:-lecture - 7 :-

Queries (4 marks)

- **Procedural** (It Tells Procedure Btao ✓)
  - Query language
  - → Formulation of **How to retrieve** & **What data retrieves** from data Base Table.
  - *** Relational Algebra
  - How to Retrieve and what to Retrieve. → K/a Procedural language
  - ↓ done in Relational Algebra ✓
- **Non Procedural**
  - Query language
  - → formulation of **what data retrieves** from data-Base Tables.
  - → Relational Calculas.
    - [uses First order logic & Predicate logic Formulas]
    - ** TRC {Tuple Relational Calculas.
      - SQL queries close to TRC
      - ↳ (IBM) ✓
    - DRC {domain Relational calculas.
      - ↳ QBE (Query By Example).
        - Queries close to DRC
        - ↳ (microsoft) ✓

:-Relational Algebra:-

Basic operators

Unary operator:
1. Projection (π)
2. selection (σ)

Binary operator:
3. cross product (×)
4. Union (∪)
5. Set difference (−)

Unary operator:
6. Rename (ρ)
7.

Derived operators

1. Join (⋈) → {×, σ, π}
2. Intersection (∩) → {−}
3. division (/ or $\frac{a}{b}$) → {×, π, −}

π : Projection

π(R) Project required attributes from R → Also select distinct Tuple or Record ✓

σ : Selection.

σ(R) : Retrieves records of R those are satisfied condition (R)

R

A	B	C
4	6	8
8	5	4
7	3	4
7	3	4

(Bag)

$\pi_{BC}(R)$

B	C
6	8
5	4
3	4

** distinct Tuples

$\sigma_{A>5}(R)$

A	B	C
8	5	4
7	3	4

** distinct Tuples

** Ques- Which is true??

① π is commutative

② σ is commutative.

(a) (I) (b) (II) (c) I & II (d) None ✓

Soln ① $\pi_{List_2}(\pi_{List_1}(R)) = \pi_{List_1}(\pi_{List_2}(R))$

$\pi_{AB}(\pi_{ABC}(R)) \neq \pi_{ABC}(\pi_{AB}(R))$

Not commutative

**** Ex1) $\pi_{AB}(\pi_{ABC}) \neq \pi_{ABC}(\pi_{AB})$

Ex2) $\sigma_{>7}(\sigma_{>5}) = \sigma_{>5}(\sigma_{>7})$

② $\sigma_{C_2}(\sigma_{C_1}(R)) = \sigma_{C_1}(\sigma_{C_2}(R)) = \sigma_{(C_1 \wedge C_2)}(R)$

Centre = Delhi, Stream = CS, Stream = CS, Centre = Delhi

So ** σ is Commutative while π is Not holding Commutative property.

Cross Product :-

R x S :- All attributes of R followed by all attributes of S.
& Each record of R pairs with Every record of S.

R

A	B	C
4	2	5
6	5	8
3	4	5

S

C	D
5	7
7	8

Find R x S

A	B	C	C	D
4	2	5	5	7
4	2	5	7	8
6	5	8	5	7
6	5	8	7	8
3	4	5	5	7
3	4	5	7	8

If ① → ① Ist table have x No. of attributes;
② IInd table have y No. of attributes

Then (I x II) have (x+y) attributes. ★★★

If ② — ① Ist Rel^{n} have m distinct Tuples
② IInd Rel^{n} have n distinct Tuples.

Then I x II have (m * n) Tuples ★★★

Ques Relation R with "n" distinct tuples, Relation S with "0" Tuples.
Then How many tuples in Result of R * S: ____

Ans '0' zero

Ex

R

A	B
4	2
7	2

S

B	C

} 0 No tuples.

No Pairing occurs ✓

R × S

A	B	B	C

} Empty.

So, R × S result empty record set if either R or S rel^n with 0 tuples.

JOINS:- (Various Joins are there)

- ⋈ ← Natural Join
- ⋈$_c$ ← conditional Join
- ⟕ ← left outer Join
- ⟖ ← Right outer Join
- ⟗ ← Full outer Join.

