1. **What do you understand By Database?**

**Ans.** A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

1. **What is Normalization?**

**Ans. Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.

Normalization rules divides larger tables into smaller tables and links them using relationships.

The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

## **Database Normal Forms**

Here is a list of Normal Forms in SQL:

* 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)
* 6NF (Sixth Normal Form)

1. **What is Difference between DBMS and RDBMS?**

**Ans.** The following are the important differences between DBMS and RDBMS-

| **DBMS** | **RDBMS** |
| --- | --- |
| [DBMS](https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/) stores data as file. | [RDBMS](https://www.geeksforgeeks.org/rdbms-architecture/) stores data in tabular form. |
| Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| No relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present. | Normalization is present. |
| DBMS does not support distributed database. | RDBMS supports distributed database. |
| It stores data in either a navigational or hierarchical form. | It uses a tabular structure where the headers are the column names, and the rows contain corresponding values. |
| It deals with small quantity of data. | It deals with large amount of data. |
| Data redundancy is common in this model. | Keys and indexes do not allow Data redundancy. |
| It is used for small organization and deal with small data. | It is used to handle large amount of data. |
| Not all Codd rules are satisfied. | All 12 Codd rules are satisfied. |
| Security is less | More security measures provided. |
| It supports single user. | It supports multiple users. |
| Data fetching is slower for the large amount of data. | Data fetching is fast because of relational approach. |
| The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| Low software and hardware necessities. | Higher software and hardware necessities. |
| Examples:[XML](https://www.geeksforgeeks.org/xml-basics/), Window Registry, Forxpro, dbaseIIIplus etc. | Examples: [MySQL](https://www.geeksforgeeks.org/architecture-of-mysql/), [PostgreSQL](https://www.geeksforgeeks.org/what-is-postgresql-introduction/), SQL Server, Oracle, Microsoft Access etc. |

1. **What is MF cod Rule of RDBMS System?**

## **Ans.** MF cod rule of RDBMS system:-

## Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

## Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

## Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

## Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

## Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

## Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be up datable by the system.

## Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

## Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

## Rule 9: Logical Data Independence

The logical data in a database must be independent of its user’s view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

## Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

## Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

## Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

1. **What do you understand By Data Redundancy?**

**Ans.** Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables. Unfortunately, data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information.

1. **What is DDL Interpreter?**

**Ans.** DDL Interpreter interprets the DDL statements and records the generated statements in the table containing meta data.

* DDL Interpreter DDL expands to Data Definition Language. DDL Interpreter as the name suggests interprets the DDL statements such as schema definition statements like create, delete, etc. The result of this interpretation is a set of a table that contains the meta-data which is stored in the data dictionary.

1. **What is DML Compiler in SQL?**

**Ans.** A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database. A DML is typically a sub-language of a larger database language like SQL, with the DML containing some of the language's operators.

1. **What is SQL Key Constraints writing an example of SQL Key Constraints?**

**Ans.** SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

* The following constraints are commonly used in SQL:
* [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value.

Example:-

CREATE TABLE Colleges (

college\_id INT NOT NULL,

college\_code VARCHAR(20) NOT NULL,

college\_name VARCHAR(50)

);

* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different.

Example:-

CREATE TABLE Colleges (

college\_id INT NOT NULL UNIQUE,

college\_code VARCHAR(20) UNIQUE,

college\_name VARCHAR(50)

);

* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.

Example:-

CREATE TABLE Colleges (

college\_id INT PRIMARY KEY,

college\_code VARCHAR(20) NOT NULL,

college\_name VARCHAR(50)

);

* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables.

Example:-

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id int REFERENCES Customers(id)

);

* [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition.
* Example:-

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

amount int CHECK (amount >= 100)

);

* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified.
* Example:-

CREATE TABLE College (

college\_id INT PRIMARY KEY,

college\_code VARCHAR(20),

college\_country VARCHAR(20) DEFAULT 'US'

);

* [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly.

