* Join operation :-

The " " in one of the most useful and commonly used operation to extract information from two or more relation. The join operation denoted by '⋈' is used to join two relation for a new relation on the basis of a common attribute present in the two operand relationship.

STUDENT

Rollno	name	Dno
1	aa	101
2	ab	102
3	bb	NULL

DEPARTMENT

Dno	Dname
101	CS
102	Mgmt
103	Law

i) Student $\bowtie_{\text{student.Dno} = \text{Department.Dno}}$ Department

Rollno	name	Dno	Dno	Dname
1	aa	101	101	CS
2	ab	102	102	Mgmt

ii) Student $\bowtie_{\text{student.Dno} < \text{Department.Dno}}$ Department

Rollno	name	Dno	Dno	Dname
1	aa	101	102	Mgmt
1	aa	101	103	Law
2	ab	102	103	Law

- Equi join :-

" " operation is one in which the join condition consist only of the equility condition.

$$R_1 \bowtie_{R_1.\text{attributes } 1 = R_2.\text{attributes } 3} R_2$$

- Natural join :-

If one of the two identical attributes is removed from the result of

equi join it is known as a ~~not~~ natural join. It is denoted by

$R_1 *_{R_1.\text{attribute } 1 = R_2.\text{attribute } 2} R_2$

Q.

i) Display student rollno and ~~dep~~ Dno from the relation student.

ii) Display the student details whose rollno is 2.

iii) " rollno and name for the students whose Dno is 101

iv) Display student details with their department details.

v) " name, Dno and Dname for all students.

vi) " Rollno, " and " " the " whose rollno is 2.

Ans :- i)

$\pi_{\text{rollno, Dno}}(\text{STUDENT})$

Ans :- ii)

$\sigma_{\text{rollno} = 2}(\text{STUDENT})$

Ans (iii) :-

$\pi_{\text{rollno, name}}(\sigma_{\text{Dno} = 101}(\text{STUDENT}))$

Ans (iv) :- $\text{STUDENT} \bowtie_{\text{student.Dno} = \text{department.Dno}} \text{DEPARTMENT}$

Ans ~~(v) :- STUDENT student.~~

Ans (v) :- $\pi_{\text{name, Dno, Dname}}(\text{STUDENT} \bowtie_{\text{student.Dno} = \text{department.Dno}} \text{DEPARTMENT})$

- Outer join :-

" " selects all the tuples satisfying the join condition along with the tuples for which no tuples from the relation satisfy the join condition. There are three type of outer join operations namely :-

i) left outer join
ii) right " "
iii) full " "

→ * Left outer join :-

The " " " includes all the tuples from both the relations satisfying the join condition along with all the tuples in the left relation that do not have a corresponding " in the right " . The tuples from left relation not satisfying the join condition are.

(a) Concanated with the ~~tuple~~ tuples having null values for all the values from the right relation.

$$R_1 \bowtie_{\langle condition \rangle} R_2$$

For eg :-

Student

Roll no	name	Dno
1	aa	101
2	ab	102
3	bb	103
4	ba	NULL
5	cc	NULL

department

Dno	Dname
101	CS
102	mgmt
103	Law
104	msw

Student ⋈ $_{student.Dno = department.Dno}$ Department

Rollno	name	Dno	Dno	Dname
1	aa	101	101	CS
2	ab	102	102	mgmt
3	bb	103	103	Law
4	ba	NULL	NULL	NULL
5	cc	NULL	NULL	NULL

Right outer join :-

The " " " includes all the tuples from both the relations satisfying the join condition along with all the tuples in the right relation that do not have a corresponding " in the left ". The tuples from right relation not satisfying the join condition are concatenated with the tuples having null values for all the values from the left relation.

$$R_1 \bowtie_{<condition>} R_2$$

For eg:

Student $\bowtie_{student.Dno = department.Dno}$ Department

Roll no	Name	Dno	Dno	Dname
1	aa	101	101	CS
2	ab	102	102	mgmt
3	bb	103	103	Law
Null	Null	Null	104	MSW

Full outer join :-

The " " " combines the results of both the left and the right outer join. the resultant relation contains all records from both the relations with null values for the missing corresponding tuples on either side

$$R_1 \bowtie_{<condition>} R_2$$

For eg:

Student ⋈ student.Dno = department.Dno Department

Rollno	name	Dno	Dno	Dname
1	aa	101	101	CS
2	ab	102	102	mgmt
3	ba	103	103	Law
4	bb	null	null	null
5	cc	null	null	null
null	null	null	104	mcu

* Division operation :-

The " " denoted by '÷', is useful for queries of the form for all objects having all the specified properties. Consider a relation R_1 having exactly two attributes A and B and a " R_2 " just just one " B with the same domain as in R_1. The division operation $R_1 \div R_2$ is defined as the set of all values of attribute A such that for every value of attribute B in R_2, there is a tuple in R_1.

