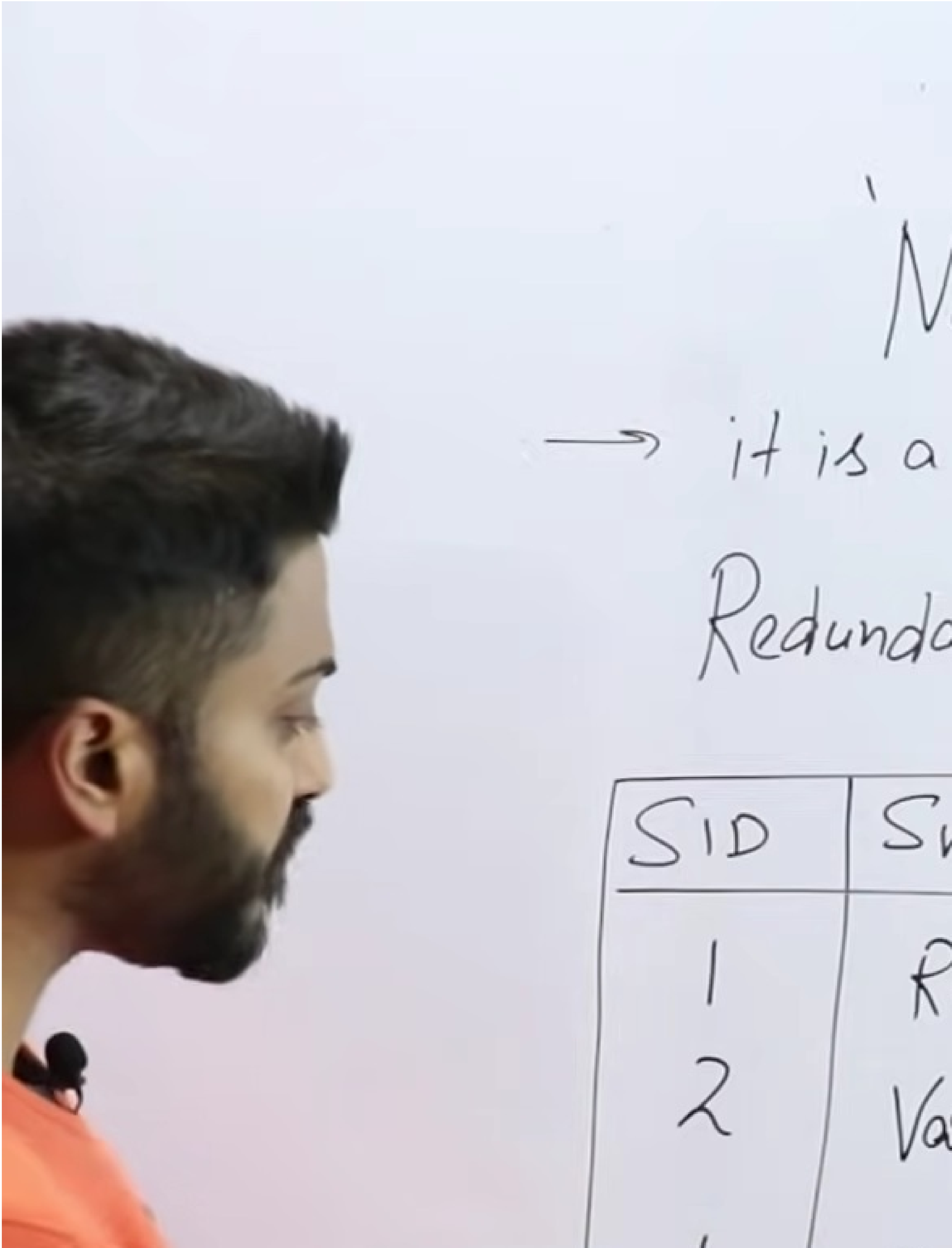
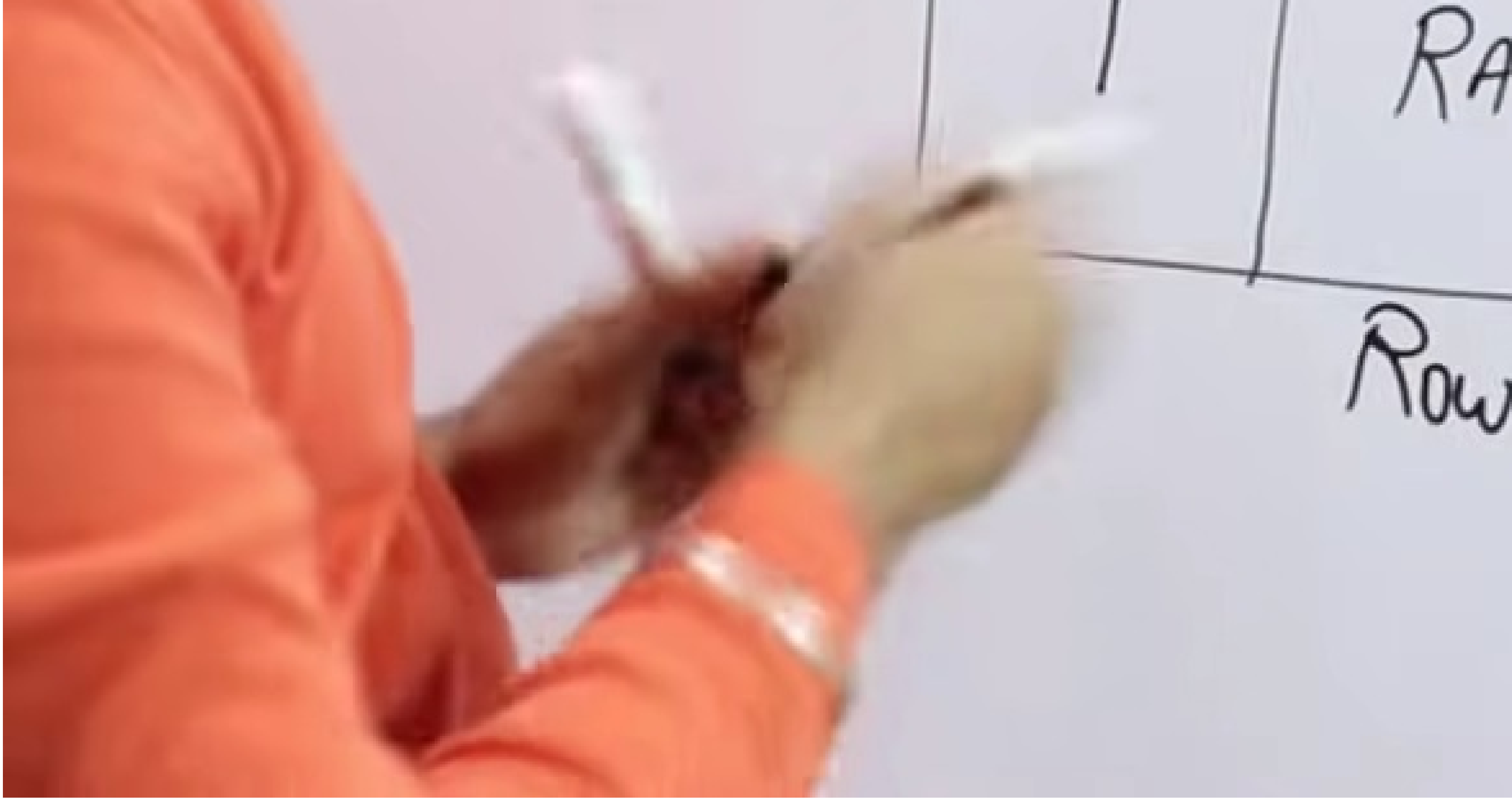
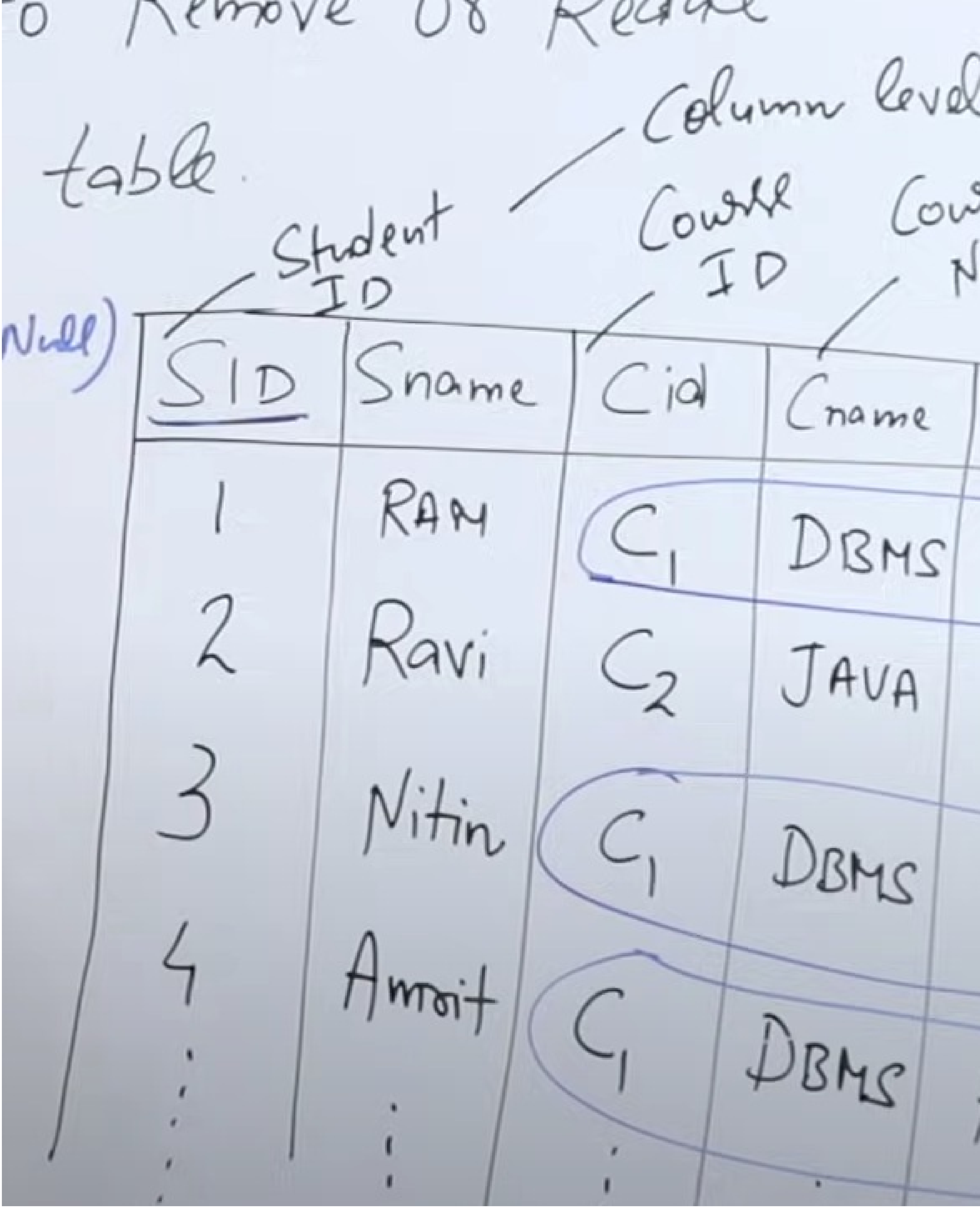
DBMS

NORMALISATION

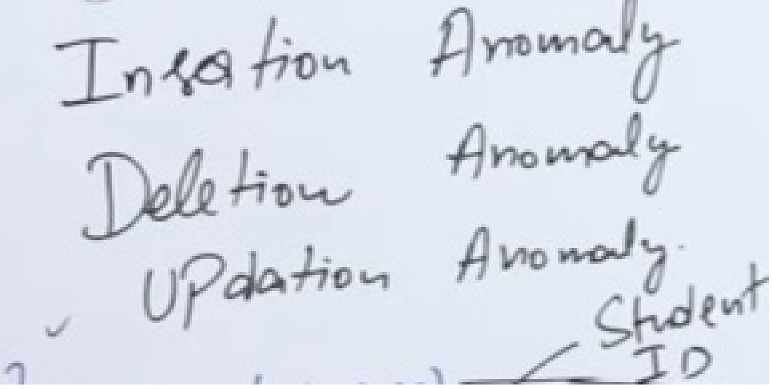




In order to remove the row level duplicacies, we introduce sid as primary key.

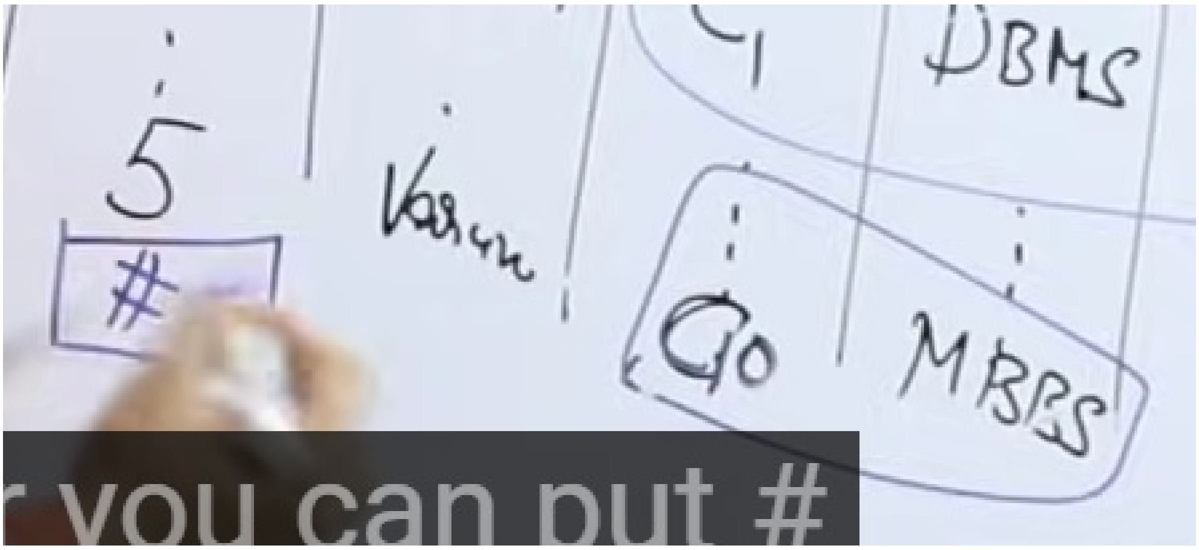


There are 3 anomalies that occur due to column level duplicacy :



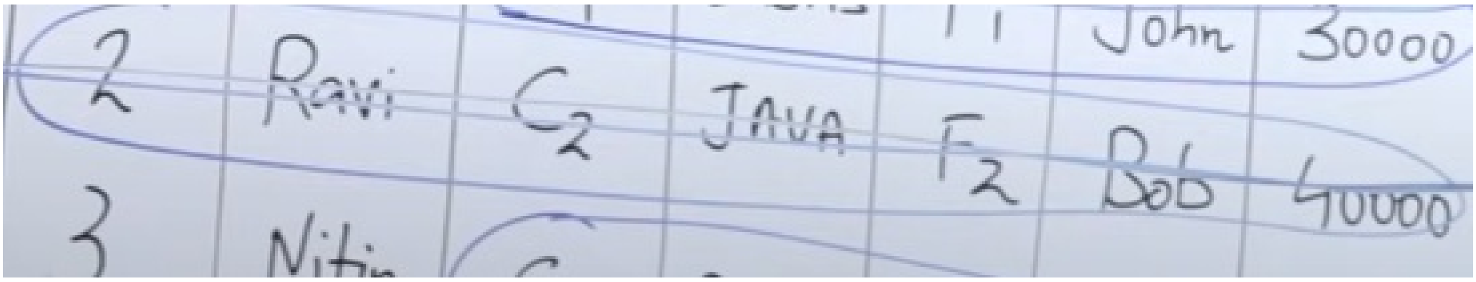
- INSERTION ANOMALY

If I want to introduce and insert a new course in the table , I cant without the mention of sid



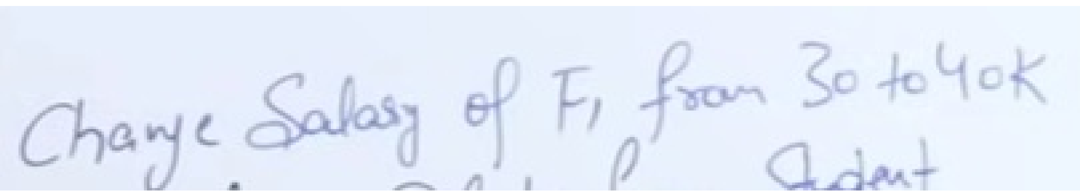
- DELETION ANOMALY

Delete from student where sid =2;



This particular row will be deleted but all the data related to the course c2 will also be deleted which was just present in this particular row.

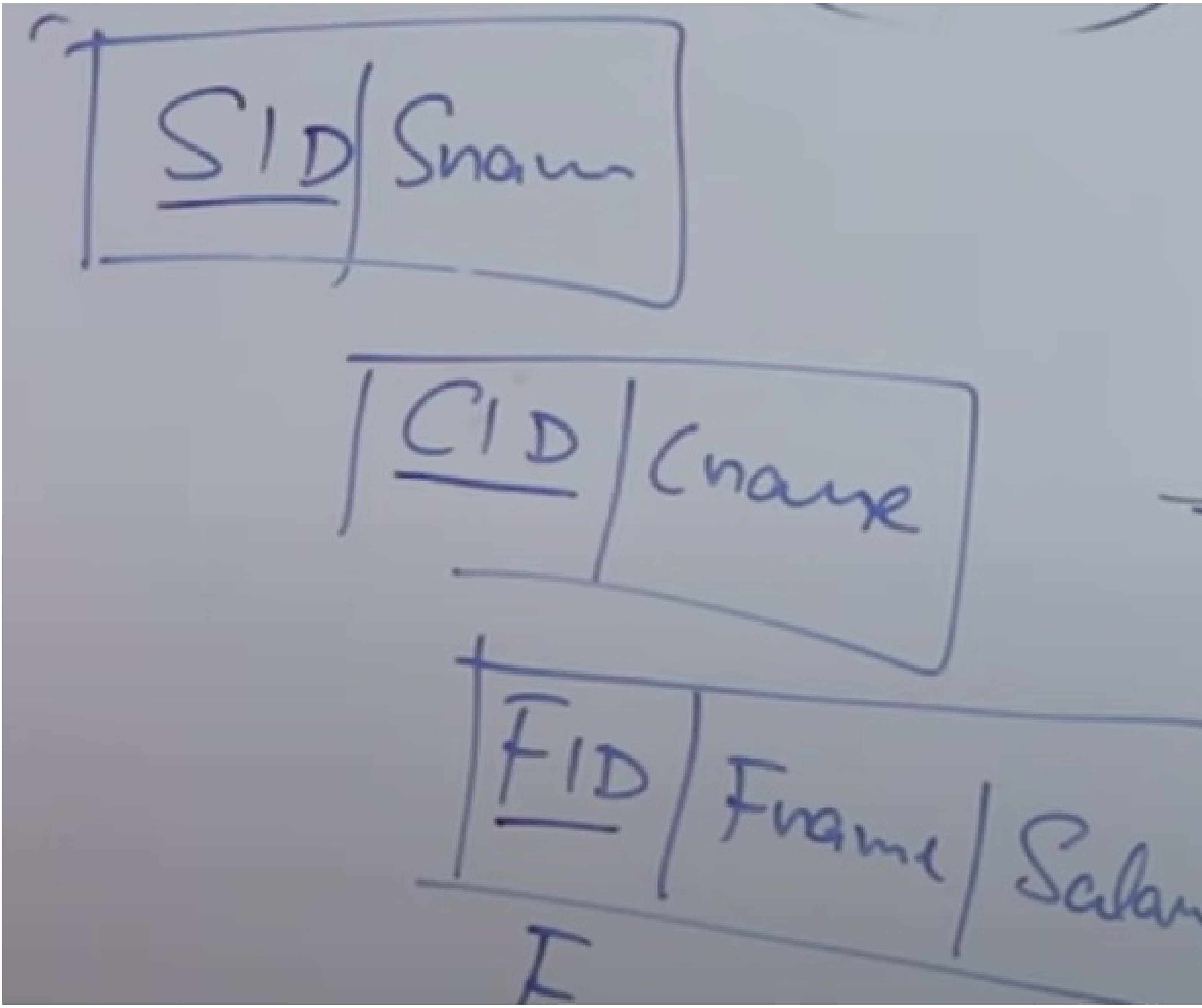
- UPDATION ANOMALY



The changes will happen for multiple rows int the table for a single fid which will take lots of time.

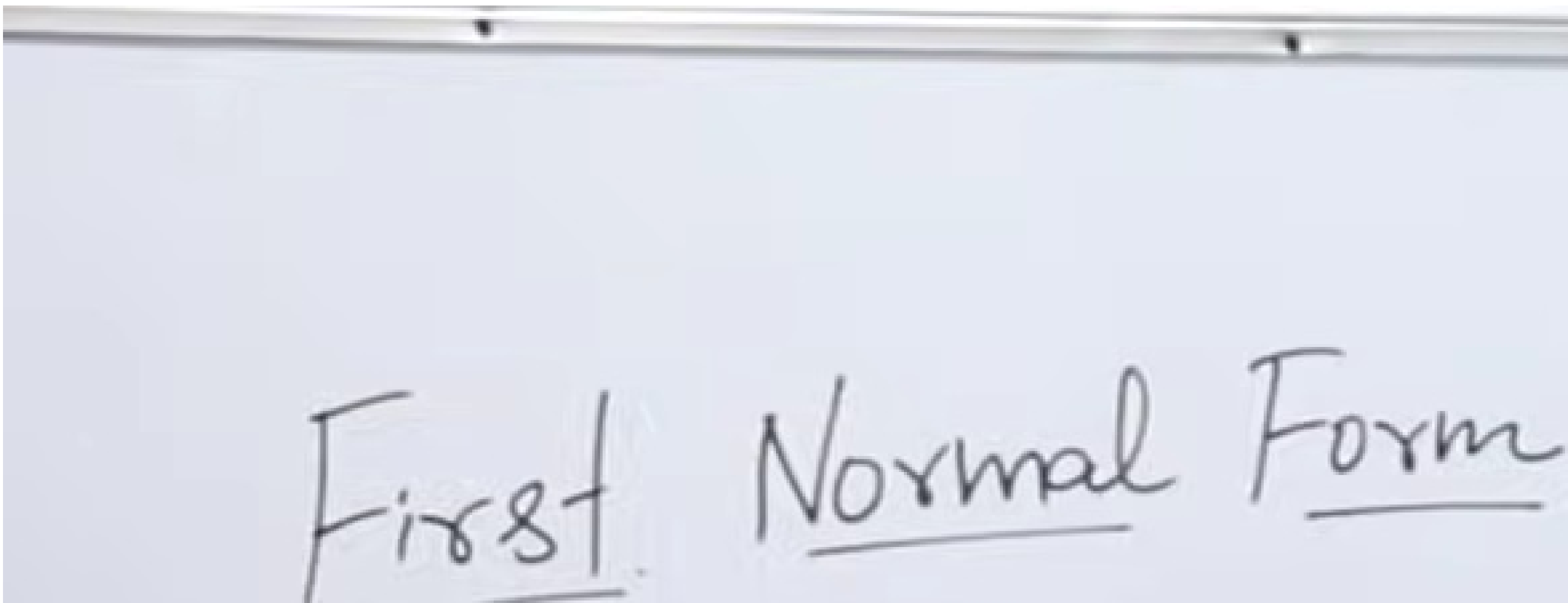
NORMALIZATION SOLVES ALL THESE ANOMALIES.

