IAT 1 SOLUTION: 7TH SEM: 2022-26 BATCH ,

Scheme 2022

Sub:	BDA					Sub Code:	BCS714D	Branch:	CSE
Date:	29 SEPT	Duration:	90 mins	Max Marks:	50	Sem / Sec:	A,B,C		

Q1 a)	List and explain types of data. Each type-1+1+1 example - 2 In Big Data Analytics (BDA), there are two main ways to categorize data: by its structure (structured, semi-structured, and unstructured) and by the type of analysis performed (descriptive, diagnostic, predictive, and prescriptive). Structured data is organized in a predefined format, like a database, while unstructured data has no specific format, such as text messages or videos. Semi-structured data falls in between, using tags to create a hierarchy, like XML or JSON. The analytics types focus on different goals: descriptive summarizes past data, diagnostic investigates the causes, predictive forecasts the future, and prescriptive recommends actions. Types of data by structure	5

	● **Structured data:** Data that is highly organized and fits into a fixed schema with a predefined structure, like rows and columns. ● **Examples:** Relational databases, spreadsheets, or customer transaction records. ● **Use:** Easy to store, query, and analyze quantitatively. ● **Semi-structured data:** Data that is not stored in a traditional database but has some organizational properties, like tags or markers. ● **Examples:** JSON, XML files, emails, or web pages. ● ● **Use:** Offers more flexibility than structured data while still having some level of organization. ● **Unstructured data:** Data that has no predefined format or organization. ● **Examples:** Text documents, social media posts, audio files, and videos. ● **Use:** Can be rich with information, but it is challenging to organize and analyze without advanced tools.	

1B)	**Define Bigdata. List and explain characteristics of data.** Define big data , characteristics- big data definition - 2 marks, characteristics- 3 marks Big Data refers to extremely large and complex datasets that are difficult to store, process, and analyze with traditional methods. Its key characteristics are often described by the "5 Vs": Volume (massive amount of data), Velocity (high speed at which data is generated and processed), Variety (multiple data formats), Veracity (accuracy and reliability of data), and Value (the usefulness of the data). Characteristics of Big Data ● **Volume:** This refers to the sheer amount of data being generated. Big data involves massive datasets that can be measured in terabytes or petabytes, far exceeding the capacity of traditional databases. ● **Velocity:** This is the speed at which new data is generated and needs to be processed. Data streams in continuously from sources like social media, sensors, and mobile devices, requiring real-time or near-real-time analysis. ● **Variety:** This describes the different types of data that make up the dataset. Big data includes structured data (like in a database), unstructured data (like text, images, and video), and semi-structured data (like JSON or XML files).	5

	● **Veracity:** This characteristic addresses the accuracy, quality, and reliability of the data. Due to the numerous sources and high volume, big data can contain noise, errors, and biases, making it crucial to have processes to ensure data quality. ● **Value:** This refers to the ability to convert the raw data into valuable insights. The goal of big data is to extract meaningful information to make better business decisions, improve operations, and drive innovation.	
2A	**What is Shared Nothing Architecture? Explain its advantages.** Shared nothing architecture- 3marks advantages- 2 marks **Shared nothing architecture is a distributed computing model where each node has its own processor, memory, and storage, and nodes communicate only over a network. This model's main advantage is its high scalability, as more nodes can be added to handle increased loads without affecting others. It also offers high availability and fault tolerance, as the failure of one node does not bring down the entire system.** ● **Shared Disk Architecture: In contrast to shared nothing, shared disk architecture uses multiple nodes that share a common storage area. This can lead to performance bottlenecks as nodes contend for access to the shared storage.** ● **Shared Memory Architecture: Shared memory architecture is another distributed system model**	5

	where nodes can access the same memory. This is not an effective approach for large-scale distributed systems due to memory contention issues.	
2B	**Scenario: Online Food Delivery App (like Swiggy/Zomato)** 2(b) each is of 1 marks and 1 mark is for detailed explanation **Read the below Scenarios and Identify type of analytics.** **1. The company looks at last month's data: "We delivered 50,000 orders, and 15% were delayed." It summarizes trends and KPIs like average delivery time, number of orders per city, and customer ratings.** **2. The team investigates: "Why were 15% of deliveries delayed?", They find that delays mostly occurred during heavy rain and traffic congestion in metro cities.** **3. Using machine learning models, they predict: "Next month, during the festival season, we expect a 30% spike in orders and delivery delays may increase by 10%."** **4. The system recommends: "Add 200 temporary delivery partners during festivals, reroute deliveries using live traffic data, and offer incentives for faster deliveries. "It not only predicts the problem but also suggests the best possible actions.** **Type of Analytics: Descriptive Analytics** **Explanation:**	5

