter, heat map, dashboard, network graph).
- **Insight:** reveal trends, patterns, clusters, outliers and correlations faster than tables; support interactive drill-down and easy communication to decision-makers.
- Tools: Tableau, Power BI, D3.js, matplotlib.

Q98	**Analyse the impact of varying the number of principal components retained after PCA on the quality of data representation and the explanatory power of the reduced dimensions.**	**[5]**

Few components	Many components
More compression and noise removal, faster models	Closer to the original data
Lower explained variance, more information lost, poorer reconstruction	**Higher explained variance** but more dimensions, more noise kept, less reduction

Choose k so that the **cumulative explained variance is about 90 to 95%** or at the **elbow** of the scree plot (see the figure in Q94, left).

Q99 Analyse clustering with MapReduce. [10]

Clustering with MapReduce (k-means style)

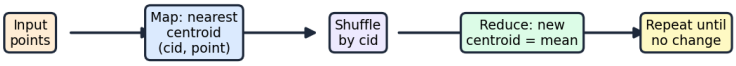


Figure: k-means clustering with MapReduce

k-means in MapReduce (one MapReduce job per iteration):

```
map(point):     c = nearest centroid to point;  emit(c.id, point)
combine(id, pts): emit(id, (sum of pts, count))        # optional
reduce(id, pts):  emit(id, mean of pts)                 # new centroid
# driver: repeat jobs until centroids stop moving
```

- Centroids are small, so they are shared with every mapper (distributed cache).
- Points are read and assigned in parallel, so the method scales to huge datasets; the combiner cuts network traffic.
- **Canopy first:** a Canopy job gives rough centres and cuts distance calculations; **CURE / hierarchical** methods also run on samples and partitions.
- **Drawback:** each iteration re-reads data from disk (Spark is faster for iterative work).

Q100 Illustrate how the PCY algorithm is used in big data analytics. [10]

Use case: market-basket analysis for a retailer with billions of baskets, to find pairs of products bought together (cross-selling, shelf placement, recommendations). Counting every pair would not fit in memory, so PCY is used.

1. **Pass 1** (Map): each mapper reads baskets and emits (item, 1) and (bucket of pair, 1). Reducers sum to get item counts and bucket counts.
2. Select frequent items and build the **bitmap of frequent buckets**.
3. **Pass 2** (Map): emit (pair, 1) only for pairs whose items are frequent and bucket bit is 1; Reduce sums and keeps pairs with count ≥ s.
4. Use the frequent pairs to build **association rules** (support, confidence) and extend to bigger itemsets.

Benefit: far fewer candidate pairs than plain A-Priori, so less memory and faster mining (see the PCY figure in Q92).

Q101 Apply PCY algorithm on the following transactions to find candidate set. Use buckets and the concept of mapreduce to solve the problem. Threshold: 2, Hash function: (i × j) mod 10. T1:{1,2,3} T2:{2,3,4} T3:{3,4,5} T4:{4,5,6} T5:{1,3,5} T6:{2,4,6} T7:{1,3,4} T8:{2,4,5} T9:{3,5,6} T10:{1,2,4} T11:{2,3,5} T12:{3,4,6} [10]

Pass 1, Map: for every transaction emit (item, 1) for each item, and for each pair emit (bucket, 1) where bucket = (i × j) mod 10. **Reduce:** sum the counts.

Item	1	2	3	4	5	6
Count	4	6	8	8	6	4

All items have count ≥ 2, so **all six items are frequent**.

Bucket	Pairs hashed to it (with occurrences)	Count	Frequent? (≥ 2)
0	(2,5)×2, (4,5)×3, (5,6)×2	7	Yes (bit 1)
2	(1,2)×2, (2,6)×1, (3,4)×4	7	Yes (bit 1)

Bucket	Pairs hashed to it (with occurrences)	Count	Frequent? (≥ 2)
3	(1,3)×3	3	Yes (bit 1)
4	(1,4)×2, (4,6)×3	5	Yes (bit 1)
5	(1,5)×1, (3,5)×4	5	Yes (bit 1)
6	(2,3)×3	3	Yes (bit 1)
8	(2,4)×4, (3,6)×2	6	Yes (bit 1)

Buckets 1, 7 and 9 receive no pair. Bitmap (buckets 0 to 9) = **1 0 1 1 1 1 1 0 1 0** in the order 0, 1, 2 ... 9 is: 0→1, 1→0, 2→1, 3→1, 4→1, 5→1, 6→1, 7→0, 8→1, 9→0.