ONE OF THE SOLUTIONS WOULD BE :



(dividing into 3 separate tables)

- FIRST NORMAL FORM



→ Table should

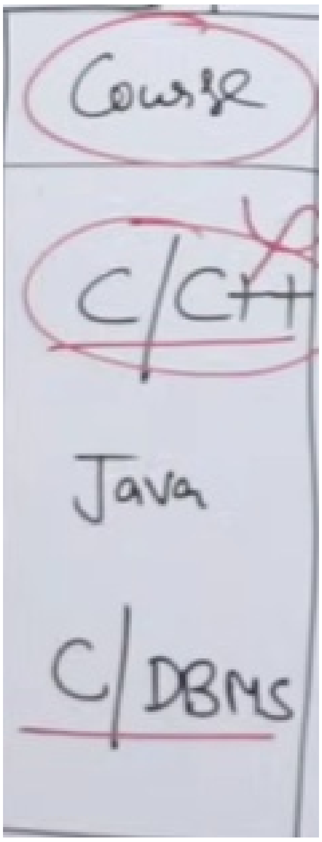
any multivalued

Not in

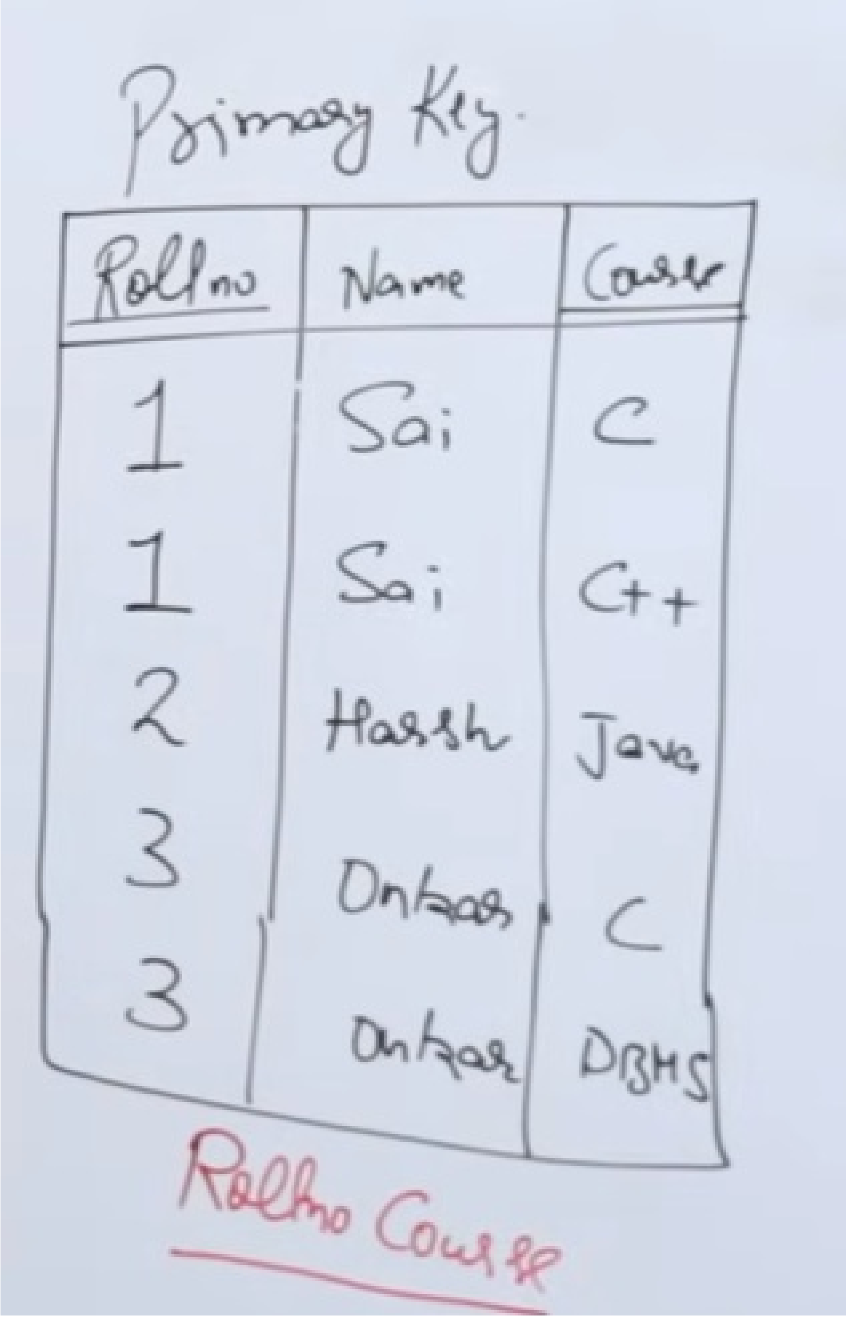
Student

Rollno	Name
1	Sai
2	Harsh
3	Onkar

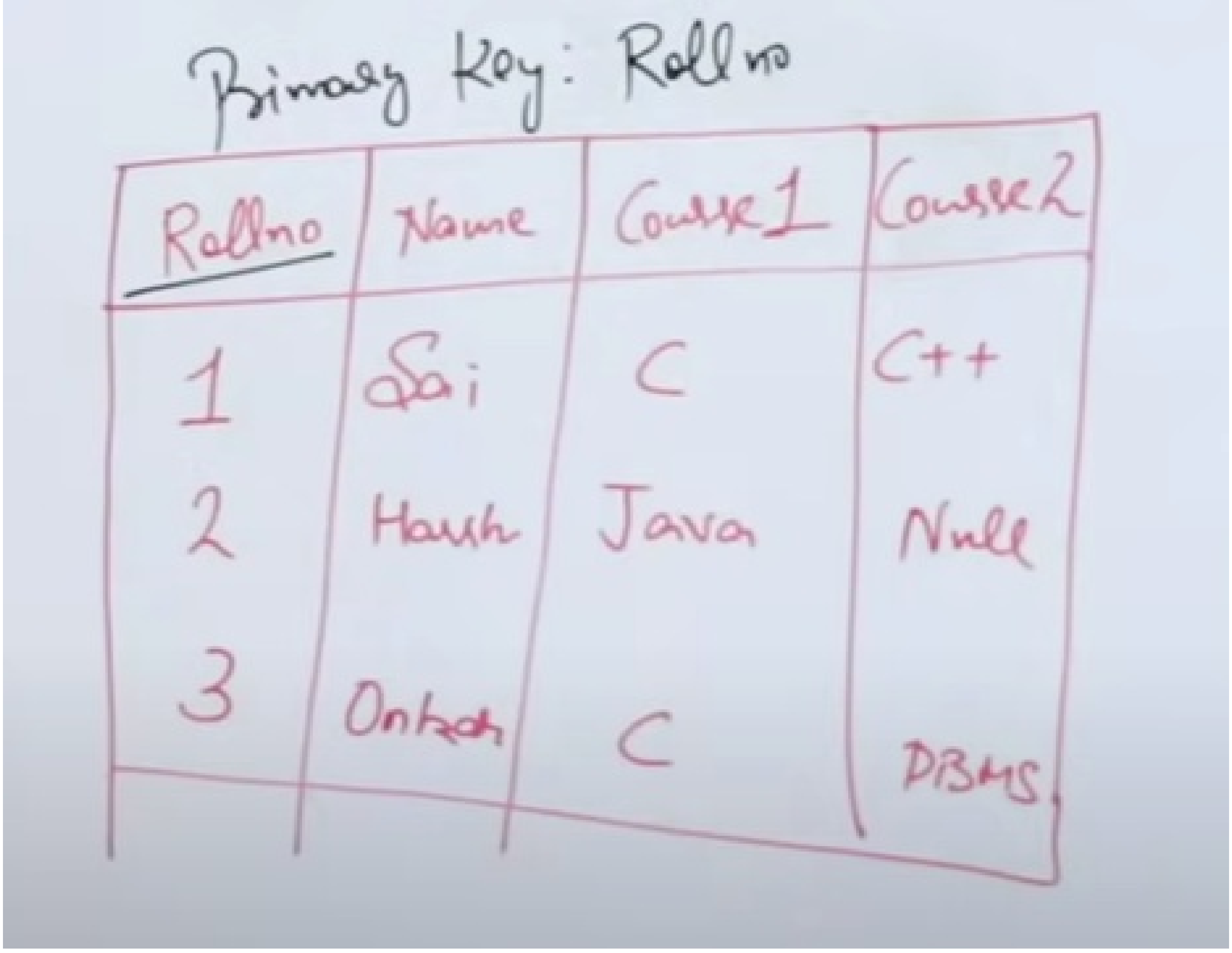
Course is a multivalued attribute therefore not in 1st NF



3 SOLUTIONS

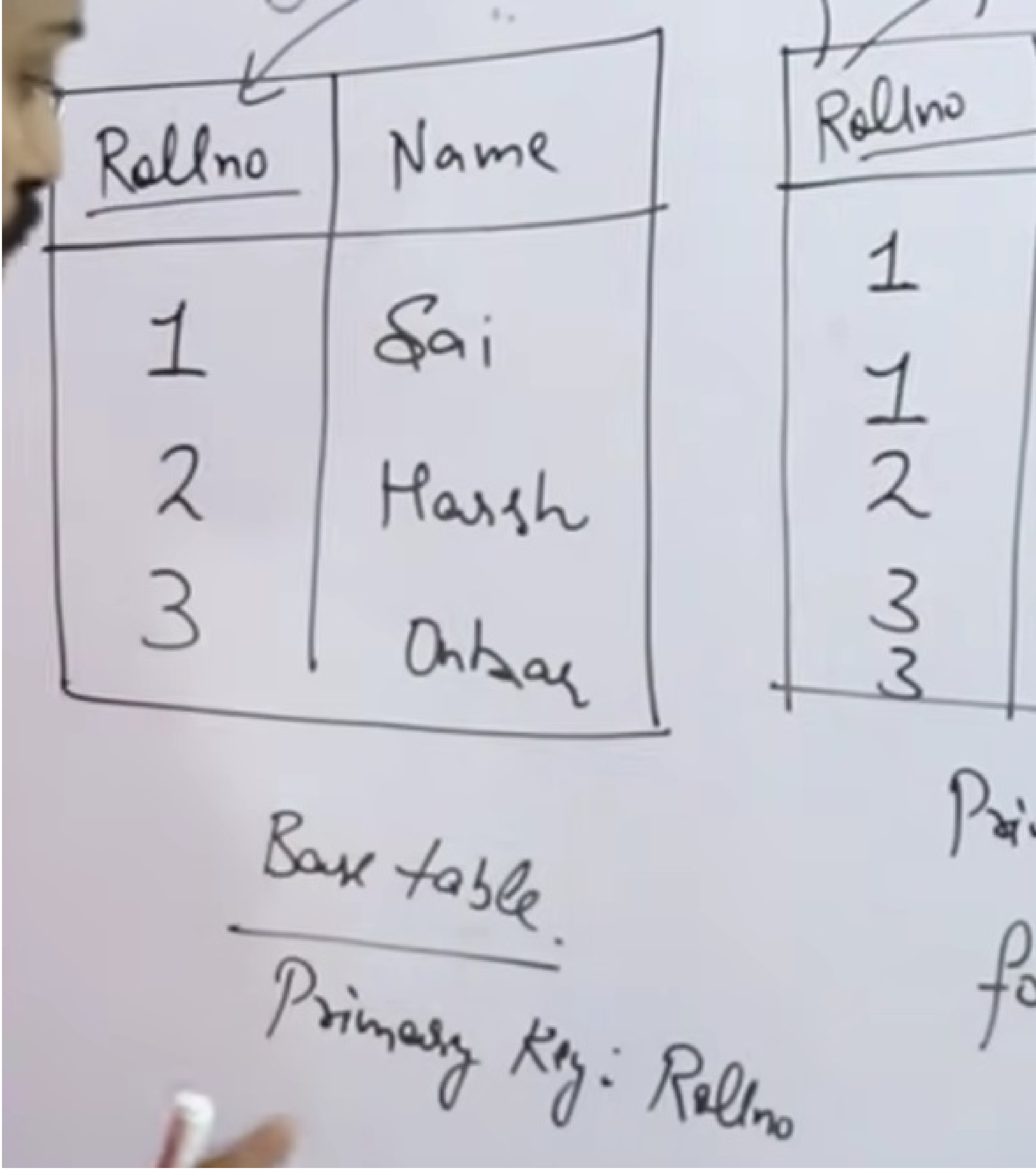
1. 

COMPOSITE PRIMARY KEY :ROLLNO COURSE

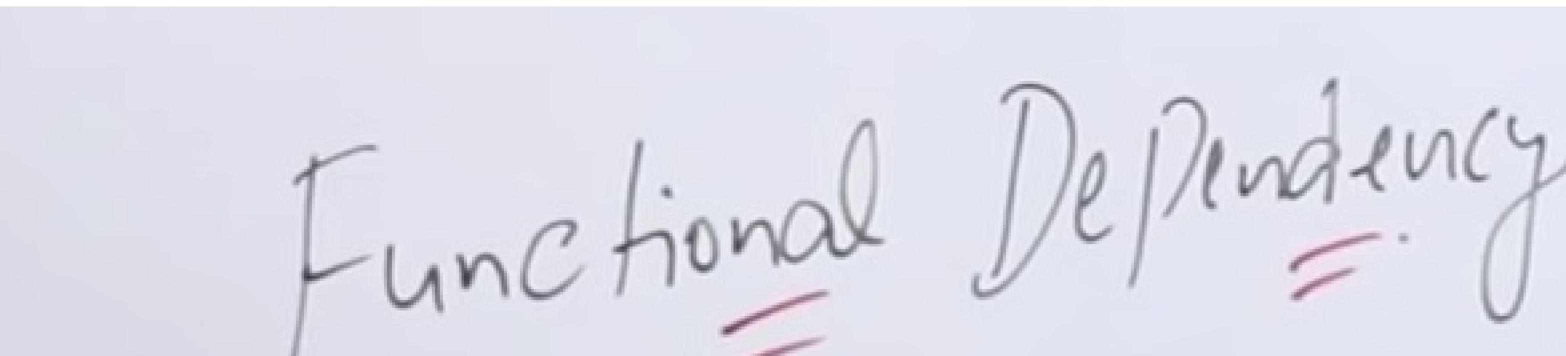
2. 

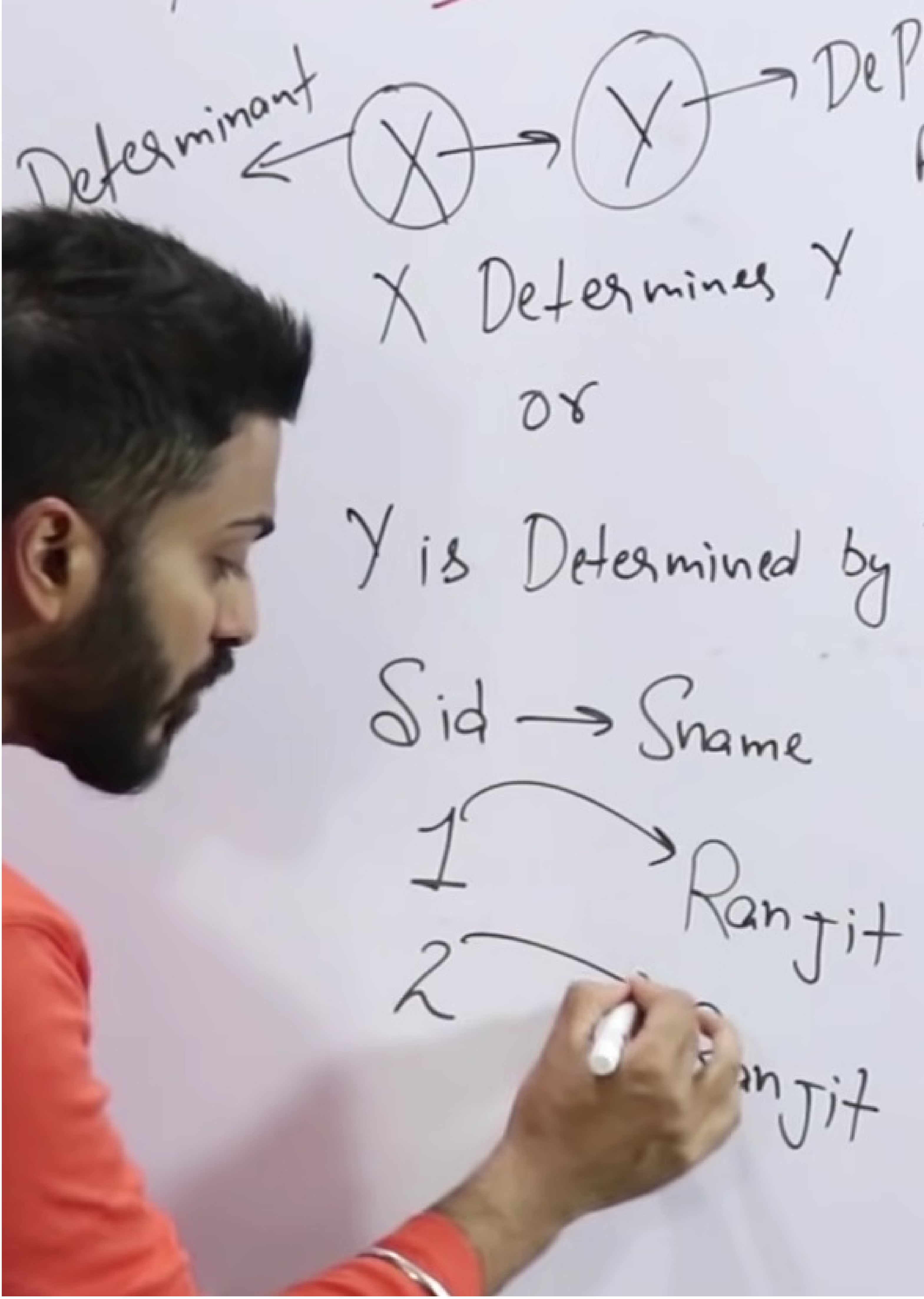
This is not the most optimum solution kyunki aisa ho skta ki 15 courses hon max aur ek koi row mei sirf ek hi course ho to baaki saari values ko null krna pdega which is not a good representation

3) BEST METHOD



FUNCTIONAL DEPENDANCIES – tells the dependency of attributes

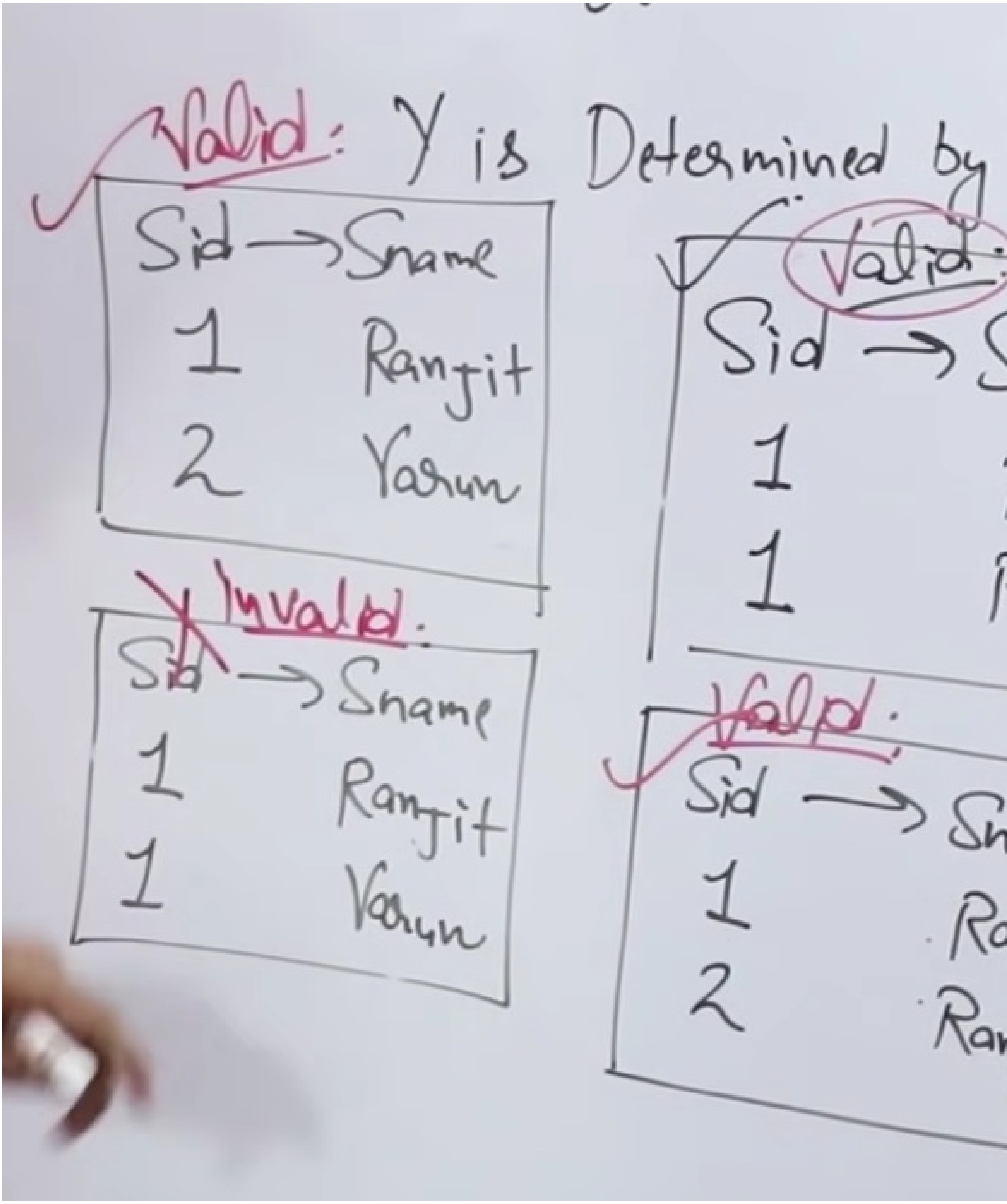



Determinant
X
Y
X Determines Y
or
Y is Determined by
Sid → Sname
1
Ranjit
2

If my table has 2 snames with the same value as Ranjit

How do I know that both tha ranjits are same or different ? By looking at their SIDs , different SIDs denote different tuples , same SIDs denote same tuple