Example:-

-- create tableCREATE TABLE Colleges (

college\_id INT PRIMARY KEY,

college\_code VARCHAR(20) NOT NULL,

college\_name VARCHAR(50)

);

-- create indexCREATE INDEX college\_indexON Colleges(college\_code);

1. **What is save point? How to create a save point write a Query?**

**Ans.** A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

The syntax for a SAVEPOINT command is as shown below. :-SAVEPOINT SAVEPOINT\_NAME; This command serves only in the creation of a SAVEPOINT among all the transactional statements.

* **Creating Save points: Example:-**

To update Band's and Green's salary in the sample table hr.employees, check that the total department salary does not exceed 314,000, then reenter Green's salary,

Enter:-

UPDATE employees

SET salary = 7000

WHERE last\_name = 'Band;

SAVEPOINT band\_Sal;

UPDATE employees

SET salary = 12000

WHERE last\_name = 'Greene';

SAVEPOINT green\_Sal;

SELECT SUM(salary) FROM employees;

ROLLBACK TO SAVEPOINT band\_Sal;

UPDATE employees

SET salary = 11000

WHERE last\_name = 'Greene';

COMMIT;

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

**Ans.** A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

* **Syntax:-**

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

[trigger\_body]

**Task:-**

1. **. Create Table Name : Student and Exam**

|  |  |  |
| --- | --- | --- |
| **Roll no.** | **Name** | **Branch** |
| 1 | Jay | Computer Science |
| 2 | Suhani | Electronic and Com |
| 3 | Kriti | Electronic and Com |

**Primary key**

**Foreign key**

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll no.** | **S\_code** | **Marks** | **P\_code** |
| 1 | CS11 | 50 | CS |
| 1 | CS12 | 60 | CS |
| 2 | EC101 | 66 | EC |
| 2 | EC102 | 70 | EC |
| 3 | EC101 | 45 | EC |
| 3 | EC102 | 50 | EC |

**Query:-**

Create Database School;

Create Table Student(Roll\_no int Auto\_increment,

Name varchar(100),

Branch varchar(100),

Primary key(Roll\_no)

);

Create Table Exam(Roll\_no int Auto\_increment,

S\_code varchar(100),

Marks int,

P\_code varchar(100),

Foreign key(Roll\_no)References

Student(Roll\_no)

);

Insert into Student(Roll\_no,Name,Branch)

Values(1,”Jay”,”Computer Science”),

(2,”Suhani”,”Electronic and Com”),

(3,”Kriti”,”Electronic and Com”);

Insert into Exam(Roll\_no,S\_code,Marks,P\_code)

Values(1,”CS11”,50,”CS”),

(1,”CS12”,60,”CS”),

(2,”EC101”,66,”EC”),

(2,”EC102”,70,”EC”),

(3,”EC101”,45,”EC”),

(3,”EC102”,50,”EC”);

1. **. Create Table given below**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First Name** | **Last Name** | **Address** | **City** | **Age** |
| Mickey | Mouse | 123 Fantasy Way | Anaheim | 73 |
| Bat | Man | 321 Cavern Ave | Gotham | 54 |
| Wonder | Woman | 987 Truth Way | Paradise | 39 |
| Donald | Duck | 555 Quack Street | Mallard | 65 |
| Bugs | Bunny | 567 Carrot Street | Rascal | 58 |
| Wiley | Coyote | 999 Acme Way | Canyon | 61 |
| Cat | Woman | 234 Purrfect Street | Hairball | 35 |
| Tweety | Bird | 543 | Itotltaw | 28 |

**Query:-**

Create Database Person;

Create Table Biodata(First\_Name varchar(100),

Last\_Name varchar(100),

Address varchar(100),

City varchar(100),

Age int );

INSERT INTO biodata(First\_Name,Last\_Name,Address,City,Age)

VALUES("Mickey","Mouse","123 Fantasy way","Anaheim",73),

("Bat","Man","321 Cavern Ave","Gotham",54),

("Wonder","Woman","987 Truth Way","Paradise",39),

("Donald","Duck","555 Quack Street","Mallard",65),

("Bugs","Bunny","567 Carrot street","Rascal",58),

("Wiley","Coyote","999 Acme Way","Canyon",61),

("Cat","Woman","234 Purrfect Street","Hairball",32),

("Tweety","Bird","543","Itotlaw",28);