Pass 2, Map: emit (pair, 1) only if both items are frequent and the pair's bucket bit is 1. **Reduce:** sum.

Candidate pair	Bucket	Count	Frequent (≥ 2)?
(1,2)	2	2	Yes
(1,3)	3	3	Yes
(1,4)	4	2	Yes
(1,5)	5	1	No
(2,3)	6	3	Yes
(2,4)	8	4	Yes
(2,5)	0	2	Yes
(2,6)	2	1	No
(3,4)	2	4	Yes
(3,5)	5	4	Yes
(3,6)	8	2	Yes
(4,5)	0	3	Yes
(4,6)	4	3	Yes
(5,6)	0	2	Yes

Candidate set (pairs that pass both tests): 14 pairs shown above. **Frequent pairs (12):** (1,2), (1,3), (1,4), (2,3), (2,4), (2,5), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6). Pairs (1,5) and (2,6) occur only once, so they are not frequent.

Note: in this small data every non-empty bucket is frequent, so the hash table prunes nothing here; the bucket step becomes useful when there are many more pairs than buckets can hold.

Q102 **Apply different classification algorithms to a sample dataset. Describe how you would implement algorithms such as Decision Trees, SVM, K-NN and Logistic Regression. Explain the steps involved in each algorithm and how you would evaluate their performance using metrics like accuracy, precision, and recall.** **[10]**

Sample dataset: Iris, or a customer dataset with features and a class label (e.g. buys / does not buy). Steps common to all: load data, clean and scale, split into train/test (70/30), fit the model, predict on test data, evaluate.

Algorithm	Steps
Decision Tree	Choose the best split by Gini or entropy at each node, grow the tree, prune, then predict by following the path
SVM	Scale features, pick a kernel, find the maximum-margin hyperplane (tune C and gamma), predict by the side of the hyperplane
K-NN	Scale features, choose k, compute distance from a test point to all training points, take a majority vote of the k nearest
Logistic Regression	Fit weights with the sigmoid and log-loss by gradient descent, predict the class when probability ≥ 0.5

