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 purposes of normalization in SQL are to eliminate redundant (repetitive) data and ensure data is stored logically.

1. What is difference between DBMS and RDBMS?

Ans:

DBMS:

DBMS does not support distributed database.

DBMS stored data as file.

Normalization is not present.

It stores data in either a navigational or hierarchical form.

No relationship between data.

Examples: XML, Window Registry, etc.

RDBMS:

RDBMS supports distributed database.

RDBMS stores data in tabular form.

Normalization is present.

It uses a tabular, structures where the headers are the column names and the rows contain corresponding values.

Data is stored in the form of tables which are related to each other.

1. What is MF Cod Rule of RDBMS Systems?

Ans:

Rule 1: Information Rule: The data stored in a database may it be user data or metadata must be a value if 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 updatable 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 rules 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 means having multiple copies of same data in the database. This problem arises when a database is not normalized.

1. What is DDL Interpreter?

Ans: It processes the DDL statements into a set of tables containing meta data.

1. What is DML Compiler in SQL?

Ans: It processes the DML statements into low level instruction (machine language). So that they can be executed.

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

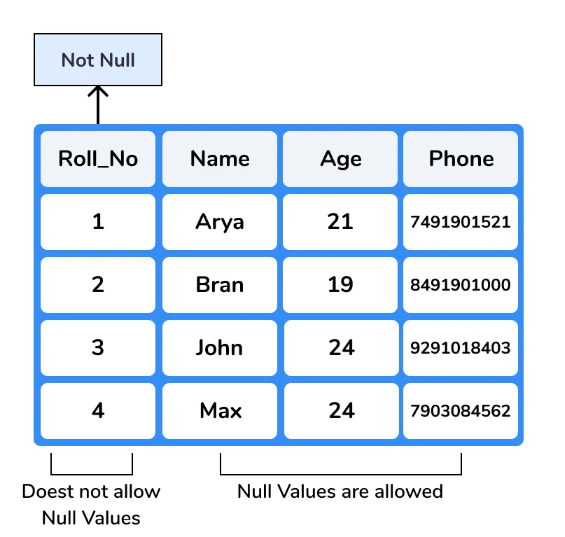
Ans: Constraints in SQL means we are applying certain conditions or restrictions on the database.

### **Constraints available in SQL are:**

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY

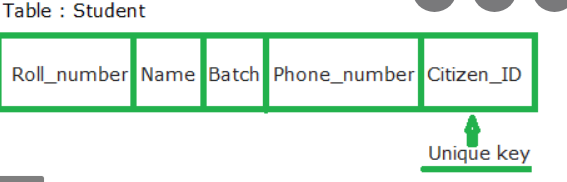
### **NOT NULL**

* NULL means empty, i.e., the value is not available.
* Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.
* There must exist a value in the column to which the NOT NULL constraint is applied.



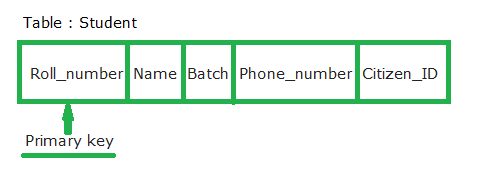
### **UNIQUE**

* Duplicate values are not allowed in the columns to which the UNIQUE constraint is applied.
* The column with the unique constraint will always contain a unique value.
* This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.
* Using the UNIQUE constraint, you can also modify the already created tables.



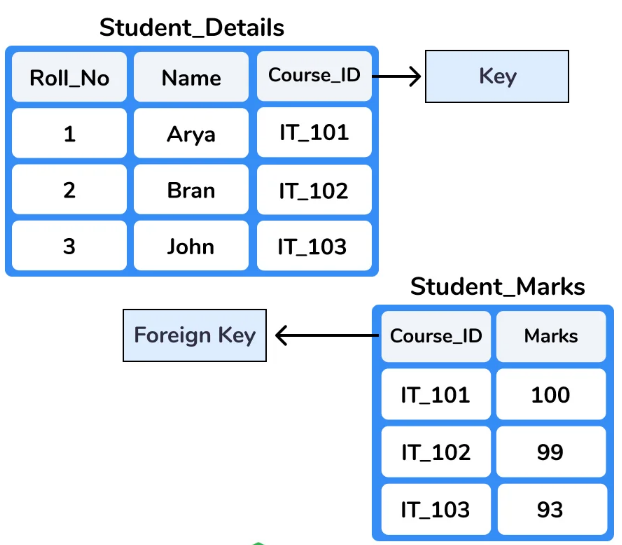
### **PRIMARY KEY**

* PRIMARY KEY Constraint is a combination of NOT NULL and Unique constraints.
* NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint.
* The column to which we have applied the primary constraint will always contain a unique value and will not allow null values.



### **FOREIGN KEY**

* A foreign key is used for referential integrity.
* When we have two tables, and one table takes reference from another table, i.e., the same column is present in both the tables and that column acts as a primary key in one table. That particular column will act as a foreign key in another table.



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

Ans: Savepoint is a command in SQL that is used with the rollback command.

It is a command in transaction control language that is used to mark the transaction in a table.

Syntax for savepoint:

SAVEPOINT SAVEPOINT\_NAME;

Queries:

SAVEPOINT SP1;

// savepoint is created.

DELETE FROM STUDENT WHERE AGE=20;

// deleted.

