# Big Data Analytics - Important Questions and Answers

## 1. What is a NoSQL Database? Explain different features of NoSQL Database.

A NoSQL database is a non-relational database that stores and retrieves data in a way that is different from traditional RDBMS systems. It is designed to handle large volumes of structured, semi-structured, and unstructured data, and is highly scalable, flexible, and efficient for modern applications such as social media, IoT, and real-time analytics.

**Features of NoSQL Databases:**

• Schema-free – NoSQL databases do not require a fixed table structure. • Horizontal Scaling – They allow scaling by adding more servers instead of upgrading one machine. • High Performance – Optimized for high-speed data reads and writes. • Distributed Storage – Data is stored across multiple nodes or clusters. • Flexible Data Models – Supports different data formats like key-value, document, column-family, and graph.

## 2. Differentiate between RDBMS and NoSQL Database.

| RDBMS | NoSQL | | --- | --- | | Data is stored in tables (rows and columns). | Data is stored in collections or key-value pairs. | | Follows ACID properties. | Follows BASE properties. | | Uses SQL for queries. | Uses various query languages depending on type. | | Fixed schema. | Schema-free and flexible. | | Scaling is vertical (by adding resources to one system). | Scaling is horizontal (by adding more machines). | | Best for structured data. | Best for unstructured or semi-structured data. |

## 3. NoSQL Business Drivers

Traditional RDBMS systems cannot handle modern business data challenges. NoSQL databases solve problems related to Volume, Velocity, Variability, and Agility.

**1. Volume:** Handles large datasets using clusters of low-cost computers. Enables horizontal scaling and parallel processing.

**2. Velocity:** Processes data in real time for fast applications like social media and e-commerce.

**3. Variability:** Supports diverse data formats (structured, semi-structured, unstructured) without requiring schema changes.

**4. Agility:** Allows quick adaptation to new requirements with flexible design and easy scaling.

## 4. Explain CAP Theorem in Detail.

The CAP Theorem states that in a distributed database, only two of the following three properties can be guaranteed at the same time: Consistency, Availability, and Partition Tolerance.

**1. Consistency:** Every node in the system has the same data at the same time. **2. Availability:** Every request gets a response, even if some nodes fail. **3. Partition Tolerance:** The system continues working even if communication between nodes is broken.

Example: - Cassandra focuses on Availability and Partition Tolerance. - MongoDB balances all three based on configuration.

## 5. Different Architectural Patterns in NoSQL

NoSQL databases can follow different data storage patterns like Key-Value Store, Document Store, Column Family Store, and Graph Store.

**Graph Data Store:**

Stores data in nodes and relationships. Ideal for social networks, recommendation systems, etc. Example: Neo4j database, where each user or entity is a node connected by relationships.

**Column Family Store:**

Stores data in columns instead of rows. Great for analytical applications. Example: Apache Cassandra or HBase.

## 6. Explain Vector and List Object in R.

**Vector:** A vector is a sequence of data elements of the same type. Example: x <- c(1, 2, 3, 4). **List:** A list is a collection of different types of elements. Example: mylist <- list(1, "Apple", TRUE).

## 7. Variables in R (with examples)

A variable in R is used to store data values. The assignment operator <- or = is used to assign values. Example: x <- 10 name <- "R Programming" LogicalVar <- TRUE

## 8. Types of Operators in R

1. Arithmetic Operators (+, -, *, /, ^) 2. Relational Operators (==, !=, >, <, >=, <=) 3. Logical Operators (&, |, !) 4. Assignment Operators (<-, =) 5. Misc Operators (%in%, :, etc.)

## 9. R Script to Sort Values

Vector: v <- c(23, 45, 10, 34, 89, 20, 67, 99) Ascending order: sort(v) Descending order: sort(v, decreasing = TRUE) Output: Ascending → 10 20 23 34 45 67 89 99 Descending → 99 89 67 45 34 23 20 10

## 10. DGIM Algorithm with Example

The DGIM algorithm is used for counting 1’s in a data stream over a sliding window efficiently. It uses buckets to summarize bits, reducing memory usage. Example: For stream 1 0 1 1 0 1, DGIM maintains bucket timestamps to count recent 1’s without storing all data.

## 11. Data Stream Management System (DSMS)

A DSMS processes continuous, high-speed, and real-time data streams. It provides results instantly instead of storing data first like DBMS.

**Key Components:** 1. Streams Entering – Continuous flow of data (e.g., sensor readings). 2. Stream Processor – Core part that processes incoming data using standing queries. 3. Output Streams – Generates live results for dashboards or alerts. 4. Ad-hoc Queries – On-demand queries for instant insights. 5. Limited Working Storage – Uses sliding windows for recent data. 6. Archival Storage – Stores old data for future analysis.

**Applications:** Real-time monitoring, stock trading, fraud detection, IoT data analytics.

**Diagram (Textual Representation):** Streams → Stream Processor → Output Stream / Archival Storage ← Ad-hoc Queries