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3)
for m in [DecisionTreeClassifier(), SVC(), KNeighborsClassifier(5), LogisticRegression()]:
    print(classification_report(yte, m.fit(Xtr, ytr).predict(Xte)))
```

Evaluation: confusion matrix, **accuracy** = (TP+TN)/all, **precision** = TP/(TP+FP), **recall** = TP/(TP+FN), F1; use cross-validation and compare the models on the same test data. (Details in the ML key.)

Q103 **Compare and contrast the strengths and limitations of simple linear regression and multiple linear regression models in terms of their applicability and complexity.** **[10]**

	Simple linear regression	Multiple linear regression
Equation	y = b0 + b1·x	y = b0 + b1·x1 + b2·x2 + ... + bn·xn
Predictors	One	Two or more
Strengths	Simple, easy to interpret and visualise, fast	Uses more information, usually more accurate, controls for several factors
Limitations	Ignores other factors, may under-fit	Multicollinearity, over-fitting, needs more data, harder to interpret
Complexity	Low	Higher (more parameters, feature selection)
Applicability	One clear driver (marks vs study hours)	Real problems with many drivers (house price from area, rooms, location)

Both assume a **linear** relationship, independent errors and constant variance.

Q104 **Given a dataset with a high number of features, outline the steps you would take to perform dimension reduction using PCA. What considerations would guide your choices?** **[10]**

1. **Clean** the data and handle missing values.
2. **Standardise** each feature (mean 0, standard deviation 1).
3. Compute the **covariance matrix**.
4. Find its **eigenvalues and eigenvectors** (or use SVD).
5. Sort the components by eigenvalue (variance explained).
6. Keep the top k components.
7. **Project** the data: $Z = X \cdot W_k$.

- **Choosing k:** cumulative explained variance (90 to 95%), scree-plot elbow, or what the next model needs.
- **Considerations:** scaling is essential, PCA is linear only, components are hard to interpret, outliers distort it, for huge data use incremental or randomised PCA (or MapReduce/Spark MLlib), and apply PCA only after splitting the data to avoid leakage.

Q105	How does the SON (Savasere, Omiecinski, and Navathe) algorithm leverage MapReduce to mine frequent patterns? What advantage does MapReduce bring to this process?	[10]

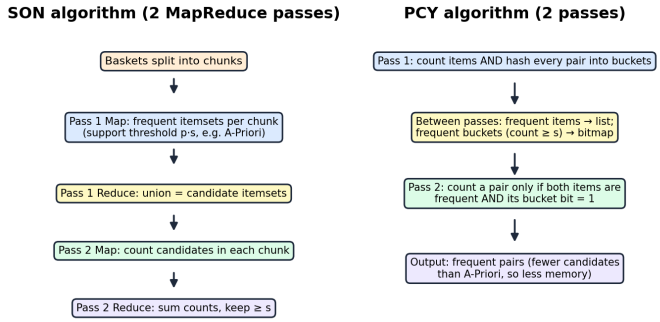


Figure: SON algorithm (left)

Pass 1: Map runs an in-memory frequent-itemset algorithm on its chunk (threshold p·s) and emits (itemset, 1). **Reduce** removes duplicates, giving the **candidate itemsets**.

Pass 2: Map counts how often each candidate occurs in its chunk and emits (candidate, count). **Reduce** sums the counts and outputs the itemsets with total ≥ s.

- **Advantages of MapReduce:** chunks are processed in parallel on many machines; data locality; scales to datasets far larger than memory; fault tolerance (failed chunks re-run); only two passes.
- **No false negatives** (an itemset frequent overall is frequent in at least one chunk).

CO6: Big Data Analytics Applications (Q106 to Q126)

Q106	What are dead ends in the context of link analysis on the web?	[2]

Dead ends are **pages with no out-links**. A random surfer reaching one cannot move on, so PageRank **leaks away** and all scores tend to 0. Fixes: teleportation (random jump), or removing dead ends recursively.

Q107	Explain the difference between content-based recommendations and collaborative filtering in recommendation engines?	[2]

	Content-based	Collaborative filtering
Uses	Features of items and the user's own profile	Ratings or behaviour of similar users / items
Example	You liked sci-fi, so more sci-fi	Users like you also liked this movie
Weakness	Limited variety	Cold start for new users and items

Q108	What is clustering in the context of social network graphs? Name a common algorithm used for clustering social network graphs.	[2]

Clustering means finding **communities**: groups of nodes that are densely connected to each other and sparsely connected to others. Common algorithm: **Girvan-Newman** (removes edges of high betweenness); also Louvain.