	● The scenario involves **looking at past data** (e.g., last month’s performance). ● It focuses on **summarizing trends**, **key performance indicators (KPIs)** such as delivery counts, delays, and ratings. ● **Descriptive analytics** answers the question **“What happened?”** — it helps understand **historical performance** using reports, summaries, and dashboards. The team investigates: “Why were 15% of deliveries delayed?”, They find that delays mostly occurred during heavy rain and traffic congestion in metro cities. **Type of Analytics: Diagnostic Analytics** **Explanation:** ● This scenario focuses on **understanding the reasons behind an outcome** — in this case, **why 15% of deliveries were delayed**. ● The team identifies **causal factors** such as **heavy rain** and **traffic congestion**.	

- **Diagnostic analytics** answers the question **“Why did it happen?”**, using techniques like root cause analysis, correlation analysis, and drill-down of data.

Using machine learning models, they predict: “Next month, during the festival season, we expect a 30% spike in orders and delivery delays may increase by 10%.”

Type of Analytics: Predictive Analytics

Explanation:

- This scenario uses **machine learning models** to **forecast future trends** — predicting a **30% increase in orders** and a **10% rise in delays**.

- **Predictive analytics** answers the question **“What is likely to happen?”**

- It relies on **historical data, statistical models, and machine learning algorithms** to make **data-driven predictions** about future events.

The system recommends: “Add 200 temporary delivery partners during festivals, reroute

deliveries using live traffic data, and offer incentives for faster deliveries. “It not only

predicts the problem but also suggests the best possible actions.

Type of Analytics: Prescriptive Analytics

Explanation:

	● This scenario goes **beyond prediction** — it **recommends specific actions** (e.g., adding delivery partners, rerouting, offering incentives). ● **Prescriptive analytics** answers the question **“What should we do?”** ● It uses **optimization models, simulations, and AI-driven recommendations** to suggest the **best course of action** for achieving desired outcomes.	
3 (a)	**List and Explain Different types of NoSQL Databases with example.** Types - 4 marks, example -2 marks The four main types of NoSQL databases are: key-value stores, which store data as simple key-value pairs; document databases, which store data in document-like structures such as JSON or BSON; wide-column stores (also called column-oriented databases), which store data in columns rather than rows for efficient querying of specific column data; and graph databases, which use nodes and edges to represent and store data with complex relationships.	6
3B	**(b) List and Explain difference between ACID property and CAP theorem.**	4

ACID Properties

- **Atomicity: Transactions are all-or-nothing. Either all operations within a transaction succeed, or none of them do.**
- **Consistency: A transaction brings the database from one valid state to another. It ensures that the database's integrity constraints are not violated.**
- **Isolation: Concurrent transactions do not interfere with each other. The execution of one transaction should not be visible to other transactions until it is committed.**
- **Durability: Once a transaction is committed, the changes are permanent and will survive system failures.**

CAP Theorem

- **Consistency: All nodes in the distributed system have the same data at the same time. If a read is performed, it returns the most recent write.**
- **Availability: Every request receives a response, without guarantee that it is the most recent write.**
- **Partition Tolerance: The system continues to operate even if there is a network failure (a "partition") between nodes.**