VALID AND INVALID CASES :



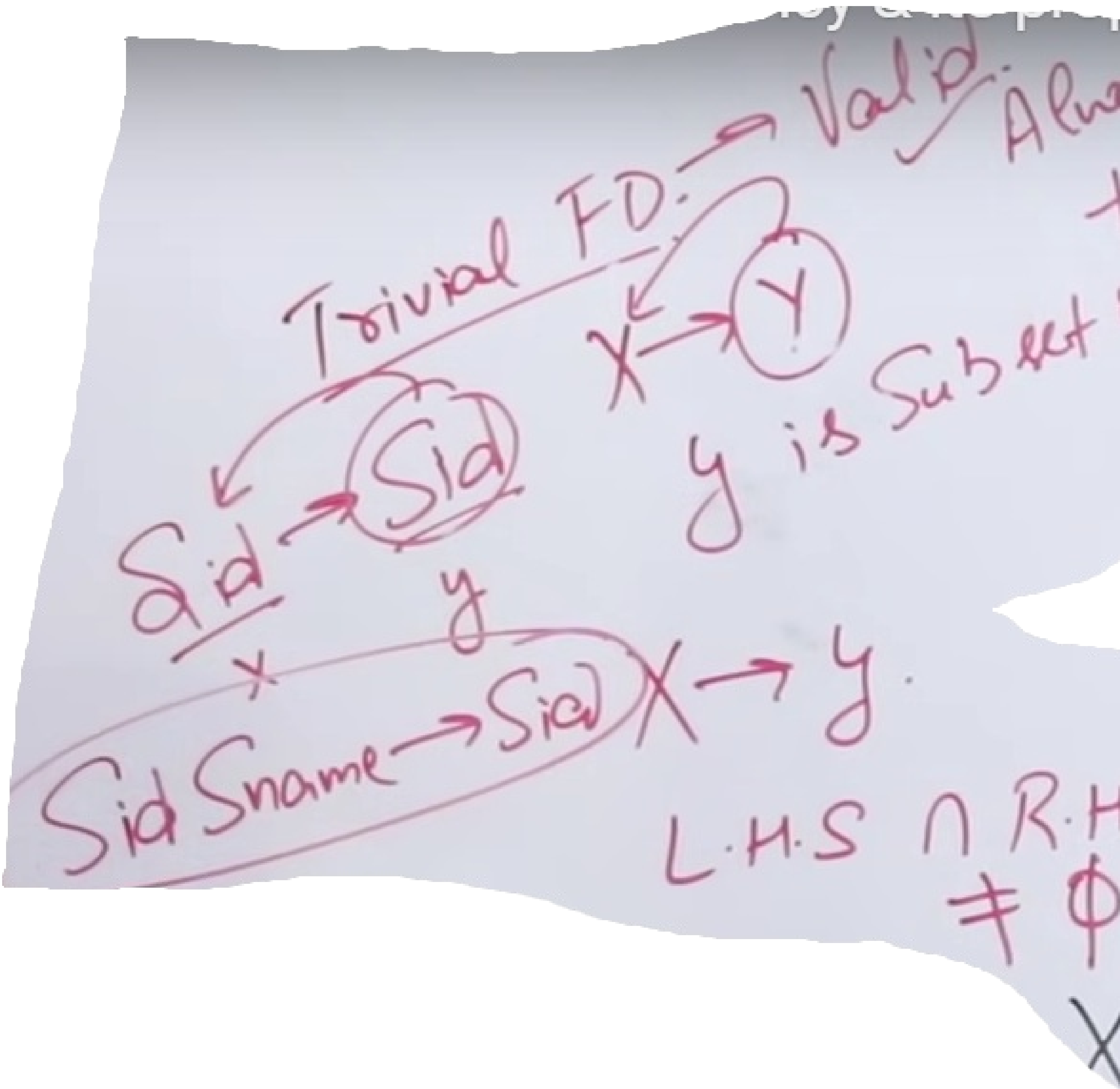
FUNCTIONAL DEPENDENCIES ARE OF 2 TYPES

- TRIVIAL
- NON TRIVIAL

TRIVIAL:

If X->Y is a functional dependency then this implies Y will be a subset of X. This FD is always valid kyunki Y already subset hai XX ka to vo ek tarah se khudse hi determine horha .

In trivial FDs lhs ^ rhs != phi

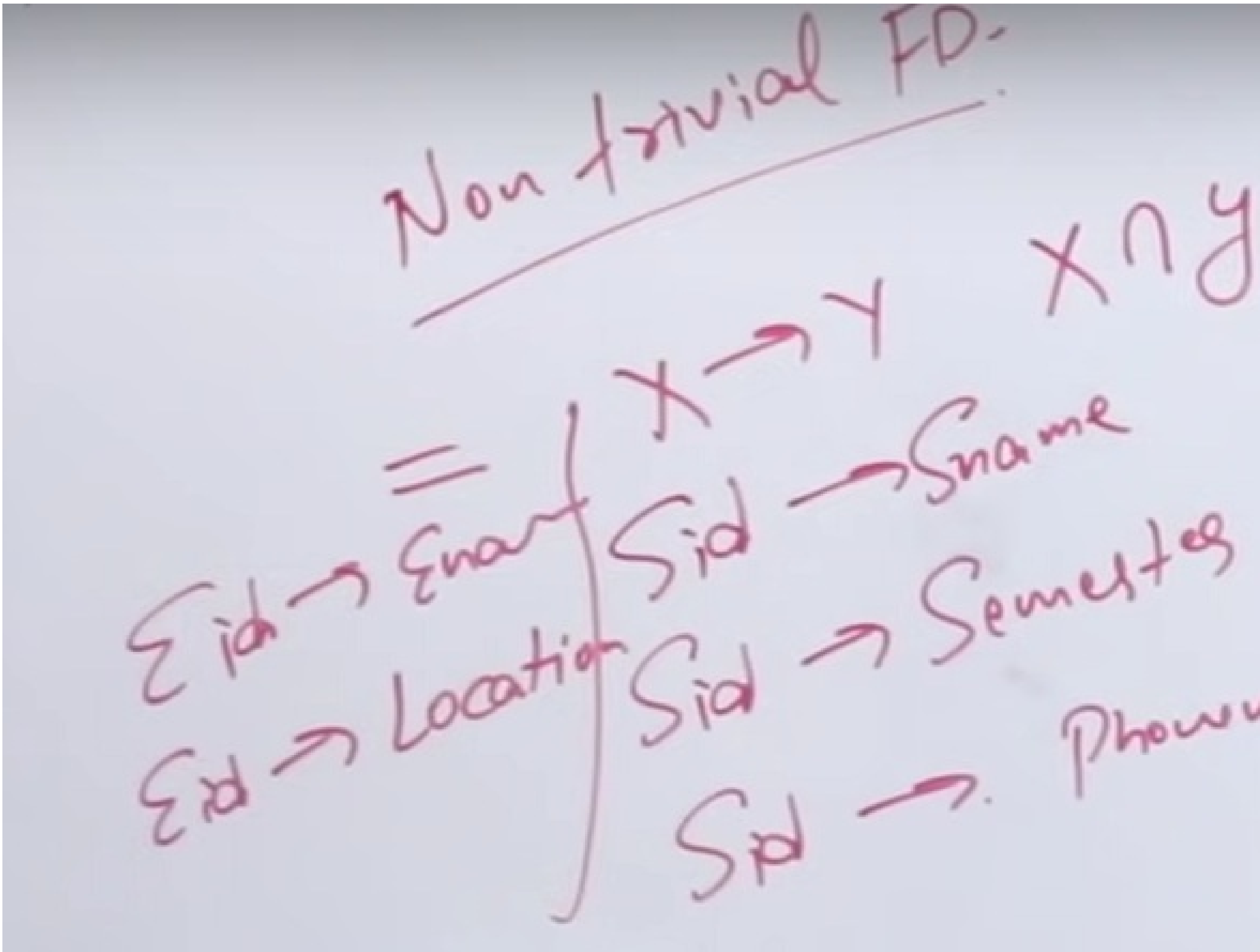


NON TRIVIAL :

If X->Y is an FD then this implies X^Y = phi

Eg : SID -> Sname , SID -> Semester , Eid -> Ename

This FD is not always valid isme cases dekhne pdte kaunsa valid hai aur kaun nhi



PROPERTIES OF FD ****

Properties of FD:

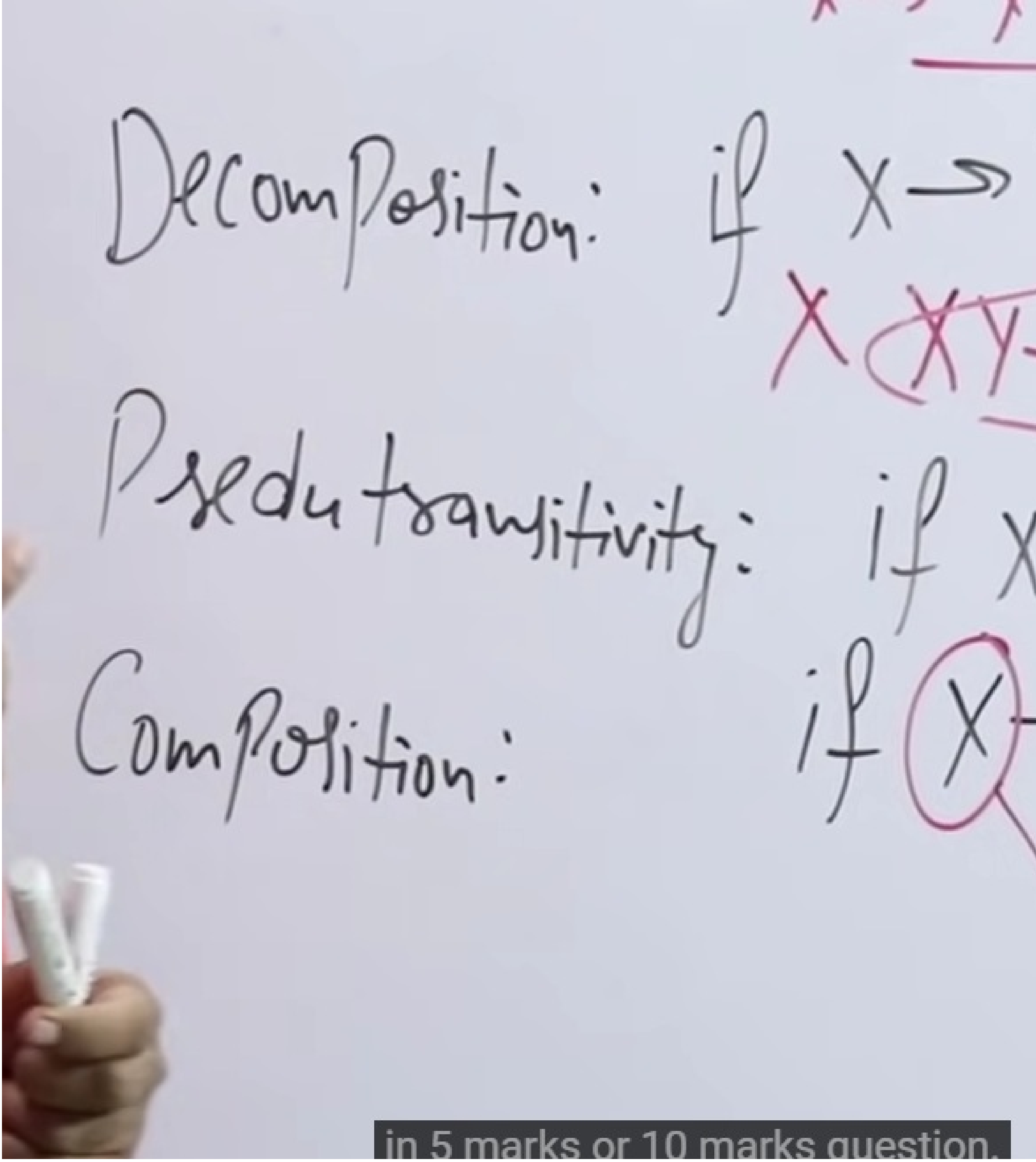
Reflexivity: if y is subset

Augmentation: if $X \to Y$,

** Transitive: if $X \to$

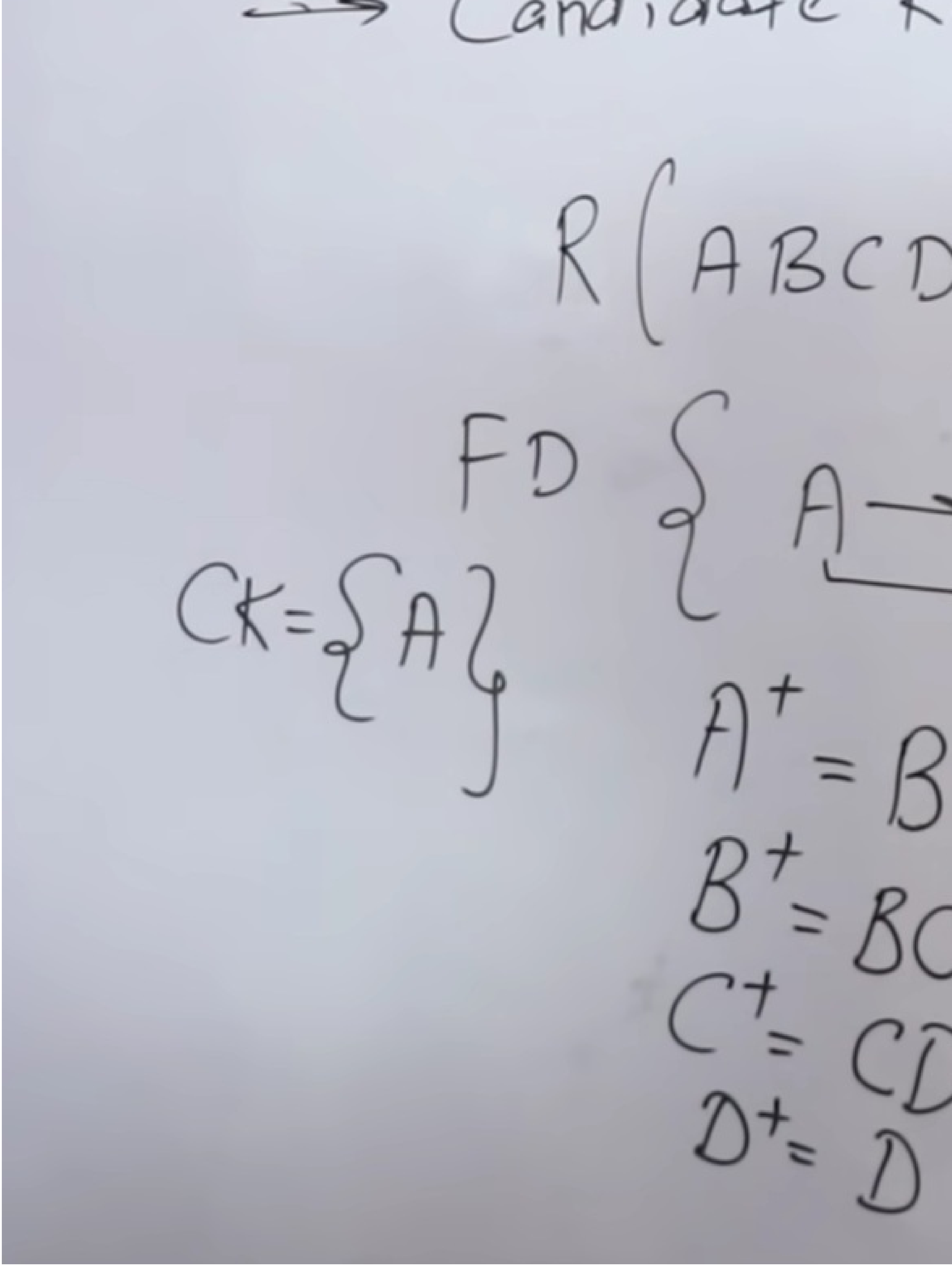
Union: if $X \to Y$

$X \to$



CLOSURE METHOD *** : Used to find all the possible candidate keys in a relation.

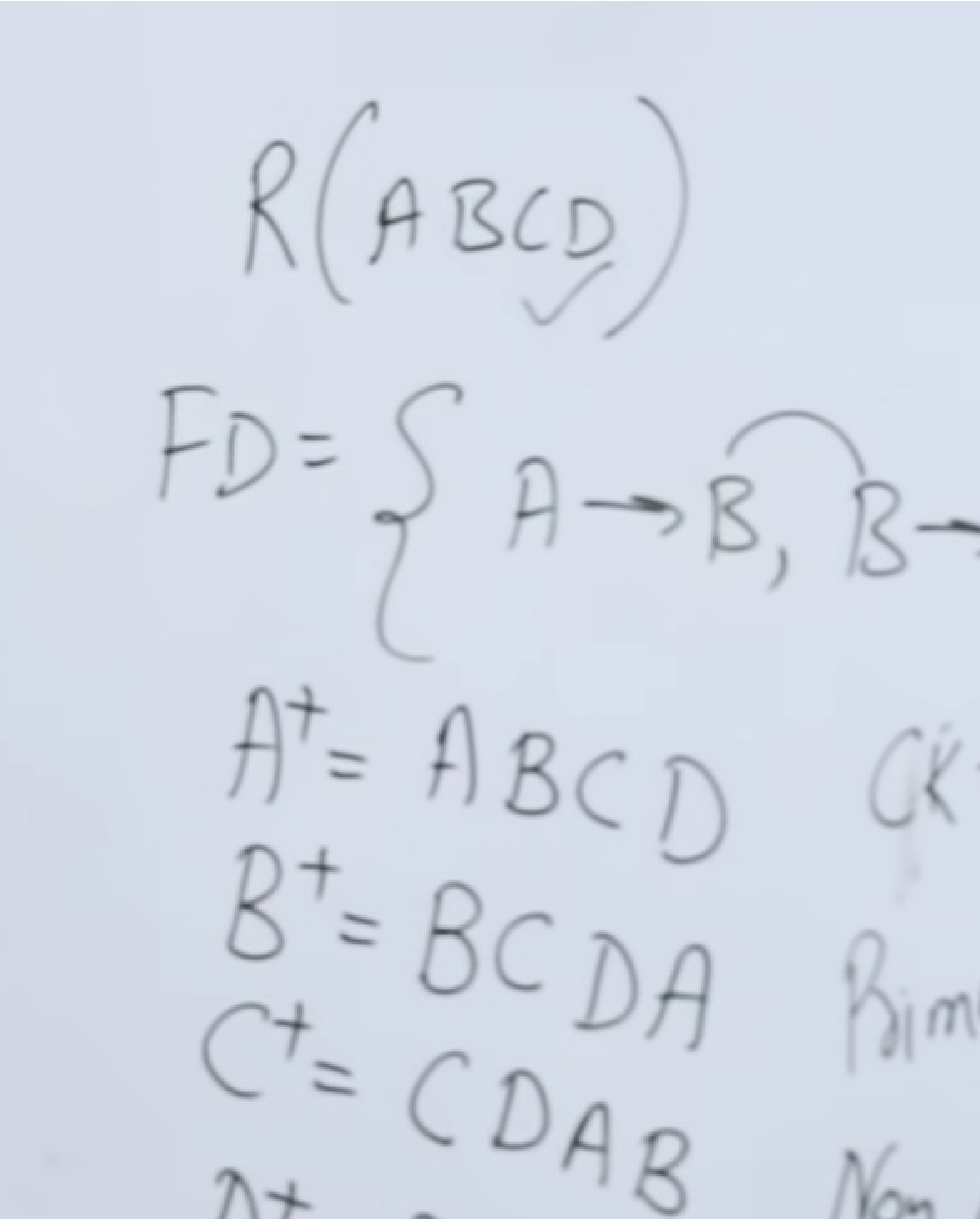
Candidate Key : an attribute or minimal group of attributes that can uniquely determine all the attributes in a relation

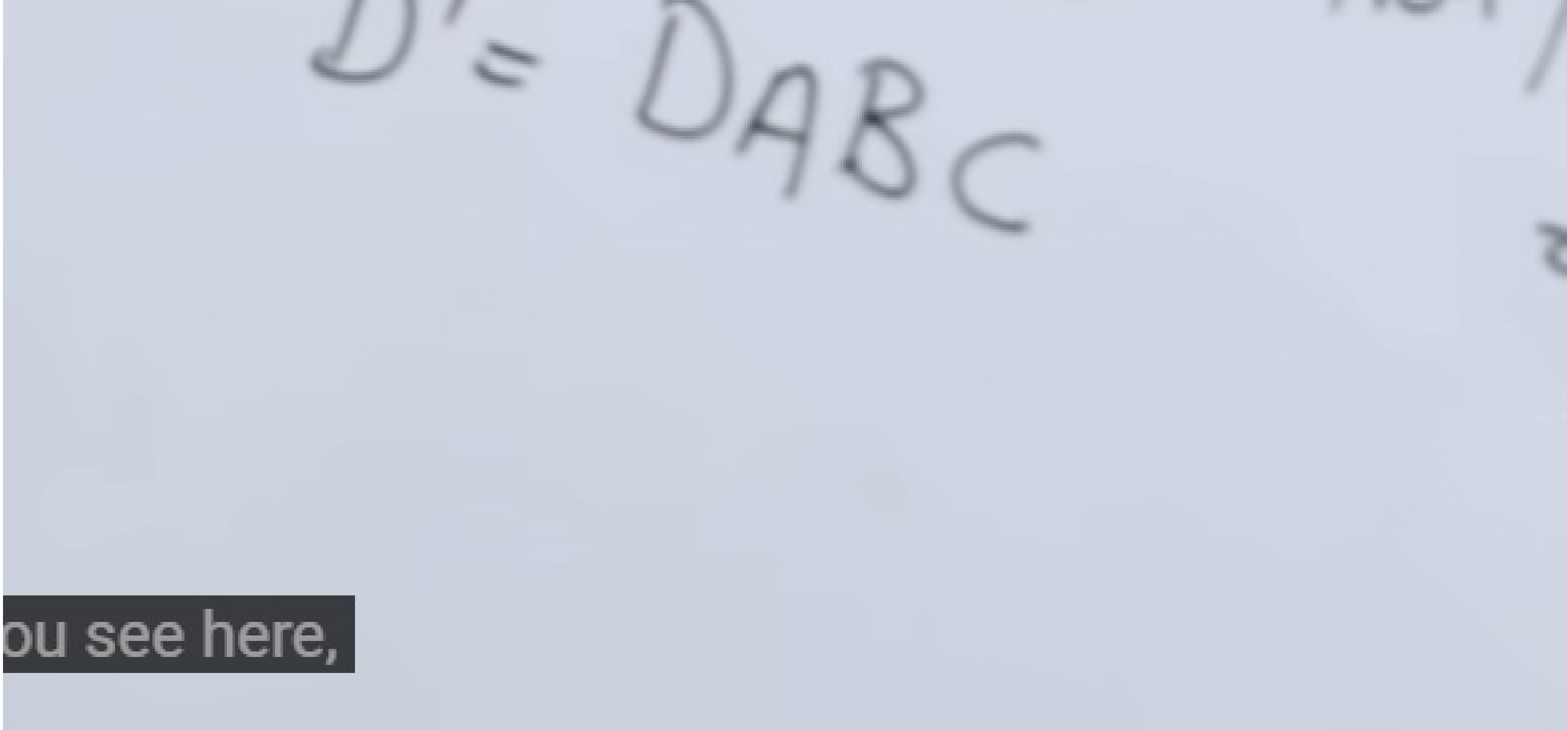


A+ means the closure of A i.e what A can determine by its own

Here (ab) would be a super key not a candidate key cause candidate keys are minimal super keys and we have already found a single attribute ck.

