Q1: "What is Data?"
Ans: "Data is raw facts and figures without any context. It can be numbers, text, images, or symbols that have not yet been processed into meaningful information."

Q2: "State the differences between data and information."
Ans: "Data is raw and unprocessed, while information is processed, organized, and meaningful data that helps in decision-making."

Q3: "What is Database?"
Ans: "A database is a structured collection of data stored electronically, which can be accessed, managed, and updated efficiently."

Q4: "What is DBMS?"
Ans: "DBMS (Database Management System) is software that allows users to create, manage, and manipulate databases while ensuring data integrity and security."

Q5: "State the differences between Database & Database Management System."
Ans: "A database is the collection of data, while a DBMS is the software that manages the database, enabling data storage, retrieval, and manipulation."

Q6: "State the differences between Database & Data Structures."
Ans: "A database stores persistent, organized data for multiple users, while data structures are in-memory formats used in programs to store and manage temporary data."

Q7: "What are the advantages of DBMS?"
Ans: "Advantages include data consistency, reduced redundancy, data security, easy data access, backup and recovery, and multi-user support."

Q8: "What are the advantages of DBMS over File System?"
Ans: "DBMS reduces data redundancy, enforces data integrity, allows concurrent access, supports complex queries, and provides security and backup mechanisms compared to file systems."

Q9: "What are the disadvantages of DBMS?"
Ans: "Disadvantages include high initial cost, complexity, hardware requirements, and overhead due to DBMS software management."

Q10: "What is the characteristics of DBMS?"
Ans: "Characteristics include self-describing nature, data abstraction, support for multiple views, transaction management, and concurrency control."

Q11: "Explain different languages present in DBMS."
Ans: "DBMS languages include DDL (Data Definition Language) for schema creation, DML (Data Manipulation Language) for data operations, DCL (Data Control Language) for access control, and TCL (Transaction Control Language) for transaction management."

Q12: "What is CRUD in database?"
Ans: "CRUD stands for Create, Read, Update, and Delete—basic operations that can be performed on database data."

Q13: "Define 3 schema architecture in DBMS?"
Ans: "The 3-schema architecture includes: Internal schema (physical storage), Conceptual schema (logical structure), and External schema (user views), providing data abstraction."

Q14: "What is Data Independence?"
Ans: "Data independence is the ability to change the schema at one level without affecting other levels, such as changing storage without affecting applications."

Q15: "How many levels of abstraction can be achieved in DBMS?"
Ans: "Three levels: Physical level (how data is stored), Logical level (what data is stored), and View level (how users see the data)."

Q16: "What is Instance and Schema?"
Ans: "A schema is the overall logical structure of the database, describing how data is organized. An instance is the actual data stored in the database at a particular moment."

Q17: "What is the role of Data Dictionary?"
Ans: "A Data Dictionary stores metadata—information about the database such as structure, constraints, and relationships—and helps in data management and validation."

Q18: "What are the languages used in DBMS?"
Ans: "Languages in DBMS include:

- **DDL (Data Definition Language) – defines schema**
- **DML (Data Manipulation Language) – manipulates data**
- **DCL (Data Control Language) – controls access**
- **TCL (Transaction Control Language) – manages transactions**
- **Query Language (e.g., SQL) – retrieves data"**

Q19: "How many types of users are there in DBMS? What is the difference between DDL and DML?"
Ans: "Users in DBMS:

1. **Database Administrator (DBA)**
2. **Application Programmers**
3. **End Users (Casual, Naive, Sophisticated)**
   Difference: DDL defines the database structure/schema (e.g., CREATE, ALTER), whereas DML manipulates the data (e.g., INSERT, UPDATE, DELETE)."

Q20: "What are the five major functions of a database administrator?"
Ans: "DBA functions:

1. **Schema definition and modification**
2. **Data storage and retrieval management**
3. **Security and authorization management**
4. **Backup and recovery planning**
5. **Performance monitoring and tuning"**

Q21: "Compare relational, network, and hierarchical data models by stating the advantages and disadvantages of every model."
Ans: "Relational Model: Uses tables; easy to use, supports SQL; can have performance issues for very large datasets.

Network Model: Uses records and links; efficient for complex relationships; complex to design.
Hierarchical Model: Tree structure; fast access for parent-child; poor for many-to-many relationships."

Q22: "What is an Entity?"
Ans: "An entity is a real-world object or concept about which data is stored in a database, e.g., Student, Employee."