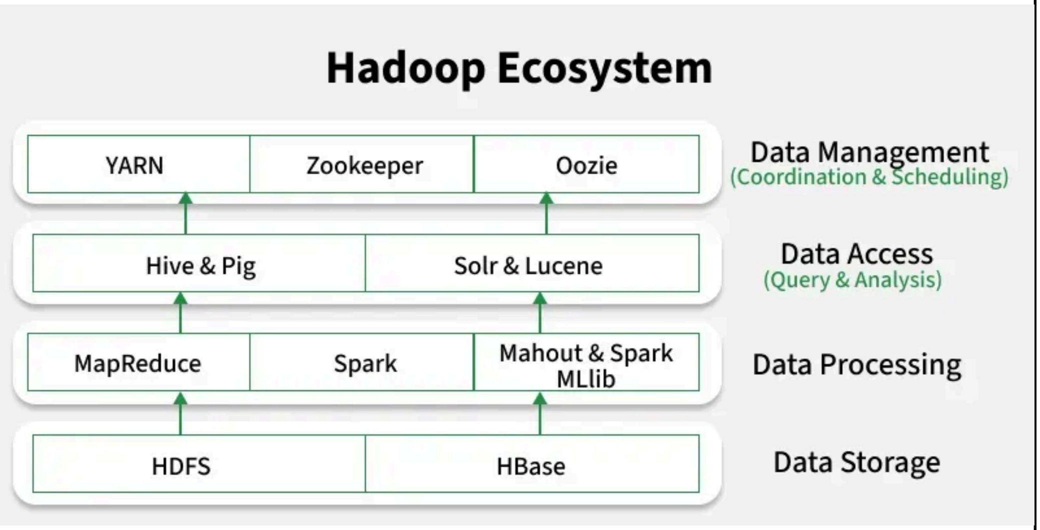
Key Differences

Feature	ACID Properties	CAP Theorem
Scope	Focuses on the internal integrity of a single database transaction.	Focuses on the trade-offs in a distributed system that experiences network partitions.
Consistency	ACID Consistency: Ensures the database follows its rules, moving from one valid state to another.	CAP Consistency: Ensures all nodes have the same data. In a partition, a choice must be made between this and availability.
Purpose	Guarantees reliability and data integrity for individual transactions.	Describes fundamental limitations and trade-offs in distributed systems.
Example	A bank transfer, where both the debit and credit must succeed or fail together.	A social media feed that might show slightly stale data to ensure users can still see posts

during a network outage.

4 (a) With Diagram explain Hadoop ecosystem.

4(a) Hadoop ecosystem - diagram 3 marks explanation 3 marks



Hadoop is an open-source framework for storing and processing large-scale data across distributed clusters using commodity hardware. The Hadoop Ecosystem is a suite of tools and technologies built around Hadoop's core components (HDFS, YARN, MapReduce and Hadoop Common) to enhance its capabilities in data storage, processing, analysis and management.

Components of Hadoop Ecosystem

Hadoop Ecosystem comprises several components that work together for efficient big data storage and processing:

	• HDFS (Hadoop Distributed File System): Stores large datasets across distributed nodes. • YARN (Yet Another Resource Negotiator): Manages cluster resources and job scheduling. • MapReduce: A programming model for batch data processing. • Spark: Provides fast, in-memory data processing. • Hive & Pig: High-level tools for querying and analyzing large datasets. • HBase: A NoSQL database for real-time read/write access. • Mahout & Spark MLlib: Libraries for scalable machine learning. • Solr & Lucene: Tools for full-text search and indexing. • Zookeeper: Manages coordination and configuration across the cluster. • Oozie: A workflow scheduler for managing Hadoop jobs.	
	(b) Explain PIG,HIVE,SPARK,OOZIE.	

4(b) Pig, hive, spark oozie- 1 marks each

- **PIG:** Apache Pig is a platform for analyzing large datasets. It uses a high-level, procedural data flow language called Pig Latin, which simplifies the writing of complex data processing tasks. Pig Latin scripts are then translated into MapReduce jobs, allowing users to process data in Hadoop without writing Java MapReduce code directly. Pig is particularly useful for ETL (Extract, Transform, Load) operations and for processing semi-structured or unstructured data.
- **HIVE:** Apache Hive is a data warehouse software built on top of Hadoop. It provides a SQL-like query language called HiveQL (HQL) to query and manage large datasets residing in distributed storage like HDFS. Hive allows users familiar with SQL to interact with big data, abstracting away the complexities of MapReduce. It is commonly used for data summarization, ad-hoc queries, and data analysis for reporting.
- **SPARK:** Apache Spark is a fast and general-purpose distributed processing engine for large-scale data analytics. Unlike Hadoop's disk-based processing, Spark leverages in-memory computation, leading to significantly faster performance for many workloads. It offers APIs in Java, Scala, Python, and R, and supports various processing paradigms including batch