① Natural Join (⋈)

$R \bowtie S = \pi_{\text{distinct attributes}} (\sigma_{\text{equality b/w common attributes of R \& S}} (R \times S))$

Example:-

R

A	B	C
4	2	5
6	5	8
3	4	5

S

C	D
5	7
7	8

** $\bowtie \rightarrow (\times, \sigma, \pi)$

** $\% \rightarrow (\times, \pi, -)$

$$R \bowtie S = \pi_{ABCD}\left(\sigma_{R.C=S.C}(R \times S)\right)$$

①

A	B	C	C	D
4	2	5	5	7
4	2	5	7	8
6	5	8	5	7
6	5	8	7	8
3	4	5	5	7
3	4	5	7	5

⇒

②

A	B	C	C	D
4	2	5	5	7
3	4	5	5	7

③

A	B	C	D
4	2	5	7
3	4	5	7

Example ②

R(A B C) S(B C D E)

$$\pi_{ABCDE}\left(\sigma_{\substack{R.B=S.B \\ \wedge \\ R.C=S.C}}(R \times S)\right) \checkmark$$

*** Qu: if R(A B), S(C D)

$R \bowtie S = ??$

Sol: If No common attributes.

then

$R \bowtie S = R \times S$ → when No common attribute are their.

Conditional Join :- ($\bowtie_c$)

$$R \bowtie_c S = \pi(\sigma_c(R \times S))$$ ✓

Example

R

A	B	C
4	2	5
6	5	8
3	4	5

S

C	D
5	7
7	8

Find $R \bowtie_{R.C>S.C} S$

Sol"

Not merging

A	B	C	C	D
6	5	8	5	7
6	5	8	7	8

→ $R \bowtie_{R.C>S.C} S$ ✓

Outer Joins :-

① left outer join ($\bowtie$) → Natural Join

$$(R \bowtie S) = (R \bowtie S) \cup$$ (records of R those are failed join condition with Null condition in remaining attr)

Ex for above R & S Table ✓

$R \bowtie S$ →

A	B	C	D
4	2	5	7
3	4	5	8
6	5	8	Null

Extra attached due to Left outer join

② Right Outer Join

$R \overline{\bowtie} S = (R \bowtie S) \cup$ (records of "S" those are failed Join condition with Null cond" in remaining attributes)

from Previous Table

$R \bowtie S =$

A	B	C	D
4	2	5	7
3	4	5	7
N	N	7	8

③ Full outer Join :- (Left Outer Join ∪ Right Outer Join)

$R \overline{\bowtie} S = (R \overline{\bowtie} S) \cup (R \overline{\bowtie} S)$

from Previous Table :-

$R \overline{\bowtie} S \equiv$

A	B	C	D
4	2	5	7
3	4	5	7
6	5	8	Null
Null	Null	7	8

Ex from previous R & S Tables

$R \overline{\bowtie}_{R.C>S.C} S$

A	B	C	G	D
6	5	8	5	7
6	5	8	7	5
4	2	5	Null	Null
3	4	5	Null	Null

Join Queries :-

⇒ R(A), S(B)

Retrieve "A" values of R, those are more than some "B" value of S.

(any / at least one)

Sol"

R

A
10
20
30
40

S

B
15
25
30

$\pi_{A}(\sigma_{R.A > B.B}(R \times S)) \Rightarrow$

A	B
20	15
30	15
30	25
40	15
40	25
40	30

$\pi_{A}(R \bowtie_{R.A > S.B} S)$

*** Learn

For at least one / some / any

use $R \bowtie S$

$R \bowtie_{C} S$

$\sigma_{C}(R \times S)$

∴ Rename :- (ρ) used to rename table names (or) attribute names

→ To avoid ambiguity we go for Renaming.

Ex

$\pi(\sigma(stud \times stud))$

Stud.sid, Stud.age > stud.age

sid	age	sid	age

[Ambiguity so Renaming use Here]

*** $\rho(temp, stud)$ → One student Table rename as Temp

Temp

sid	sname	age

*** (or) $\rho(stud)_{I,N,A}$ ⇒ attribute rename

stud

I	N	A

(or)

ρ(stud) ⇒ stud ⇒

I	Sname	A

Instance Renaming: sid → I, age → A

Some attribute Renaming

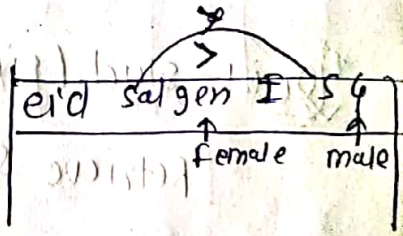
Ques Emp (eid sal gen)

Retrieve female emp id's whose salary more than salary of some male employee.

