DBMS Advantages

- **Security**

A Database Management System (DBMS) provides a better platform for data privacy and security policies thus, helping companies to improve Data Security

- **Integrity**

Due to the Database Management System we have access to well-managed and synchronized forms of data thus it makes data handling very easy and gives an integrated view of how a particular organization is working

- **Backup**

DBMS solves this problem of taking backup again and again because it allows automatic backup and recovery of database.

- **Redundancy**

provides the feature to prevent the input of duplicate items in the database. for e.g. – If there are two same students in different rows, then one of the duplicate data will be deleted

- **Concurrency**

If two users are accessing data simultaneously and they both want to update values of same record, then it may create concurrency. DBMS has the power to control concurrency so that no transactions are lost.

- **Data sharing**

provides a platform for sharing data across multiple applications and users, which can increase productivity and collaboration.

Roles in a Database System
Explaining each of the following roles:

System Analyst

- is a bridge between business needs and technology solutions. They meticulously analyze an organization's processes, systems, and data to identify opportunities for improvement and innovation

Database Designer

- are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. Database designers typically interact with each potential group and user and develop a view of the database that meets the data and processing requirements of these groups.

Database Developer

- These professionals design and create the database schema, tables, views, and queries. They work closely with the DBAs to ensure that the database is designed and implemented to meet the needs of the organization.

DBA (Admin)

- The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.

Application Developer

- These professionals create applications that interact with the database. They work closely with the database developers and DBAs to ensure that the applications are efficient and use the database resources appropriately.

BI Developer

- Business Intelligence (BI) Developer plays a crucial role in transforming data into actionable insights that drive strategic decision-making within an organization. With the growing importance of data in today's business landscape, BI Developers are tasked with designing, developing, and maintaining BI solutions that enable stakeholders to access and analyze data effectively.

Relational vs Non-Relational (e.g., MongoDB, Cassandra)

	Relational	Non-Relational
Definition	A relational database stores and allows access to related data points through structured schemas and SQL	type of database that does not rely on the traditional tabular structure of rows and columns found in relational databases. Instead, it uses flexible data models such as key-value pairs, documents, graphs, and wide-column stores.
Key Features	• Structured schema • ACID compliance • SQL support • Table relationships	• Scalability • Flexibility in Data Models • Performance • Schemaless Design • High Availability and Fault Tolerance • Cost-Effectiveness
Flexibility	Less flexible with rigid schema definitions	Highly flexible due to schema-less or dynamic schema
When to Use	Choose a relational database when your data has clear structure, relationships, and you need strong consistency especially for financial systems, inventory, or user records.	• Applications requiring real-time data processing. • Scenarios with massive and rapidly growing datasets. • Use cases involving unstructured or semi-structured data. • Agile development environments where data models evolve frequently.
Examples	MySQL, PostgreSQL, SQL Server	MongoDB, Cassandra, Redis, DynamoDB

Centralized vs Distributed vs Cloud Databases

	Centralized	Distributed	Cloud Databases
Definition	It is a database that is stored, located as well as maintained at a single location only.	It is a database that consists of multiple databases which are connected with each other and are spread across different physical locations.	are managed database services hosted on cloud platforms, offering scalability and resource flexibility
Advantages	• Integrity of data • Security • Easy access to all information • Data is easily portable	• High performance because of the division of workload. • High availability because of the readiness of available nodes to do work. • Independent nodes and better control over resources	• Scalability: Easily scales up or down based on demand. • Cost-Effective: Reduces upfront hardware costs and maintenance expenses. • Accessibility: Accessible from anywhere, facilitating remote work and collaboration. • Backup & Recovery: Automated backup and disaster recovery. • Managed Services: Cloud providers manage maintenance, updates, and patches.
Disadvantages	• Data searching takes time • In case of failure of a centralized server, the whole database will be lost. • If multiple users try to access the data at the same time then it may create issues.	• It is quite large and complex so difficult to use and maintain. • Difficult to provide security • Issue of data integrity • Increase in storage and infrastructure requirements • Handling failures is a quite difficult task	• Latency Issues: Potential delays due to internet connectivity dependencies. • Security Concerns: Security & privacy issues depending on the provider's protocols. • Limited Control: Users have less control over the physical database and hardware. • Vendor Lock-in: Potential difficulties in migrating to another service. • Cost Predictability: Costs can escalate with increased usage and data growth.
Examples	• A desktop or server CPU • A mainframe computer.	• Apache Ignite • Apache Cassandra • Apache HBase • Amazon SimpleDB • Clusterpoint • FoundationDB.	• Amazon RDS, • Google Cloud SQL

Cloud Storage and Databases

01. WHAT IS CLOUD STORAGE

Cloud Storage is a method of storing digital data on remote servers managed by third-party providers. These servers are located in off-site data centers and are accessible via the internet. The providers are responsible for hosting, managing, and securing the data, ensuring it is always available.