R_1

A	B
a_1	b_1
a_1	b_2
a_1	b_3
a_1	b_4
a_2	b_1
a_2	b_2
a_3	b_2
a_4	b_2
a_4	b_4

R_2

B
b_2

R_3

B
b_2
b_4

R_4

B
b_1
b_2
b_4

$R_1 \div R_2$

A
a_1
a_2
a_3
a_4

$R_1 \div R_3$

A
a_1
a_4

$R_1 \div R_4$

A
a_1

Aggregate function :-

" " take a collection of values as input, process them and return a single value as the return. The aggregate function are - sum, max, min, count, Avg. The $\mathcal{I}$ is denoted be '$\mathcal{I}$'.

$$\mathcal{I}_{\text{sum (attribute)}} (R)$$

Q: Write queries in relational algebra for the following based on book database

i) Retreive the city, ph_no of the author whose name is Anjana Das

ii) Retreive name, address and ph_no of all the publishers located in kannur district.

iii) Retreive the title and price of all the textbook with a page_count > 600

iv) Retreive ISBN, Title and price of all the books belonging to either novel or language book category.

v) Retreive the ID, name, address and ph_no of publishers publishing novels.

vi) Retreive the title and price of all the books published by oxford publication.

Ans:-

i) $\pi_{\text{city, ph\_no}} (\sigma_{\text{author\_name = 'Anjana Das'}} (\text{AUTHOR}))$

ii) $\pi_{name, address, ph\_no}(\sigma_{district = 'Kamrup'}(PUBLISHER))$

iii) $\pi_{title, price}[\sigma_{page\_count > 600}(BOOK) \bowtie_{book.page\_count = textbook.page\_count} (TEXTBOOK)]$

iv) $\pi_{ISBN, title, price}$

v) $\pi_{ID, name, address, ph\_no}(BOOK \bowtie_{book.publisher = novel.publisher} NOVEL)$

vi) $\pi_{title, price}(\sigma_{publisher = 'oxford'}$

vi) $\pi_{title, price}[\sigma_{publisher = 'oxford'}(BOOK)]$

Ar-

i) $\pi_{city, ph\_no}(\sigma_{name = "Anjana Das"}(AUTHOR))$

ii) $\pi_{name, address, ph\_no}(\sigma_{district = "Kamrup"}(PUBLISHER))$

iii) $\pi_{title, price}(\sigma_{category = "textbook" \wedge page\_count > 600}(BOOK))$

iv) $\pi_{ISBN, title, price}(\sigma_{category = "language" \vee category = "novel"}(BOOK))$

v) $\pi_{PID, name, address, ph\_no}[\sigma_{category = "novel"}(PUBLISHER \bowtie_{PUBLISHER.PID = BOOK.PID} BOOK)]$

vi) ~~$\pi_{title, price}[\sigma_{Pname = "oxford"}(P$~~

vi) $\pi_{title, price}[\sigma_{Pname = "oxford\ publication"}(BOOK \bowtie_{BOOK.PID = PUBLISHER.PID} PUBLISHER)]$

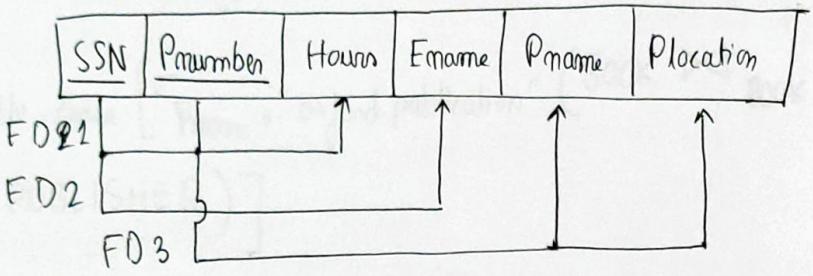
---

* Normalization :-

1) Remove Redundancy
2) " Anomalies

* 2nd normal form (2NF):

The 2nd normal form based on the concept of full functional dependency. A functional dependency $X \rightarrow Y$ in a full " " if removal of any attribute A from X means that the dependency does not hold any more, i.e for any attribute $A \in X$, $(X - \{A\})$ does not functional determine Y. A functional dependency $X \rightarrow Y$ is a partial dependency if some attribute $A \in X$ can be removed from X and " still holds, i.e for some $A \in X$, $(X - \{A\}) \rightarrow Y$.



- FD1 = {SSN, Pnumber} → {Hours}
- FD2 = {SSN} → {Ename}
- FD3 = {Pnumber} → {Pname, Plocation}

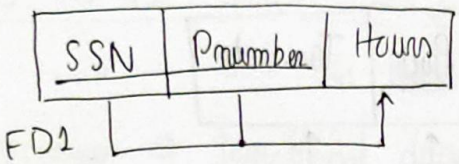
Q. Whether the relation in in 2NF or not.

Soln-

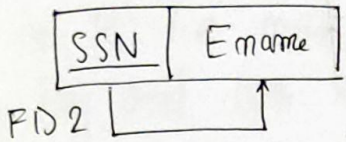
The relation in in 1NF but the relation in not in 2NF because of FD2 and FD3, because FD2 and FD3 holds partial dependency.