Soln Emp(eid sal gen) | Emp(eid sal gen)

(female) (male)

$$\pi_{eid}\left(\sigma_{sal>s}\left(\sigma_{gen=female}(Emp) \times \rho_{I,S,G}\left(\sigma_{gender=male}(Emp)\right)\right)\right)$$



(or)

$$\pi_{eid}\left(\sigma_{gen=female}(Emp) \bowtie_{sal>s} \rho_{I,S,G}\left(\sigma_{gen=male}(Emp)\right)\right)$$

Que Emp (eid Sal dNO)

Retrieve eid whose salary more than salary of some department "5" employee.

Soln

$$\pi_{eid}\left(Emp \underset{Sal > S}{\bowtie} \rho_{I,S,D}\left(\sigma_{dNO=5}(Emp) \right) \right)$$

Conditional Join going for :-

(⋈) → some / any / at least one.

Que :- stud (sid, age) dept (did, age)

Retrieve sid's whose age is more than age of some department

Soln

$$\pi_{sid}\left(stid \underset{Studage > Dept.age}{\bowtie} Dept \right)$$

(lecture - 8)

for (1) R(A) > some S(B)

↑ Retrieve 'A' value's of R those are more than some B value of S.

↳ $\pi_A (R(A) \bowtie_{R.A > S.B} S(B))$

(2) R(A) ∀ S(B)

Retrieve A values of R those are more than every B value of S.

Soln R(A) − { value of A which is less than or equal to some value of B }.

$\pi_A(R(A)) - \{ \pi_A (R(A) \bowtie_{R.A \leq R.B} R(B)) \}$.

• [All values of A] − [A ≤ some B] ✓

Example

for R(A)

A
10
20
30
40

S(B)

B
15
25
30

⇒

A
10
20
30
40

− [10, 20, 30] = (40) Ans ✓

V.Imp Qu

Emp (eid dno sal)

Retrieve eid's whose salary are more than every emp salary of dept 5.

Soln

Emp(eid dno sal) Emp(eid dno sal)

↑ Choose dept 5 employees

$\{\sigma_{dept=5}(Emp)\}$

(All Employees) − ¬(Sal > every sal of dept 5)

(All employee's) − (eid sal ≤ some salary of dept 5)

$\pi_{eid}(emp) - \pi_{eid}(Emp \underset{Sal \leq S}{\bowtie} \rho_{I,D,S}(\sigma_{dNo=5}(Emp)))$

V.Imp Qu

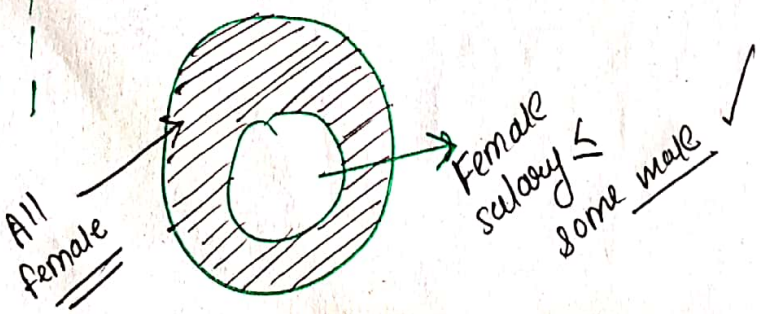
Emp (eid, sal, gender)

Retrieve eid's of female employee's whose salary is more than every salary of male employees.

Soln :-

$\pi_{eid}(\sigma_{g=f}(Emp)) - \pi_{eid}(\sigma_{g=f}(Emp) \underset{Sal \leq S}{\bowtie} \rho_{I,S,G}(\sigma_{g=m}(Emp)))$

(All Female eid) − (female emp sal ≤ some sal of male employee)



Que Emp(eid, sal)

Retrieve eid's whose gets maximum salary.

Soln { eid's whose salary maximum }

{ eid's whose salary >= every salary of emps } ≡ { all eids of emp } − { eid's whose salary < some salary of emp's } ✓

≡ $\pi_{eid}(emp) - \pi_{eid}(Emp \underset{sal<s}{\bowtie} \rho_{I,S}(Emp))$

- A > every B ≡ {All A} − {A < some B}.
- A >= every B ≡ {All A} − {A < some B}.
- R(A, ---)
  max value of A
- ** A value >= every A ≡
  {All A values} − {A < some A}.

example:-

Emp

eid	sal
e1	40
e2	80
e3	60
e4	40
e5	80

Emp

I	S
e1	40
e2	80
e3	60
e4	40
e5	80

All eid

eid
e1
e2
e3
e4
e5

−

eid
e1
e3
e4

≡ (e2 e5) ✓

Que Emp(eid sal)

eid's whose gets min salary.