Q23: "What integrity rules exist in a DBMS?"
Ans: "Integrity rules:

1. **Entity Integrity – primary key must be unique and not null**
2. **Referential Integrity – foreign key must match primary key in another table**
3. **Domain Integrity – attribute values must fall within a defined domain**
4. **User-defined Integrity – any additional business rules"**

Q24: "What are the components of E-R Model?"
Ans: "Components: Entity, Attribute, Relationship, Entity Set, Relationship Set, Key"

Q25: "Briefly discuss the components of E-R Model?"
Ans: "Entity – object/concept; Attribute – property of entity; Relationship – association between entities; Key – unique identifier; Entity Set – collection of similar entities; Relationship Set – collection of similar relationships"

Q26: "Briefly discuss the following terms used in DBMS: Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, Relation Instance, Relation Key, Attribute Domain."
Ans: "Attribute – column of a table; Tables – collection of rows/tuples; Tuple – a row; Relation Schema – structure of table; Degree – number of attributes; Cardinality – number of tuples; Column – same as attribute; Relation Instance – current rows in table; Relation Key – uniquely identifies a tuple; Attribute Domain – allowed values for an attribute"

Q27: "What is the role of ER Model to create a database?"
Ans: "ER Model provides a conceptual blueprint of data, relationships, and constraints, which is used to design the logical schema of a database."

Q28: "Give an example of each relationship:
a. one-to-one – Person and Passport
b. one-to-many – Department and Employees
c. many-to-many – Students and Courses"

Q29: "What are the differences between generalization and specialization?"
Ans: "Generalization combines similar entities into a higher-level entity (bottom-up), whereas specialization divides an entity into sub-entities (top-down) based on differences."

Q30: "What is Aggregation? State with an example."
Ans: "Aggregation is a higher-level abstraction where a relationship between entities is treated as a single entity.
Example: 'Project' is assigned to a 'Department'; we can aggregate the relationship 'Works_On' between Employee and Project into a higher-level entity."

Q31: "What are the various types of attributes? Discuss about them."
Ans: "Types of attributes:

1. **Simple – cannot be divided (e.g., Name)**
2. **Composite – can be divided (e.g., Full Name → First Name + Last Name)**
3. **Derived – calculated from other attributes (e.g., Age from DOB)**
4. **Single-valued – holds single value (e.g., Gender)**
5. **Multi-valued – can hold multiple values (e.g., Phone Numbers)**
6. **Key attribute – uniquely identifies an entity (e.g., Employee ID)"**

Q32: "What are CODD's Rules? Briefly discuss each rule."
Ans: "Codd's 12 rules define characteristics of RDBMS:

1. **Information Rule – all data represented as table values**
2. **Guaranteed Access – each data element accessible via table, row, column**
3. **Systematic Treatment of NULL – supports missing info**
4. **Dynamic Online Catalog – metadata accessible via query**
5. **Comprehensive Data Sub-language – supports DDL, DML, DCL**
6. **View Updating – views updatable**
7. **High-level Insert, Update, Delete – support set operations**
8. **Physical Data Independence – schema changes don't affect apps**
9. **Logical Data Independence – logical changes don't affect apps**
10. **Integrity Independence – constraints stored in catalog**
11. **Distribution Independence – apps unaffected by distributed data**
12. **Non-subversion Rule – cannot bypass integrity constraints"**

Q33: "How many rules should a software satisfy to qualify as a RDBMS?"
Ans: "It should satisfy all 12 of Codd's rules to be considered a full RDBMS."

Q34: "What is Integrity rule? Discuss briefly about them?"
Ans: "Integrity rules ensure accuracy and consistency:

1. **Entity Integrity – primary key cannot be null**
2. **Referential Integrity – foreign keys must match primary key values**
3. **Domain Integrity – attribute values must be from valid domain**
4. **User-defined Integrity – application-specific rules"**

Q35: "What is Relational Algebra?"
Ans: "Relational Algebra is a procedural query language used to operate on relations (tables) using operations like SELECT, PROJECT, UNION, etc."

Q36: "What is query language? How many types of query languages are present?"
Ans: "Query language is used to request data from a database. Types:

1. **Procedural – user specifies what and how (e.g., Relational Algebra)**
2. **Non-procedural – user specifies what only (e.g., SQL)"**

Q37: "What is the role of relational algebra in query language?"
Ans: "Relational algebra provides a theoretical foundation for query languages like SQL and defines operations to manipulate relations systematically."

Q38: "Briefly discuss various set operations and relational algebra operations."
Ans: "Set operations: UNION, INTERSECTION, DIFFERENCE

Relational algebra operations: SELECT (σ), PROJECT (π), JOIN, DIVISION, RENAME (ρ), SET operations (∪, ∩, –)"

Q39: "What is a JOIN operation? / Explain joining."
Ans: "JOIN operation combines rows from two or more tables based on a related column between them. It helps retrieve related data stored in multiple tables."