Therefore, we will decompose Emp_project table.

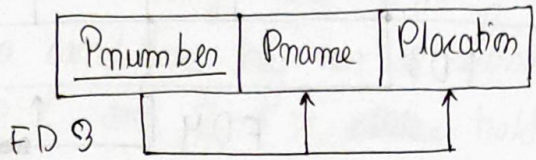
EP1



EP2



EP3

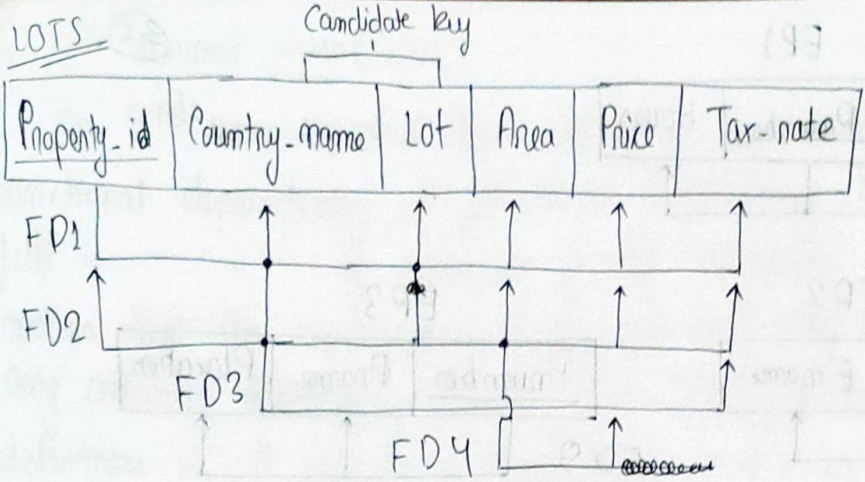


Now, the relation EP1, EP2 and EP3 are in 2NF since they hold full functional dependency.

---

1. select * from tablename;
2. ~~select~~ Alter table table_name add column_name datatype(size) (size);
3. Alter table table_name modify column_name datatype(size);
4. select * from table_name where name like 'A%';
   " '%B';
   " 'A%B';
   " 'A__';
   " '___B';
5. select distinct column_name from table_name;
6. select * from table_name order by column_name;
7. select * from table_name order by column_name desc;
8. select * from table_name where column_name in (attribute1, attribute2, attribute3, ...);
9. select * from table_name where column_name between a and b;

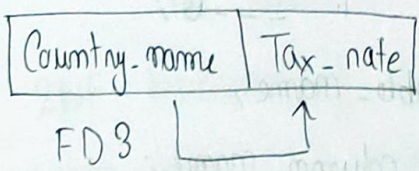
LOTS



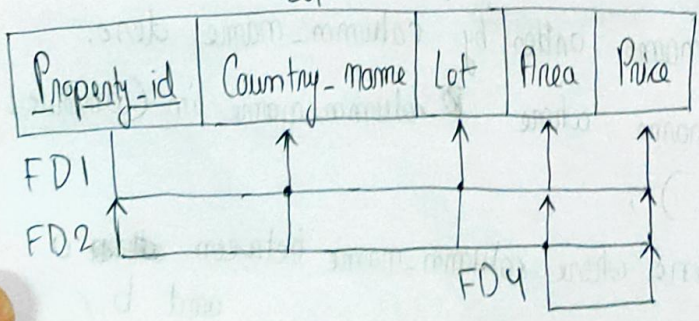
- FD1 : {Property_id} → {Country_name, Lot, Area, Price, Tax-rate}
- FD2 : {Country_name, Lot} → {Property_id, Area, Price, Tax-rate}
- FD3 : {Country_name} → {Tax-rate}
- FD4 : {Area} → {Price}

The relation is in 1NF but the relation is not in 2NF since FD3 violates the property of 2NF as it holds partial dependency so will decompose the relation Lot 1 and Lot 2

Lot 1



Lot 2



* 3nd Normal Form (3NF):

" " " is based on the concept of transitive dependency a functional dependency $X \rightarrow Y$ in a relation schema R is a transitive " if there a set of attributes Z in R i.e neither a candidate key nor a subset of any key and both $X \rightarrow Z$ and $Z \rightarrow Y$ holds.

Or

A relation schema R is in 3NF whenever a functional dependency holds in R either (a) X is a super key of R or (b) A is a prime attribute of R.