1. **. Create Table given below: Employee and Incentive**

**Table Name: Employee**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Employee\_id** | **First\_Name** | **Last\_Name** | **Salary** | **Joining\_date** | **Department** |
| 1 | John | Abraham | 1000000 | 01-JAN-13  12.00.00  AM | Banking |
| 2 | Michael | Clarke | 800000 | 01-JAN-13  12.00.00  AM | Insurance |
| 3 | Roy | Thomas | 700000 | 01-FEB-13  12.00.00  AM | Banking |
| 4 | Tom | Jose | 600000 | 01-FEB-13  12.00.00  AM | Insurance |
| 5 | Jerry | Pinto | 650000 | 01-FEB-13  12.00.00  AM | Insurance |
| 6 | Philip | Mathew | 750000 | 01-JAN-13  12.00.00  AM | Services |
| 7 | TestName1 | 123 | 650000 | 01-JAN-13  12.00.00  AM | Services |
| 8 | TestName2 | Lname% | 600000 | 01-FEB-13  12.00.00  AM | Insurance |

**Table Name: Incentive**

|  |  |  |
| --- | --- | --- |
| **Employee\_ref\_id** | **Incentive\_date** | **Incentive\_amount** |
| 1 | 01-FEB-13 | 5000 |
| 2 | 01-FEB-13 | 3000 |
| 3 | 01-FEB-13 | 4000 |
| 1 | 01-JAN-13 | 4500 |
| 2 | 01-JAN-13 | 3500 |

**Query:-**

Create Database Company;

Create Table Employee(Employee\_id int AUTO\_INCREMENT,

First\_Name varchar(100),

Last\_Name varchar(100),

Salary int,

Joining\_date varchar(100),

Department varchar(100),

PRIMARY KEY(Employee\_id)

);

Insert Into employee(Employee\_id,First\_Name,Last\_Name,Salary,Joining\_date,Department)

VALUES(1,"John","Abraham",1000000,"01-JAN-13 12.00.00 AM","Banking"),

(2,"Michael","Clarke",800000,"01-JAN-13 12.00.00 AM","Insurance"),

(3,"Roy","Thomas",700000,"01-FEB-13 12.00.00 AM","Banking"),

(4,"Tom","Jose",600000,"01-FEB-13 12.00.00 AM","Insurance"),

(5,"Jerry","Pinto",650000,"01-FEB-13 12.00.00 AM","Insurance"),

(6,"Philip","Mathew",750000,"01-JAN-13 12.00.00 AM","Services"),

(7,"TestName1","123",650000,"01-JAN-13 12.00.00 AM","Services"),

(8,"TestName2","Lname%",600000,"01-FEB-13 12.00.00 AM","Insurance");

CREATE TABLE Incentive (Employee\_ref\_id INT,

Incentive\_date DATE,

Incentive\_amount INT,

FOREIGN KEY (Employee\_ref\_id)

REFERENCES Employee(Employee\_id)

);

Insert Into incentive(Employee\_ref\_id,Incentive\_date,Incentive\_amount)

VALUES(1,"01-FEB-13",5000),

(2,"01-FEB-13",3000),

(3,"01-FEB-13",4000),

(1,"01-JAN-13",4500),

(2,"01-JAN-13",3500);

1. **Get First\_Name from employee table using Tom name “Employee**

**Name”.**

**Query:-** SELECT First\_Name

FROM Employee

WHERE First\_Name = 'Tom';

1. **Get FIRST\_NAME, Joining Date, and Salary from employee table.**

**Query:-** SELECT First\_Name, Joining\_Date, Salary

FROM Employee;

1. **Get all employee details from the employee table order by First\_Name**

**Ascending and Salary descending?**

**Query:-** SELECT \*

FROM Employee

ORDER BY First\_Name ASC, Salary DESC;

1. **Get employee details from employee table whose first name contains”J”.**

**Query:-** SELECT \*

FROM Employee

WHERE First\_Name LIKE '%J%';