Q109	Define a dangling node in a web graph	[2]

A dangling node is a web page with **no outgoing links** (same as a dead end). PageRank is usually redistributed from it equally to all pages.

Q110 **List and briefly describe the common types of social network graphs?** **[2]**

- **Undirected graph:** mutual friendship (Facebook).
- **Directed graph:** follower relation (Twitter / X).
- **Weighted graph:** edge strength (number of messages).
- **Bipartite graph:** two kinds of nodes (users and items, authors and papers).
- **Collaboration / email / telephone networks**.

Q111 **Explain utility matrix with example** **[2]**

A **utility matrix** has users as rows and items as columns; each cell holds the user's rating of the item, and most cells are **blank (unknown)**. A recommender tries to predict the blanks.

	Matrix	Titanic	Avengers
Asha	5	?	2
Ravi	?	4	5
Meena	4	5	?

Q112 **Explain how PageRank is used in search engines to rank web pages. What does a higher PageRank indicate?** **[2]**

PageRank scores a page by the **number and quality of pages linking to it** (the chance that a random surfer is on it). The search engine combines it with relevance to the query to order results. **Higher PageRank = more important and trusted page**.

Q113 **Analyze the fundamental concept of the PageRank algorithm and evaluate its significance in web search and ranking of web pages.** **[5]**

- **Idea:** a link is a vote; votes from important pages count more. A random surfer follows links with probability β and jumps to a random page with probability $1 - \beta$.
- **Formula:** $PR(p) = (1 - \beta)/N + \beta \cdot \Sigma\, PR(q)/L(q)$, summed over pages q that link to p ($L(q)$ = out-links of q).
- **Computation:** repeat until values converge, or solve $r = \beta \cdot M \cdot r + (1 - \beta)/N$ (the principal eigenvector).
- **Significance:** a link-based importance score, hard to fake, the foundation of Google; teleportation solves dead ends and spider traps. **Limit:** link spam, static, topic-blind.

Q114 Explain structure of web? Explain spider trap in detail? [5]

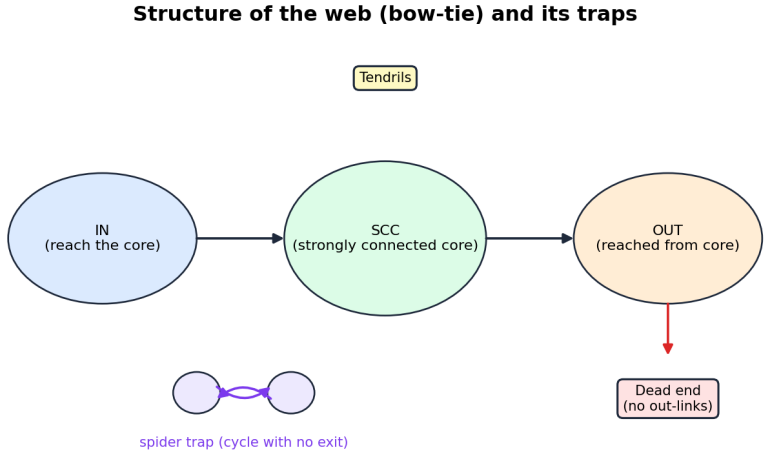


Figure: Bow-tie structure of the web, a dead end and a spider trap

- **Structure:** a **strongly connected core (SCC)**; **IN** pages that reach the core; **OUT** pages reachable from the core; **tendrils**; and disconnected pieces.
- **Spider trap:** a group of pages whose links all point only inside the group (for example, two pages linking to each other). Importance flows in and **never flows out**, so these pages absorb all PageRank and the rest of the web gets near 0.
- **Solution: taxation / teleportation**: with probability $1 - \beta$ the surfer jumps to a random page, so rank cannot be trapped.

Q115 Explain the fundamental characteristics that define social networks as graphs. What are nodes and edges in this context? [5]

- **Nodes (vertices)** are the entities: people, pages, organisations. **Edges (links)** are relationships: friendship, follow, message, collaboration.
- **Characteristics:** a collection of entities and relationships; edges can be directed, undirected or weighted; **community structure** (dense groups); **small-world** property (short paths between any two people); **power-law degree distribution** (a few hubs, many low-degree nodes); high clustering.

Q116 How does the process of direct discovery of communities work in the analysis of social network graphs? [5]

Direct discovery looks for **dense subgraphs** such as complete bipartite graphs $K_{s,t}$ (s people each connected to the same t others), without any prior grouping.

1. Treat each node as a **basket** whose items are its neighbours.
2. Run **frequent-itemset mining** (A-Priori / PCY) to find sets of t nodes (an itemset) that occur together in at least s baskets.
3. Each frequent itemset of size t with its s supporting nodes forms a community $K_{s,t}$.