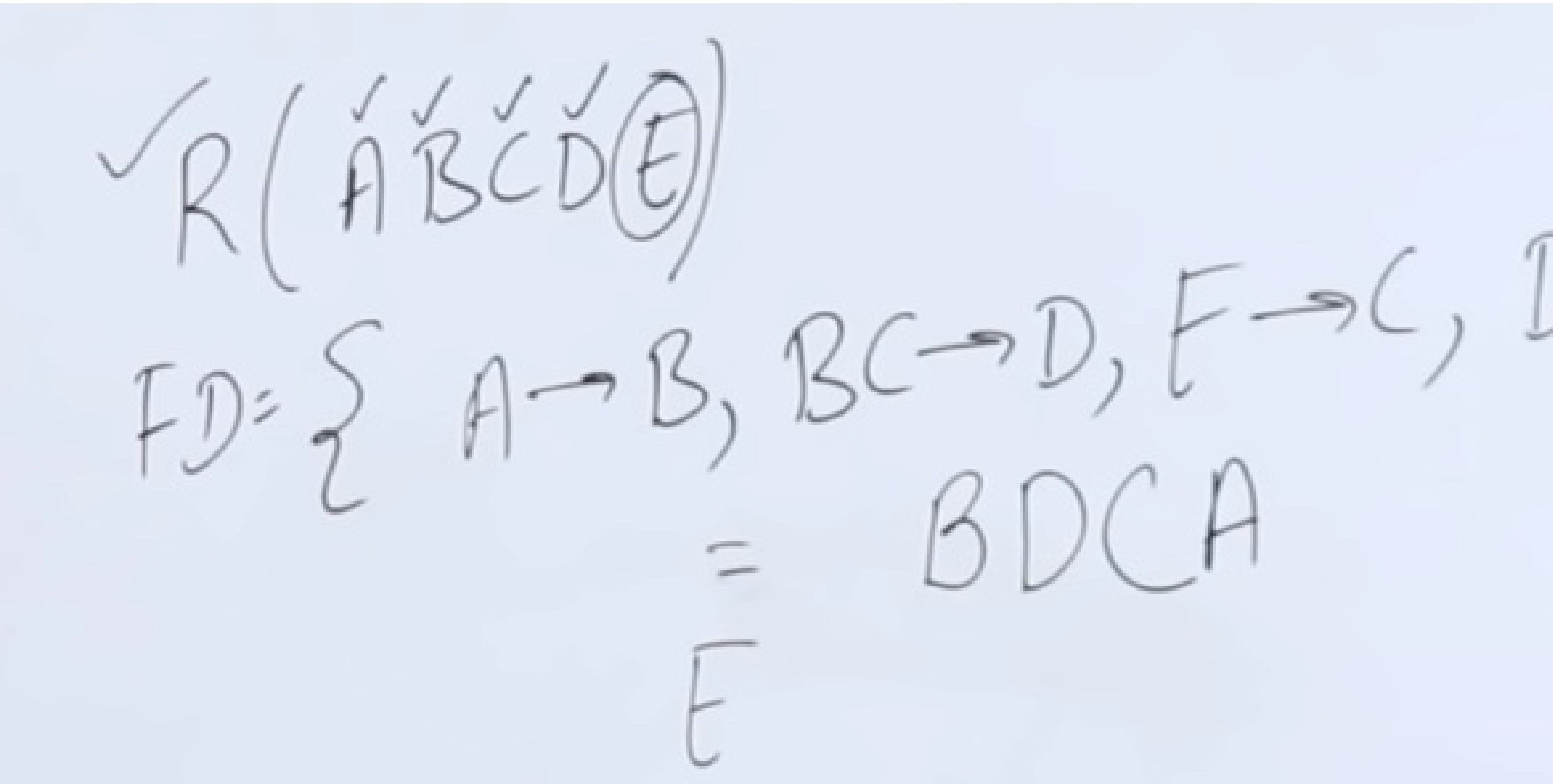
- Prime attribute : the attributes of the relation which are used in the making of all the candidate keys. In the above eg Prime attribute ={A} , Non-Prime attribute = {B,C,D}.





For complex questions to find the cks follow these steps :

- Find all the rhs and note all the attributes WHICH CAN BE DETERMINED

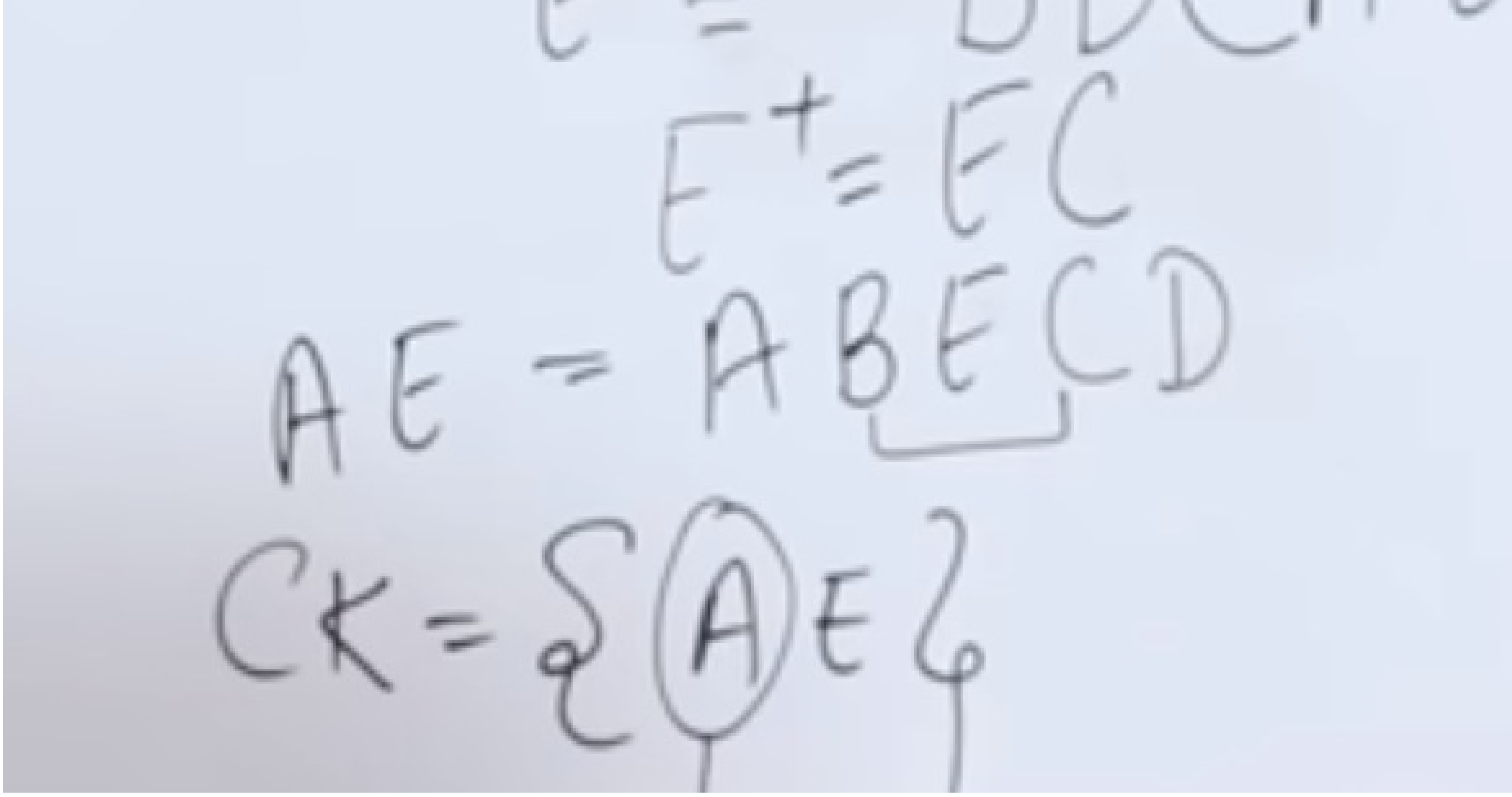


Here , BCDA are to the right and hence it is possible to determine them

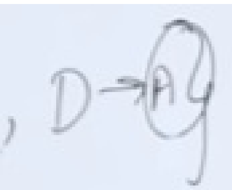
But E can not be determined kaise bhi

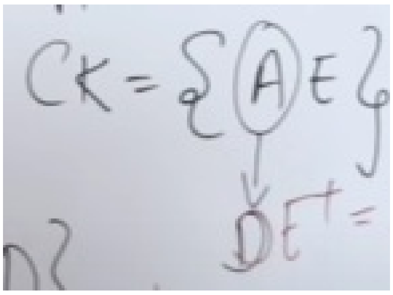
So multiply E both sides as E can determine itself.

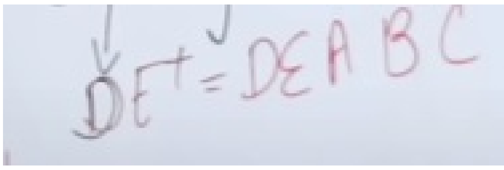
- Now we will always include the attribute which can not be determined by anyone but itself in our answer



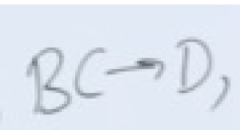
- Now that we have got our first candidate key we can check if A or E are present right mei kahi pe bhi in our functional dependency



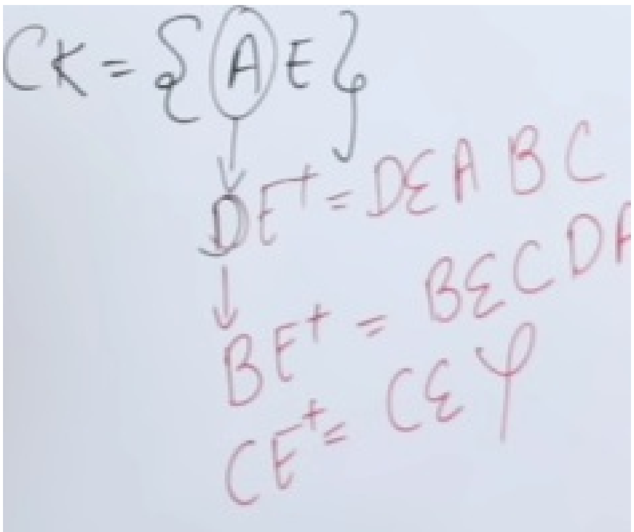




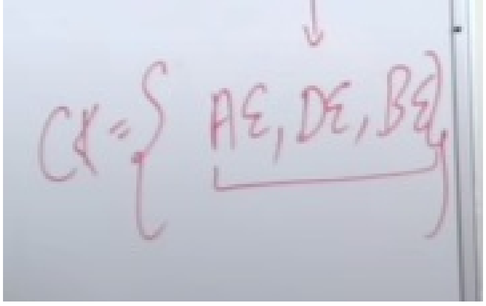
- Now check if D is present right side mei kahi bhi



so we ca replace D with B and with C as well



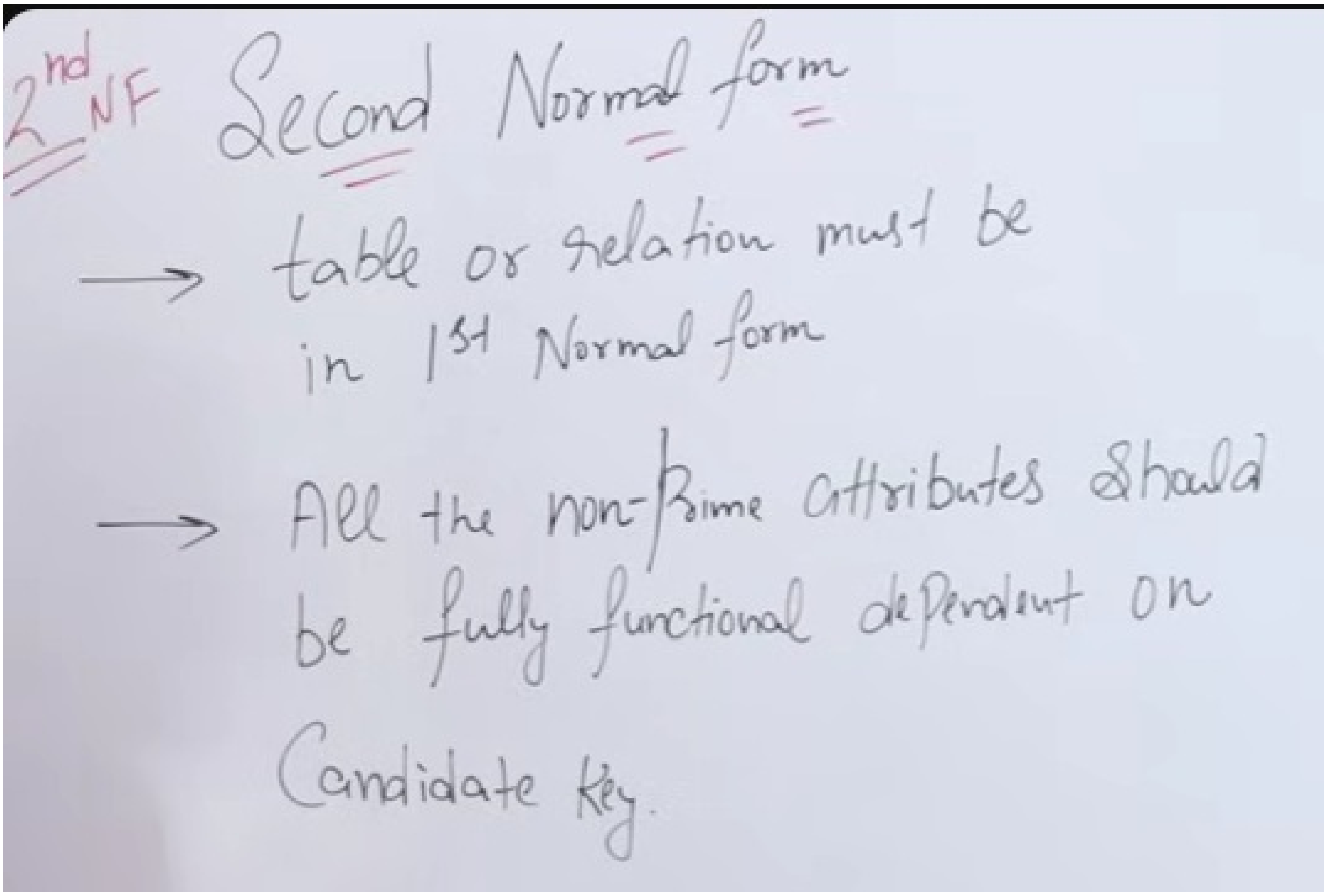
CE cant be Candidate key because it cant determine all the attributes .

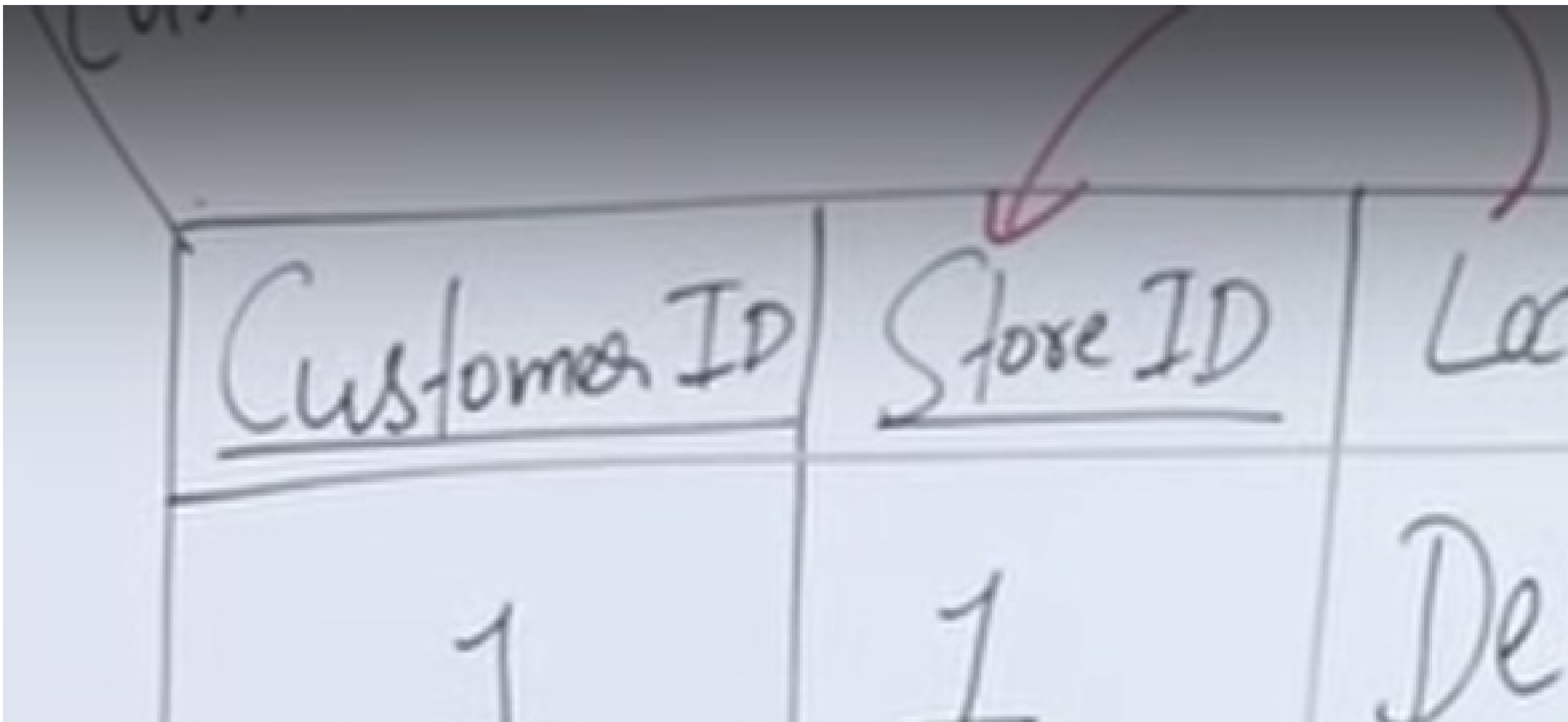


-> Final answer

Prime Attributes ={A,B,D,E} Non-Prime ={C}

- SECOND NORMAL FORM





1	1	
1	3	Mu
2	1	Del
3	2	Ban
4	3	Mum

Candidate Key: Customer ID

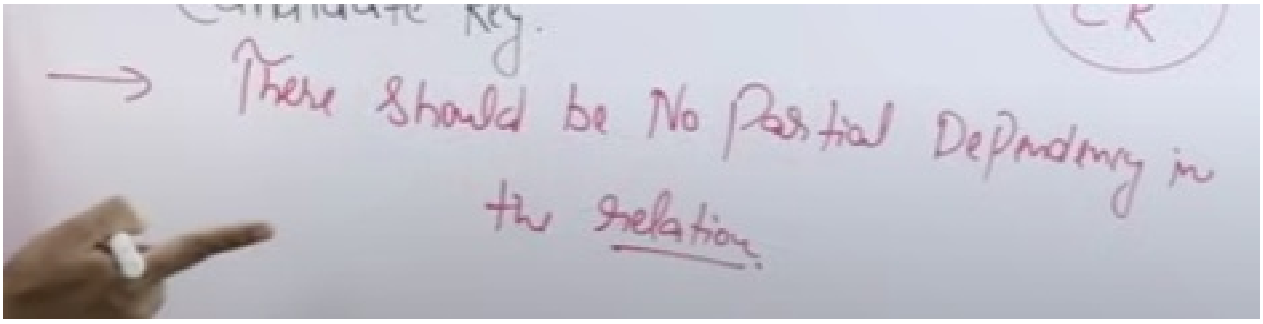
Prime Attributes: Custm I
Store ID

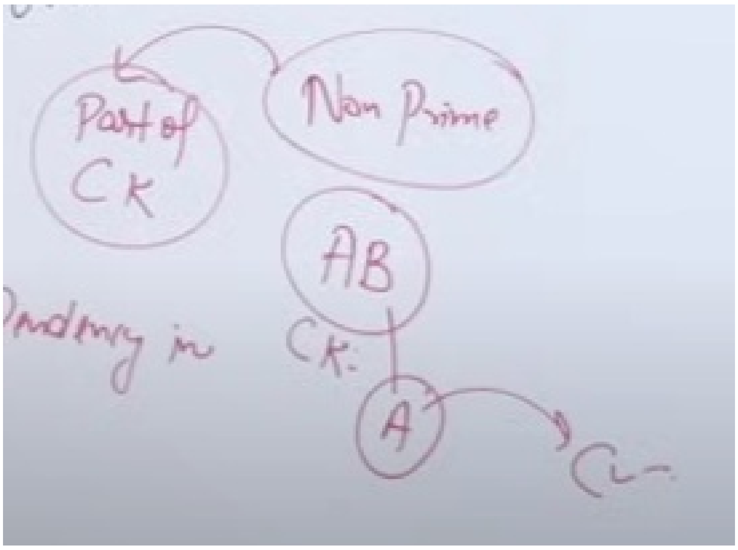
Non Prime: Location

In order to solve this , we divide the table in 2 tables with candidate leys for the first table being ={CustomerId StoreId} and for the second ={StoreId}.

