Q1.Distinguish between Name node and Data node?
→

Feature	NameNode	DataNode
Role	Manages the metadata of the HDFS (directory tree, file names, permissions, etc.).	Stores the actual data blocks in HDFS.
Type	Master node	Slave node
Function	Keeps track of where data is stored across DataNodes.	Stores and retrieves blocks of data when requested by clients or NameNode.
Data Handling	Does not store actual data, only metadata (file system namespace and block locations).	Stores the actual blocks of data files.
Failure Handling	If the NameNode fails, the HDFS becomes inaccessible (Single point of failure, mitigated by Secondary NameNode or High Availability setup).	If a DataNode fails, data can still be accessed from replicas stored in other DataNodes.
Communication	Communicates with DataNodes to keep track of the data blocks. Also interacts with clients to provide metadata information.	Communicates with the NameNode and clients to read/write blocks of data. Periodically sends a heartbeat to the NameNode.
Memory Requirements	Requires more memory as it stores the entire metadata in RAM for quick access.	Requires less memory as it deals only with data storage and periodic communication.
Number in Cluster	Typically one (or two in a high-availability setup).	Many DataNodes per cluster for distributed data storage.
Criticality	High – Its failure affects the entire cluster.	Lower – Failure of individual DataNodes does not impact the overall cluster due to replication.
Data Integrity	Maintains file system namespace and ensures replication policies are followed.	Ensures data blocks are replicated as per the NameNode’s instructions.

Q2. Mention four characteristics of big data. Elaborate these characteristics with respect to social media websites?
→
The four key characteristics of big data, often referred to as the **4 Vs**, are **Volume, Velocity, Variety, and Veracity**. Let's elaborate on these characteristics in the context of social media websites:

1. Volume:

- **Definition:** Refers to the sheer size of data generated, collected, and stored.
- **In Social Media:** Social media platforms generate enormous amounts of data daily. Platforms like Facebook, Instagram, Twitter, and TikTok handle billions of posts, messages,

images, and videos shared by users worldwide. This volume is continuously growing, requiring scalable storage and processing solutions.

2. Velocity:

- **Definition:** Refers to the speed at which data is generated, processed, and analyzed.
- **In Social Media:** The data flow on social media is extremely rapid. Posts, likes, comments, shares, and reactions happen in real time. Social media platforms must handle these high-velocity interactions to ensure a smooth user experience, deliver real-time recommendations, and monitor trends or viral content.

3. Variety:

- **Definition:** Refers to the different types of data formats being generated.
- **In Social Media:** Social media platforms deal with diverse data types, including structured data (user profiles), unstructured data (images, videos, text posts), and semi-structured data (hashtags, emojis, and metadata). Handling and analyzing these various formats is a challenge, but it provides rich insights into user behavior and trends.

4. Veracity:

- **Definition:** Refers to the quality, accuracy, and trustworthiness of the data.
- **In Social Media:** Social media data often comes with challenges of trust and accuracy. User-generated content may include spam, fake accounts, or misinformation. Platforms must constantly verify the authenticity of data, ensure content is reliable, and filter out noise, like bots or irrelevant posts, to extract meaningful insights.

These characteristics illustrate the complexity of managing big data on social media platforms and emphasize the importance of effective data processing and analytics strategies.

Q3. Explain how node failure is handle in Hadoop?
→
Hadoop handles node failures through the following mechanisms:

1. **DataNode Failure:**
    - Data is replicated across multiple DataNodes (default 3 replicas). If a DataNode fails, the NameNode detects the failure and re-replicates the data to maintain the required number of copies.
    - Clients can read data from other replicas on different nodes.
2. **NameNode Failure:**
    - In **High Availability (HA)** setups, two NameNodes exist: Active and Standby. If the Active NameNode fails, the Standby automatically takes over, ensuring uninterrupted operation.
    - Metadata is backed up using **FsImage** and **EditLogs**.
3. **Task Failure:**
    - Failed tasks are rescheduled on different nodes by the JobTracker (or ResourceManager in YARN) to ensure job completion without disruption.

4. **Client Failure:**
    - If a client fails, jobs continue to run on the cluster, and the client can reconnect later for results.

Hadoop’s fault tolerance ensures continued operation despite node failures through replication and task rescheduling.

