1. What do you understand By Database.

- Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data.
- **Tables:**
- Also known as relations, these are the basic storage units of a database. Each table consists of rows and columns, where each row represents a single record, and each column represents a field or attribute of that record.
- **Schemas:**
- The plan or formulation of database is known as schema.
- Schema gives the names of the entities and attributes. It specifies relationship among them.
- Schema includes the definition of database name, record type, components that make up records.
- Schema can be categorized in two ways:
- Logical schema (Table, View)
- Physical Schema (Secondary storage)
- **Records:**
- A record is a collection of logically related fields. For examples, collection of fields (id, name, society & city) forms a record for customer.
- **Fields:**
- A group of related records. Files are frequently classified by the application for which they are primarily used (employee file).
- **Primary Key:**
- A primary key is a unique identifier for each record in a table, ensuring that no two records have the same value.
- **Foreign Key:**
- A foreign key is a field in a table that refers to the primary key of another table, establishing relationships between tables.
- **Unique:**
- A column must have unique values. This is required to identify all records stored in table uniqly, A column, defined as a UNIQUE, cannot have duplicate values across all records. In other words, such column must contain unique values.

2. What is Normalization?

- Definition: In relational database design, the process of organizing data to minimize redundancy. Normalization usually involves dividing a database into two or more tables without losing information and defining relationships between the tables.
- **There are three types:**
  1. **First Normal Form (1NF)**
  2. **Second Normal Form (2NF)**
  3. **Third Normal Form (3NF)**

❖ **First Normal From(1NF)**

- A relation R is in first normal form (1NF) if and only if all domains contain atomic values only.
- • A relation R is in first normal form (1NF) if and only if it does not contain any composite or multi valued attributes or their combinations.

❖ **Second Normal Form (2NF)**

- A relation R is in second normal form (2NF) if and only if it is in 1NF and every nonprime attribute of relation is fully dependent on the primary key.

❖ **Third Normal Form (3NF)**

- A relation R is in third normal form (3NF) if and only if it is in 2NF and no any nonprime attribute of a relation is transitively dependent on the primary key.

3. What is Difference between DBMS and RDBMS?

DBMS	RDBMS
Data stored is in the file format	Data stored is in table format
Individual access of data elements	Multiple data elements are accessible together
No connection between data	Data in the form of a table are linked together
There is normalisation	Normalisation is not achievable
No support for distributed database	Support distributed database
Data stored is a small quantity	Data is stored in a large amount
Data redundancy is common	Here, redundancy of data is reduced with the help of key and indexes in RDBMS
DBMS supports a single user	RDBMS supports multiple users
There is only low security while handling data	It features multiple layers of security while handling data
The software and hardware requirements are low	The software and hardware requirements are highe
XML, Microsoft Access.	Oracle, SQL Server

4. What is MF Cod Rule of RDBMS Systems?

- **Rule 0: The Foundation Rule**
- The DB must be structured in a relational manner so that the system's relational capabilities can manage the DB.
- **Rule 1: The Information Rule**
- A DB comprises a variety of data, which must be recorded in the form of columns and rows in each and every cell of a table.

- **Rule 2: The Guaranteed Access Rule**

- A relational DB's primary key value, column name, and table name can be used to conceptually retrieve any single or precise data (the atomic value).

- **Rule 3: The Systematic Treatment of Null Values**

- The treatment of Null values in DB records is defined by this rule. No value in a cell, missing data, unsuitable information, unknown data, the primary key that should not be null, etc., are all examples of null values in DBs.

- **Rule 4: The Dynamic/Active Online Catalog on the basis of the Relational Model**

- A DB dictionary is a logical representation of the whole logical structure of a descriptive DB that needs to be stored online. It grants users access to the DB and uses a query language that is comparable to that of the DB.

- **Rule 5: The Comprehensive Data SubLanguage Rule**

- The relational DB supports a variety of languages, and in order to access the DB, the language has to be linear, explicit, or a well-defined syntax, character strings. It must support the following operations: view definition, integrity constraints, data manipulation, data definition, as well as limit transaction management. It is considered a DB violation if the DB permits access to the data and information without the use of any language.

- **Rule 6: The View Updating Rule**

- A view table can theoretically be updated, and DB systems must update them in practice.

- **Rule 7: The Relational Level Operation (or High-Level Insert, Delete, and Update) Rule**
- In each level or single row, a DB system should adhere to high-level relational operations (for example, update, insert, and delete). The DB system also includes operations like intersection, union, and minus.

- **Rule 8: The Physical Data Independence Rule**
- To access a DB or an application, all stored data must be independent physically. Each piece of data should not be reliant on another piece of data or an application. External applications that access data from the DB will have no effect if data is altered or the physical structure of a given DB is modified.

- **Rule 9: The Logical Data Independence Rule**
- It's similar to the independence of physical data. It indicates that any modifications made at the logical level (or the table structures) should not have an impact on the user's experience (application). For example, if a table is split into two separate tables or into two table joins in order to produce a single table, the application at the user view should not be affected.

- **Rule 10: The Integrity Independence Rule**
- When we are using SQL to put data into table cells, a DB must guarantee integrity independence. All the entered values must not be changed, and the integrity of the data should not be reliant on any external component or application. It's also useful for making each front-end app DB-independent.

- **Rule 11: The Distribution Independence Rule**
- This rule denotes that a DB must function properly even if it's stored in multiple locations and used by various end-users. Let's say a person uses an application to access the DB. In such a case, they must not be aware that another user is using the same data, and thus, the data they always obtain is only available on one site. The DB can be accessed by end-users, and each user's access data must be independent in order for them to run SQL queries.