Both the tables will be in the second normal form.

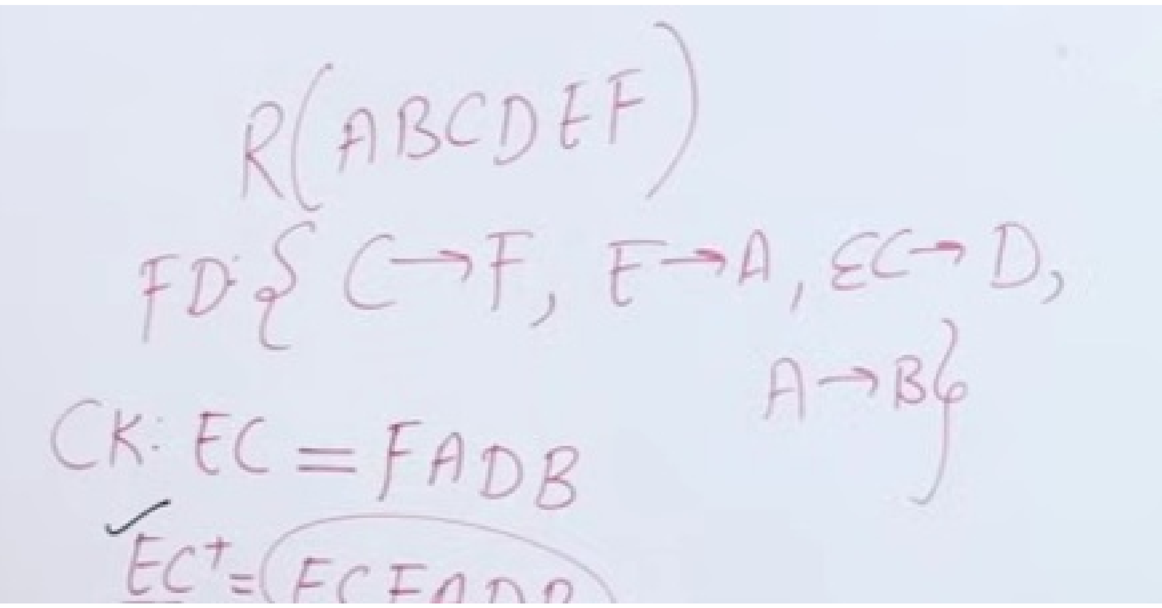
The second point of the second nf can be written as :

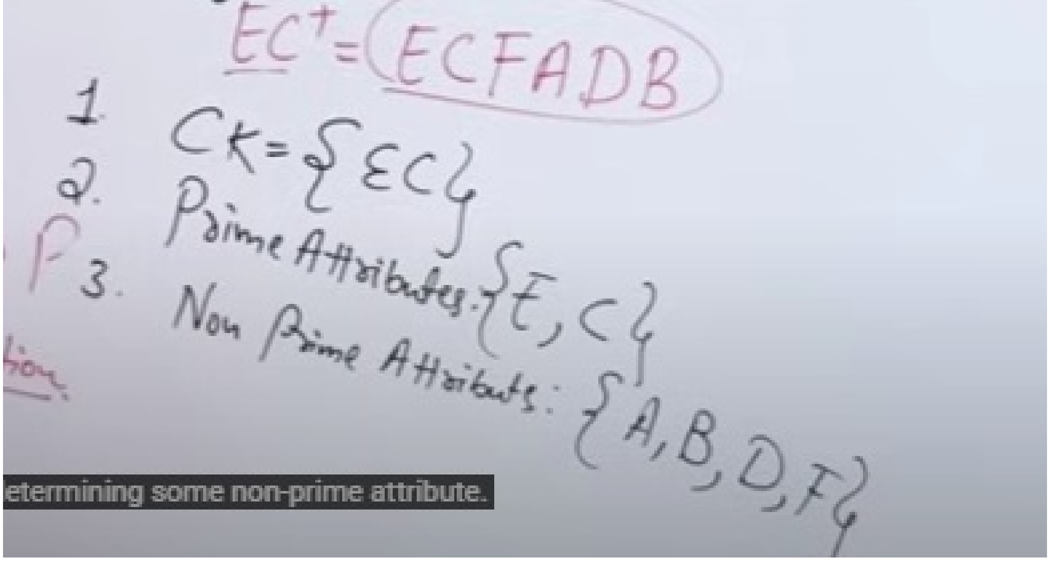




eg : ab is candidate key and a is determining c -> partial dependency

eg :





e and c both can determine a non prime attribute i.e partial dependency exists ! -> the relation is not in the second normal form

Partial Dependency : LHS should be the **proper subset** of CK and RHS should be a Non Prime Attribute.

1. Given a relation R(A, B, C, D) and Functional Dependency
given R is in 2NF? If not convert it into 2 NF.

Ans :

a) R1(B, C)

b) R2(A, B, D)

4. Given a relation R(A, B, C, D, E) and Functional Depend
whether the given R is in 2NF? If not convert it into 2 NF.

Ans :

Finally, the decomposed tables which are in 2NF:

a. **R1(A, B, E)**

b. **R2(C, D)**

c. **R3(A, C)**

3. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functi

TU, P → X, W → Y }, determine whether the given R is in 2N

Ans :

Since due to FD: PQ → R, PS → VW, QS → TU, P → X our table wa

R1(P, Q, R) (Now in table R1 FD: PQ → R is Full F D, hence R1 is i

R2(P, S, V, W) (Now in table R2 FD: PS → VW is Full F D, hence R

R3(Q, S, T, U) (Now in table R3 FD: QS → TU is Full F D, hence R

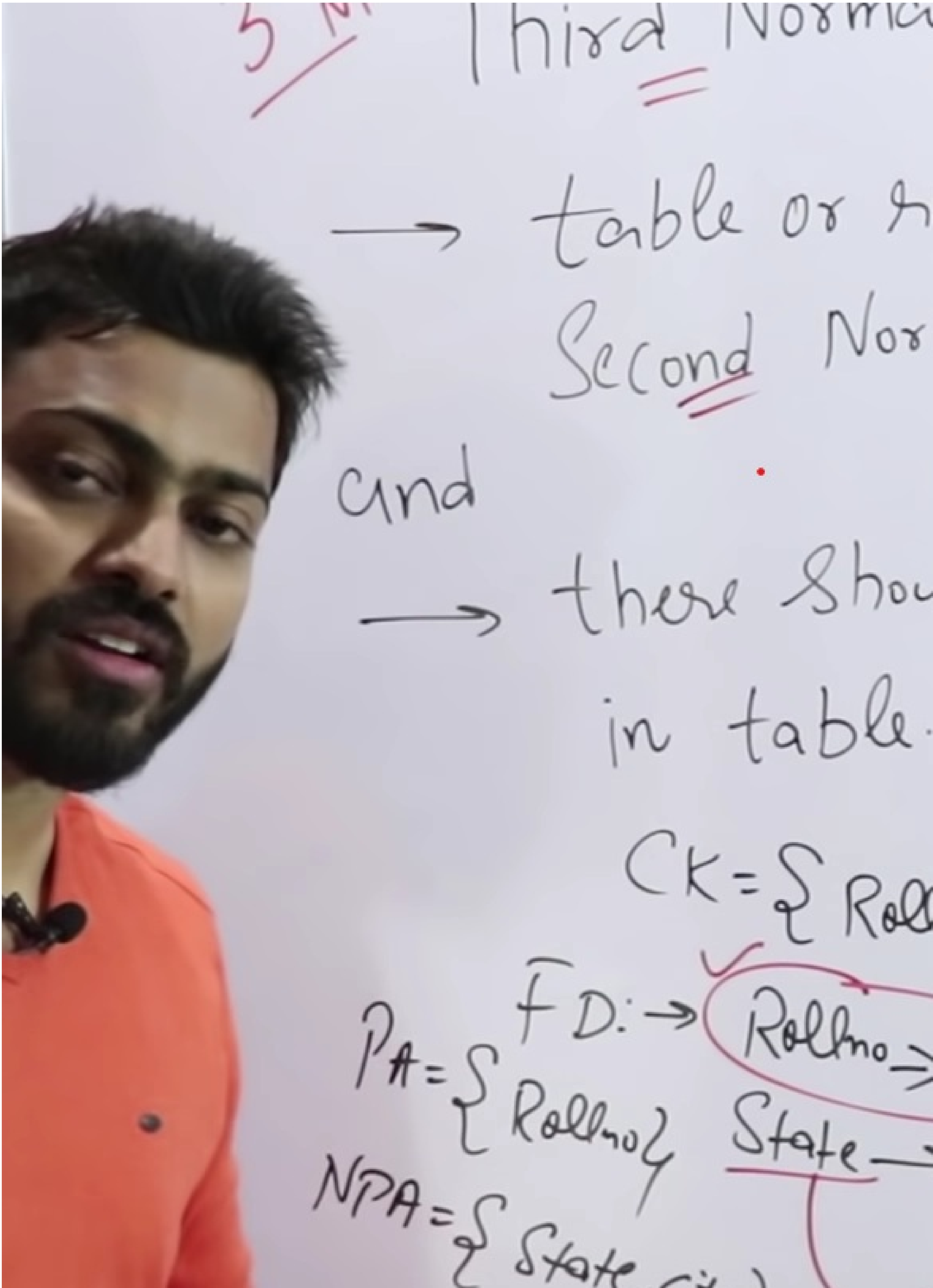
R4(P, X) (Now in table R4 FD : P → X is Full F D, hence R4 is in 2

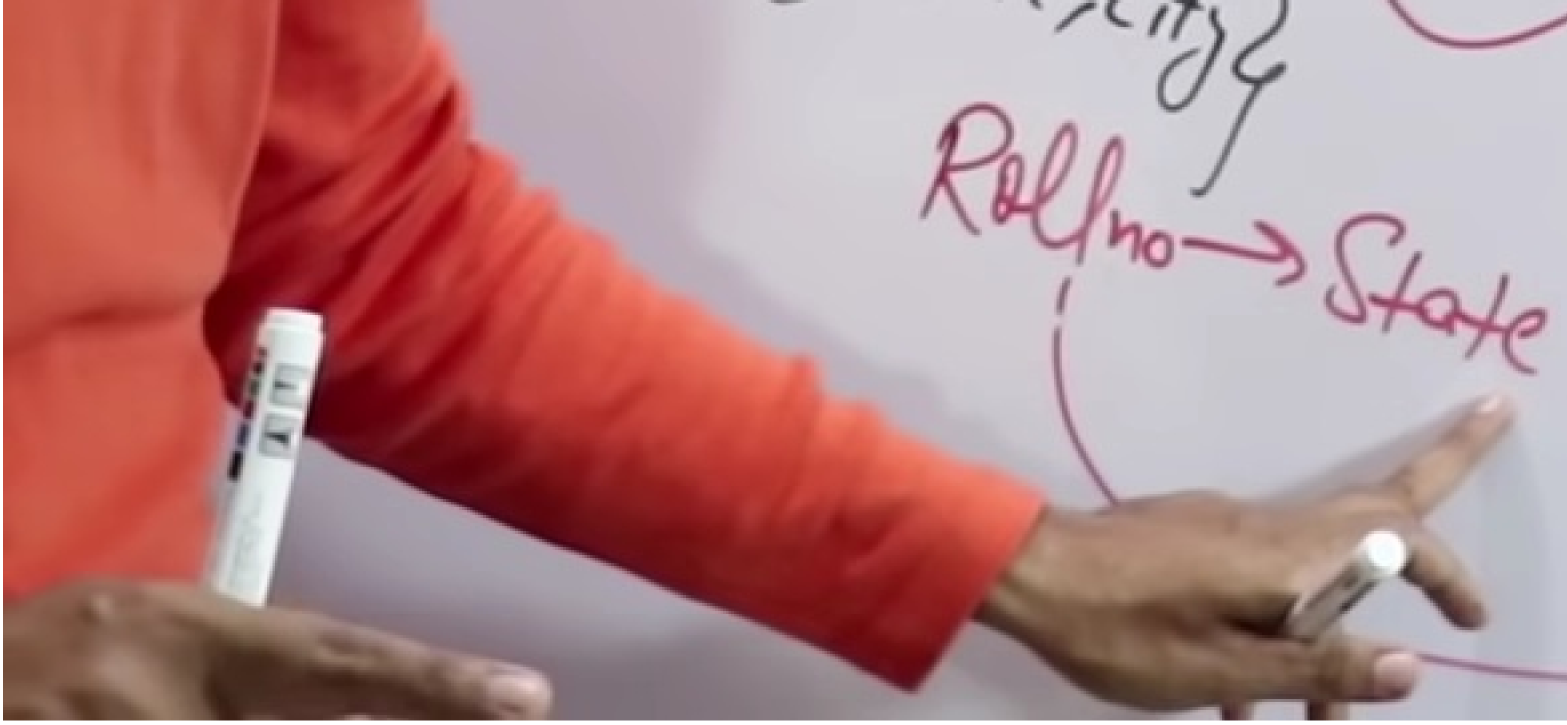
R5(W, Y) (Now in table R5 FD: W → Y is Full F D, hence R2 is in 2

And create one table for the key, since the key is PQS.

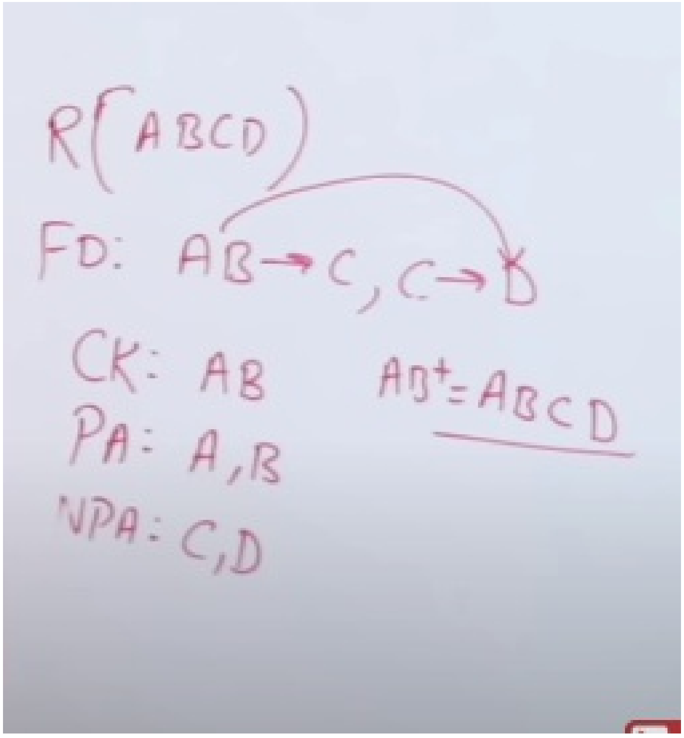
R6(P, Q, S)

- THIRD NORMAL FORM

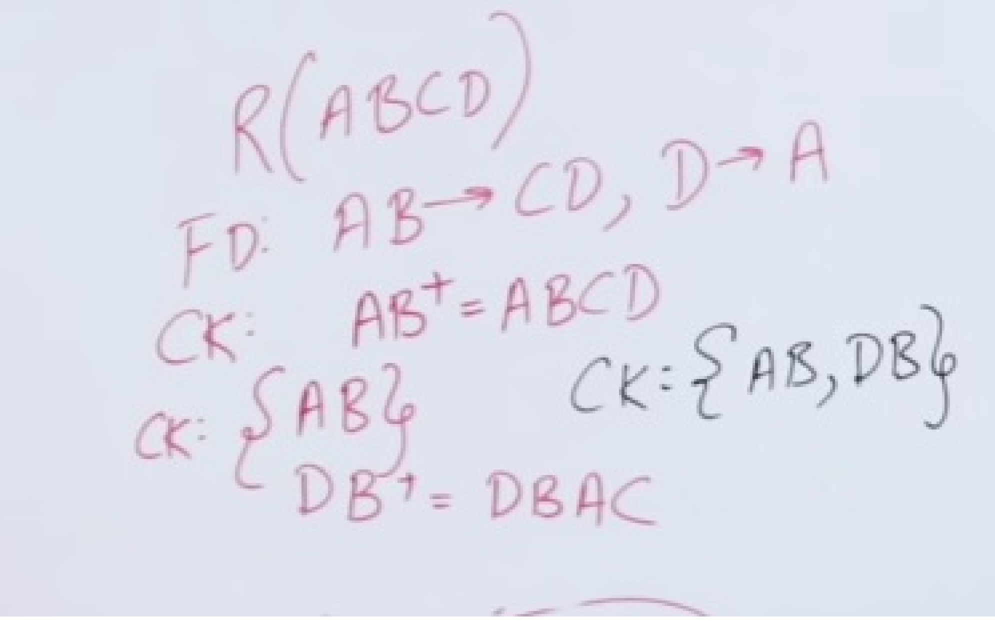




Example 1:

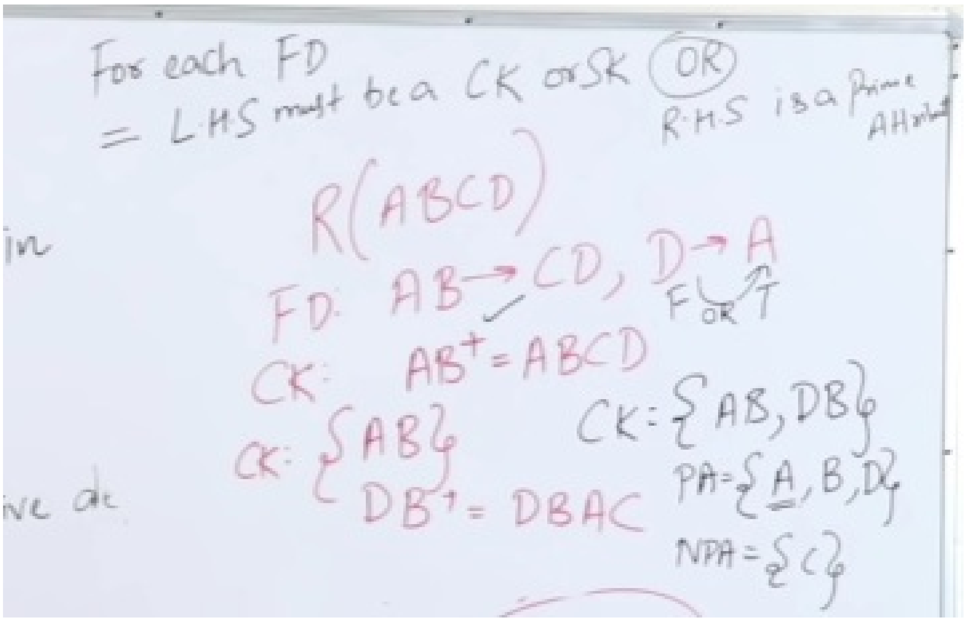


Example 2:



For a Table to be said in third normal form :

LHS of all FDs must be a super key or candidate key OR RHS of all FDs may be a Prime attribute



ABOVE TABLE IS IN THIRD NORMAL FORM

QUESTIONS FOR THIRD NF

Question 1: Given a relation R(X, Y, Z) and Functional Depende
determine whether the given R is in 3NF? If not convert it into 3 N

Ans :

R1(X, Y)

R2(Y, Z)

Question 2: Given a relation R(X, Y, Z, W, P) and Functional De
and Z → W}, determine whether the given R is in 3NF? If not conv

Ans :

Convert the table R(X, Y, Z, W, P) into 3NF:

Since all the FD = { X → Y, Y → P, and Z → W} were not in 3NF, I

R1(X, Y) {Using FD X → Y}

R2(Y, P) {Using FD Y → P}

R3(Z, W) {Using FD Z → W}

And create one table for Candidate Key XZ

R4(X, Z) { Using Candidate Key XZ }

All the decomposed tables R1, R2, R3, and R4 are in 2NF(as the
as in 3NF.

Question 3: Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Fur → R, P → ST, Q → U, U → VW, and S → XY}, determine whether th it into 3 NF.

Ans:

<u>**Convert the table R(X, Y, Z, W, P) into 3NF:**</u>

Since all the FD = { P → ST, Q → U, U → VW, and S → XY } were r

R1(P, S, T) {Using FD P → ST }

R2(Q, U) {Using FD Q → U }

R3(U, V, W) { Using FD U → VW }

R4(S, X, Y) { Using FD S → XY }

R5(P, Q, R) { Using FD PQ → R, and candidate key PQ }

All the decomposed tables R1, R2, R3, R4, and R5 are in 2NF(as well as in 3NF.

