dbms?
dabase management system.a set of applications or programs that enables users to create and maintain database. it provides tool or interface for performing various operations such as inserting, updating, deleting, etc into a database.

rdbms?
relational database management system. store data in form o f tables thus making it easier to access and store data efficiently.
ex - mysql, oracle db...

database?
The database is a collection of inter-related or organised data which is used to retrieve, insert and delete the data efficiently.

data?
collection of information.

advantages of dbms?
controls database redundancy
data sharing
easily maintained
reduce development and maintainance time.

Types of DBMS architecture??

schema - structural representation of the database.It defines the arrangement of the data in the database and provides the blue print for how data is stored accessed and related to each other.

different languages in dbms?
ddl - data definition language.define the database.
ex - create, alter, drop, truncate, rename...
dml - data manipulation language. manipulate the data.
ex - select, update, insert, delete...
dcl - data control language.deals with permission and controls of database system.
ex - grant, revoke...
tcl - transaction control language.deal with transaction of database.
ex - commit, rollback, savepoint.

acid properties in dbms?
atomicity - all changes in data must be performed successfully or not at all.
consistent - data must be consistent before and after the transaction.

isolation - no other process can change the data while trasaction is going on. each transaction is occurring independently of the others.
durable - data doesn't get lost in case of system failure or restart and is present in the same state as it was before the system failure or restart.

ER (Entity Relationship) diagram in DBMS:
Entity - any object, class, person or place.represented as rectangles.
weak entity - an entity that depends on another entity.
Attributes - Property of an entity.represented by eclipse.
simple - represents a single, atomic value.
composite - multiple simple attributes.
multi-valued - can have multiple values.
derived - not stored directly but can be derived from other attributes or data in database.
key attributes - attribute that uniquely identify each row in table. primary and composite key are key attributes.
Relationship - describe relations between entities.represented by diamond or rhombus.
one-to-one.
one-to-many.
many-to-one.
many-to-many.

Functional Dependency - A relationship that exists between two attributes.
X -> Y
The left side (X) of FD is known as a determinant, the right side of the production is known as a dependent.It means that Y is funtionally dependent on X.
1. Trivial Functional Dependency

- A -> B has a trivial dependency if B is a subset of A.
- A -> A, B -> B etc are trivial.
  2. Non-Trivial Functional Dependency
    - A -> B is non-trivial dependency if B is not a subset of A.

> Non-Trivial Dependency:
> Suppose you find that the attribute Employee_ID determines the attribute Employee_Name. This means that for each unique Employee_ID, there is only one corresponding Employee_Name. This is a non-trivial dependency because it provides new information about how Employee_ID and Employee_Name are related. It's not obvious or trivial that the employee ID would determine the employee's name.

> Complete Non-Trivial Dependency:
> Now, let's say you observe that the combination of Employee_ID and Department_ID determines the attribute Employee_Name. In this case, you can't break down the dependency into smaller parts while retaining the same meaning. Both Employee_ID and Department_ID are needed together to determine Employee_Name. If you tried to remove either Employee_ID or Department_ID from the left-hand side of the dependency, you would lose the ability to determine Employee_Name uniquely. This is a complete non-trivial dependency.

Inference Rule(IR): logical deduction or transformation that allows you to draw new conclusions based on existing rules

1. Reflexive Rule:
   if Y is a subset of X.
   X -> Y
2. Augmentation Rule:
   X -> Y then XZ -> YZ
3. Transitive Rule:
   X -> Y and Y -> z then X -> Z
4. Union Rule:
   X -> Y and X -> Z then X -> YZ
5. Decomposition Rule:
   X -> YZ then X -> Y and X -> Z
6. Pseudo Transitive Rule:
   X -> Y and YZ -> W then XZ -> W

> Normalisation: process of decomposing the relations into relations with fewer attributes.

- Elimination of Data Redundancy: Redundant data occurs when the same piece of information is stored in multiple places in the database. This redundancy wastes storage space and can lead to inconsistencies if the same data is updated in one place but not in others. Normalization helps in breaking down the data into smaller, related tables, each representing a single subject, which reduces redundancy.
- Data Consistency: When data is stored redundantly, inconsistencies can arise when updates are made in one place but not propagated to all instances of that data. Normalization minimizes these inconsistencies by ensuring that each piece of data is stored in only one place, reducing the chance of conflicts.
- Anomaly Prevention: Anomalies are inconsistencies or irregularities that can occur when data is not properly organized. These include insertion, update, and deletion anomalies. Normalization

helps prevent anomalies by structuring data in a way that each piece of information is stored in its appropriate place without duplication.

- Improved Data Integrity: Data integrity refers to the accuracy and reliability of the data stored in the database. By eliminating redundancy and anomalies, normalization contributes to better data integrity. Ensuring that data accurately represents the real-world entities it represents is essential for making informed business decisions.
- Simplified Updates: In a normalized database, when updates need to be made, they typically affect only one place where the data is stored. This simplifies the update process and reduces the chances of errors that could occur when updating redundant data.
- Efficient Querying: Normalization can help improve the efficiency of querying and data retrieval. Smaller, well-structured tables are easier for the DBMS to manage and optimize for query execution. This can lead to faster query performance and improved response times.
- Scalability and Flexibility: A normalized database schema provides a solid foundation for scalability. As the database grows and new requirements emerge, it's easier to accommodate changes and additions to the data structure without causing major disruptions.
- Maintainability: A normalized database is generally easier to maintain and modify. Changes to the schema are more straightforward, and developers can work with a clear understanding of the relationships between different data elements.

> Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

> Anomalies : inconsistencies or irregularities that can occur in a database when the data is not properly organized or structured.
> It is of three types:

- Insertion Anomaly: Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- Deletion Anomaly: The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- Updatation Anomaly: The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

> Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.