Q40: "Give simple example to explain various types of join present in relational algebra? Inner Join & Outer Join."
Ans: "Example:
Tables: Students(S_ID, Name), Marks(S_ID, Subject, Score)

- **Inner Join: Students ⋈ Marks → only students with marks**
- **Left Outer Join: Students ⟕ Marks → all students, marks NULL if absent**
- **Right Outer Join: Students ⟖ Marks → all marks, students NULL if absent**
- **Full Outer Join: Students ⟗ Marks → all students and marks, NULL if no match"**

Q41: "What are the differences between Tuple Relational Calculus and Domain Relational Calculus?"
Ans: "Tuple Relational Calculus (TRC) – query specifies desired tuple(s); Domain Relational Calculus (DRC) – query specifies values for attributes. TRC works with tuples, DRC works with attribute domains."

Q42: "What is a key and discuss the role of various types of keys in DBMS? How to add foreign key?"
Ans: "Key uniquely identifies a record. Types:

- **Primary Key – unique, not null**
- **Candidate Key – eligible to be primary key**
- **Alternate Key – candidate key not selected as primary**
- **Foreign Key – links tables**
- **Composite Key – combination of attributes**
  Adding foreign key in SQL: ALTER TABLE Child ADD CONSTRAINT fk_name FOREIGN KEY (column) REFERENCES Parent(column);"

Q43: "What is Lossless and Lossy Join Decomposition?"
Ans: "Lossless decomposition ensures no data is lost when a relation is decomposed.
Lossy decomposition may result in loss of information after join operations."

Q44: "What is Normalization?"
Ans: "Normalization is the process of organizing database tables to reduce redundancy and improve data integrity by dividing tables into smaller relations."

Q45: "State the need of Normalization? Difference between left join."
Ans: "Need: Eliminates redundancy, avoids update anomalies, ensures data consistency.
Difference between joins: Left Join returns all records from left table and matched records from right table; unmatched right table records appear as NULL."

Q46: "How normalization differ from denormalization?"
Ans: "Normalization organizes data to reduce redundancy and improve integrity. Denormalization intentionally introduces redundancy to improve read performance."

Q47: "What are the different types of anomalies?"
Ans: "Types of anomalies:

1. **Insertion anomaly – difficulty adding data without existing info**
2. **Deletion anomaly – deleting data unintentionally removes other info**
3. **Update anomaly – updating data requires multiple changes, risking inconsistency"**

Q48: "What are the various types of normalization levels or normalization forms?"
Ans: "Normalization forms:
1NF – First Normal Form
2NF – Second Normal Form
3NF – Third Normal Form
BCNF – Boyce-Codd Normal Form
4NF – Fourth Normal Form
5NF – Fifth Normal Form (Projection-Join Normal Form)"

Q49: "What is the First Normal Form (1NF) in database normalization?"
Ans: "1NF requires that each column contains atomic (indivisible) values and each record is unique."

Q50: "What is an atomic value, and how does it relate to 1NF?"
Ans: "Atomic value is a single, indivisible piece of data. 1NF requires all table fields to contain only atomic values, no repeating groups or arrays."

Q51: "What is the Second Normal Form (2NF), and why is it an improvement over 1NF? How can you normalize a table from 1NF to 2NF?"
Ans: "2NF eliminates partial dependency, meaning every non-key attribute depends on the full primary key. Improvement: reduces redundancy.
To normalize: separate table into multiple tables so that each non-key attribute depends on the full primary key."

Q52: "Explain partial dependency and provide an example."
Ans: "Partial dependency occurs when a non-key attribute depends on part of a composite primary key.
Example: Table(StudentID, CourseID, StudentName) – StudentName depends only on StudentID, not on (StudentID + CourseID)."

Q53: "What is the Third Normal Form (3NF), and how does it further refine the database schema?"
Ans: "3NF removes transitive dependency; non-key attributes depend only on primary key. Refines schema by reducing redundancy and ensuring data consistency."

Q54: "Explain transitive dependency and provide an example."
Ans: "Transitive dependency occurs when a non-key attribute depends on another non-key attribute.
Example: Table(EmployeeID, DeptID, DeptName) – DeptName depends on DeptID, which depends on EmployeeID."

Q55: "What is the Boyce-Codd Normal Form (BCNF), and when is it applied in database design?"
Ans: "BCNF is a stricter version of 3NF; every determinant must be a candidate key. Applied when 3NF still has anomalies due to overlapping candidate keys."