Soln { eid's whose salary < every emp salary }.

$$\equiv \{ \text{All emp's of emp} \} - \{ \text{eid's whose salary} > \text{Some emp salary} \}.$$

$$\Rightarrow \boxed{\pi_{eid}(Emp) - \pi_{eid}\left(Emp \underset{sal>s}{\bowtie} \rho_{I,S}(Emp)\right)}$$

Que Emp (eid sal dno)

Eid's who gets maximum salary for each department.

Soln $\left\{ \begin{matrix} \text{eid's whose} \\ \text{sal} \geq \text{every emp} \\ \text{of same dept} \end{matrix} \right\} = \left\{ \begin{matrix} \text{all} \\ \text{Emp} \\ \text{id's} \end{matrix} \right\} - \left\{ \begin{matrix} \text{eid's whose sal} \\ < \text{some emp of same} \\ \text{dept} \end{matrix} \right\}$

$$\Rightarrow \pi_{eid}(emp) - \pi_{eid}\left(Emp \underset{\substack{sal<s \\ \wedge \\ dno=D}}{\bowtie} \rho_{I,S,D}(Emp)\right)$$

Que family (Parent child child dob)

Parent	child	child dob
A	B	1995
A	C	1998
A	D	1996
E	f	1990
E	G	1996

Retrieve youngest child of each parent.

Soln $\left\{ \pi_{child}^{(family)} - \left(\pi_{child} \left(family \underset{\substack{childdob < cd \\ \wedge \\ Parent = P}}{\bowtie} \rho_{P,C,cd}(family) \right) \right) \right\}$

$\left\{ \begin{array}{l} \text{Retrieve Childs} \\ \text{whose DOB} \\ > \text{Every child DOB} \\ \text{of Same parent} \end{array} \right\} \equiv \left\{ \begin{array}{l} \text{all} \\ \text{childs} \end{array} \right\} - \left\{ \begin{array}{l} \text{childs whose DOB} \\ < \text{some child DOB} \\ \text{of same parent} \end{array} \right\}.$

Division (/ or ÷)

Enroll

sid	cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C2
S3	C1

Course

cid
C1
C2
C3

all courses

uses in division

{ Retrieve sid's Enrolled every course of course relation }.

π (Enroll) / π_{cid} (course) ≡ Result

sid	cid
s1	c1
s2	c2
s1	c3
s2	c1
s2	c2
s3	c1

cid
c1
c2
c3

sid
s1

Expansion of Division (/)

$\{ \pi_{sid,cid}(E) / \pi_{cid}(C) \Rightarrow$ sid's enrolled Every Course $\}$

These steps are very much important please concentrate on this your division operator problem will be finished within one minute

a) sid's not enrolled for Every Course

[sid's enrolled proper subset of all courses]

$\pi_{sid}(\pi_{sid}(E) \times \pi_{cid}(C) - \pi_{sid,cid}(E))$

sid
s1
s2
s3

×

cid
c1
c2
c3

All possible pair b/w sid's of enroll rel && all courses.

sid	cid
s1	c1
s1	c2
s1	c3
s2	c1
s2	c2
s2	c3
s3	c1
s3	c2
s3	c3

−

sid	cid
s1	c1
s1	c2
s1	c3
s2	c1
s2	c2
s3	c1

⇒

s2	c3
s3	c2
s3	c3

⇒

sid
s2
s3

Std Not enrolled course

P.T.O

Divison can Express :-

Using π, ×, − basic operation

③ ① ②

(b) { sid's enrolled Every courses } = { sid enrolled some courses } − { sid not enrolled for every course }.

$$\pi_{sid\,cid}(E) / \pi_{cid}(C) \equiv \pi_{sid}(E) - \pi_{sid}\left(\pi_{sid}(E) \times \pi_{cid}(C) - \pi_{sid\,cid}(E)\right)$$

→ (X, −, π, π) finally

$\pi_{AB}(R) / \pi_B(S)$ ⇒ it gives A value which pairs with every value of B.

a1	b1
a1	b2
a1	b3
a2	b2
a2	b3

b1, b2, b3

Ans ✓ (a1)

A value which paired by proper subset of B valued (But Not all B paired)

