Relationships:
are nothing but relations between the tables in the database. We divide the big table into two parts and then they relate in some way. That's relation is known as relationship.

Types: One to One => One husband and one wife. One person has one CNIC
One to Many => teacher to students is example
Many to Many => Multiple classes has multiple students and multiple students has multiple classes. But this is not work in the relational databases.

One to One design: Whenever you find the one to one relationship then we can design that realtionship in table by 2 ways:
1) Make Attribute
For Example, you have the relationship of user and credit card. Suppose the company allows only one card to one user so this is one to one relation-ship. So we can make its attribute in the user table like:

userId	userName	cardNumber

Here we simply make the card number as attribute of user and this is frst method of desiging one to one.

2) If you have more details for the other entity for which you are going to make the attribute then use 2nd method which is to make 2nd table with the name of 2nd entity. Let's take above example and here we need card issue date and expiry date. For this, instead of making attributes for them also we should go with other table name Credit Card and give cardId to User Table and userId to Credit Card table.

userId	userName	cardId

cardId	cardNumber	cardIssue	cardExpiry	userId

NOTE: In this, we make the primary key of user table which is userId to foriegn key of credit card table and cardId to user table.

2) One to Many Design: Let's take above example but now company of credit card allows the user to have multiple cards. so this is reltionship of one to many. Its desgin going to very similar but in this we just make forign key of user table in card table. Because in this scnerio we have multiple cards and we can't given multiple cards id to user table. so instead we just give the user id to card table and then every card will show the id of its user which tells that this card belongs to this user and this other one also belong to the same user.

userId	userName

cardId	cardNumber	userId

Example: cardId 1 belongs to userId 1, cardId 24 also belong to userId 1. cardId 12 belongs to userId 47777.

Parent and Child: userId is primary key of user table which is going to card table and become foriegn key. So user is parent because it has primary key and card is child of user because forign key points to primary key of user table. Parent Child is not available in single table of one to one.

MANY to MANY Design: As we above discuessed that the design of many to many is complex in database. Why? Suppose we have class and students example. In this example, Class Table may have names of students which are enroll in this particular class. But what if someone drops then the space of that student become null. And Students table contain the name of student and the classes whcich he enrolled in it. Suppose one student is enrolled in all classes and other one is only enroll in one class then the second students other classes name attributes become null.

stuName	c1	c2	c3	c4	c5
Ali	enrolled	enrolled	enrolled	enrolled	enrolled
Talha	enrolled	nulll	nulll	null	null

So these null value is problem in many to many. Other problem is that both are parent at a time if we apply parent child rule in it.

But the solution is available for many to many. The solution is to break the many to many relationship into 1 to many and many to 1
Simply create intermediate Table which will contain the classId and studentId and shows the particular of id student is enrolled in this class id. This table is act like a child for both parents here like class and students table.

Class

classId	name
12	Math
20	Physics
30	Chem

Enroll (intermediary)

classId	studentId
30	3
12	1
20	2

Student

studentId	name
1	Talha
2	Ahmad
3	Ali

1 to many to 1

NOTE: Inntermediary shows that this student id has this class id. Here is ease now. If any student get drop then now the whole row regarding the student will delete. And here duplication is not allow.

So these all about the binary relationship... Binary means 2 and we are doing on two tables relations. Multiple table can have relation but that we do later.

KEYS: Keys are defined to speed up access to data and, in many cases, to create links between different tables.
=> Should be unique
=> Never Changing
=> Never Null

Primary key Index: In which if we want to search instead of searching each row one by one. But when we use index, we will order the data in a way that database knows how to get certain data:
Select * from where name = 'Talha'; // This will search one by one
But when we use index then we order data in such a way that database knows how to get certain data. so it will jump to T and fetch Talha.

LookUp table: This table is nothing but we make it for storing the set of values which are already defined and we have to use it again and again in our database. So, we make the sperate table for this and use them by using their Id. e.g: GYM has members and they can buy membership of gold, silver and platinum. So these are defined now, so we can make seperate table of it use forigen key to access them. If we do not do that then there will be problem of duplication and database get slow. e.g: Member Ali is gold and others members are also gold then gold is repeating. So this is consuming our database more. so we can simply use the ID of gold from the membership table where we defined all these things and can use them in member table.

Others example can be like: => States, Gender etc.

Super Key and Candiadate Key: The main keys are primary and the foreign keys. but when we go for high level database desgin then we use the concept of super key and candidate key also.