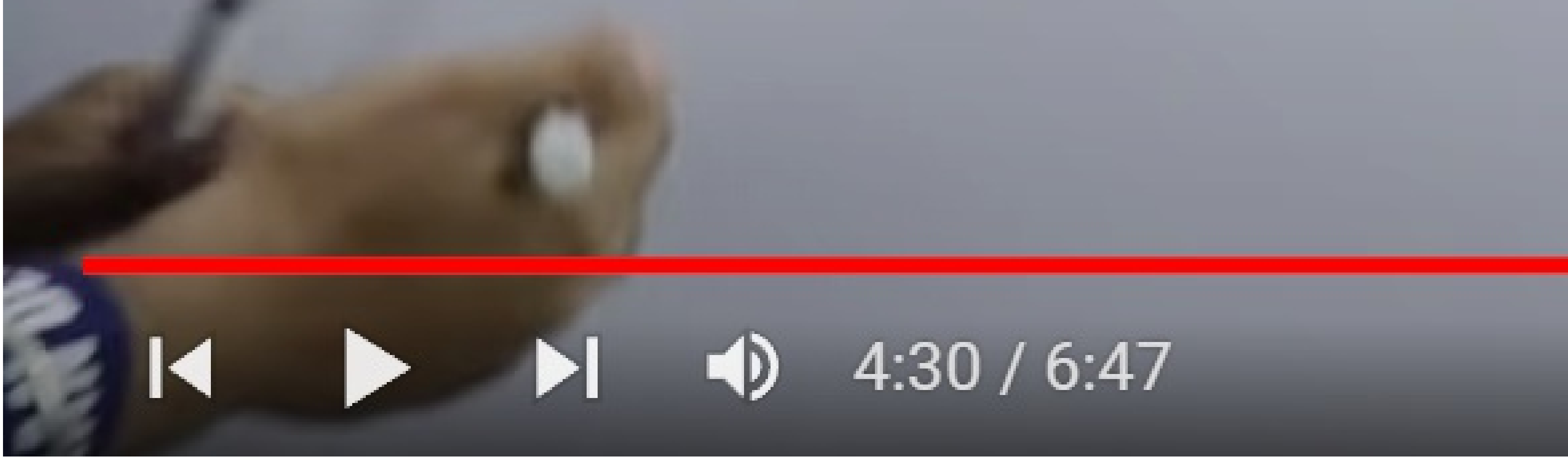
- BOYCE CODD NORMAL FORM (modification of third normal form)

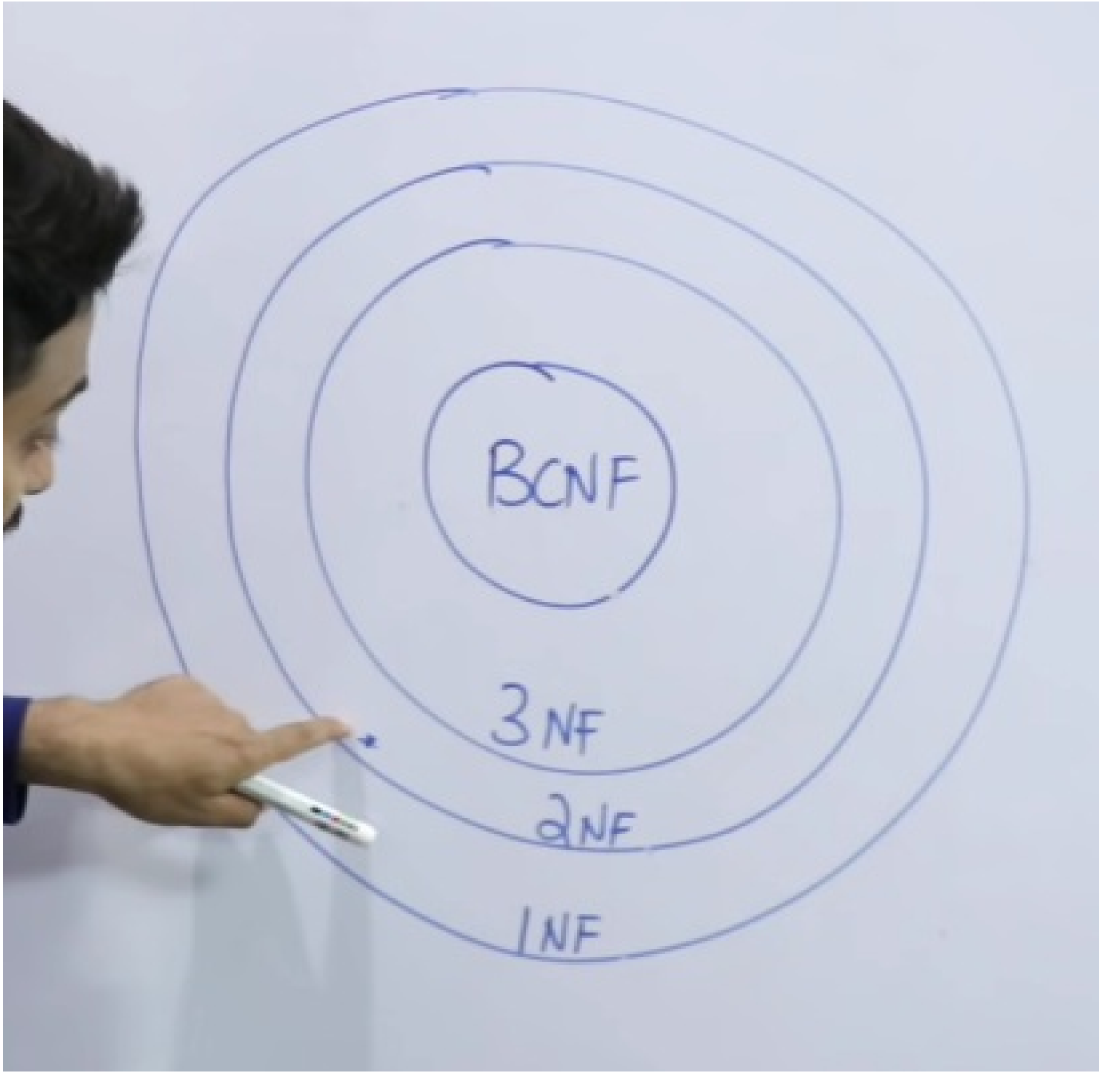
1. 3NF MEI HONA CHHAIYE
2. Lhs mei candidate key ya super key hi honi chahiye

3NF:

Student

Rollno	Name
1	Ravi
2	Varun
3	Ravi
4	Rahul





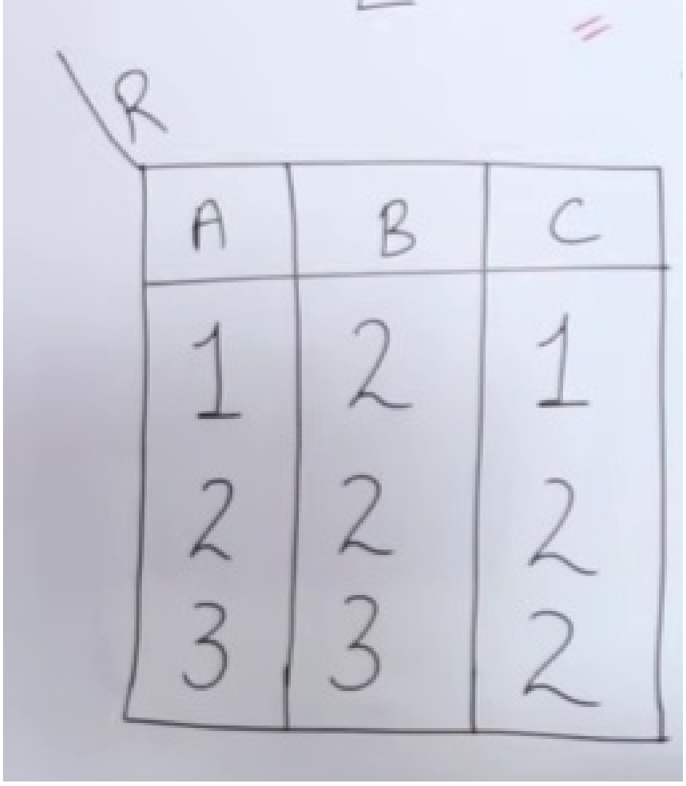
LOSSLESS/LOSSY JOIN DECOMPOSITION

While normalising a table , decomposition of the table is done. We need to check whether the decomposition is lossless or lossy.

There are 2 rules to be followed at the time of decomposition

one of them states that decomposition should be lossless

second one is dependency decomposition preservation

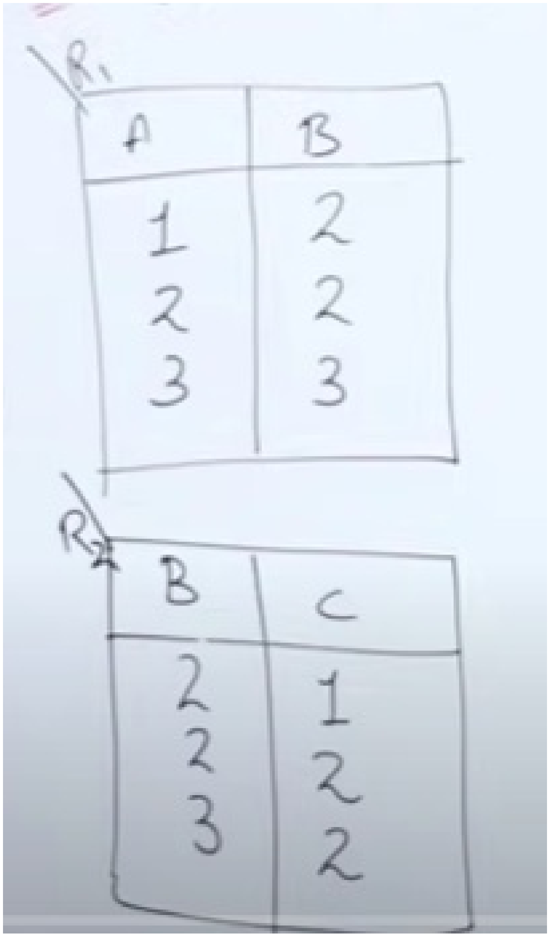


R

A	B	C
1	2	1
2	2	2
3	3	2

Lets decompose this table in to 2:

The tables should have at least one common attribute



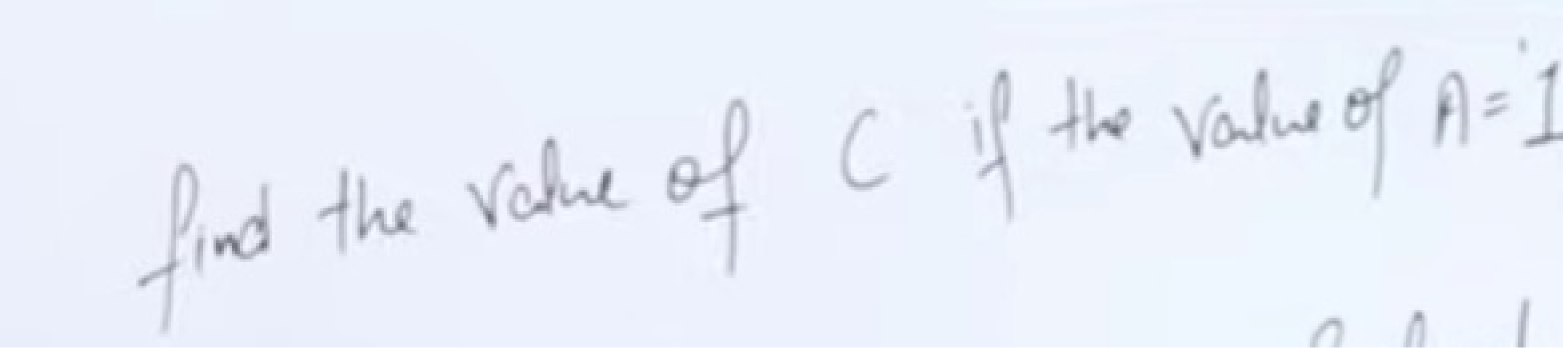
R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

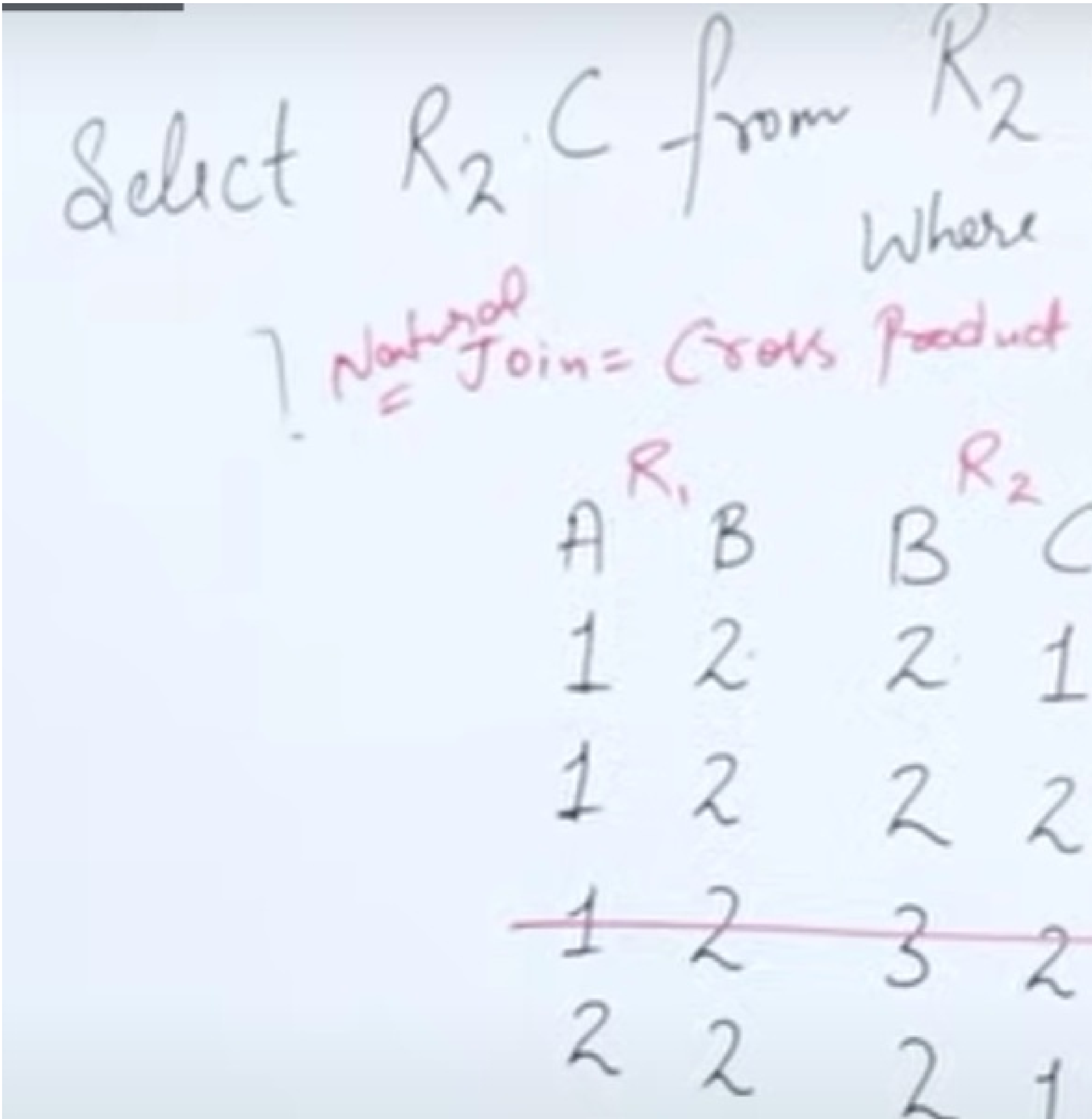
If the following instruction was to be carried out :

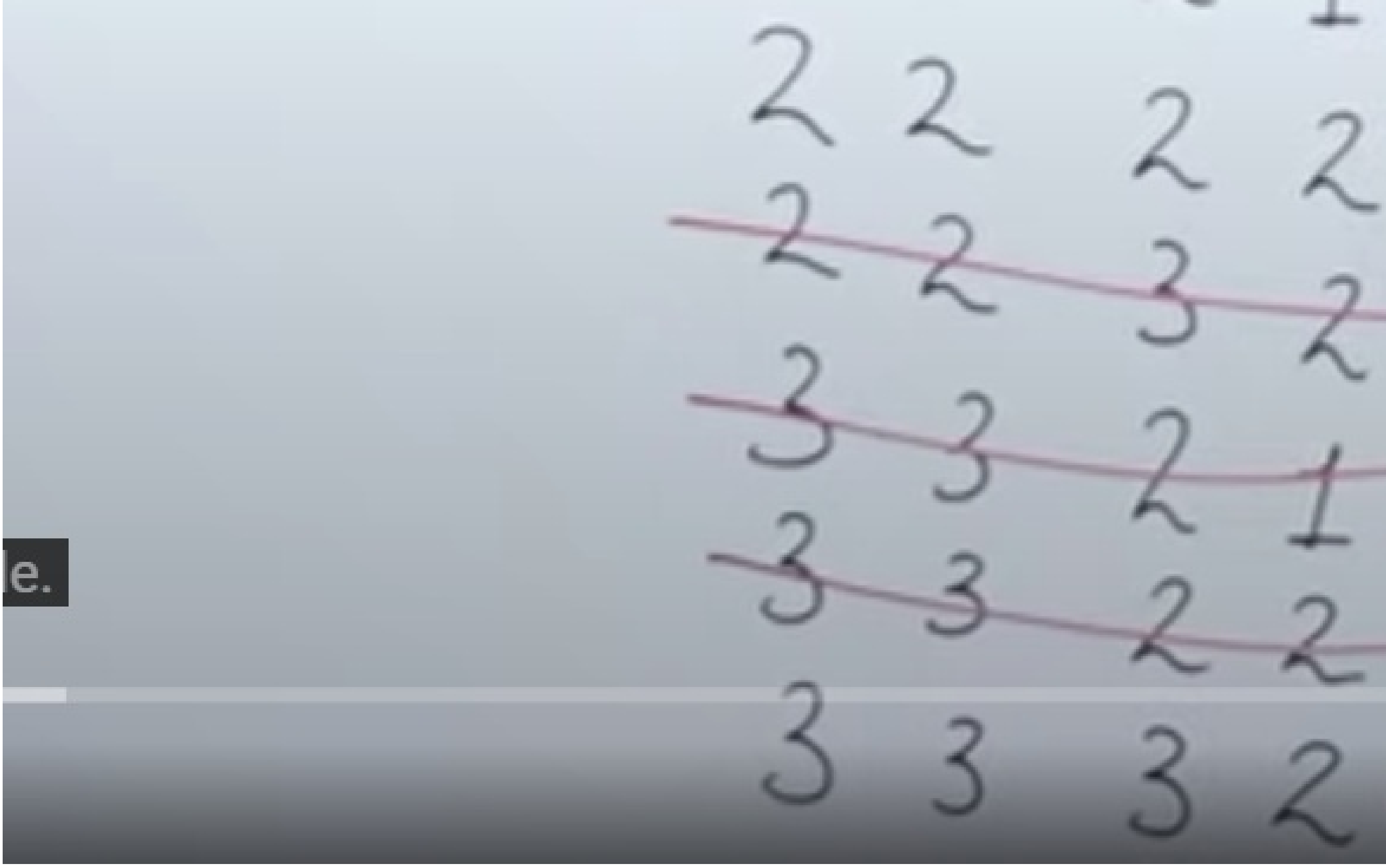


Now to execute such a query we will need to join the 2 tables:

Select R2.C from R2 natural join R1 where R1.A = ‘1’ ;

Natural join explanation :





Natural join also removes duplicate columns.

After the query is executed result :

R'

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

Now idhar we can see C column has 2 values for A=1 which is not correct if we look at the original table -> FLAW!! -> LOSSY DECOMPOSITION

The table has 5 rows instead of 3

DATA INCONSISTENCY HOGYI

The extra 2 tuples in the joined table are called spurious tuples

BUT WHY IS THIS PROBLEM OCCURING?
when dividing the table we kept B as the common attribaccute.