02. HOW CLOUD IS RELATED TO DATABASES

loud Storage operates by using remote servers to save various types of data, such as files, business data, videos, or images. Users upload data to these servers through an internet connection, where it is stored on virtual machines within physical servers. To ensure availability and redundancy, cloud providers often distribute data across multiple virtual machines in different data centers worldwide.
When storage needs increase, the cloud provider can allocate more virtual machines to handle the additional load. Users can access their data through an internet connection using software like web portals, browsers, or mobile apps via an application programming interface (API)

03. ADVANTAGES OF CLOUD STORAGE

Total Cost of Ownership: Shifts expenses from capital expenditure to operational expenditure, allowing for budget adjustments
Elasticity: Scalable storage that can be adjusted based on organizational needs
Flexibility: Offers various ways to store and access data, deploy resources, and design IT infrastructure
Security: Provides robust security measures, including physical security at data centers and advanced software security
Sustainability: Operates on sustainable energy through renewable resources
Redundancy: Ensures data recovery and business continuity by replicating data across multiple servers

04. DISADVANTAGES OF CLOUD STORAGE

Compliance: Some industries have stringent requirements for data storage and access
Latency: Network traffic congestion or slow internet connections can cause delays
Control: Relinquishes some control over data management to the cloud service provider
Outages: Although rare, outages can make stored data temporarily unavailable

DATABASE ENGINES AND LANGUAGES

WHAT IS A DATABASE ENGINE?

s the core component of a database management system (DBMS) that handles data storage, retrieval, and manipulation. It processes SQL commands, manages data integrity, performs transactions, and optimizes query performance.

01.

EXAMPLES: SQL SERVER, MYSQL, ORACLE, POSTGRESQL

- SQL Server: Microsoft
- MySQL: Oracle Corporation
- Oracle Database: Oracle Corporation
- PostgreSQL: Open-source

02.

WHAT LANGUAGES DO THEY USE? (E.G., T-SQL, PL/SQL, ANSI SQL)

SQL Server: T-SQL (Transact-SQL)
MySQL: ANSI SQL + MySQL-specific extensions
Oracle Database: PL/SQL (Procedural Language/SQL)
PostgreSQL: PL/pgSQL (PostgreSQL Procedural Language) and ANSI SQL

03.

IS THERE A RELATIONSHIP BETWEEN THE ENGINE AND THE LANGUAGE?

Yes, there is a strong relationship between a database engine and the language it uses. Each engine supports ANSI SQL (a standardized version of SQL), but also includes its own extensions to provide additional functionality.

04.

CAN ONE LANGUAGE WORK ACROSS DIFFERENT ENGINES?

- Basic ANSI SQL (e.g., SELECT, INSERT, UPDATE) works across most database engines.
- Advanced or procedural SQL dialects (e.g., T-SQL, PL/SQL) are not portable across engines without modification.
- Cross-platform compatibility often requires writing standard-compliant SQL and avoiding vendor-specific features.

05.

CAN WE TRANSFER A DATABASE BETWEEN ENGINES?

IS IT POSSIBLE TO MIGRATE A DATABASE FROM SQL SERVER TO MYSQL, OR ORACLE TO POSTGRESQL?

Yes, databases can be migrated between engines like:

- SQL Server → MySQL
- Oracle → PostgreSQL

This process is often used for cost savings, open-source adoption, or cloud platform compatibility. However, the migration is rarely one-click and involves multiple phases including schema conversion, data transfer, and logic rewriting.

WHAT ARE THE CHALLENGES OF ENGINE-TO-ENGINE MIGRATION?

1. Different SQL Dialects
    - SQL Server uses T-SQL, Oracle uses PL/SQL, and PostgreSQL uses PL/pgSQL.
    - These dialects differ in syntax, procedural logic, and function behavior.
2. Data Type Incompatibility
    - Example: Oracle's NUMBER doesn't map directly to PostgreSQL's types.
    - SQL Server's DATETIME has different behavior than MySQL's equivalent.
3. Stored Procedures and Triggers
    - Procedural code like functions, triggers, and stored procedures must often be rewritten.
    - Example: Oracle's BEFORE INSERT triggers may not have a direct translation in MySQL.
4. Indexes and Constraints
    - Index types (e.g., bitmap indexes in Oracle) may not exist in the target engine.
    - Constraint naming and behavior might differ.
5. Query Optimization and Execution Plans
    - Optimizers work differently across engines. Queries may behave or perform differently.
6. Security Models
    - User roles, permissions, and authentication methods vary by engine.
7. Tooling Differences
    - Backup/restore mechanisms and logging systems are not standardized.

WHAT SHOULD WE CONSIDER BEFORE TRANSFERRING (DATA TYPES, TRIGGERS, STORED PROCEDURES, ETC.)?

Data Types: Ensure target engine supports or has an equivalent for source data types.

Stored Procedures: Review and rewrite using the procedural language of the target DB.

Triggers & Functions: Map functionality carefully; test behavior post-migration.

Indexes: Rebuild indexes using best practices in the target engine.

Constraints: Recreate with compatible logic.

Data Integrity: Use checksums or row counts to ensure no data is lost or altered.

LOGICAL VS. PHYSICAL SCHEMA

What is the Logical Schema in database design?	s the abstract design of a database. It defines the structure of the data
What is the Physical Schema?	describes how the database is actually implemented in a specific DBMS, including data types, indexing, partitioning, storage formats, and performance-related settings.
What's the difference between them?	logical schema: • That describes the data without regard to how they will be physically implemented in the database. • Defines the data elements and their relationship. Physical Schema: • That represents how the actual database is built • Developing the actual database
Why is it important to understand both?	• Separation of concerns: Logical schema lets designers focus on data modeling without worrying about hardware or DBMS. • Better collaboration: Analysts, designers, and DBAs can work on different layers. • Efficient performance: Physical schema lets DBAs optimize queries and storage. • Scalability: Logical models remain stable even when physical implementations change for scaling or performance.
Example: Show how one entity (e.g., Student) would appear in both logical and physical schemas.	Logical Schema: This defines the structure of the data without worrying about how it will be implemented in a specific DBMS. It's more abstract and focuses on entities, attributes, and relationships. Physical Schema: This defines how the logical design is actually stored in the database, considering data types, constraints, indexing, and storage specifics.