SAVEPOINT SP2;

// savepoint created.

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

Ans: Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs.

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

Ans: create table student

(rollno int primary key,

name varchar (255),

branch varchar (255));

insert into student (rollno, name, branch) values

(1, ”jay”, ”computer science”),

(2, ”suhani”, ”electronic and comp”),

(3, ”kriti”, ”electronic and comp”);

create table exam

(roll\_no int,

foreign key(roll\_no) references student(rollno),

s\_code varchar(255),

marks int,

p\_code varchar(255));

insert into exam

(roll\_no,s\_code,marks,p\_code)values

(1,"cs12",60,"cs"),

(2,"ec101",66,"ec"),

(2,"cs102",70,"ec"),

(3,"ec101",45,"ec"),

(3,"ec102",50,"ec");

1. Create Table Name: Student and Exam

Ans: create table info

(first\_name varchar(255),

lasr\_name varchar(255),

address varchar(255),

city varchar(255),

age int);

insert into info

(first\_name,lasr\_name,address,city,age) values

("mickey","mouse","123 fantasy way","anaheim",73),

("bat","man","321 cavem ave","gotham",54),

("wonder","women","987Truth 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","women","234 purefect street","hairball",32),

("tweety","bird","543","itotiaw",28);

1. Create table given below: Employee and Incentive

Ans: Table name: Employee:-

create table employee

(employee\_id int primary key,

first\_name varchar(255),

last\_name varchar(255),

salary bigint,

joining\_date timestamp,

department varchar(255));

insert into employee

(employee\_id,first\_name,last\_name,salary,joining\_date,department) values

(1,"john","abraham",1000000,"2013-01-01 12:00:00","Banking"),

(2,"michael","clarke",800000,"2013-01-01 12:00:00","insurance"),

(3,"roy","thomas",700000,"2013-01-01 12:00:00","Banking"),

(4,"tom","jose",600000,"2013-01-01 12:00:00","insurance"),

(5,"jerry","pinto",650000,"2013-01-01 12:00:00","insurance"),

(6,"philip","mathew",750000,"2013-01-01 12:00:00","services"),

(7,"testname","123",650000,"2013-01-01 12:00:00","services"),

(8,"testname2","lname",600000,"2013-01-01 12:00:00","insurance");

Table name: Incentive

create table incentive

(employee\_ref\_id int references employee(employee\_id),

incentive\_date date,

incentive\_amount int);

insert into incentive

(employee\_ref\_id,incentive\_date,incentive\_amount)values

(1,"2013-02-01",5000),

(2,"2013-02-01",3000),

(3,"2013-02-01",4000),

(1,"2013-02-01",4500),

(2,"2013-02-01",3500);

1. Get First Name From employee table

Ans: select First\_ name from employee:

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

Ans: select first \_ name, Joining \_ date, salary from employee;

1. Get all employee details from the table order by First \_ Name Ascending and Salary descending?

Ans: select \*from employee order by First \_ name asc;

Select \* from employee order by salary desc;

1. Get employee details from employee table whose first name contains ‘J’.

Ans: select \*from employee where First \_ name like ‘%j%’;

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

Ans: select Department, max [salary] from employee group by Department order by salary asc;

1. Select first \_name, incentive amount from employee and incentive table for those employees who Have incentives and incentive amount greater than 3000.

Ans: select employee. First \_ name, incentive. incentive \_ amount from employee inner join incentive on employee. Employee\_ id= incentive. Incentive \_ amount and incentive \_ amount > 3000;

1. create After Insert trigger on Employee table which insert records in view table

Ans: create trigger tr

After insert

On employee

For each row

Insert into view

[e\_id, f\_name, l\_name,salary, j\_date, department) values

(employee\_id, first\_name, last\_name, salary\_ joining\_date, department);

1. Create table given below: salesperson and customer

Ans: table name: salesperson

Create table salesperson

(sno int primary key,

Sname varchar(255),

City varchar(255),

Comm varchar(255));

Insert into salesperson

(sno, sname, city, comm) values

(1001,”peel”,”London”,”0.12”),

(1002,”serresl”,”sanjose”,”0.13”),

(1004,”motika”,”London”,”0.11”),

(1007,”rafkinl”,”barcelona”,”0.15”),

(1003,”axelrod”,”new york”,”0.1”),

Table name: customer

Create table customer

(CNM it primary key,

CNAME varchar(255),

City varchar(255),

Rating int,

Sno int references salesperson(sno));

Insert into customer

(CNM, CNAME, city, rating, sno) values

(201,”hoffman”,”London”,100,1001),

(202,”giovance”,”roe”,200,1003),

(203,”ilu”,”sanjose”,300,1002),

(204,”grass”,”barcelona”,100,1002),

(206,”celemens”,”London”,300,1007),

(207,”perria”,”roe”,100,1004);

1. All orders for more than sno 1003.

Ans: select \* from customer where sno>1003;

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

Ans: select sname.city from salesperson where city=”London and comm>0.11;

1. All salespeople either in Barcelona or in London

Ans: select \* from salesperson where city=”London” or city=”Barcelona”;

1. All salespeople with commission between 0.10 and 0.12 (Boundary values should be excluded).

Ans: select \* from salesperson where comm>0.1 and comm<0.12;

1. All customer excluding those with rating <=100 unless they are located in Rome.

Ans: select \* from customer where rating<=100 and city=”roe”;