Q4. What are the 3 V’s of Big Data? Give two Big data case studies indicating respective V’s with justification.
→

The **3 V's of Big Data** are:

1. Volume:

- **Definition:** Refers to the sheer size of data generated, collected, and stored.
- **In Social Media:** Social media platforms generate enormous amounts of data daily. Platforms like Facebook, Instagram, Twitter, and TikTok handle billions of posts, messages, images, and videos shared by users worldwide. This volume is continuously growing, requiring scalable storage and processing solutions.

2. Velocity:

- **Definition:** Refers to the speed at which data is generated, processed, and analyzed.
- **In Social Media:** The data flow on social media is extremely rapid. Posts, likes, comments, shares, and reactions happen in real time. Social media platforms must handle these high-velocity interactions to ensure a smooth user experience, deliver real-time recommendations, and monitor trends or viral content.

3. Variety:

- **Definition:** Refers to the different types of data formats being generated.
- **In Social Media:** Social media platforms deal with diverse data types, including structured data (user profiles), unstructured data (images, videos, text posts), and semi-structured data (hashtags, emojis, and metadata). Handling and analyzing these various formats is a challenge, but it provides rich insights into user behavior and trends.

Case Study 1: Netflix – Personalized Recommendations

- **Volume:** Netflix processes petabytes of data daily from millions of users worldwide, including watching habits, search history, and user preferences.
- **Velocity:** Data from users' interactions is collected and processed in real time to deliver immediate recommendations.
- **Justification:** Netflix uses this data to generate personalized recommendations, constantly refining its algorithms based on new data.

Case Study 2: Twitter – Real-time Sentiment Analysis

- **Variety:** Twitter deals with various types of data, such as text (tweets), images, videos, and hashtags.
- **Velocity:** Tweets are generated and shared at an incredibly fast rate (thousands of tweets per second), requiring real-time analysis for tasks like sentiment tracking and trend detection.
- **Justification:** Twitter processes diverse, rapidly generated data to analyze user sentiment, track trends, and respond to viral content in real time.

Q5. Explain map reduce execution pipeline.

→

The **MapReduce execution pipeline** consists of several stages that take raw input data, process it, and generate the desired output in a distributed manner. Here's a breakdown of the key steps in the pipeline:

1. Input Splitting:

- The input data is divided into smaller chunks called **splits**. Each split is processed independently by a map task.
- The split size is typically the size of an HDFS block (64 MB or 128 MB).

2. Mapping:

- Each split is passed to a **Map** function. The Map function processes the data and generates key-value pairs as intermediate output.
- The number of Map tasks is determined by the number of splits. Each task runs in parallel on different nodes.

3. Shuffling and Sorting:

- The output of the Map tasks is **shuffled** to ensure that all values associated with the same key are sent to the same **Reduce** task.
- During this phase, the intermediate key-value pairs are also **sorted** by key to prepare for reduction.
- The shuffling and sorting happen in parallel across the nodes running Map tasks.

4. Partitioning:

- The framework partitions the intermediate data by key to distribute it to different Reduce tasks.
- The partitioner function ensures that the same keys go to the same reducer.

5. Reducing:

- The sorted and shuffled key-value pairs are passed to the **Reduce** function. The Reduce function processes each key and aggregates or summarizes the associated values (e.g., counting, summing).

- Each Reduce task runs in parallel across nodes and processes a portion of the intermediate data.

6. Final Output:

- The output of the Reduce tasks is written to HDFS in the form of key-value pairs.
- The final output is stored in multiple files, with each file representing the output of a single Reduce task.

Q6. Distinguish between traditional and big data approach.