Candidate keys are those which are least to uniquly identify the rows. e.g; userId OR userName OR email etc. Now each of them are candidate keys and when we select the most uniquly identify attribute for our table then it is primary and we use it as forign and we the reaming attributes in candiate keys set are known as alternatives keys.

Super Keys: Combination of candidate keys are super keys like userId + userName + email, now its a super key becausse it is a set of all candidate keys.
userId => Candidate key
userName => Candidate key
email => Candiadate key

Set of each candidate key which is "userId + userName + email " is super key.

Selection of primary key:
Those candidate key which has following properties you can make one of them as primary key.
1) Unique
2) Never Changing
3) Never Null

Surrogate Key: The key which is gemerated when we insert the data which we want to insert in table. e.g; auto generated id by adding 1 is surrogate key whcih is after the data is inserted or which is not natural.

Natural key: Name shows that it is natural. like first name, last name. All the data which we already have.

Not Null: is the restriction that if someone dont have value than he will not be able to put information in this table where not null restriction is.

Diffrenece in primary and the foreign keys are:
Foriegn key can be null because sometimes there maybe no realtionship
Relationship maybe get change so maybe update because forign key refrences get change

OK so if you want to must have relationship or foreign key then you can use characterstic of not null which make it required.
NOTE: Terms name may change in otther DBMS but concept remain same.
Foriegn Key Constraints:
These are forign key constraints that refer to parent.
On Delete
On Update
These are forign key constraints that refer to child.
Restrict
Cascade
Set Null

Ok so we perform like on delete with restrict OR on update with cascade

On delete	Restrict	it will show error whenever try to delete primary key and keep PK remain same
On delete	Cascade	it will delete primary key from parent as well as child
On delete	Set Null	It will delete PK and make the child null, no any refernce
On Update	Restrict	If we change id 504 to 508 then it will come back to 504 because restrict
On update	Cascade	It change 504 to 508 in parent as well as in childs (NOTE: PK never changes but for security)
on update	set null	if we update from 504 to 508 then again children will set to null BUT BUT as we learn about not null characteristic so childern have this characterstic so we for this condition the opertion fails and value set back to 504 and give error.

Simple keys: are those which have one column or attribute e.g; username
Composite Keys: are combination of multiple attributes. e.g; username, address, userId
Compund Keys: Some persons say that composite keys are actually the compund keys but some say they are change so we are going to discuss defination of other persons who said compund keys and composite keys are different:

Compound Keys: Combination of keys (foriegn keys pointing to primary keys) is known as compund keys. e.g; userId, classId as in intermediary table
Composite keys: Combination of maybe keys but with atleast one normal attribute. e.g; userId, classId, name.

NOTE: Some persons use the surrogate key before the compund key. e.g;
enrollment id (surrogaet key) classId userId
althogh not a requirement of surrogate key in certain RDBMS
Some RDBMS may no get well with composite and compound key and you can use simple keys.

Relationships:
are nothing but relations between the tables in the database. We divide the big table into two parts and then they relate in some way. That's relation is known as relationship.

Types: One to One => One husband and one wife. One person has one CNIC
One to Many => teacher to students is example
Many to Many => Multiple classes has multiple students and multiple students has multiple classes. But this is not work in the relational databases.

One to One design: Whenever you find the one to one relationship then we can design that realtionship in table by 2 ways:
1) Make Attribute
For Example, you have the relationship of user and credit card. Suppose the company allows only one card to one user so this is one to one relation-ship. So we can make its attribute in the user table like:

userId	userName	cardNumber

Here we simply make the card number as attribute of user and this is frst method of desiging one to one.

2) If you have more details for the other entity for which you are going to make the attribute then use 2nd method which is to make 2nd table with the name of 2nd entity. Let's take above example and here we need card issue date and expiry date. For this, instead of making attributes for them also we should go with other table name Credit Card and give cardId to User Table and userId to Credit Card table.

userId	userName	cardId

cardId	cardNumber	cardIssue	cardExpiry	userId

NOTE: In this, we make the primary key of user table which is userId to foriegn key of credit card table and cardId to user table.

2) One to Many Design: Let's take above example but now company of credit card allows the user to have multiple cards. so this is reltionship of one to many. Its desgin going to very similar but in this we just make forign key of user table in card table. Because in this scnerio we have multiple cards and we can't given multiple cards id to user table. so instead we just give the user id to card table and then every card will show the id of its user which tells that this card belongs to this user and this other one also belong to the same user.

userId	userName	

cardId	cardNumber	userId

Example: cardId 1 belongs to userId 1, cardId 24 also belong to userId 1. cardId 12 belongs to userId 47777.