Emp_dept

Ename	SSN	Bdate	Address	Dnumber	Dname	Dmgr_SSN

FD1

FD2

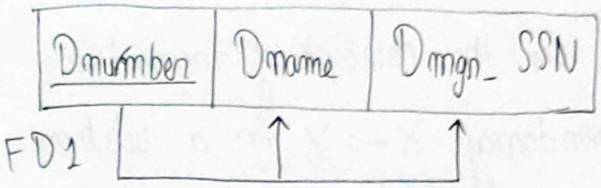
FD1 = {SSN} → {Ename, Bdate, Address, Dnumber}

FD2 = {Dnumber} → {Dname, Dmgr_SSN}

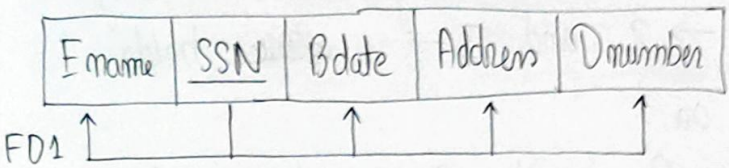
Since, the relation holds full functional dependency so the relation in in 2NF.

In this relation Dnumber in not the key. The relation holds transitive dependency so the relation is not in 3NF and so we will decompse the relation into two relation FD1 and FD2.

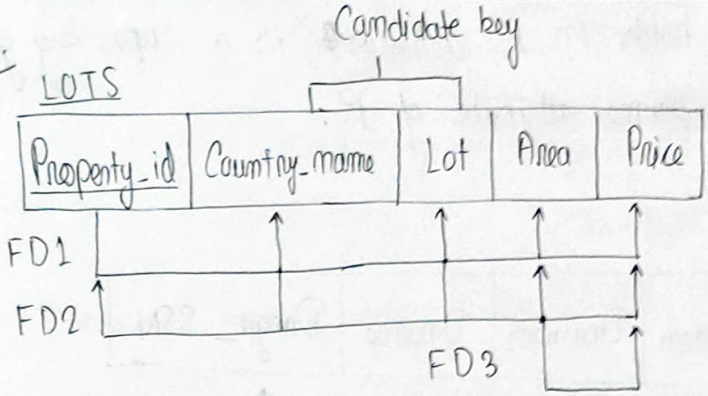
ED1



ED2



Q.

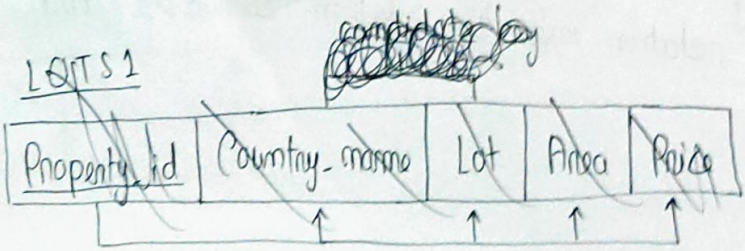


Check whether the relation is in 3NF or not.

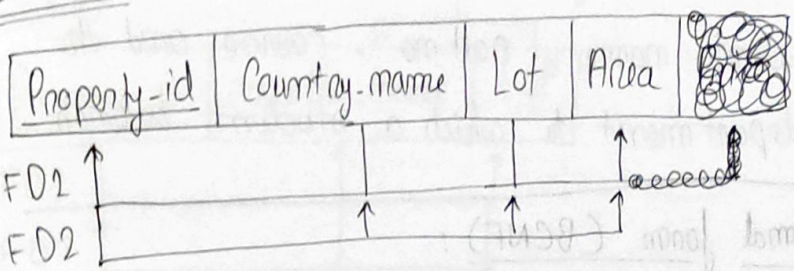
Sol^-

The given relation is in 1NF and 2NF but not in 3NF. because of FD3, in this relation neither area is a key nor Price is a prime attribute. The relation holds transitive dependency so the relation is not in 3NF and so we will decompose the relation into two relation LOTS1, and LOTS2

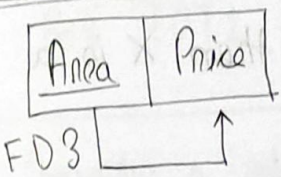
LOTS1



LOTS 1



LOTS 2



Now the relation LOTS 1 and LOTS 2 are in 3NF.

(*)

- Select avg (salary) from table_name group by column_name;

- Select avg (salary) from employee group by department having max (salary) > 20000;

Q. Create a student table with the following attributes S_name, S_roll, course, department & total_marks. (atleast 3 course and department). S_roll = primary key. Another table department with attributes D_id & D_name.

i) Display the no. of students in each course.

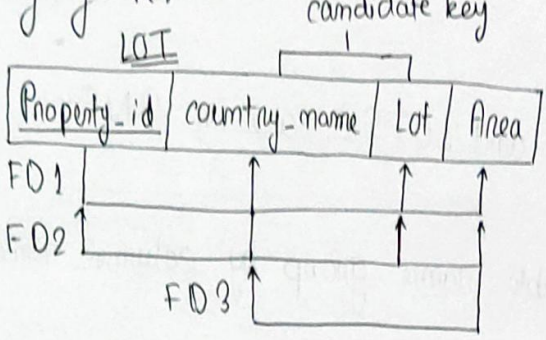
ii) " " S_name with the highest mark from each department.

iii) Display the D_name with minimum three students

iv) " " Students name, roll no, course and the name of the department to which a student belongs.

