Short Answer Questions

1. What is a functional dependency?
A functional dependency occurs when the value of one attribute (or a group of attributes) determines the value of another attribute. In a relation R, attribute B is functionally dependent on attribute A (written A → B) if, for every valid instance of A, the value of A uniquely determines the value of B.

2. Define 1NF (First Normal Form).
A relation is in First Normal Form (1NF) if all attributes contain only atomic (indivisible) values, and each record is unique. There should be no repeating groups or arrays.

3. What is a partial dependency?
A partial dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key (not the whole key) in a relation with a composite key.

4. Explain the concept of transitive dependency.
A transitive dependency exists when a non-prime attribute depends on another non-prime attribute, which in turn depends on the primary key. For example, if A → B and B → C, then A → C is a transitive dependency.

5. Define 2NF (Second Normal Form).
A relation is in 2NF if it is in 1NF and has no partial dependencies; that is, every non-prime attribute is fully functionally dependent on the entire primary key.

6. What is the purpose of 3NF (Third Normal Form)?
3NF eliminates transitive dependencies to ensure that non-prime attributes are only dependent on candidate keys. This reduces redundancy and improves data integrity.

7. How does Boyce-Codd Normal Form (BCNF) differ from 3NF?
BCNF is a stricter version of 3NF. A relation is in BCNF if, for every non-trivial functional dependency A → B, A is a superkey. 3NF allows non-superkey determinants if the dependent is a prime attribute, but BCNF does not.

8. What are the advantages of database normalization?

- Reduces data redundancy
- Improves data integrity and consistency
- Makes it easier to maintain and update data
- Prevents anomalies (insert, update, delete)

9. What are some potential disadvantages of normalizing to higher normal forms?

- Increases complexity in query operations due to more table joins
- May reduce performance in read-heavy applications
- Can complicate understanding and design for newcomers

10. Give an example of a relation that is in 2NF but not in 3NF.
Consider the relation `Student(CourseID, StudentID, InstructorName)`

- Candidate key: (CourseID, StudentID)
- Functional dependencies:
    - (CourseID, StudentID) → InstructorName
    - CourseID → InstructorName

This relation is in 2NF (no partial dependency), but not in 3NF due to the transitive dependency:
(CourseID, StudentID) → CourseID → InstructorName

Critical Thinking Prompts

1. Compare and contrast 3NF and BCNF. When might you choose to normalize to BCNF even if it introduces redundancy?
3NF allows a non-superkey as a determinant if the dependent is a prime attribute, whereas BCNF requires all determinants to be superkeys.
BCNF is often used when anomalies still exist after 3NF due to violations caused by candidate keys. For example, in a multi-candidate key scenario, BCNF may eliminate redundancy better. One might choose BCNF even if it introduces some redundancy if data anomalies (especially updates or deletions) are more critical than storage or performance.

2. Argue for or against the statement: "Normalization always leads to better database performance."
Against:
Normalization reduces redundancy and ensures integrity but may not always improve performance. In read-heavy systems (e.g., reporting dashboards), joins across normalized tables can slow down queries.
Example: A sales analytics dashboard may benefit from denormalization to avoid multiple joins,

improving speed at the cost of some redundancy.
Conclusion: Normalization is about data quality, not necessarily performance. Optimization depends on use case.

3. Designing a library database: Functional dependencies and normalization to 3NF
Attributes: BookID, Title, AuthorID, AuthorName, MemberID, MemberName, IssueDate
Functional dependencies:

- BookID → Title, AuthorID
- AuthorID → AuthorName
- MemberID → MemberName
- (BookID, MemberID) → IssueDate

Normalization Steps:

- 1NF: Ensure atomicity (e.g., no multiple authors in one row)
- 2NF: Remove partial dependencies
    - Separate Book(BookID, Title, AuthorID)
    - Author(AuthorID, AuthorName)
    - Member(MemberID, MemberName)
    - Issued(BookID, MemberID, IssueDate)
- 3NF: Remove transitive dependencies
    - Already achieved since all non-key attributes depend only on keys

4. Practical limitations of higher normal forms (4NF, 5NF)
4NF deals with multi-valued dependencies and 5NF with join dependencies. These are complex and rare in most applications.
Limitations:

- Higher complexity in design
- Less support in ORMs and common database tools

- Often unnecessary unless handling advanced scenarios (e.g., product configuration systems)
  Thus, 3NF or BCNF is typically sufficient for most systems.

5. Improving a poorly performing database with normalization or denormalization
Normalization:

- Use it if the database has too much redundant data and suffers from update anomalies
- Clean schema improves maintainability and avoids data corruption

Denormalization:

- Use it in performance-critical applications with frequent reads (e.g., reports, dashboards)
- Combine tables to reduce JOINs
  Example: Materialized views or summary tables in a data warehouse are denormalized for speed

Conclusion:
Both strategies have their place. The right choice depends on the system's read/write patterns and performance needs.