	processing, interactive queries (Spark SQL), real-time streaming (Spark Streaming), machine learning (MLlib), and graph processing (GraphX). ● **OOZIE:** Apache Oozie is a workflow scheduler system for managing Apache Hadoop jobs. It allows users to define and schedule a sequence of actions (like Pig jobs, Hive queries, MapReduce jobs, or Spark applications) as a Directed Acyclic Graph (DAG). Oozie ensures that these jobs run in a specific order, handling dependencies between them and providing features for scheduling based on time or data availability. It is crucial for automating and coordinating complex data pipelines within the Hadoop ecosystem.	

5 (a) With neat Diagram explain File Read &amp;File Write anatomy with respect to HDFS.

5 (A) File read and file write anatomy - diagram 3 marks

Explanation 3 marks

Big data is nothing but a collection of data sets that are large, complex, and which are difficult to store and process using available data management tools or traditional data processing applications. Hadoop is a framework (open source) for writing, running, storing, and processing large datasets in a parallel and distributed manner. It is a solution that is used to overcome the challenges faced by big data.

Hadoop has two components:

- HDFS (Hadoop Distributed File System)
- YARN (Yet Another Resource Negotiator)

In this article, we focus on one of the components of Hadoop i.e., HDFS and the anatomy of file reading and file writing in HDFS. HDFS is a file system designed for storing very large files (files that are hundreds of megabytes, gigabytes, or terabytes in size) with streaming data access, running on clusters of commodity hardware(commonly available hardware that can be obtained from various vendors). In simple terms, the storage unit of Hadoop is called HDFS.

Some of the characteristics of HDFS are:

- Fault-Tolerance
- Scalability
- Distributed Storage
- Reliability
- High availability
- Cost-effective
- High throughput

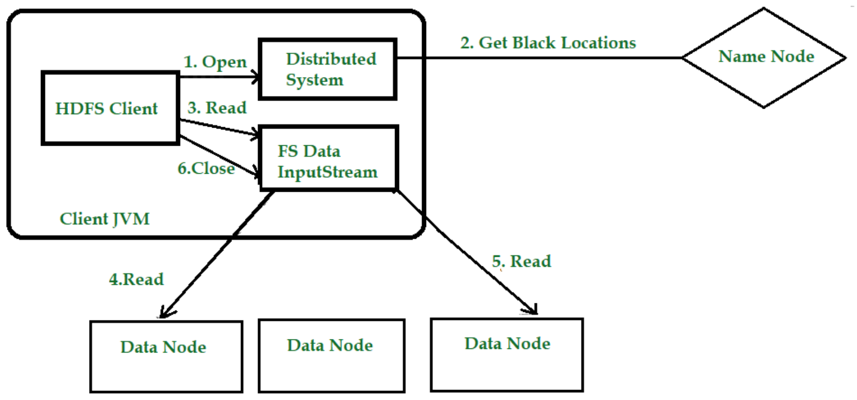
Building Blocks of Hadoop:

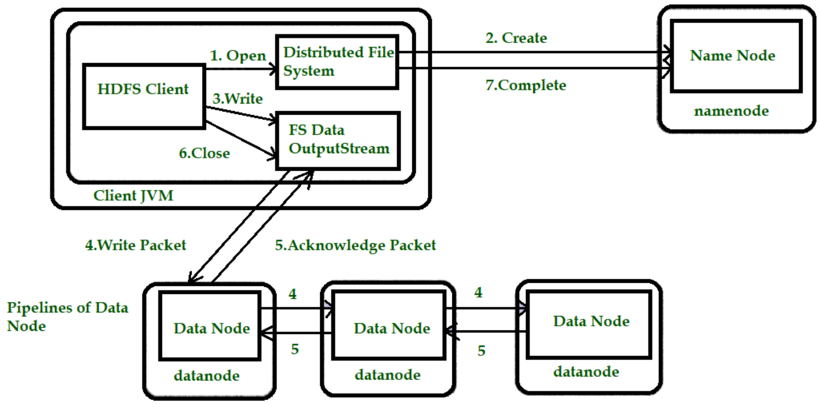
1. Name Node
2. Data Node
3. Secondary Name Node (SNN)
4. Job Tracker

