JOINS:

In the database, our data is divided into the tables and on the user interface we have to show the data as one table (called as view table) from multiple tables. This can be done with the help of join clause. For Example:

USERS

userId	name
123	Talha

COMMENTS

userId	comment
123	HyE

View will be generated using join clause:

name from user table comment from comment table

Talha	HyE

Inner Join: Suppose you have 2 tables, one is customers and second is card. When you generate new table using the inner join then it will give you those rows which have common in both tables. Means those customers which have card and those card which have customers and eliminate the others cards which dont have customer and those customer which dont have card. e.g

Select fn, ln, amtPaid from customers innerjoin card on customers.customerId = card.customerId
This will generate the rows which are common in both tables based on the customerId which PK and FK and give First name and last nmae from customer table and amoint paid from card table.

After Join

FN	LN	AMTPaid
Talha	Riaz	12300

Ok what if we have to apply the inner join on the more than two tables. Lets take example of three tables. Suppose we have a customer table, card table and lookup table whcih contain the specific types of cards. So how will it work? Ok First it will apply the 1st join condition between the customer and card table and generate the new one. And then apply the 2nd join condition on the new one table and the third one. That's how table will get smaller with more joins.

Customer

userId	FN	LN

Card

cardId	userId	amtPaid

Card Type

cardTypeId	cardId	type

NEW TABLE

FN	LN	amtPaid	cardType

First based on the userId join fill the condition and generate table without type, Then 2nd condition satisfied with third table and add type of card w.r.t to cardId.

Because its inner join so it will definatly exclude somethings while applying its conditions:
On first Condition:
1) Customers with no cards.
2) Cards with no customers.
On second Condition:
1) Cards with no cardType
2) CardType with no card

Lets make not null characteristic in the card tabel on the userId column. Now it will remove the Cards with no Customers in new table.
Lets make not null characterisic in the cardType table on the cardType. Now it will remove the cards with no card type in new table

NOTE: There is no any kind of relation between customer table and card type table but customer table has relation with card table and that card table has the relation with the card type table.

Lets have another example for better understanding:
User Table and the comment Table and the Video table.

User

userId	userName

Comment

commentId	userId	videoId	msg

Video

videoId	videoTitle

NOTE: Here user table has no any relationship with the video table. But throgh the comment table.

Join condition no 1: User.userId = Comment.userId
Join condition no 2: Comment.videoId = Video.videoId

Which column should have the not null condition?
In comment table userId and videoId not null. Because comment should have user and also a video on which comment is.

What will be exclude by the Inner Join condition?
User Table also have the users which do not comment but they are user so the inner join condition will exclude the those users which do not comment.
Video Table also have the videos on which nobody post a comment yet. so these videos also get exclude by the inner join condition.

So new Table will be:

userName	msg	video title

Query will be:
SELECT userName, video title, msg FROM User INNER JOIN COMMENT ON User.userId = Comment.userId INNER JOIN Video ON Video.videoId = Comment.videoId

OUTER JOINS:
1) Left joins => return all the rows from left table and matched (Acc to join condition) rows from the right table
2) Right => return all the rows from right table and matched (Acc to join condition) rows from the left table
3) Full => return matched values and unmatched values from either or both tables.
How will decide which table is left or right?
In query the first table name is actually the name of left table and most developer use left join and in case for right they swicth the table name to 1st. instead of using right join.
Select *From ..... Here is table name which is 1st
NOTE: With the not null characteristic the outer join purpose may be go down because then table must have associaton between each row.

Alias: is a nickname or fancy name of the table or columns that help to use SELECT easily and to present columns in good way. And we use as before giving alias.

Table Name: User as U and then we can write U.userId Column Name: First_Name as "First Name"
Query Like: SELECT email as contact, first_name as "First Name", last_name as "Last Name" FROM User

Self Join: The table need to join to itself. You can say we make the copu of table and then join it with original one. But this is just for the illustration, in actual there is only one table which is joining to itself.
For Example: We have table of enrolllment.

This is table and here we have given a query to solve which is:

student	course
S1	C1
S2	C2
S1	C2

Find student which is enroll in more than course.
So It is possible with other methods also but we are learning self join so we do with self join.
Here we will make alias of this Enrollment table so that we can treat one table as 2 and can perform join operaion on it.

Query:

```
SELECT student FROM enrollment as T1, enrollment as T2
WHERE T1.student = T2.student and T1.course <> T2.course
```

We make alias with name T1 and T2, and apply condition that student should be equal in both same and course should not equal to same. so it will show same student with different course enrollment.

That's how self join work by making alias