1. **Get department wise maximum salary from employee table order by**

**salary ascending?**

**Query:-** SELECT Department, MAX(Salary) AS Max\_Salary

FROM Employee

GROUP BY Department

ORDER BY Max\_Salary ASC;

1. **Select First\_Name, incentive amount from employee and incentives table**

**For those employees who have incentives amount greater than 3000.**

**Query:-**  SELECT Department, MAX(Salary) AS Max\_Salary

FROM Employee

GROUP BY Department

ORDER BY Max\_Salary ASC;

1. **Create After Insert trigger on employee table which insert records in**

**View table.**

**Query:-** SELECT First\_Name, Incentive\_Amount

FROM Employee e

JOIN Incentive ON Employee\_ID = Employee\_Ref\_ID

WHERE Incentive\_Amount > 3000;

1. **. Create Table given below: Salesperson and Customer**

**Table:-1**

**TABLE NAME-SALESPERSON**

|  |  |  |  |
| --- | --- | --- | --- |
| **(PK)S No** | **SNAME** | **CITY** | **COMM** |
| 1001 | Peel | London | .12 |
| 1002 | Serres | San Jose | .13 |
| 1004 | Motika | London | .11 |
| 1007 | Rafkin | Barcelona | .15 |
| 1003 | Axelrod | New York | .1 |

**Table:-2**

**TABLE NAME-CUSTOMER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **(PK)CNM.** | **CNAME** | **CITY** | **RATING** | **(FK)S No** |
| 201 | Hoffman | London | 100 | 1001 |
| 202 | Giovanne | Roe | 200 | 1003 |
| 203 | Liu | San Jose | 300 | 1002 |
| 204 | Grass | Barcelona | 100 | 1002 |
| 206 | Clemens | London | 300 | 1007 |
| 207 | Pereira | Roe | 100 | 1004 |

**Query:-**

CREATE TABLE Salesperson (SNo INT AUTO\_INCREMENT,

SName VARCHAR(50),

City VARCHAR(50),

Comm int,

PRIMARY KEY (SNo)

);

INSERT INTO Salesperson (SNo, SName, City, Comm)

VALUES (1001, 'Peel', 'London', 0.12),

(1002, 'Serres', 'San Jose', 0.13),

(1004, 'Motika', 'London', 0.11),

(1007, 'Rafkin', 'Barcelona', 0.15),

(1003, 'Axelrod', 'New York', 0.1);

CREATE TABLE Customer (CNM INT AUTO\_INCREMENT,

CName VARCHAR(50),

City VARCHAR(50),

Rating INT,

SNo INT,

PRIMARY KEY(CNM),

FOREIGN KEY (SNo) REFERENCES Salesperson(SNo)

);

INSERT INTO Customer (CNM, CName, City, Rating, SNo)

VALUES (201, 'Hoffman', 'London', 100, 1001),

(202, 'Giovanne', 'Roe', 200, 1003),

(203, 'Liu', 'San Jose', 300, 1002),

(204, 'Grass', 'Barcelona', 100, 1002),

(206, 'Clemens', 'London', 300, 1007),

(207, 'Pereira', 'Roe', 100, 1004);

**Retrieve the below data from above table:-**

1. **All orders for more than $1000.**

**Query:-**  SELECT \*

FROM Customer

WHERE Rating > 1000;

1. **Names and cities of all salespeople in London with commission above 0.12.**

**Query:-**  SELECT SName, City

FROM Salesperson

WHERE City = 'London' AND Comm > 0.12;

1. **All salespersons either in Barcelona or in London:**

**Query:-** SELECT \*

FROM Salesperson

WHERE Comm > 0.10 AND Comm < 0.12;

1. **All salespersons with commission between 0.10 and 0.12 (boundary values excluded):**

**Query:-** SELECT \*

FROM Salesperson

WHERE Comm > 0.10 AND Comm < 0.12;

1. **All customers excluding those with a rating <= 100 unless they are located in Rome:**

**Query:-** SELECT \*

FROM Customer

WHERE Rating > 100 OR City = 'Rome';