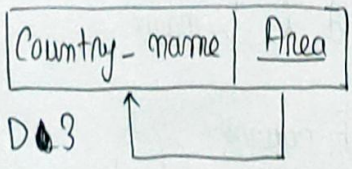
- **Boyce Codd Normal form (BCNF):**

A relation schema R is in BCNF if whenever a Functional dependency $X \rightarrow A$ holds in R then X is a superkey of R.



The relation is in 1NF. The relation is also in 2NF as all functional dependency holds full functional dependency. FD1 and FD2 follows the property of 3NF but FD3 violates it since neither Area is a superkey nor country_name is a prime attribute, so the relation is not in 3NF. So we will decompose the " into two relation LOT1 and LOT2.

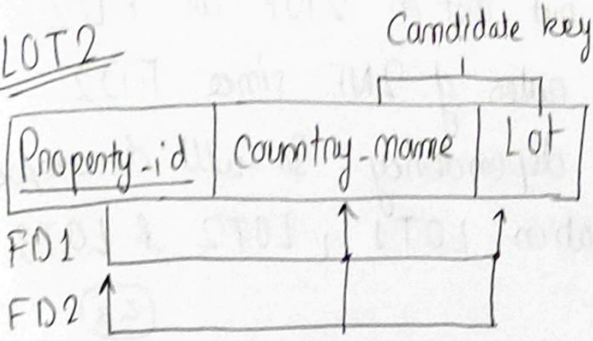
LOT1



LOT2

Property

LOT2



Both LOT1 and LOT2 are in BCNF.

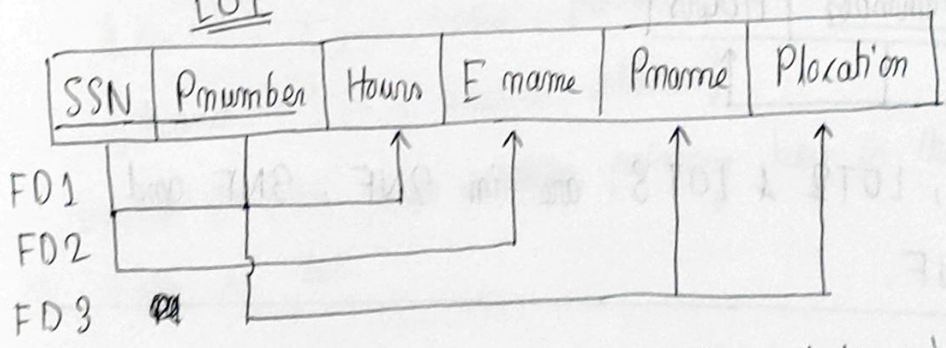
Q. Consider a relation R = { SSN, Pnumber, Hours, Ename, Pname, Plocation } and FD are

FD1 = { SSN, Pnumber } → { Hours }

FD2 = { SSN } → { Ename }

FD3 = { Pnumber } → { Pname, Plocation }

LOT



The relation is in 1NF but the relation is not in 2NF as FD2 violates the rules of 2NF since FD2 holds partial functional dependency. So we will decompose the relation into two relation LOT1 and LOT2

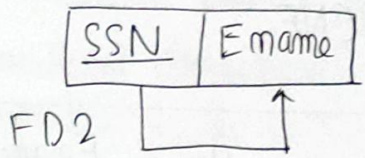
LOT1

SSN

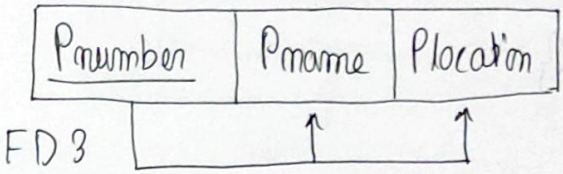
The relation is in 1NF but not in 2NF as FD2 and FD3 violates the rules of 2NF since FD2 and FD3 holds partial dependency. So will decompose the relation into three relation LOT1, LOT2 & LOT3.

(43)

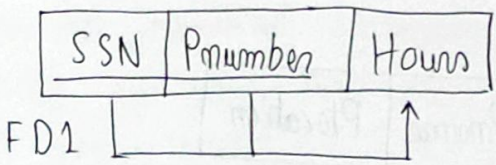
LOT1



LOT2



LOT3



LOT1, LOT2 & LOT3 are in 2NF, 3NF and also BCNF.

Q

Q. Consider the following relation employee_details (e_id, branch_id, e_name, branch_address, position, working-hours) assume that the primary key of the given relation is {e_id, branch_id} additional dependency are

{e_id} → {e_name, position}

{branch_id} → {branch_address}

based on the given ~~relation~~ primary key in the relation in 2NF or not.

• 4th Normal form (4NF).

• Multi-valued dependency :-

A multi valued dependency $X \rightarrow Y$ specify to relation schema R where X & Y are both subsets of R specifies the following constraints on any relation state r of R. If two tuples T_1 and T_2 exist in r such that $T_1[X] = T_2[X]$, then two tuples T_3 and T_4 should also exist in r with the following properties where we use Z to denote $(R - (X \cup Y))$.