5. Task Tracker

Anatomy of File Read in HDFS

Let's get an idea of how data flows between the client interacting with HDFS, the name node, and the data nodes with the help of a diagram. Consider the figure:





Step 1: The client creates the file by calling create() on DistributedFileSystem(DFS).

Step 2: DFS makes an RPC call to the name node to create a new file in the file system's namespace, with no blocks associated with it. The name node performs various checks to make sure the file doesn't already exist and that the client has the right permissions to create the file. If these checks pass, the name node prepares a record of the new file; otherwise, the file can't be created and therefore the client is thrown an error i.e. IOException. The DFS returns an FSDataOutputStream for the client to start out writing data to.

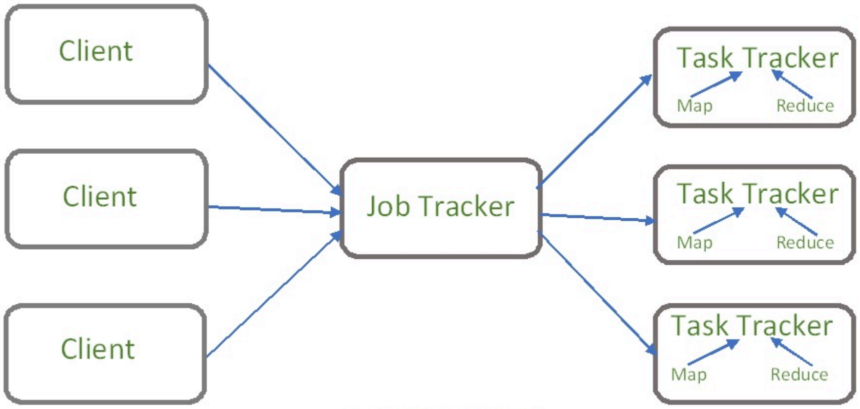
Step 3: Because the client writes data, the DFSOutputStream splits it into packets, which it writes to an indoor queue called the info queue. The data queue is consumed by the DataStreamer, which is liable for asking the name node to allocate new blocks by picking an inventory of suitable data nodes to store the replicas. The list of data nodes forms a pipeline, and here we'll assume the replication level is three, so there are three nodes in the pipeline. The DataStreamer streams the packets to the

	primary data node within the pipeline, which stores each packet and forwards it to the second data node within the pipeline. Step 4: Similarly, the second data node stores the packet and forwards it to the third (and last) data node in the pipeline. Step 5: The DFSOutputStream sustains an internal queue of packets that are waiting to be acknowledged by data nodes, called an "ack queue". Step 6: This action sends up all the remaining packets to the data node pipeline and waits for acknowledgments before connecting to the name node to signal whether the file is complete or not. HDFS follows Write Once Read Many models. So, we can't edit files that are already stored in HDFS, but we can include them by again reopening the file. This design allows HDFS to scale to a large number of concurrent clients because the data traffic is spread across all the data nodes in the cluster. Thus, it increases the availability, scalability, and throughput of the system.	

Explain YARN Architecture with diagram.

5(b) YARN ARCHITECTURE- 2 marks diagram , 2 marks explanation

YARN stands for "**Yet Another Resource Negotiator**". It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN was described as a "Redesigned Resource Manager" at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing.