There are some rules for dividing the table :

- Common attribute should be a ck or sk of either R1 or R2 or both.
- R1 U R2 == R -> AB U AC =ABC
- R1^R2 != phi -> AB ^ AC = A i.e divide krne pe atleast ek common attribute ho

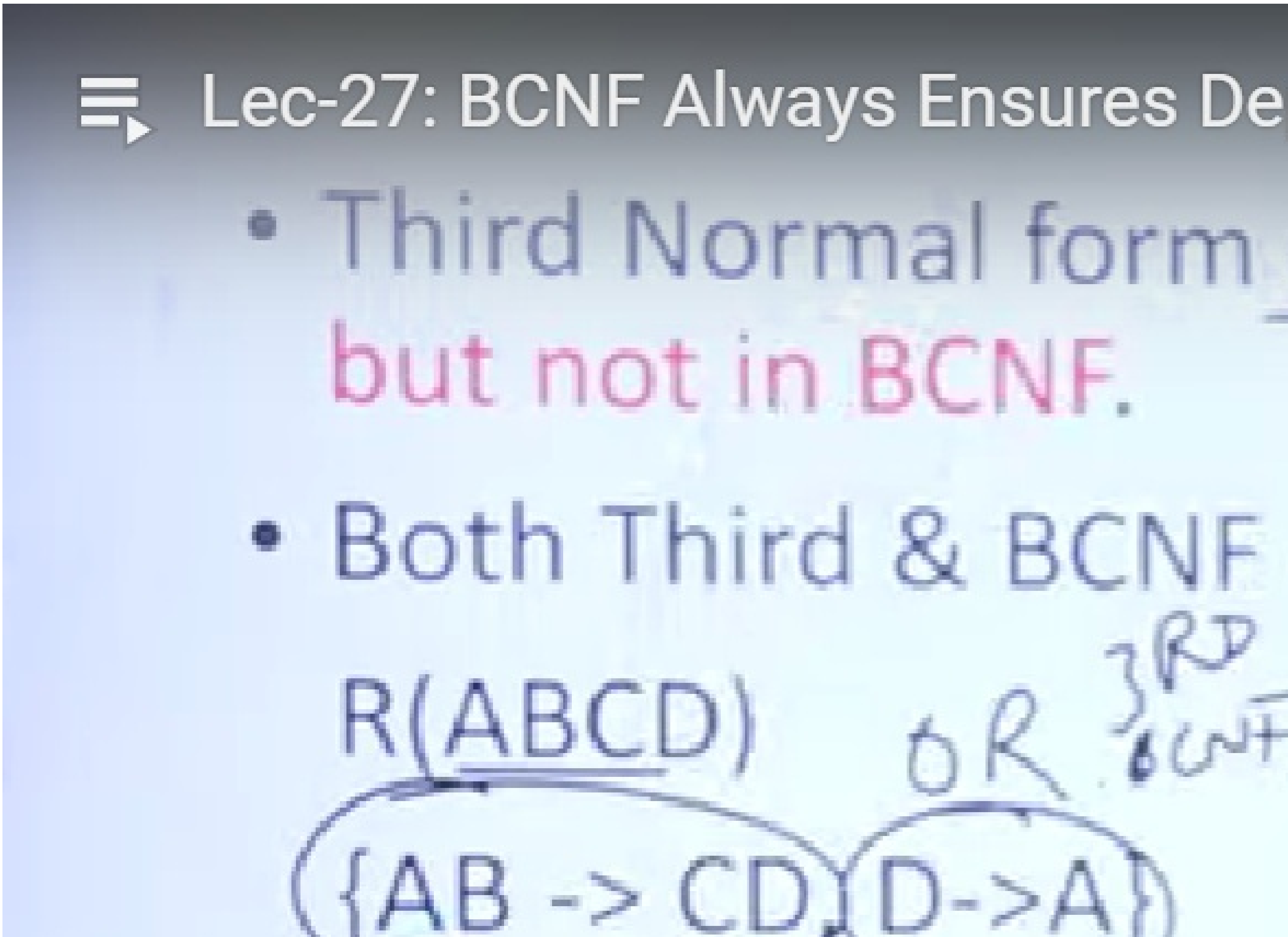
If R1(A,B) and R2(A,C) there wouldn't be any lossy decomposition

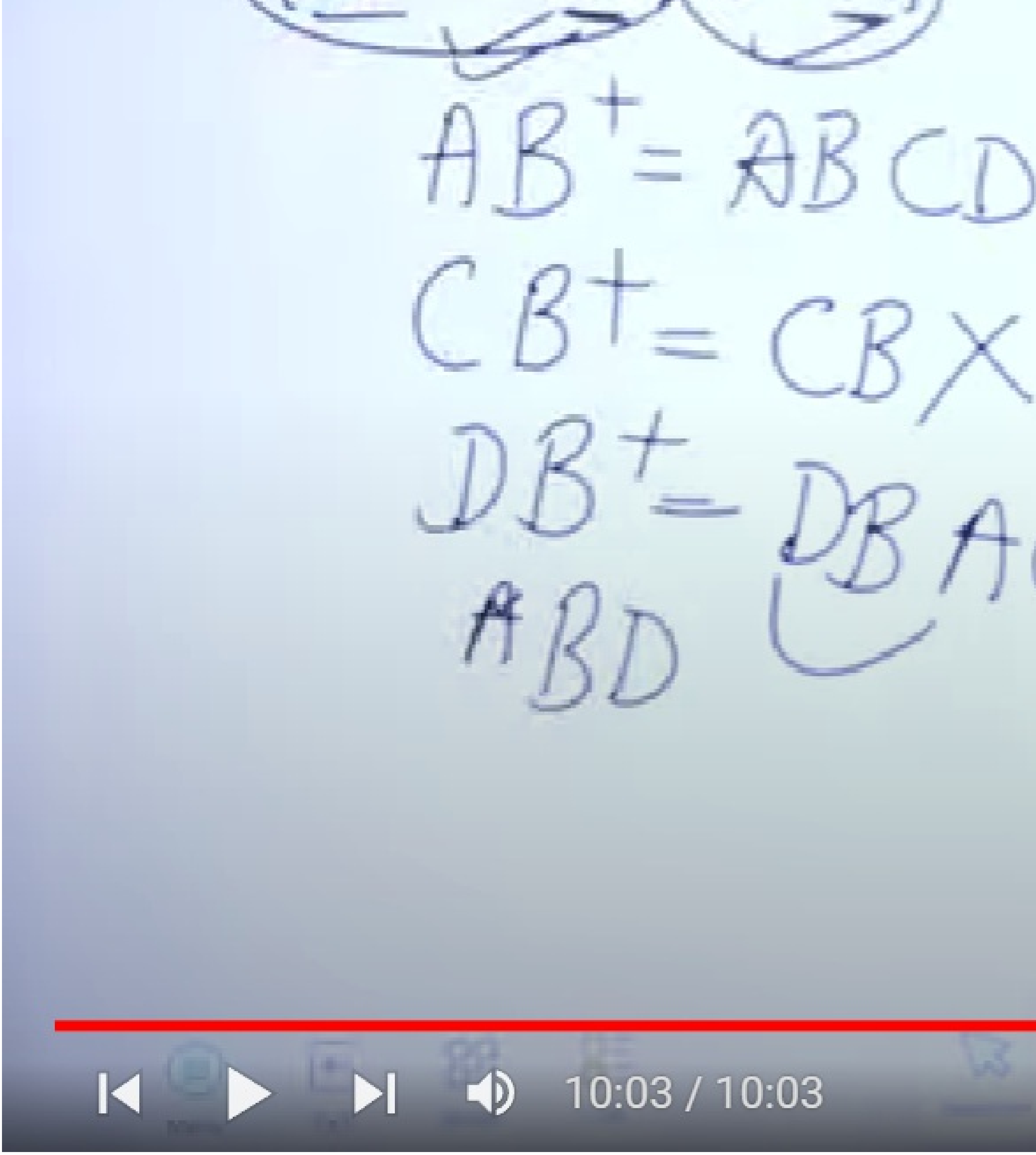
THESE SAME CONDITIONS ARE USED IN THE FIFTH NORMAL FORM.

DOES BCNF ENSURE DEPENDENCY PRESERVING DECOMPOSITION?

Dependencies should be preserved while dividing the table

Original dependencies should be preserved





Dependency was not preserved in this eg as R2 R1 mei AB ka closure doesn’t give CD like in the original one.

So conclusion : **BCNF doesn’t always ensure dependency preserving decomposition**

MINIMAL COVER

We need to check whether the given functional dependencies are irreducible or not

For the following Functional dependencies, find the C

$\{ A \to B,\ C \to B,\ D \to ABC,\ AC \to D \}$

Step 1 : saare right vaalon ko tod tod k likhdo if they aren't single

$A \to B,\ C \to B,\ D \to A,\ D \to B,\ D \to C,\ AC \to D$

Step 2 : remove all the redundant fds

$D \to ABC,\ AC \to D\}$

$A^+ = A$
$C^+ = C$
$D^+ = DBC$
$D^+ = DABC$

Step1

$A \to B,\ C \to B,\ \boxed{D \to A},\ \boxed{\cancel{D \to B}},\ \boxed{D \to C},\ AC \to D$

Step2

Redundant

$D^+ = DAB$
$AC^+ = ACB$

$\{A \to B,\ C \to B,\ D \to A,\ D \to C,\ AC \to D\}$

Step 3 : lhs mei single attribute krna hai

Like int his eg the fd AC -> D

Lets check by deleting A

C k closure mei pr A kahi nhi aara so we cant delete A

Lets check by deleting C

A k closure mei sirf B aara hai

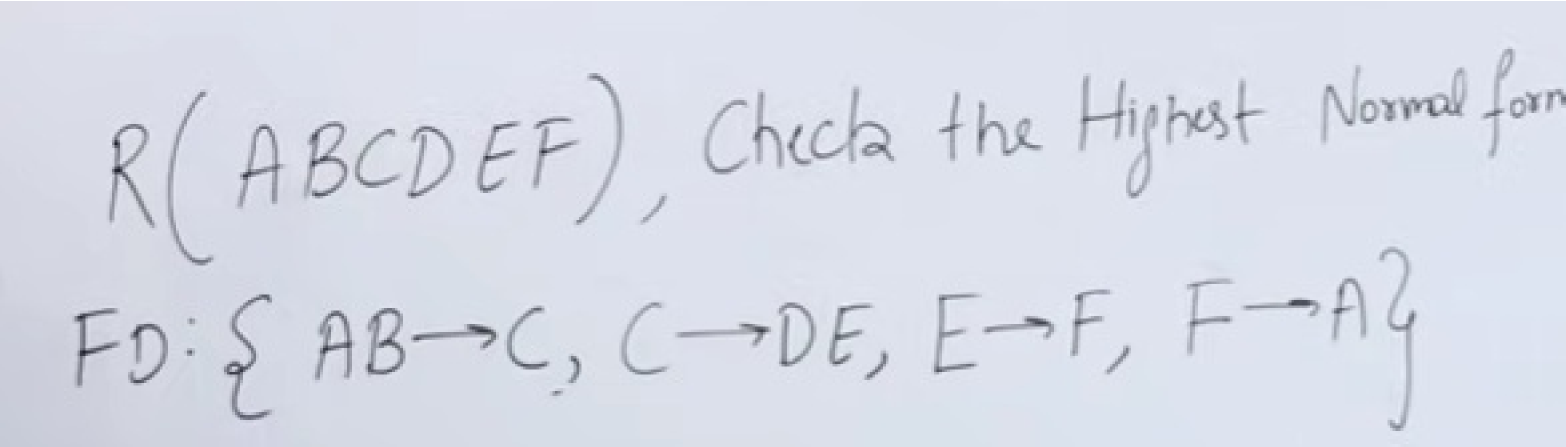
So we cant delete any attribute from AC

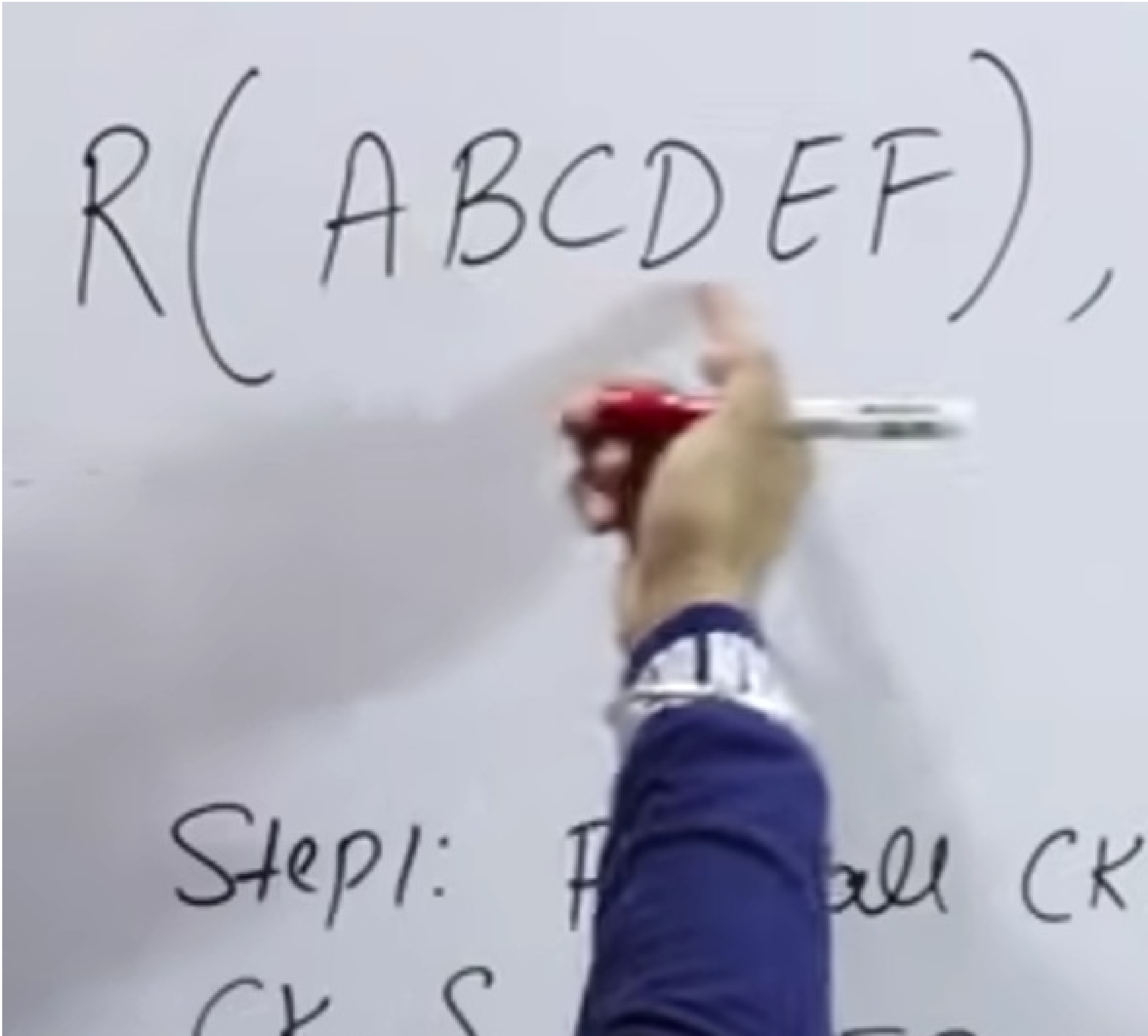
Step 4 : D-> A and D-> C ko combine krke likhdo vaapis

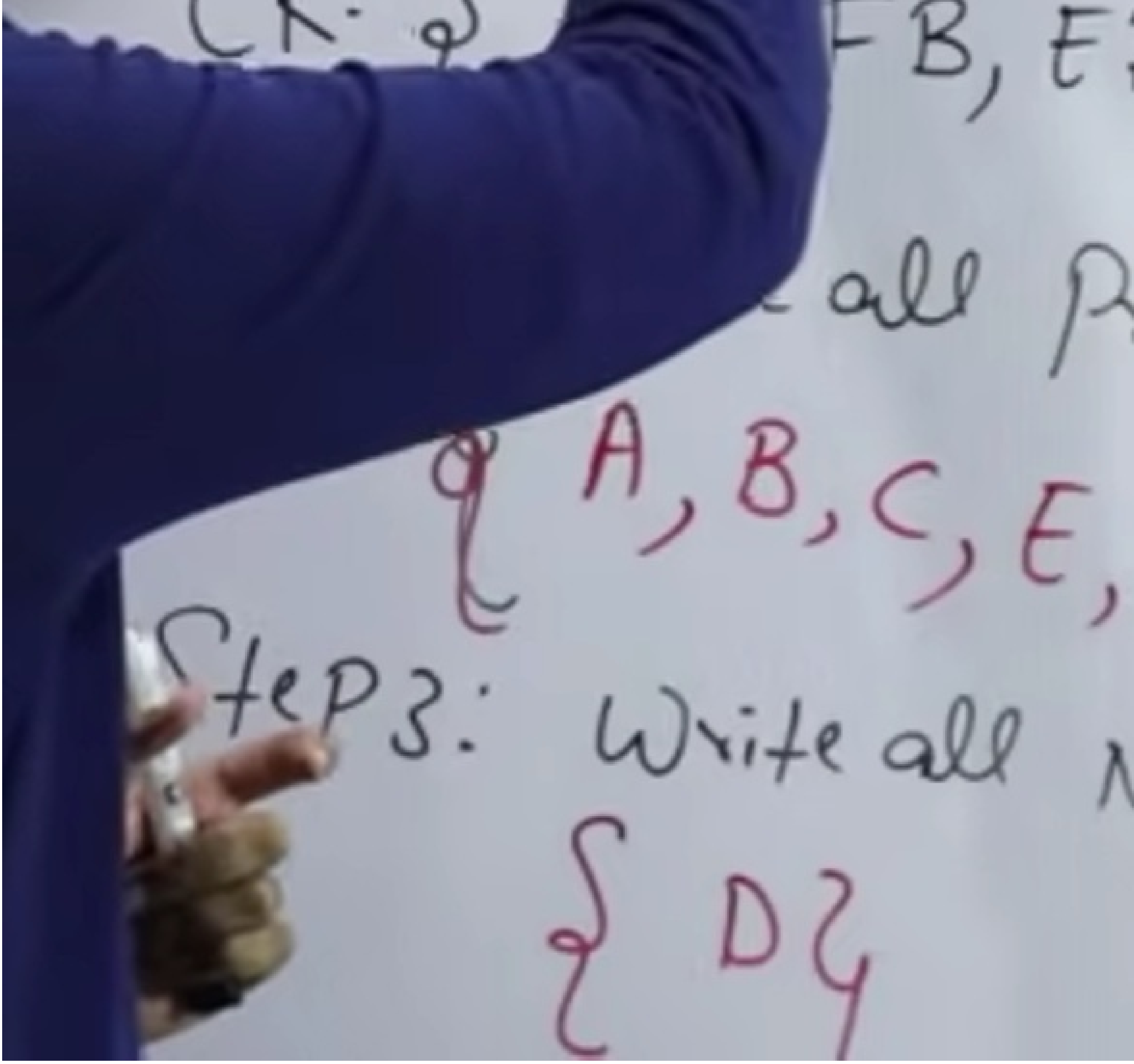
So resultant minimal cover wold be :

A->B , C->B, D->AC , AC->D

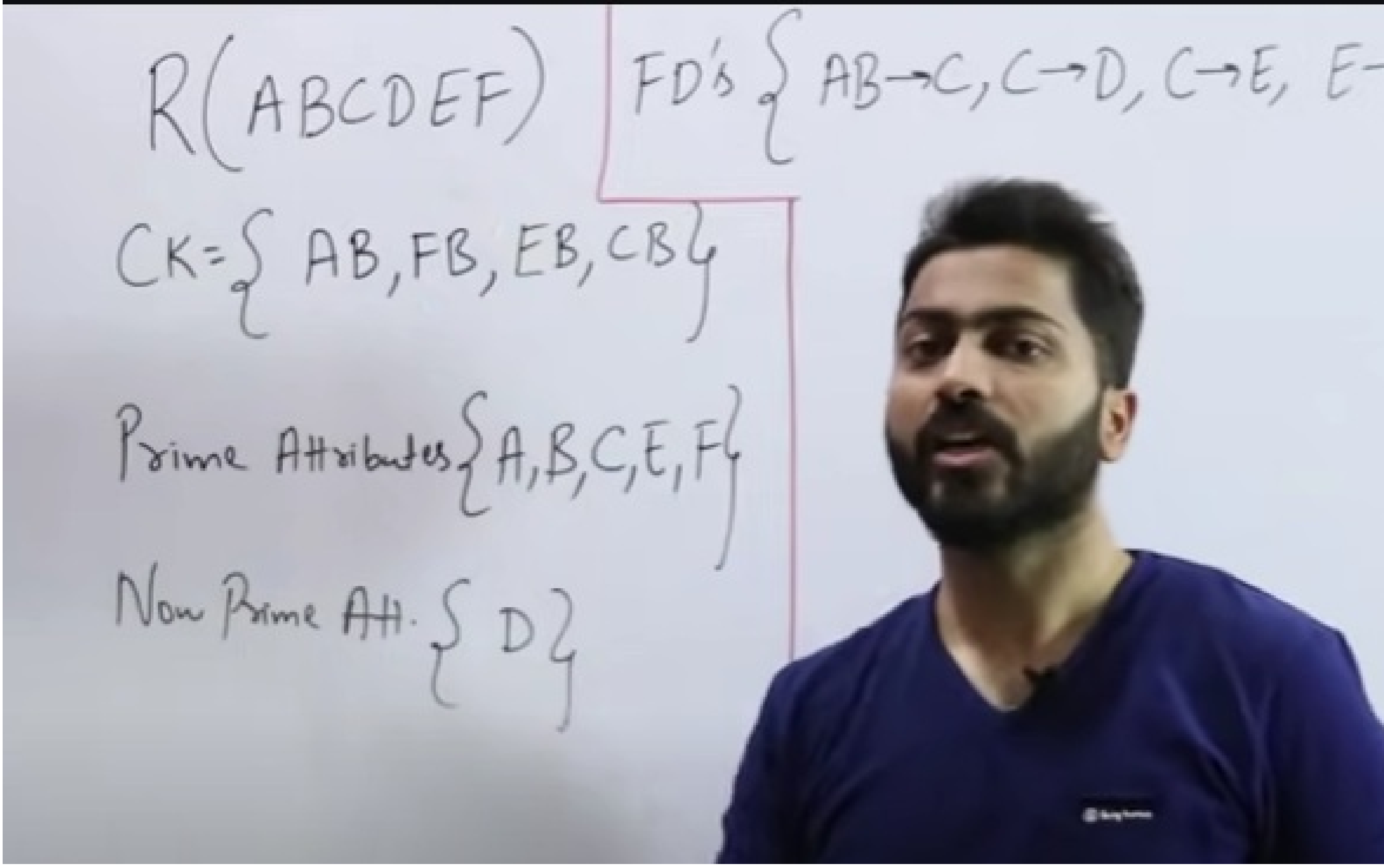
QUESTION







HOW TO FIND THE NORMAL FORM OF A RELATION?