Q117 What is the primary objective of recommendation systems, and why are they valuable for businesses and users? [5]

- **Objective:** predict which items a user will like and suggest them.
- **Businesses:** more sales, engagement and loyalty, cross-selling.
- **Users:** discover relevant items, save time, less information overload, personalised experience.

Q118 **Compare and contrast the direct discovery of communities in social network graphs with other community detection algorithms. What are the strengths and weaknesses of each approach?** **[5]**

Approach	Strengths	Weaknesses
Direct discovery (frequent itemsets, $K_{s,t}$)	Finds exact dense subgraphs, scalable with frequent-itemset tools, overlapping communities	Needs s and t, finds only complete bipartite patterns, misses loose communities
Girvan-Newman (betweenness)	Intuitive, gives a hierarchy	Very slow ($O(m^2n)$), poor for big graphs
Modularity / Louvain	Fast, good quality	Resolution limit, usually non-overlapping
Spectral clustering	Strong theory, good partitions	Needs eigenvectors (expensive), k required

Q119 **Describe the concept of topic-sensitive PageRank and how it differs from the standard PageRank algorithm.** **[5]**

Topic-sensitive PageRank changes the teleport step: instead of jumping to any page, the surfer jumps only to a **chosen set S of pages about a topic** (e.g. sports). $\mathbf{r = \beta \cdot M \cdot r + (1 - \beta) \cdot e_S / |S|}$.

	Standard PageRank	Topic-sensitive PageRank
Teleport to	Any page uniformly	Only pages in topic set S
Vectors	One for the whole web	One per topic (precomputed)
Result	Same ranking for every query	Ranking biased to the query / user topic

Q120 **Compute the page rank of each page in the following graph after 3 iterations (A → B, A → C, B → C, C → A).** **[10]**

Graph for Q120

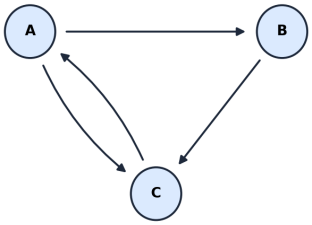


Figure: Graph for Q120

Use the basic PageRank (no damping, because none is given) with initial PR = 1/3 each. A has 2 out-links (B, C), B has 1 (C), C has 1 (A). Rules: PR(A) = PR(C); PR(B) = PR(A)/2; PR(C) = PR(A)/2 + PR(B).

Iteration	PR(A)	PR(B)	PR(C)
0	0.333	0.333	0.333
1	0.333	0.167	0.500
2	0.500	0.167	0.333
3	**0.333**	**0.250**	**0.417**

After 3 iterations: **PR(A) = 0.333, PR(B) = 0.25, PR(C) = 0.417** (the values keep settling towards A = 0.4, B = 0.2, C = 0.4). C is the most important page.

Q121 **Apply the principles of link spam detection. What are common strategies used by link spammers, and how can PageRank be used to identify them?** **[10]**

Link spam = artificially creating links to raise a page's PageRank. A common structure is a **spam farm**: a **target page**, many **supporting pages** that all link to it, and some links from **accessible** (genuine) pages such as blog comments.

- **Strategies:** spam farms, link exchanges, buying links, comment and forum spam, hijacking links, cloaking.
- **Detection using PageRank-based ideas: TrustRank** (a topic-sensitive PageRank that teleports only to a small set of trusted seed pages; spam pages get low trust), and **spam mass** (the fraction of a page's PageRank that comes from spam: PageRank minus TrustRank).
- Pages with high PageRank but very low TrustRank are flagged as spam; search engines then demote them.

Q122	**Apply different types of recommendation systems (content-based, collaborative filtering, hybrid) to recommend products in an online bookstore. Provide specific examples and discuss the expected benefits and limitations of each type in this context.**	**[10]**

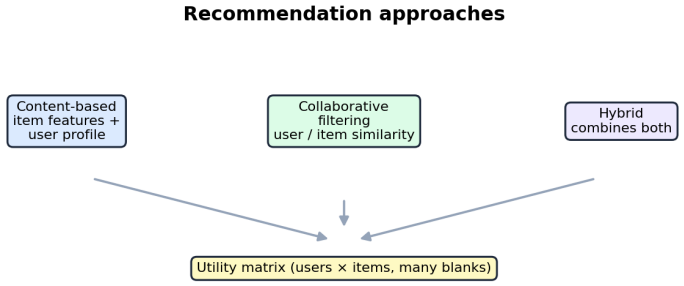
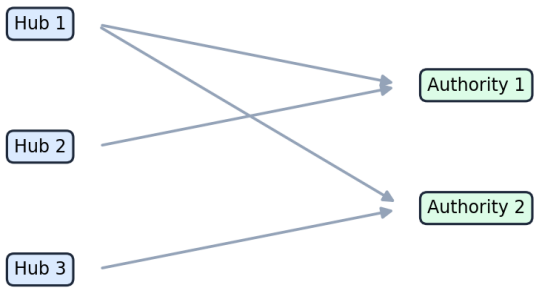


Figure: Types of recommendation systems

Type	Bookstore example	Benefits	Limitations
Content-based	A reader of fantasy novels by an author is shown similar genre / author books	Works for new books, no other users needed, explainable	Filter bubble, needs good book features, no surprise
Collaborative filtering	“Readers who bought this also bought ...”	Finds unexpected books, no item features needed	Cold start for new users and books, sparse data, popularity bias