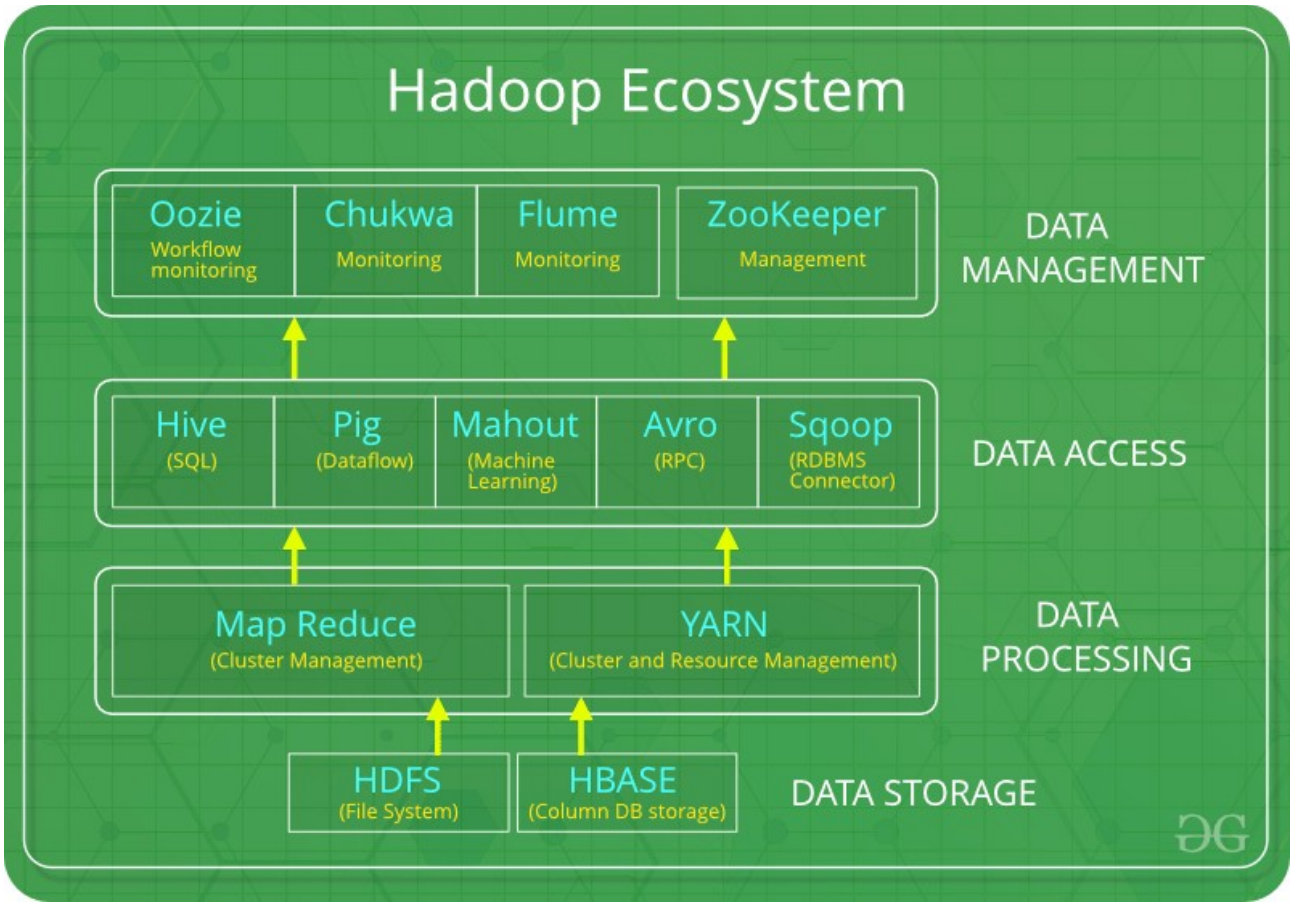
→

Traditional Data	Big Data
Traditional data is generated in enterprise level.	Big data is generated outside the enterprise level.
Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
Traditional database system deals with **structured data**.	Big data system deals with structured, semi-structured,**database**, and **unstructured data**.
Traditional data is generated per hour or per day or more.	But big data is generated more frequently mainly per seconds.
Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.
Data integration is very easy.	Data integration is very difficult.
Normal system configuration is capable to process traditional data.	High system configuration is required to process big data.
The size of the data is very small.	The size is more than the traditional data size.
Traditional data base tools are required to perform any data base operation.	Special kind of data base tools are required to perform any **database schema** based operation.
Normal functions can manipulate data.	Special kind of functions can manipulate data.
Its data model is strict schema based and it is static.	Its data model is a flat schema based and it is dynamic.
Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.
Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.

Traditional Data	Big Data
It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
Its data sources includes ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

Q7. Draw Hadoop ecosystem and explain any 5 components of Hadoop ecosystem.