* $T_3[X] = T_4[X] = T_1[X] = T_2[X]$
* $T_3[Y] = T_1[Y]$ and $T_4[Y] = T_2[Y]$
* $T_3[Z] = T_2[Z]$ and $T_4[Z] = T_1[Z]$

If a relation holds these above properties then it holds multi-valued dependency.

• 4th Normal form (4NF) :-

A relation in in 4NF if it does not hold multi-valued dependency.

Emp

	X	Y	Z
	Ename	Pname	Dname
t_1	Smith	X	John
t_2	Smith	Y	Amma
t_3	Smith	X	Amma
t_4	Smith	Y	John

The relation Emp follows all the rules of multi-valued dependen-

ey. so, the relation is not in 4NF. So we decompose the relation into two " Emp1 and Emp2.

Emp1

Ename	Pname
Smith	X
smith	Y

Emp2

Ename	Dname
Smith	John
Smith	Anna

- 5th normal form (5NF) :-

A relation schema is in 5NF if it holds joint ~~dependency~~ dependency.

- Joint dependency :-

A joint dependency denoted by JD (R_1, R_2, R_3, ...) specified on relation ~~schema~~ schema R, specify a constraint on the state s (R) the constraint states that every legal states (R) should have a non-additive joint decomposition (R_1, R_2, ..., R_n). Hence every s we have

$*(\pi_{R_1}(s), \pi_{R_2}(s), \ldots, \pi_{R_n}(s)) = s$.

- Select * from Emp, Dept where Emp.Dno = Dept.Dno ;
- select * from Emp left join Dept on emp.Dno = Dept.Dno ;

~~natural join~~

Right join

- select * from Emp natural join Dept ~~where~~ where emp.Dno = Dept.Dno ;

equi join

- Lossless Join property :

Q. R = { SSN, Ename, Pnumber, Pname, Plocation, Hours }

i) R_1 = { Ename, Plocation }

R_2 = { SSN, Pnumber, Hours, Pname, Plocation }

F = { SSN → Ename, Pnumber → { Pname, Plocation }, { SSN, Pnumber } → Hours }

Check whether the relation follows lossless Join property or not.

	SSN	Ename	Pnumber	Pname	Plocation	Hours
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

Since, no row contains all 'a' symbols so it does not follow the lossless join property.

ii) R_1 = { SSN, Ename }

R_2 = { Pnumber, Pname, Plocation }

R_3 = { SSN, Pnumber, Hours }

F = { SSN → Ename , Pnumber → { Pname , Plocation } , { SSN , Pnumber } → Hours }

	SSN	Ename	Pnumber	Pname	Plocations	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	a_2 ~~b_{32}~~	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

Since we have one row which contains all 'a' symbol. So, it follows the lossless join property.

- Inference rules on functional dependency :

It is also known as armstrong's axioms.

i) Reflexive rule :

if $X \supseteq Y$, then $X \rightarrow Y$

ii) Augmentation rule :

if $X \rightarrow Y$ then $XZ \rightarrow YZ$

iii) Transitive rule :

if $X \rightarrow Y$, $Y \rightarrow Z$ then $X \rightarrow Z$

iv) Projective or decomposition rule :

if $X \rightarrow YZ$ then $X \rightarrow Y$

v) Union or additive rule :

if $X \rightarrow Y$ & $X \rightarrow Z$ then $X \rightarrow YZ$

vi) Pseudo transitive rule :

if $X \rightarrow Y$, $WY \rightarrow Z$ then $WX \rightarrow Z$

- Closure:

Formally the set of all dependencies that include 'F' as well as all dependencies that can be inferred from 'F' is called the closure of 'F'. It is denoted by 'F+'.

e.g. F = { SSN → { Ename, Bdate, Address, Dnumber }

Dnumber → { Dname, Dmgr_SSN } }

a) SSN+

Sol^n

SSN+ = { SSN } ∪ { SSN → { Ename, Bdate, Address, Dnumber } }

= { SSN } ∪ { Ename, Bdate, Address, Dnumber }

= { SSN, Ename, Bdate, Address, Dnumber }

= { SSN, Ename, Bdate, Address, Dnumber } ∪ { Dnumber }+

= { SSN, Ename, Bdate, Address, Dnumber } ∪ { Dnumber, Dname, Dmgr_SSN }

= { SSN, Ename, Bdate, Address, Dnumber, Dname, Dmgr_SSN }

b) Dname+

Sol^n.

Dname+ = { Dname } ∪ { ϕ }

= { Dname }

c) Dnumber+

Sol^n:

$Dnumber^+ = \{Dnumber\} \cup \{Dname, Dmgr\_SSN\}$

$= \{Dnumber, Dname, Dmgr\_SSN\}$

---

i) Display E_id, E_name and D_name from the given table

ii) Display E_name, E_id, D_id for computer science employee.

iii) Display employee details with their department details

iv) Remove emp " " whose " id is 102

v) Display department details with their employee details.