- **Rule 12: The Non-Subversion Rule**
- RDBMS is defined by this rule as a SQL language for storing and manipulating data in a DB. If a system uses a low-level or different language to access the DB system other than SQL, it should not bypass or subvert data integrity.

5. What do you understand By Data Redundancy?

- Data redundancy is a situation where data is stored in multiple locations, making it easier to recover in case
- of data loss or corruption.
- What are the benefits of data redundancy?
- A→ Improved data availability: With data redundancy, you can access your data even if one or more storage devices fail.
- B→ Enhanced data security: Redundant data can be used to detect and correct errors
- C→Simplified data recovery: In case of data loss or corruption, you can easily recover your data from the redundant copies.
- What are the types of data redundancy?
- A→Data Duplication: This occurs when the same data is stored in multiple locations, such as when a customer's name and address are stored in both the sales and marketing databases.
- B→Data Inconsistency: This occurs when different versions of the same data exist in different locations, such as when a customer's address is updated in one database but not in another.
- C→ Data Replication: This occurs when data is intentionally duplicated in multiple locations to improve data availability and performance, such as when data is replicated across multiple servers in a distributed database.
- D→ Data Redundancy in Data Storage: This occurs when data is stored in a way that is redundant or unnecessary, such as when a database stores both a person's age and their date of birth.
- E→ Data Redundancy in Data Transmission: This occurs when data is transmitted multiple times over a network, such as when a file is sent multiple times over the internet.

6. What is DDL Interpreter?

- **DDL (Data Definition Language)**
- It is a set of SQL commands used to create, modify and delete database objects such as tables, views, indices, etc.
- It is normally used by DBA and database designers.
- It provides commands like:
- **CREATE:** to create objects in a database.
- **ALTER:** to alter the schema, or logical structure, of the database.
- **DROP:** to delete objects from the database.
- **TRUNCATE:** to remove all records from the table.

7. What is DML Compiler in SQL?

- DML Compiler is a part of SQL Server that compiles DML statements (INSERT, UPDATE, DELETE) into an execution plan. It is responsible for optimizing the query and generating the
- execution plan. The DML Compiler is a critical component of the SQL Server query optimizer, and it plays a key role in ensuring that queries are executed efficiently.

8. What is SQL Key Constraints writing an Example of SQL Key Constraints?

- SQL Key Constraints are used to maintain the integrity of data in a database. They ensure that data in a table is unique and consistent. There are three types of key constraints:
- A→Primary Key: A primary key is a unique identifier for each row in a table. It cannot be null and must be unique.
- B→Foreign Key: A foreign key is a field in a table that refers to the primary key in another table. It ensures data consistency between two tables.
- C→ Unique Key: A unique key is a field or combination of fields that must be unique in a table. It cannot be null and must be unique.
- D→NOT NULL - Ensures that a column cannot have a NULL value

9. What is save Point? How to create a save Point write a Query?

- Save point is a point in a transaction where you can roll back to that point to create a save point.
- Syntax for Save point: -
- SAVEPOINT SAVEPOINT_NAME;
- This command is used only in the creation of SAVEPOINT among all the transactions.
- In general ROLLBACK is used to undo a group of transactions.
- Syntax for rolling back to Save point: -
- ROLLBACK TO SAVEPOINT_NAME;
- you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.
- Example: -
- From the above example Sample table1, Delete those records from the table
- which have age = 20 and then ROLLBACK the changes in the database by keeping Save points.
- Query
- SAVEPOINT SP1;
  //Save point created.
- DELETE FROM Student WHERE AGE = 20;
  //deleted
- SAVEPOINT SP2;
  //Save point created.
- Here SP1 is first SAVEPOINT created before deletion. In this example one deletion have taken place.
- After deletion again SAVEPOINT SP2 is created.

10. What is trigger and how to create a Trigger in SQL?

- A trigger in SQL is a special kind of stored procedure that automatically executes in response to certain events on a particular table or view. Triggers can be used to enforce business rules, validate data, maintain audit trails, and automate system tasks.
- In this article, you will learn about the trigger and its implementation with examples.
- A Trigger in Structured Query Language is a set of procedural statements which are executed automatically when there is any response to certain events on the particular table in the database.
- Triggers are used to protect the data integrity in the database.
- In SQL, this concept is the same as the trigger in real life. For example, when we pull the gun trigger, the bullet is fired.
- To understand the concept of trigger in SQL, let's take the below hypothetical situation: Suppose Rishabh is the human resource manager in a multinational company.
- When the record of a new employee is entered into the database, he has to send the 'Congrats' message to each new employee.
- If there are four or five employees, Rishabh can do it manually, but if the number of new Employees is more than the thousand, then in such condition, he has to use the trigger in the database.
- Thus, now Rishabh has to create the trigger in the table, which will automatically send a 'Congrats' message to the new employees once their record is inserted into the database.
- The trigger is always executed with the specific table in the database. If we remove the table, all the triggers associated with that table are also deleted automatically.
- In Structured Query Language, triggers are called only either before or after the below events: -
- INSERT Event: This event is called when the new row is entered in the table.
- UPDATE Event: This event is called when the existing record is changed or modified in the table.
- DELETE Event: This event is called when the existing record is removed from the table.
- Types of Triggers in SQL: -

- Syntax of Trigger in SQL
  CREATE TRIGGER Trigger_Name
  [BEFORE | AFTER] [Insert | Update | Delete]
  ON [Table_Name]
  [FOR EACH ROW | FOR EACH COLUMN]
  AS
  Set of SQL Statement
- In the trigger syntax, firstly, we have to define the name of the trigger after the CREATE TRIGGER keyword.
- After that, we have to define the BEFORE or AFTER keyword with anyone event.