→



Explanation of 5 Components of the Hadoop Ecosystem

1. **Hadoop Distributed File System (HDFS):**
    - **Function:** HDFS is the storage layer of Hadoop, designed to store vast amounts of data across multiple machines.
    - **Characteristics:** It is highly fault-tolerant and designed to run on commodity hardware. HDFS splits large files into blocks (typically 128 MB) and distributes them across various nodes in the cluster, ensuring data redundancy through replication (usually three copies).
2. **Yet Another Resource Negotiator (YARN):**
    - **Function:** YARN is the resource management layer of Hadoop that manages and schedules resources across the cluster.

- **Characteristics:** It allows multiple data processing engines to handle data stored in a single platform, effectively managing resources and job scheduling. YARN separates resource management from data processing, allowing different applications to run concurrently.

3. **MapReduce:**
    - **Function:** MapReduce is a programming model for processing large datasets in a distributed manner.
    - **Characteristics:** It consists of two main functions: the **Map** function, which processes and transforms data into key-value pairs, and the **Reduce** function, which aggregates the results based on keys. This model enables efficient processing of large-scale data.
4. **HBase:**
    - **Function:** HBase is a distributed, NoSQL database built on top of HDFS that provides real-time read/write access to large datasets.
    - **Characteristics:** It is designed for random, real-time access to big data and is particularly useful for applications requiring low-latency data access. HBase uses a sparse, column-oriented storage model, making it suitable for storing large amounts of structured data.
5. **Hive:**
    - **Function:** Hive is a data warehousing and SQL-like query language interface for Hadoop.
    - **Characteristics:** It allows users to write queries using a SQL-like language (HiveQL) to analyze and manage large datasets stored in HDFS. Hive abstracts the complexity of MapReduce, enabling users to work with data without needing to write low-level MapReduce code.

Q8. What are combiners, and how do they contribute to the optimization of MapReduce jobs?
→
Combiners in the MapReduce programming model are optional components that act as mini-reducers, which run on the output of the map tasks before the data is sent to the reducers. They help optimize the overall performance and efficiency of MapReduce jobs by reducing the amount of intermediate data that needs to be shuffled across the network. Here’s how combiners contribute to optimization:

How Combiners Work:

1. **Intermediate Data Reduction:**
    - After the map tasks process their input data, they produce key-value pairs as intermediate output. A combiner can aggregate these pairs on the mapper's local node before sending them to the reducer.
    - For example, in a word count job, if the mapper outputs pairs like (word1, 2), (word2, 3), and (word1, 1), the combiner can combine the pairs for word1 into (word1, 3) and output (word1, 3) and (word2, 3) instead of the original pairs.
2. **Local Execution:**

- Combiners are executed locally on the output of the map tasks. This reduces the amount of data that needs to be transferred over the network to the reducers, resulting in lower network I/O and faster processing.

3. **Optimization of Network Traffic:**
    - By aggregating intermediate results, combiners help minimize the amount of data sent across the network. This is especially beneficial in scenarios where the volume of data is large, as it reduces the time and resources required for data shuffling.

Benefits of Using Combiners:

- **Reduced Data Volume:** They significantly decrease the size of intermediate data, resulting in faster shuffle and sort phases.
- **Lower Network Load:** By sending less data across the network, they help alleviate network congestion, which can be a bottleneck in distributed systems.
- **Faster Execution Times:** Overall job execution time can be reduced as the amount of data processed in the reducer is less, leading to quicker aggregation.

Q9. Explain types of Big Data.
→
Big Data can be categorized into several types based on its characteristics, sources, and formats. The primary types of Big Data include:

1. Structured Data:

- **Definition:** This type of data is highly organized and easily searchable, typically stored in relational databases or spreadsheets.
- **Characteristics:** Structured data follows a predefined schema, making it straightforward to enter, store, query, and analyze.
- **Examples:** Customer information (names, addresses), transaction records, and inventory data.

2. Unstructured Data:

- **Definition:** Unstructured data lacks a specific format or structure, making it more challenging to analyze and process.
- **Characteristics:** This type of data does not fit into traditional data models and often requires advanced analytical techniques to derive insights.
- **Examples:** Text documents, emails, social media posts, images, videos, and audio files.

3. Semi-Structured Data:

- **Definition:** Semi-structured data falls between structured and unstructured data. It contains tags or markers to separate data elements, but does not have a rigid structure.
- **Characteristics:** While it may not fit neatly into tables, it allows for some level of organization that can be used for processing and analysis.
- **Examples:** JSON files, XML documents, and log files.