Q.

Employee

E | Ename | Address | Pid

Project

| Pid | Pname | Duration |

Ten entries in employee table.

i) Display all the employee details.

ii) Display no. of employee details for pid 101.

iii)

Employee

Eid	Ename	Address	Pid

Project

Pid	Pname	Duration

Minimum ten entries in employee table.

i) Display all the employee details with their project details.

ii) " " no. of emp enroll in its "

iii) Remove emp details enroll in project 101.

iv) Display the emp_name, address, pname and duration for all emp.

- Sub query:

Student

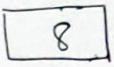
Roll no	Name	Age	marks obtained
1	aa	19	300
2	ab	20	400
3	ba	19	200
4	bb	20	420
5	cc	19	350

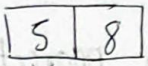
- Select * from student where marks > (select avg (marks) from student);

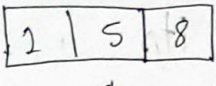
- **B tree :-**

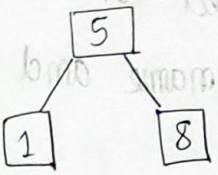
8, 5, 1, 7, 3, 12, 9, 6

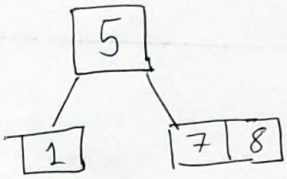
Order = 3

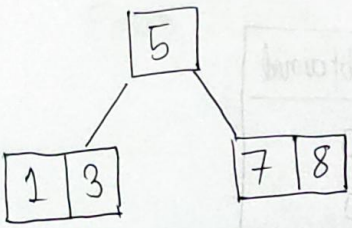
1) 

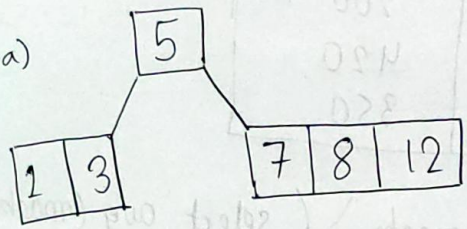
2) 

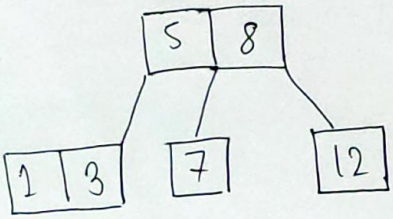
3) a) 

b) 

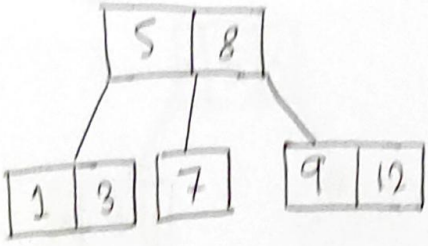
4) 

5) 

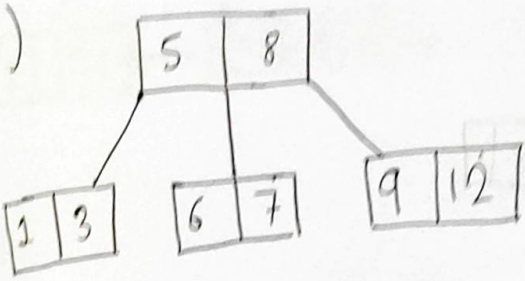
6) a) 

b) 

7)

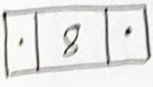


8)

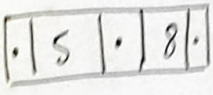


- B+ tree :

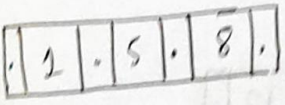
1)



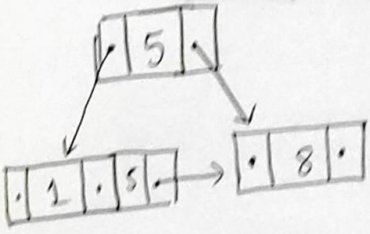
2)



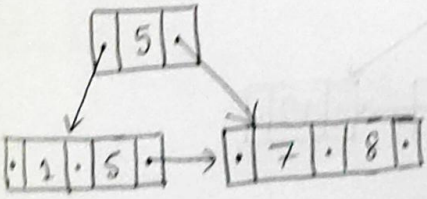
3) a)



b)

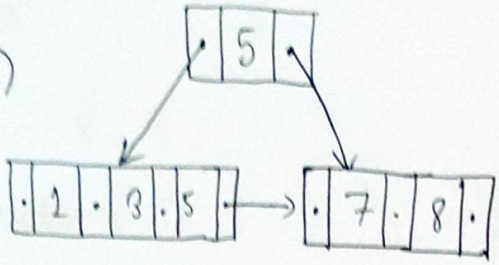


4)