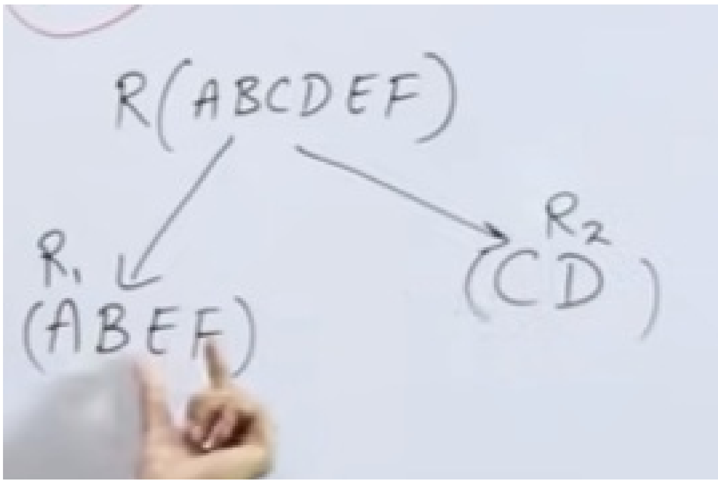


First we checked kaunsa normal form mei the above table is

So its in the first nf only and to make the table in second nf we will have to dibide the table in two halves

Here C-> D is creating a problem so usey alag krdete hain

While decomposing vo 2 rules yaad rkhne pdenge

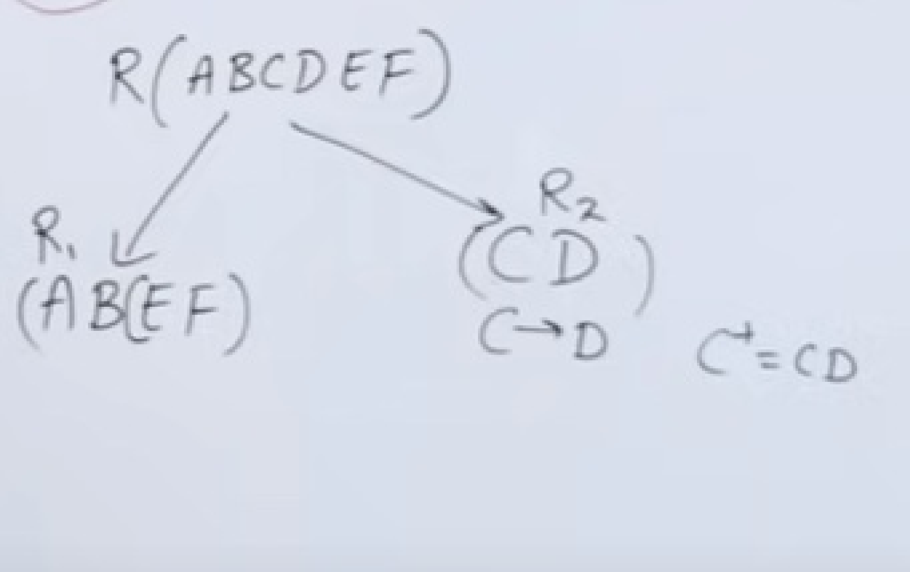


cant decompose like this

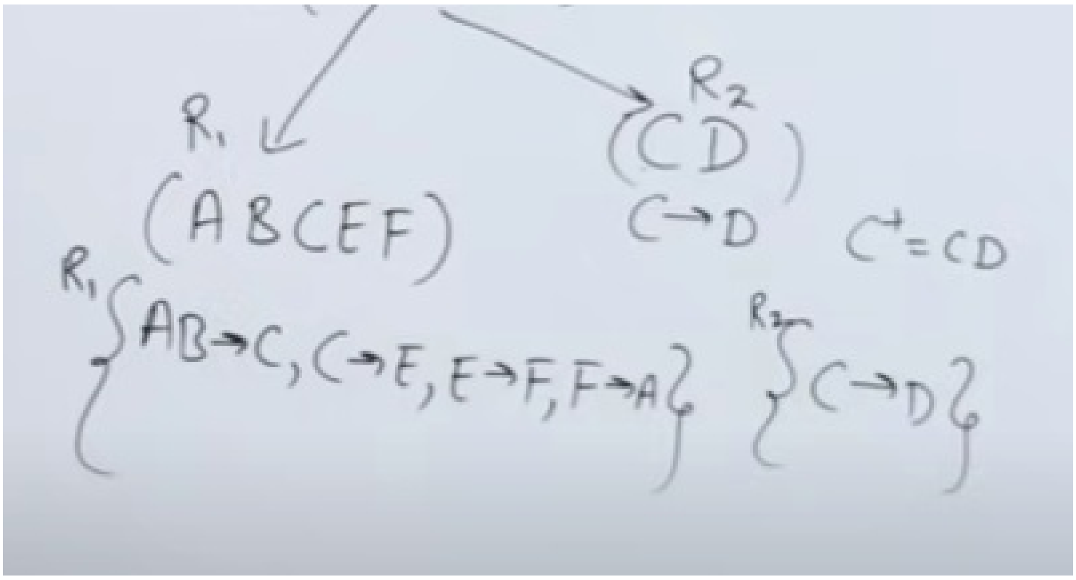
Now checking for candidate keys in R2

C ka closure indicates it's a canfidate key in R2

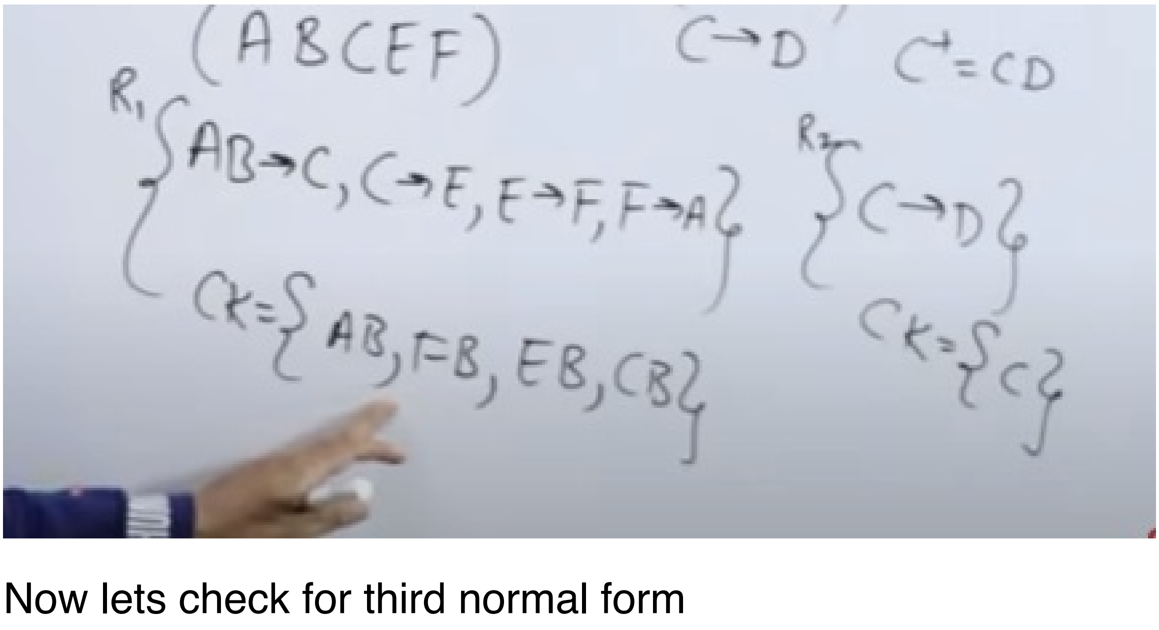
So usey common rkhdete hain R1 mei bhi



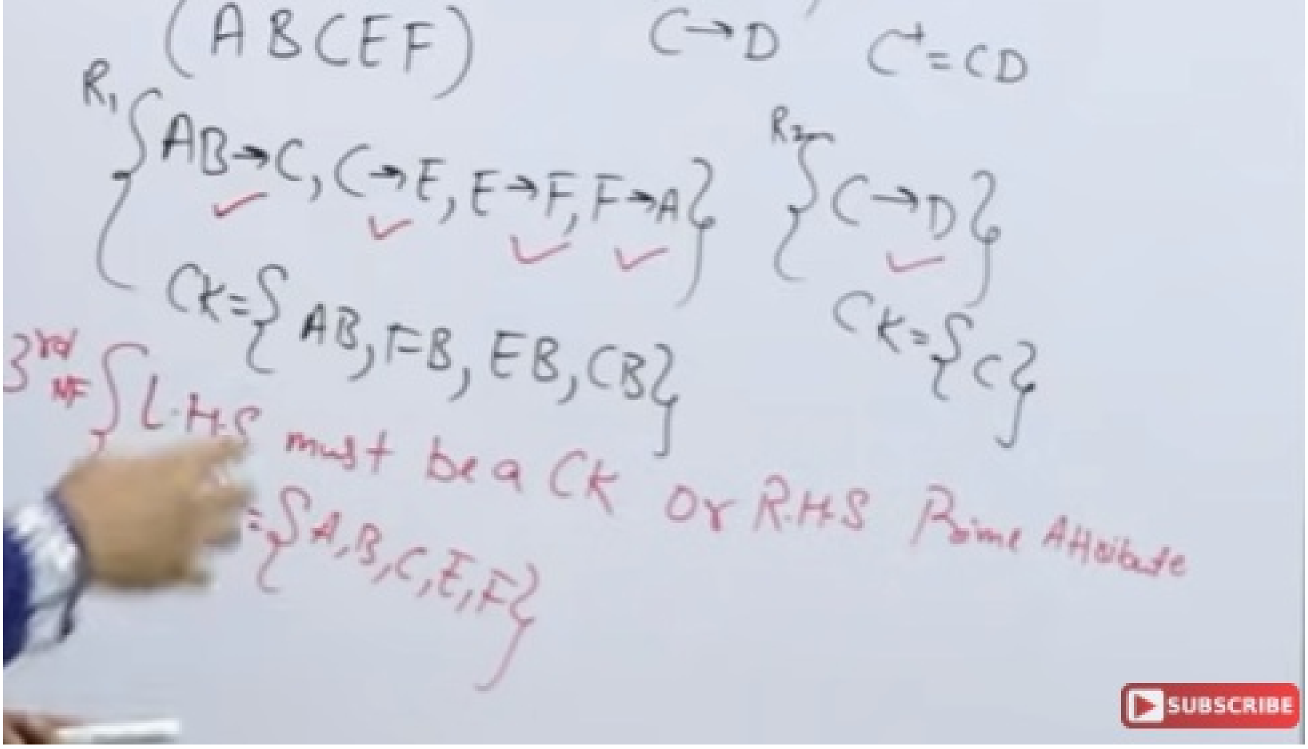
Issey functional dependency bhi hogyi preserve



Now ye second nf mei hain dono k dono tables



Now lets check for third normal form



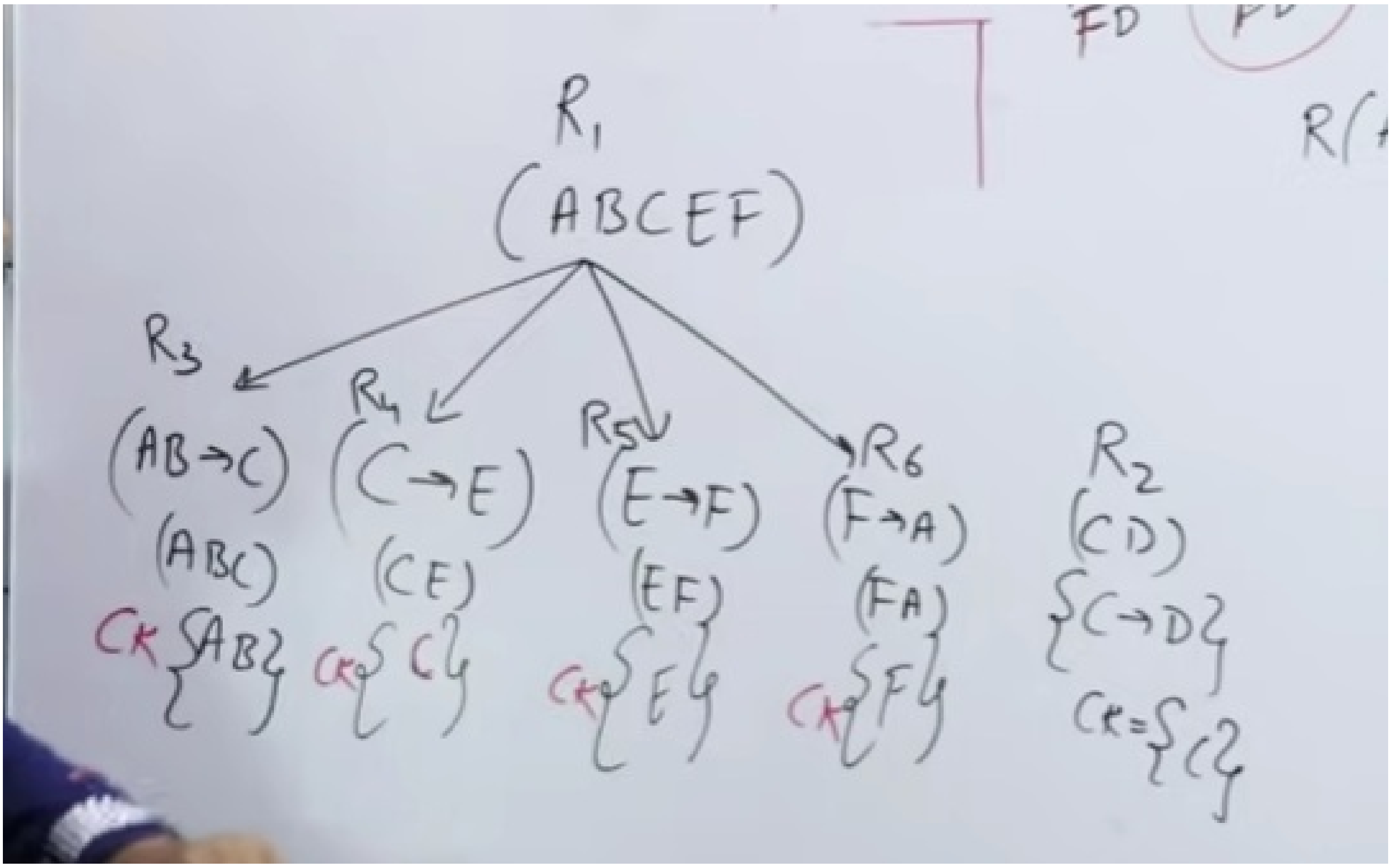
Yes it is

Lets check for bcnf also

R2 to hai bcnf mei

But R1 k C->E E->F F->A mei dikkat aarhi hai

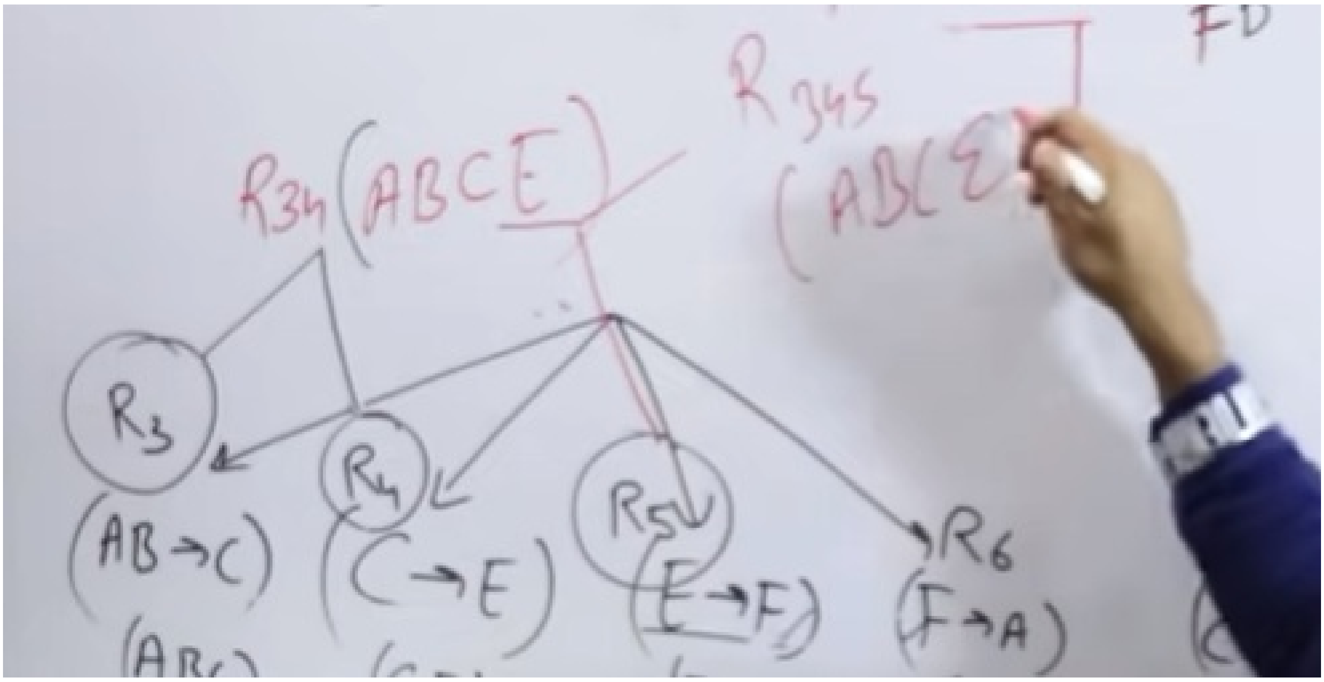
Lets decompose R1



REDUNDANCY 0 % HOGYIII

If we wanted to join R3 and R5 we wont be able to kyunki dono mei kuch common nhi so directly join nhi hopayega

For that we will have to first combine R3 and R4



COVER AND EQUIVALENCE OF FUNCTIONAL DEPENENCIES

Checking whether 2 fds are equivalent or not

$X = \{A \to B, B \to C\}$ | $Y = \{A \to B, B \to C, A \to C\}$

First check if x covers y or not

And if Y covers X or not

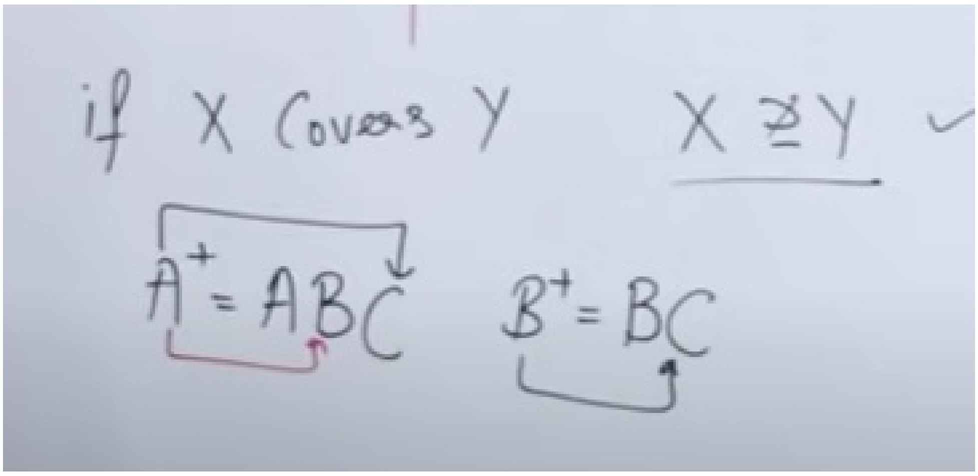
if X covers Y $\underline{X \supseteq Y}$

if Y covers X $X \subseteq Y$

If both things are true so we conclude x is equivalent to Y

$\boxed{X \equiv Y}$

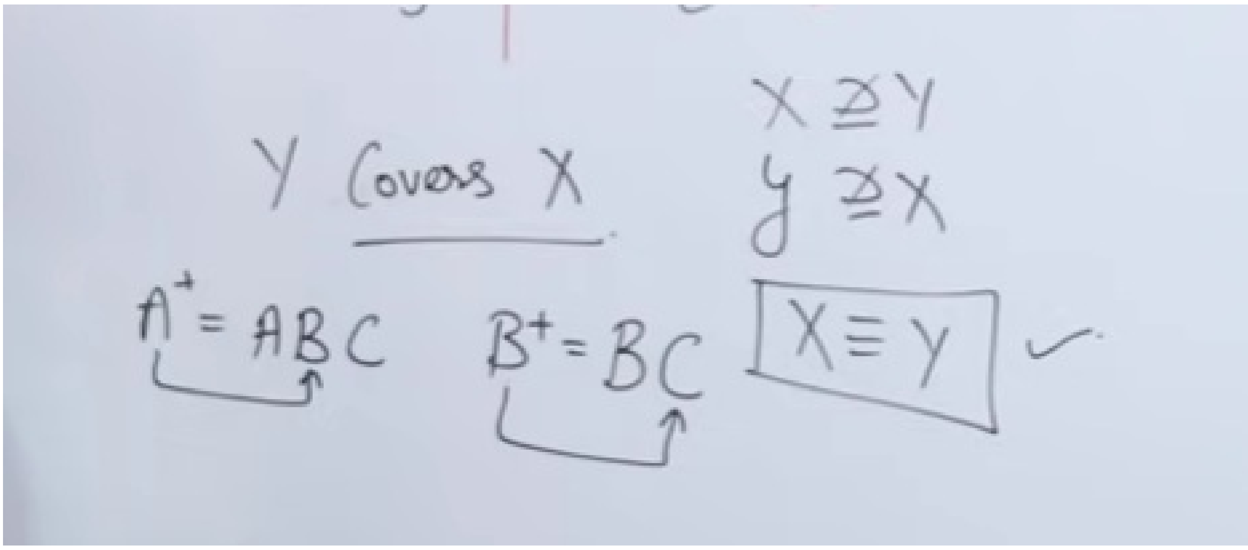
SO LET US FIRST CHECK IF X COVERS Y OR NOT

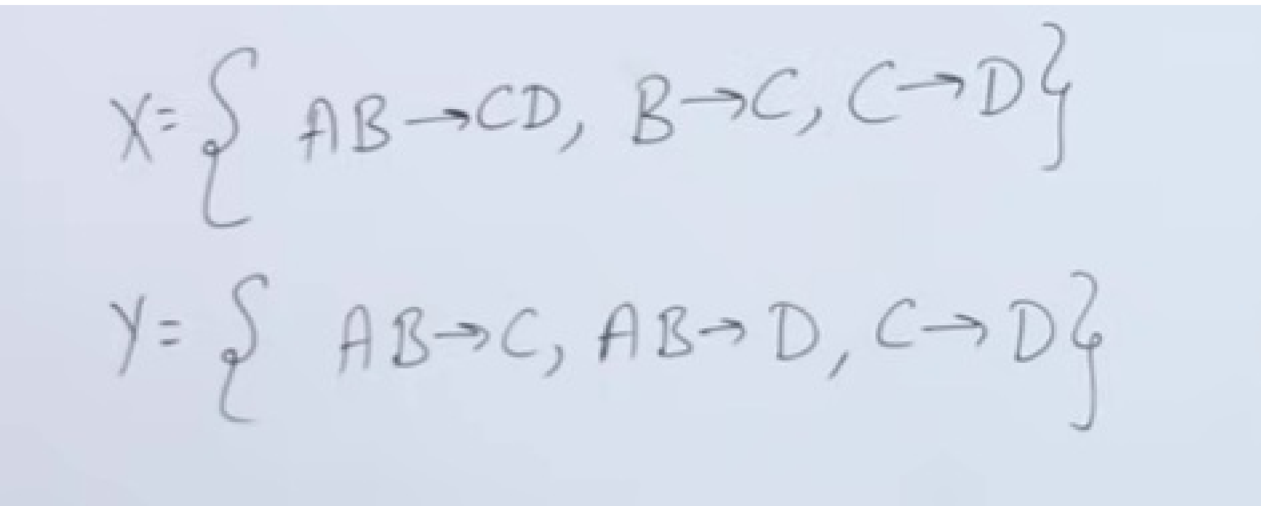


TAKE Y’S FUNCTIONAL DEPENDENCIES AND CHECK THEIR CLOSURE FROM X!

NOW WE CAN CONCLUDE THAT X COVERS Y

LETS CHECK FOR Y COVERS X





X Covers Y

$\underline{X \supseteq Y}$

$X = \{ AB-$

$Y = \{ AB$

$AB^+ = ABCD$

$C^+ = CD$

Y Covers X. $\boxed{\overset{\times}{Y \supseteq X}}$

$X = \{ AB \to CD, \boxed{B \to C}, C \to$

$Y = \{ AB \to C, AB \to D, C \to$

$AB^+ = ABCD$

$B^+ = B$

$C^+ = CD$

$\boxed{X \not\equiv Y}$