4. Time-Series Data:

- **Definition:** Time-series data is a sequence of data points collected or recorded at specific time intervals.
- **Characteristics:** It is used to track changes over time and is often analyzed to identify trends, seasonal patterns, or anomalies.
- **Examples:** Stock prices, temperature readings, server logs, and web traffic data.

5. Geospatial Data:

- **Definition:** Geospatial data relates to geographical locations and is used to represent data associated with specific locations on Earth.
- **Characteristics:** This type of data can be structured or unstructured and often involves mapping and spatial analysis.
- **Examples:** Geographic Information System (GIS) data, GPS data, and satellite imagery.

6. Streaming Data:

- **Definition:** Streaming data refers to continuous flows of data generated in real-time from various sources.
- **Characteristics:** This data type is processed in real-time or near-real-time, enabling immediate insights and responses.
- **Examples:** Social media feeds, sensor data, IoT device data, and online transaction data.

7. Multimedia Data:

- **Definition:** Multimedia data encompasses various formats of media such as images, videos, and audio files.
- **Characteristics:** It requires specialized tools for storage, processing, and analysis, often involving techniques like image processing and video analytics.
- **Examples:** Videos uploaded to streaming services, audio recordings, and image collections.

Q10. Explain Hadoop architectural model.

→

The **Hadoop architectural model** is designed to handle large-scale data processing in a distributed environment. It consists of several key components that work together to provide a framework for storing, processing, and managing big data efficiently. The architecture primarily consists of two layers: the **Hadoop Storage Layer** and the **Hadoop Processing Layer**. Below is an overview of the architectural model:

1. Hadoop Storage Layer (HDFS)

- **Hadoop Distributed File System (HDFS):**
    - HDFS is the core storage component of the Hadoop architecture. It stores large files across multiple nodes in a cluster.
    - **Key Characteristics:**
        - **Distributed Storage:** Data is split into blocks (typically 128 MB) and stored across multiple DataNodes.

        - **Replication:** Each data block is replicated (default is three copies) to ensure fault tolerance and data availability.
        - **High Throughput:** Optimized for high-throughput data access rather than low-latency access.

2. Hadoop Processing Layer (MapReduce)

- **MapReduce:**
    - MapReduce is the processing model that allows for distributed data processing across a Hadoop cluster.
    - **Key Characteristics:**
        - **Map Function:** Processes input data and generates intermediate key-value pairs.
        - **Reduce Function:** Aggregates intermediate data based on keys to produce final output.
        - **Parallel Execution:** Tasks are distributed across different nodes, allowing for efficient processing of large datasets.

3. Hadoop Common

- **Common Utilities:**
    - Hadoop Common includes the libraries and utilities needed by other Hadoop modules. It provides the necessary interfaces and shared resources across the ecosystem.

4. YARN (Yet Another Resource Negotiator)

- **Resource Management:**
    - YARN is responsible for managing resources and scheduling tasks in a Hadoop cluster.
    - **Key Components:**
        - **ResourceManager:** The master daemon that manages resources and allocates them to various applications.
        - **NodeManager:** A per-node daemon responsible for managing resources and running containers for applications.
        - **ApplicationMaster:** Manages the lifecycle of individual applications, handling job scheduling and resource negotiation.

5. Hadoop Ecosystem Components

In addition to the core components, the Hadoop ecosystem includes various tools and frameworks that enhance its capabilities:

- **Hive:** A data warehousing solution that provides a SQL-like interface for querying data stored in HDFS.
- **HBase:** A NoSQL database that runs on top of HDFS, allowing real-time read/write access to large datasets.
- **Pig:** A high-level scripting language for processing and analyzing large datasets in Hadoop.

- **Spark:** An in-memory data processing framework that can run on top of Hadoop for faster data processing.
- **Flume:** A service for collecting, aggregating, and moving large amounts of log data into HDFS.
- **Sqoop:** A tool for transferring data between Hadoop and relational databases.

