Joins :-

1) Inner Join
2) Left Join
3) Right Join
4) Full Join
5) Natural Join
6) Cross Join
7) self join

→ Consider we have two tables.

1. table1

table1
1
1
1
2
3
3
3

2. table 2

table 2
1
1
2
2
4
Null

Inner

Right join

Cross

no. records

Inner Join	Left Join	Right Join	Full Join	Natural Join	Cross Join
8	8+3=11	8+2=10	8+3+2 =13	8 (if exist com (or) column. mm) 42 (column not exist in other table	42

Inner Join :- It will written record matches with both tables only.

Left Join :- [Inner join] + [fetch any additional records in left table but not in right table].

Right join : [Inner join] + [fetch any additional reco in right table but not in left table]

Full Join:- Inner Join +

+ fetch additional record from left table not present in right table
+ fetch additional record from right table not present in left table.

Cross Join:- Every record from matc table 1. is matches or not matches but it written the all record which is multiples.

t2 = 6 record
t1 = 7 records

{ each row is matches with t2 : 6 records then for 7 record in t1

then

= 7 × 6
= 42

Interview

Natural Joins: [not support ~~MYSQL~~ Microsoft sesver SQL] but support by other support

→ we are not using on do join tables.
→ Instead it will try to matches with the common column names.

Result
→ Inner join (if the same column exist in the both table)
→ Cross join (if the same column not exist in both tables)
→ If you change one 7 column from t2 like as t1 then (alter table t1 rename column id to id-new, it written 42 record (cross join)

```
Select *
From table1   t1
Natural join table2 t2;
```

Normalization:-

→ It is a process of designing database effectively such that we can avoid data redundancy. this will help us to avoid anomalies with normalised dataset

data redundancy – data duplication

Ex: data 2-3 column is repeated with same data out 1000 records [duplicate]

Different level of Normalization:- [In order to normalize data]

each level has certain rules

1) 1NF
2) 2NF
3) 3NF
4) 4NF
⋮

} - Normal Form

most of company try to do normalization till level 3 3NF for database.

1. **1NF:-** 1) every column / attribute need to have single value

2) Each row should be unique, either through a single or multiple columns. not mandatory to have primary key.

2. **2NF:-**

1. Mst be in 1NF
2. All non-key attributes must be fully dependent on candidate key.

ex: If a non-key column is partially dependent on candidate key (subset of columns forming candidate key) then split them into seperate tables.

3. every table should have primary key & relationship b/w tables should be formed using foreign key.
4. if don't have PK then create

Candidate Key:-

→ Set of columns which uniquely identify a record.

→ A table can have multiple keys because there can be multiple set of columns which uniquely identify a record / row in a table.

Non-key columns:-

→ columns which are not part of candidate key (or) primary key.

partial dependency:-

→ If candidate key is a combination of 2 or more column then every non-key column (column which are not part of candidate key) should be fully dependent on all the columns.

→ If there is any non-key column which depends only on one of the candidate key columns then this results in partial dependency.

3. 3NF:-

→ It must in 2NF

→ Avoid Transitive dependencies. (dataset) - split them

↳

Transitive dependencies:-

→ Lets you have table T which has 3 columns namely A, B & C.

→ If A is functional dependent on B and B is functionally dependent on C then we can say that A is functionally dependent on C.

Natural Join:- Why we should not use NJ.

→ In Natural join, SQL will decide what is the join condition based on which column join would happen is decide by the SQL not by the user.

→ That can actually be a major problem when using natural join.

→ result looks like same as Inner Join, when the
→ columns that are sharing same name b/w 2 table.
otherwise it will try to do Cross Join.

Self Join :-

→ When join a table to itself.

→ you need match one record from my table with some other record of that same table to find my result then we use self join.

→ we use regular join we don't have separate keyword for self join.

Ex1:

	member_id	name	age	parent_id
① →	F1	David	4	F5
	F2	carol	10	F5
	F3	Michael	12	F5
	F4	johnson	36	
④ →	F5	Maryan	40	F6
② →	F6	Stwart	70	
	F7	Rohan	6	F4
	F8	Asha	8	F4

Query:- problem statement

↳ Write a query to fetch the child name and their age corresponding to their parent name name & parent_id.

```
Select child.name as child_name, child.age as child_age
parent.name = parent_name
parent.age = parent_age
From family as child
join family as parent on child.parent_id =
                         parent.member_id;
```

Outputs.

child_name	child_age	parent_name	parent_age
David	4	Maryam	40
Carol	10	Maryam	40
Michael	12	Maryam	40
Maryam	40	Stewart	70
Rohan	6	Johnson	36
Asha	8	Johnson	36

Suppose if you use left join then result would be records of child name even they don't have parent name. It showing null in the parent.name.