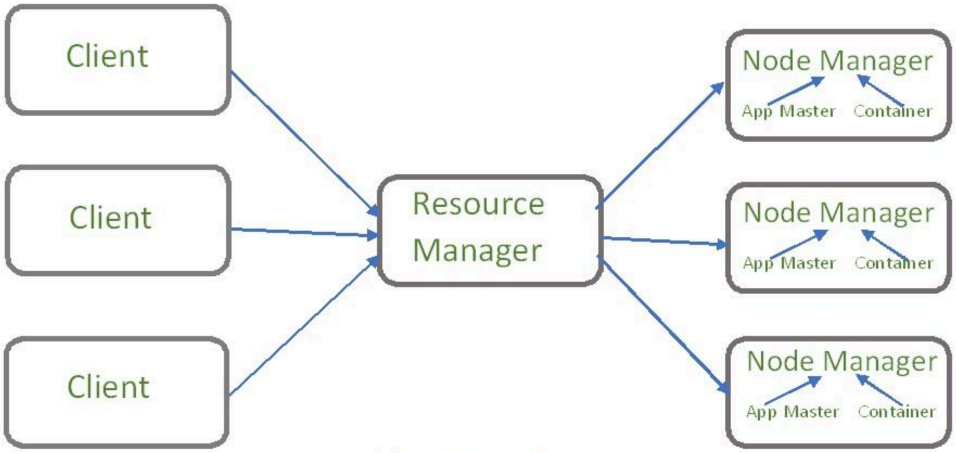


Hadoop 1.0 architecture

YARN Features

YARN gained popularity because of the following features:

	1. **Scalability:** The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters. 2. **Compatibility:** YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well. 3. **Cluster Utilization:**Since YARN supports Dynamic utilization of cluster in Hadoop, which enables optimized Cluster Utilization. 4. **Multi-tenancy:** It allows multiple engine access thus giving organizations a benefit of multi-tenancy. **Hadoop YARN Architecture**	

	 Client Client Client Resource Manager Node Manager App Master Container Node Manager App Master Container Node Manager App Master Container **Hadoop Yarn architecture**	

6(a) Explain CRUD Operation of MongoDB with example.

6(a) CRUD operations MONGO DB- 4 marks for C,R,U,D 2 marks example

CRUD stands for Create, Read, Update, and Delete. These are the four fundamental operations performed on data in a database, including MongoDB.

1. Create Operations (Insert):

This operation adds new documents to a collection. If the specified collection does not exist, MongoDB creates it.

- **Example: Inserting a single document into a `users` collection.**

JavaScript

```
   db.users.insertOne({ name: "Alice", age: 30, city: "New York" });
```

2. Read Operations (Query):

This operation retrieves documents from a collection based on specified criteria or filters.

- **Example: Finding all users with an age greater than 25.**

JavaScript

```
  db.users.find({ age: { $gt: 25 } });
```

- **Example: Finding a single user named "Alice".**

JavaScript

```
  db.users.findOne({ name: "Alice" });
```

3. Update Operations:

This operation modifies existing documents in a collection. You can update a single document or multiple documents that match a specified filter.

- **Example: Updating the `city` of the user named "Alice" to "Los Angeles".**

JavaScript

```
  db.users.updateOne(

    { name: "Alice" },

    { $set: { city: "Los Angeles" } }

  );
```

- **Example: Incrementing the `age` of all users in "New York" by 1.**

JavaScript

```
  db.users.updateMany(

    { city: "New York" },

    { $inc: { age: 1 } }

  );
```

4. Delete Operations:

This operation removes documents from a collection that match a specified filter.