Q11. Describe Matrix multiplication using MapReduce.
→
Q12. Describe Relational algebra using MapReduce
→

Q13. What is NoSQL, and how does it differ from traditional SQL databases?
→
NoSQL (Not Only SQL) refers to a category of database management systems that are designed to provide flexible data models, scalability, and high performance for handling large volumes of unstructured and semi-structured data. Unlike traditional relational SQL databases, which use structured query language (SQL) and a fixed schema, NoSQL databases allow for a more flexible approach to data storage and retrieval.

Feature	SQL Databases	NoSQL Databases
Data Model	Relational (tables with fixed schemas)	Various (document, key-value, column-family, graph)
Schema	Fixed schema (requires predefined structure)	Flexible schema (allows dynamic structure)
Query Language	SQL (structured query language)	Varies (JSON, MongoDB Query Language, etc.)
Scalability	Vertical scaling (increased power of a single server)	Horizontal scaling (adding more servers)
Transactions	ACID-compliant (Atomicity, Consistency, Isolation, Durability)	BASE (Basically Available, Soft state, Eventually consistent)
Data Relationships	Supports complex joins and relationships	Limited support for joins; optimized for denormalized data
Data Storage	Stores structured data in rows and columns	Stores unstructured or semi-structured data in various formats
Performance	May slow down with large datasets or high loads	Optimized for high-speed data access and large-scale data processing
Use Cases	Ideal for applications requiring complex queries and transactions (e.g., ↓king systems)	Suitable for big data applications, real-time analytics, content management, and social media platforms

Q14. Describe various NoSQL Business Drivers

→
NoSQL databases have gained popularity across various industries due to their ability to address specific business needs and challenges that traditional relational databases may struggle with. Here are some key **business drivers** for adopting NoSQL solutions:

1. Scalability

- **Description:** NoSQL databases are designed to scale horizontally, meaning organizations can add more servers to handle increased loads rather than upgrading existing hardware.
- **Business Impact:** This scalability is essential for businesses that experience rapid growth or fluctuating data demands, such as e-commerce platforms or social media applications.

2. Flexibility and Agility

- **Description:** NoSQL databases allow for dynamic schemas, enabling organizations to quickly adapt to changing business requirements without the need for complex migrations or schema changes.
- **Business Impact:** This flexibility is beneficial for startups and enterprises that need to iterate quickly on product features or accommodate new data types, leading to faster time-to-market.

3. High Availability and Fault Tolerance

- **Description:** Many NoSQL databases are designed with built-in replication and distribution mechanisms that ensure data availability even in the event of server failures.
- **Business Impact:** Businesses relying on continuous uptime (e.g., online services, financial transactions) can maintain operations without significant downtime, improving customer satisfaction and trust.

4. Performance Optimization

- **Description:** NoSQL databases are optimized for high-speed read and write operations, making them suitable for real-time analytics and fast data access.
- **Business Impact:** Organizations can leverage this performance for use cases like real-time monitoring, fraud detection, or recommendation systems, enhancing user experiences and operational efficiency.

5. Handling Big Data

- **Description:** NoSQL databases are designed to handle vast volumes of unstructured or semi-structured data, accommodating various data formats and types.
- **Business Impact:** Companies dealing with big data (e.g., IoT devices, social media interactions, sensor data) can efficiently store, process, and analyze this information to gain valuable insights.

6. Cost Efficiency

- **Description:** NoSQL databases often run on commodity hardware and open-source platforms, reducing the overall cost of ownership compared to traditional relational databases.

- **Business Impact:** This cost-effectiveness makes it feasible for small and medium enterprises to leverage advanced data storage solutions without significant capital investment.

7. Support for Diverse Data Types

- **Description:** NoSQL databases can accommodate various data types, including documents, key-value pairs, graphs, and more.
- **Business Impact:** Organizations can manage diverse data sources and formats (e.g., images, logs, social media feeds) in a unified manner, enabling comprehensive analytics and insights.

8. Improved Analytics and Insights

- **Description:** The ability to process and analyze large amounts of varied data helps organizations uncover trends, patterns, and insights that drive strategic decisions.
- **Business Impact:** Enhanced analytics capabilities enable businesses to make data-driven decisions, improve customer targeting, and optimize operations.

9. Rapid Development and Deployment

- **Description:** The flexibility and agility of NoSQL databases facilitate faster development cycles, allowing teams to deploy applications quickly.
- **Business Impact:** This rapid deployment is crucial for organizations aiming to stay competitive in fast-paced markets by quickly launching new features or services.