Parent and Child: userId is primary key of user table which is going to card table and become foriegn key. So user is parent because it has primary key and card is child of user because forign key points to primary key of user table. Parent Child is not available in single table of one to one.

MANY to MANY Design: As we above discuessed that the design of many to many is complex in database. Why? Suppose we have class and students example. In this example, Class Table may have names of students which are enroll in this particular class. But what if someone drops then the space of that student become null. And Students table contain the name of student and the classes whcich he enrolled in it. Suppose one student is enrolled in all classes and other one is only enroll in one class then the second students other classes name attributes become null.

stuName	c1	c2	c3	c4	c5
Ali	enrolled	enrolled	enrolled	enrolled	enrolled
Talha	enrolled	nulll	nulll	null	null

So these null value is problem in many to many. Other problem is that both are parent at a time if we apply parent child rule in it.

But the solution is available for many to many. The solution is to break the many to many relationship into 1 to many and many to 1
Simply create intermediate Table which will contain the classId and studentId and shows the particular of id student is enrolled in this class id. This table is act like a child for both parents here like class and students table.

Class

classId	name
12	Math
20	Physics
30	Chem

Enroll (intermediary)

classId	studentId
30	3
12	1
20	2

Student

studentId	name
1	Talha
2	Ahmad
3	Ali

1 to many to 1

NOTE: Inntermediary shows that this student id has this class id. Here is ease now. If any student get drop then now the whole row regarding the student will delete. And here duplication is not allow.

So these all about the binary relationship... Binary means 2 and we are doing on two tables relations. Multiple table can have relation but that we do later.

KEYS: Keys are defined to speed up access to data and, in many cases, to create links between different tables.
=> Should be unique
=> Never Changing
=> Never Null

Primary key Index: In which if we want to search instead of searching each row one by one. But when we use index, we will order the data in a way that database knows how to get certain data:
Select * from where name = 'Talha'; // This will search one by one
But when we use index then we order data in such a way that database knows how to get certain data. so it will jump to T and fetch Talha.

LookUp table: This table is nothing but we make it for storing the set of values which are already defined and we have to use it again and again in our database. So, we make the sperate table for this and use them by using their Id. e.g; GYM has members and they can buy membership of gold, silver and platinum. So these are defined now, so we can make seperate table of it use forigen key to access them. If we do not do that then there will be problem of duplication and database get slow. e.g; Member Ali is gold and others members are also gold then gold is repeating. So this is consuming our database more. so we can simply use the ID of gold from the membership table where we defined all these things and can use them in member table.

Others example can be like: => States, Gender etc.

Super Key and Candiadate Key: The main keys are primary and the foreign keys. but when we go for high level database desgin then we use the concept of super key and candidate key also.

Candidate keys are those which are least to uniquly identify the rows. e.g; userId OR userName OR email etc. Now each of them are candidate keys and when we select the most uniquly identify attribute for our table then it is primary and we use it as forign and we the reaming attributes in candiate keys set are known as alternatives keys.

Super Keys: Combination of candidate keys are super keys like userId + userName + email, now its a super key becausse it is a set of all candidate keys.
userId => Candidate key
userName => Candidate key
email => Candiadate key

Set of each candidate key which is "userId + userName + email " is super key.

Selection of primary key:
Those candidate key which has following properties you can make one of them as primary key.
1) Unique
2) Never Changing
3) Never Null

Surrogate Key: The key which is gemerated when we insert the data which we want to insert in table. e.g; auto generated id by adding 1 is surrogate key whcih is after the data is inserted or which is not natural.

Natural key: Name shows that it is natural. like first name, last name. All the data which we already have.

Not Null: is the restriction that if someone dont have value than he will not be able to put information in this table where not null restriction is.

Diffrenece in primary and the foreign keys are:
Foriegn key can be null because sometimes there maybe no realtionship
Relationship maybe get change so maybe update because forign key refrences get change

OK so if you want to must have relationship or foreign key then you can use characterstic of not null which make it required.
NOTE: Terms name may change in otther DBMS but concept remain same.
Foriegn Key Constraints:
These are forign key constraints that refer to parent.
On Delete
On Update
These are forign key constraints that refer to child.
Restrict
Cascade
Set Null

Ok so we perform like on delete with restrict OR on update with cascade