- **Example: Deleting the user named "Alice".**

JavaScript

```
  db.users.deleteOne({ name: "Alice" });
```

- **Example: Deleting all users with an age less than 20.**

JavaScript

```
  db.users.deleteMany({ age: { $lt: 20 } });
```

Consider a collection named orders has the following documents:
{ &quot;_id&quot;: 1, &quot;customer&quot;: &quot;Ravi&quot;, &quot;items&quot;: [&quot;Pizza&quot;, &quot;Burger&quot;], &quot;total&quot;: 550, &quot;city&quot;: &quot;Bengaluru&quot; }
{ &quot;_id&quot;: 2, &quot;customer&quot;: &quot;Anita&quot;, &quot;items&quot;: [&quot;Pasta&quot;], &quot;total&quot;: 300, &quot;city&quot;: &quot;Mumbai&quot; }
{ &quot;_id&quot;: 3, &quot;customer&quot;: &quot;John&quot;, &quot;items&quot;: [&quot;Pizza&quot;, &quot;Coke&quot;], &quot;total&quot;: 450, &quot;city&quot;: &quot;Bengaluru&quot; }
{ &quot;_id&quot;: 4, &quot;customer&quot;: &quot;Meera&quot;, &quot;items&quot;: [&quot;Sandwich&quot;, &quot;Juice&quot;], &quot;total&quot;: 250, &quot;city&quot;: &quot;Delhi&quot; }
Write a MongoDB query to:
1. Find all orders where the city is &quot;Bengaluru&quot;.
2. Display only the customer and total fields in the output.

1. Find all orders where the city is "Bengaluru". 2 marks each
2. Display only the customer and total fields in the output.- 2 marks each

```
db.orders.find({ city: "Bengaluru" })
```

Explanation:

- `db.orders` → refers to the **orders** collection.
- `.find({ city: "Bengaluru" })` → filters documents where the `city` field matches `"Bengaluru"`.

Output:

```
{ "_id": 1, "customer": "Ravi", "items":
["Pizza", "Burger"], "total": 550, "city":
"Bengaluru" }
{ "_id": 3, "customer": "John", "items":
["Pizza", "Coke"], "total": 450, "city":
"Bengaluru" }
```

```
db.orders.find(
  { city: "Bengaluru" },        // Filter
condition
  { _id: 0, customer: 1, total: 1 }  //
Projection fields
)
```

Explanation:

- { city: "Bengaluru" } → selects documents where the city is Bengaluru.
- { _id: 0, customer: 1, total: 1 } →
  - customer: 1 and total: 1 → include these fields in the output.
  - _id: 0 → exclude the _id field.

Output:

```
{ "customer": "Ravi", "total": 550 }
{ "customer": "John", "total": 450 }
```

Explain PIG,HIVE,SPARK,OOZIE.

PIG, HIVE, SPARK, and OOZIE — all important tools in the **Big Data ecosystem (especially Hadoop-based systems):**

🐷 1. Apache Pig

Purpose: Data processing and analysis framework
Language: Pig Latin (a high-level scripting language)
Engine: Converts Pig Latin scripts into MapReduce jobs.

🔹 **Key Points:**

- Designed for **ETL (Extract, Transform, Load)** data operations.

- Makes **complex MapReduce programming simpler**.

- Ideal for **data preprocessing**, **aggregation**, and **data cleansing**.

💡 **Example use:**
Cleaning raw log files or transforming large datasets before analysis.

🐝 2. Apache Hive

Purpose: Data warehousing and querying system
Language: HiveQL (similar to SQL)
Engine: Converts HiveQL queries into MapReduce, Tez, or Spark jobs.

🔹 **Key Points:**

- Allows users to **query large datasets using SQL-like syntax**.

- Stores data in **tables**, similar to relational databases.

- Best suited for **batch processing** and **report generation**.

💡 **Example use:**
Running SQL-style queries like:

```
SELECT city, COUNT(*) FROM orders GROUP BY
city;
```

⚡ 3. Apache Spark

Purpose: Unified analytics engine for **fast, in-memory data processing**.
Language APIs: Supports **Scala, Java, Python, R**.

🔹 **Key Points:**

- Much **faster than Hadoop MapReduce** due to in-memory computation.
- Handles **batch processing**, **streaming**, **machine learning (MLlib)**, and **graph processing (GraphX)**.
- Can run standalone or on Hadoop, Kubernetes, or Mesos.

💡 **Example use:**
Real-time analytics like monitoring food delivery trends or dynamic pricing.

🔁 4. Apache Oozie

	Purpose: Workflow scheduler system for Hadoop jobs. 🔹 **Key Points:** ● Manages and automates **job execution sequences** (Pig, Hive, MapReduce, Spark jobs). ● Supports **time-based (coordinator)** and **data-based (trigger)** workflows. ● Ensures that **complex data pipelines** run in the correct order. 💡 **Example use:** Automating a workflow such as: 1. Load data → 2. Clean using Pig → 3. Analyze using Hive → 4. Generate reports.	

5g