Hybrid	Combines genre similarity with purchase patterns	Best accuracy, reduces cold start	More complex and costly

Q123 **Analyse the importance of hubs and authorities in link analysis. How do these concepts relate to the HITS (Hyperlink-Induced Topic Search) algorithm?** **[10]**

Hubs and authorities



A good hub points to good authorities; a good authority is pointed to by good hubs

Figure: Hubs point to authorities

- **Authority:** a page with valuable content on a topic, pointed to by many good hubs.
- **Hub:** a page that is a good directory, pointing to many good authorities.
- **Importance:** two scores per page capture both quality of content and quality of links, for topic-focused (query-time) ranking.
- **HITS:** start with hub = 1 for all; repeat authority $a = L^T \cdot h$ (sum of hub scores of pages linking in) and hub $h = L \cdot a$ (sum of authority scores of pages linked to); normalise after each step until stable.

Unlike PageRank (one global score), HITS is query-dependent and gives two scores; it is more sensitive to link spam.

Q124 **Compute hub and authority scores for the matrix using the HITS algorithm with k = 3 and identify the best authority and hub node. L = [[0,1,1,1],[0,0,1,1],[1,0,0,1],[0,0,0,1]]** **[10]**

Row i = page i; a 1 in column j means page i links to page j. Start with h = (1, 1, 1, 1). Each round: $a = L^T h$, normalise by the largest value; $h = La$, normalise by the largest value.

k	Authority (nodes 1 to 4)	Hub (nodes 1 to 4)
1	(0.25, 0.25, 0.50, 1.00)	(1.00, 0.857, 0.714, 0.571)
2	(0.227, 0.318, 0.591, 1.00)	(1.00, 0.833, 0.643, 0.524)
3	**(0.214, 0.333, 0.611, 1.00)**	**(1.00, 0.829, 0.624, 0.514)**

Best authority: node 4 (highest authority score, linked to by all four pages). **Best hub: node 1** (links to nodes 2, 3 and 4).

Q125 **Consider the Web graph with six pages (A to F) and links A → B, C; B → A, D, E, F; C → A, F. PR(m) at iteration 0 is 1 and β = 0.85. Perform the PageRank algorithm and determine the rank for every page at iteration 2.** **[10]**

D, E and F have no out-links listed (dead ends), so they only receive rank. I used $PR(p) = (1 - \beta) + \beta \times \Sigma PR(q)/L(q)$, the form that matches PR = 1 at iteration 0.

Out-links: A has 2 (B, C), B has 4 (A, D, E, F), C has 2 (A, F). In-links: A from B and C; B from A; C from A; D from B; E from B; F from B and C.

Page	Formula	Iteration 1	Iteration 2
A	0.15 + 0.85 (PR(B)/4 + PR(C)/2)	0.15 + 0.85(0.25 + 0.5) = 0.7875	0.15 + 0.85(0.575/4 + 0.575/2) = **0.5166**
B	0.15 + 0.85 (PR(A)/2)	0.15 + 0.85(0.5) = 0.575	0.15 + 0.85(0.7875/2) = **0.4847**
C	0.15 + 0.85 (PR(A)/2)	0.575	**0.4847**

Page	Formula	Iteration 1	Iteration 2
D	0.15 + 0.85 (PR(B)/4)	0.15 + 0.85(0.25) = 0.3625	0.15 + 0.85(0.575/4) = **0.2722**
E	0.15 + 0.85 (PR(B)/4)	0.3625	**0.2722**
F	0.15 + 0.85 (PR(B)/4 + PR(C)/2)	0.7875	**0.5166**

Ranks at iteration 2: A = F = 0.5166 (highest), B = C = 0.4847, D = E = 0.2722 (lowest).

Q126 **Apply the principles of collaborative filtering to recommend movies to a user based on their previous movie ratings and the ratings of other similar users.** **[10]**

Idea (user-based): find users whose ratings are similar to the target user, then predict the target's rating as a similarity-weighted average of their ratings. Example: predict Asha's rating for **Inception**.

User	Avengers	Matrix	Inception	Titanic	Notebook
Asha (target)	5	4	**?**	1	2
Ravi	5	5	4	1	1
Meena	1	2	2	5	5

- **Similarity** (cosine on co-rated movies): Asha-Ravi = 48 / (6.78 × 7.21) = **0.98**; Asha-Meena = 28 / (6.78 × 7.42) = **0.56**.
- **Prediction** = (0.98 × 4 + 0.56 × 2) / (0.98 + 0.56) = **3.3**. Using only the most similar user (k = 1) gives 4.
- Asha is much closer to Ravi, so **recommend Inception**.

Steps: build the utility matrix, compute user similarities (cosine or Pearson), choose the k nearest neighbours, predict, recommend the top-rated unseen movies. **Item-based** filtering does the same using item-item similarity. **Limits:** cold start, sparsity, scalability.