Q56: "What are prime and non-prime attributes in the context of BCNF? Explain the concept of a candidate key in the context of BCNF."
Ans: "Prime attribute – part of a candidate key. Non-prime attribute – not part of any candidate key. Candidate

key – minimal set of attributes uniquely identifying a tuple; BCNF ensures all determinants are candidate keys."

Q57: "What is denormalization, and when might it be used in database design?"
Ans: "Denormalization merges tables to reduce JOINs and improve query performance; used when read speed is prioritized over storage efficiency."

Q58: "What is the purpose of the Fourth Normal Form (4NF), and when is it applied in database design?"
Ans: "4NF removes multi-valued dependencies; applied when a table contains two or more independent multi-valued facts about an entity."

Q59: "What are multi-valued dependencies, and how do they relate to 4NF?"
Ans: "Multi-valued dependency occurs when one attribute determines multiple independent values of another attribute. 4NF eliminates these dependencies to prevent redundancy."

Q60: "What is the Fifth Normal Form (5NF), and when is it used in database design?"
Ans: "5NF (Projection-Join Normal Form) removes join dependencies; it is used when a table can be decomposed into smaller tables without losing data, ensuring no redundancy due to join operations."

Q61: "What is the redundancy involved in a non-normalized (unnormalized) database, and why is it a concern?"
Ans: "Redundancy occurs when the same data is stored multiple times. Concern: increases storage, causes inconsistencies, and makes updates, deletions, or insertions error-prone."

Q62: "How does normalization help prevent data anomalies in a database?"
Ans: "Normalization organizes data to eliminate partial, transitive, and multi-valued dependencies, reducing insertion, update, and deletion anomalies."

Q63: "What are the potential downsides of over-normalization in database design?"
Ans: "Over-normalization can lead to too many tables, complex JOINs, slower query performance, and increased maintenance overhead."

Q64: "What is the difference between functional dependency and full functional dependency?"
Ans: "Functional dependency: attribute B depends on attribute A (A → B). Full functional dependency: B depends on the whole primary key, not part of it."

Q65: "What are the primary and secondary objectives of normalization in database design?"
Ans: "Primary: reduce redundancy and prevent anomalies. Secondary: improve data integrity and optimize database structure for easier maintenance."

Q66: "Explain the process of identifying functional dependencies in a database table."
Ans: "Process:

1. **Identify primary keys**
2. **Examine relationships between attributes**
3. **Determine which attributes depend on which keys**
4. **Document as A → B functional dependencies"**

Q67: "What is a non-prime attribute, and how does it relate to normalization?"
Ans: "Non-prime attribute: not part of any candidate key. Normalization focuses on ensuring non-prime attributes depend only on candidate keys to avoid anomalies."

Q68: "What is the role of dependency preservation in normalization, and how is it achieved?"
Ans: "Dependency preservation ensures all functional dependencies are retained after decomposition. Achieved by designing tables so decomposed relations can enforce original dependencies."

Q69: "What is the role of denormalization, and when might it be used despite the benefits of normalization?"
Ans: "Denormalization reintroduces redundancy to improve query performance and reduce JOIN operations; used when read speed is more important than storage efficiency."

Q70: "How do you determine the appropriate level of normalization for a specific database design?"
Ans: "Balance between data integrity and performance: normalize to eliminate redundancy and anomalies, but avoid over-normalization if it affects query speed."

Q71: "What is the purpose of a data dictionary in database design, and how does it relate to normalization?"
Ans: "Data dictionary stores metadata about tables, attributes, keys, and dependencies; it helps identify functional dependencies and guides normalization."

Q72: "Explain the concept of closure in functional dependencies and how it is used in normalization."
Ans: "Closure of an attribute set is all attributes functionally determined by it. Used to identify candidate keys and check normalization requirements."

Q73: "What is the difference between functional dependency and multivalued dependency in database normalization?"
Ans: "Functional dependency: B depends on A (A → B). Multivalued dependency: A determines multiple independent values of B and C (A →→ B, A →→ C). 4NF removes multivalued dependencies."

Q74: "What is Armstrong Axiom? What is ACID property?"
Ans: "Armstrong Axioms: rules (reflexivity, augmentation, transitivity) to infer all functional dependencies. ACID properties: Atomicity, Consistency, Isolation, Durability; ensure reliable transaction processing."

Q75: "What is transitive dependency?"
Ans: "Transitive dependency occurs when a non-key attribute depends on another non-key attribute, which depends on the primary key (A → B → C). Removed in 3NF."