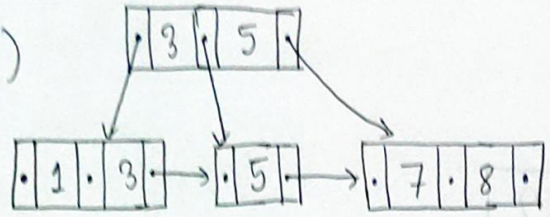


5)

a)

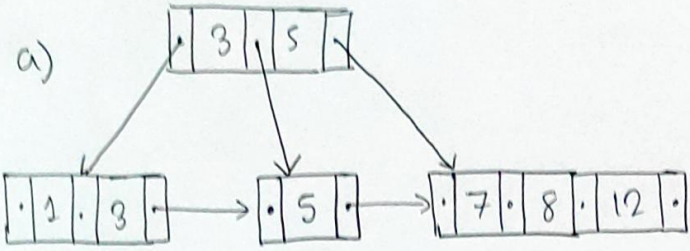


b)

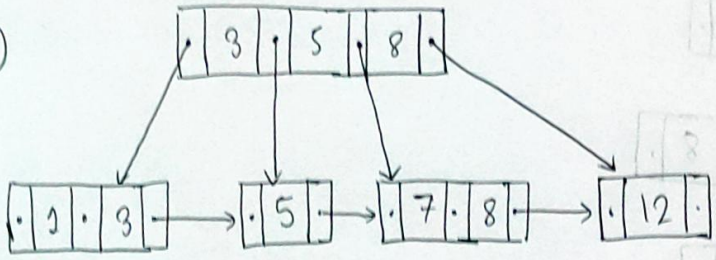


6)

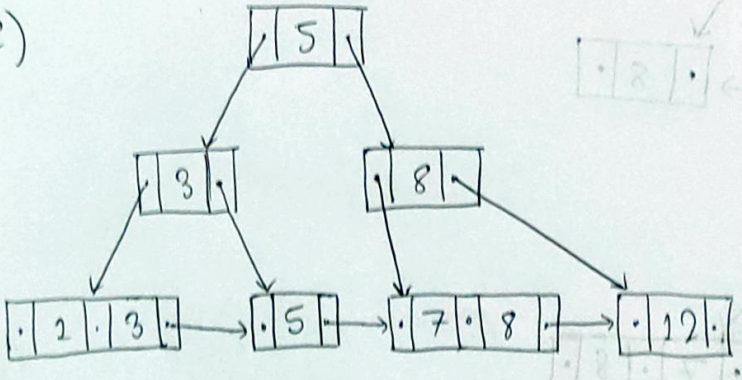
a)



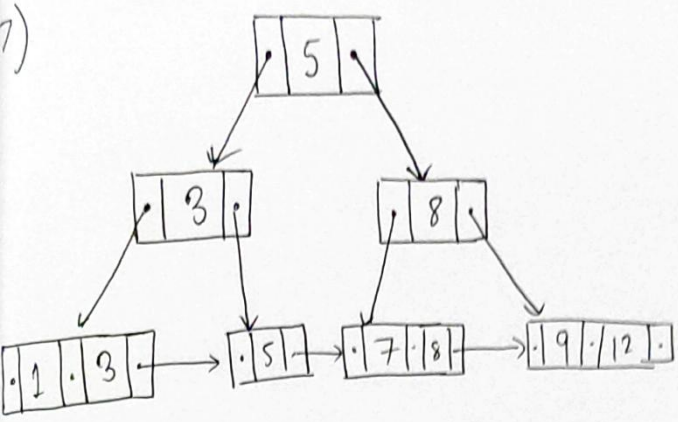
b)



c)

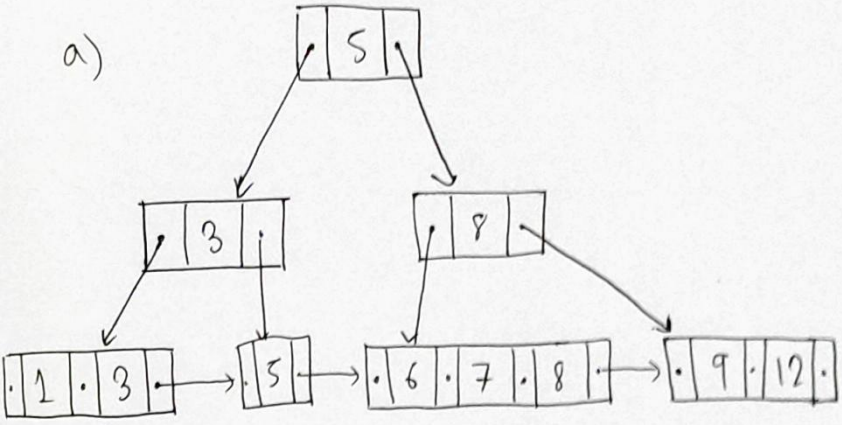


7)



8)

a)



b)