**

$$\pi_{AB}(R) / \pi_B(S) \equiv \pi_A(R) - \pi_A\left(\pi_A(R) \times \pi_B(S) - \pi_{AB}(R)\right)$$ ⇒

A
a1

Total Relation from R to S − Actual Rel^n

↳ After doing "−" we get those Not enrolled for all courses

✓ { a1 b1, a1 b2, a1 b3 }   b1, b2, b3

⊗ { a2 b2, a2 b3 }

all (AB) value

No paired with CD gives

all possible pair

given pair

Think

$$\pi_{ABCD}(R) / \pi_{CD}(S) \equiv \left\{\pi_{AB}(R) - \pi_{AB}\left(\pi_{AB}(R) \times \pi_{CD}(S) - \pi_{ABCD}(R)\right)\right\}$$

✓ { a1b1c1d1, a1b1c1d2 }   c1d1, c1d2

⊗ { a2b2c1d2

A	B
a1	b1

Scanned with CamScanner

V. Imp

Ques Enroll(sid cid) Course(cid Instr)

① Retrieve sid's who enrolled for some course taught by KORTH.

② Retrieve sid's who enrolled for every course taught by KORTH.

Soln ① $\pi_{sid}\left(\sigma_{Enrollcid=Courcecid}\left(Enroll \times \sigma_{Inst=Korth}(Course)\right)\right)$

(or)

$$\pi_{sid}\left(Enroll \bowtie \sigma_{Inst=Korth}(Course)\right)$$

② Enroll

sid	cid
S1	C1
S1	C2
S1	C3
S1	C5
S2	C2
S2	C5
S3	C4

Course

cid	Inst
C1	Korth
C2	Korth
C3	Korth
C4	Navathe
C5	Ullman

$\Rightarrow \pi_{sidcid}(Enroll) / \pi_{cid}(\sigma_{Inst=KORTH}(Course))$

gives

sid
S1

(or)

$\pi_{sid}(Enroll) - \pi_{sid}\left(\pi_{sid}(Enroll) \times \pi_{cid}(\sigma_{Inst=korth}(course)) - \pi_{sidcid}(Enroll)\right)$

V. Imp

Example:-

Workfor(Eid Pid) Project(Pid Pname)

① Retrieve eid's whose work for some DB project

** ② Retrieve eid's who work for Every DB Project.

Soln:-

① $\pi_{eid}\left(\text{Works for} \bowtie \sigma_{Pname=DB}(\text{Projects})\right)$

② $\pi_{eid,pid}(\text{Workfor}) \,/\, \pi_{Pid}\left(\sigma_{Pname=DB}(\text{Project})\right)$

(or)

$$\pi_{Eid} - \pi_{Eid}\left\{\pi_{Eid}(w) \times \pi_{Pid}\left(\sigma_{Pname=DB}(P)\right) - \pi_{eid,pid}(w)\right\}$$

↑ Possible every eid of workfor rel" pairs with all DB Projects

↑ actual given Table

Employee Id who Not work for all data-Base Project ✓

Eid's works some project ✓

use

⋈ : Some

/ : Every

Lecture - 9 :-

Set Operators

Union : ∪
Set difference : −
(Minus)
Intersection : ∩

Same in SQL

To apply R∪S, R∩S, & R−S Relation R & S must be union compatable.

⇒ R & S is union compatable iff ⓐ

① Arity of R = Arity of S
&

② Domain of each attributes of R must be same as attributes of S respectively.

Arity ⇒ # attribute
⇒ # column

Ex ① $\pi_{sid,sname}(\quad)$ $\pi_{sid}(\quad)$

Arity = 2 — Arity = 1

→ cannot apply (∪, ∩, −) ✓

② $\pi_{sid,sname}(\quad) \cup \cap \pi_{sid,age}(\quad)$

→ Not same domain → cannot apply (∪, ∩, −) ✓

③ $\pi_{sid,sname}(\quad)$ $\pi_{StudID,StudName}(\quad)$

→ Everything fine ✓ Apply (∪, ∩, −)

R∪S, R∩S, R-S set operations:-

1. Schema for a result same as schema of R
2. Resulted record set is distinct record set

Same for SQL

Ex

R

A	B	C
2	4	6
3	5	7
3	5	7
4	5	6

S

D	E	f
2	4	6
2	4	6
4	5	6
3	5	9

R∪S

A	B	C
2	4	6
3	5	7
4	5	6
3	5	9

} either record from R or S

R∩S

A	B	C
2	4	6
4	5	6

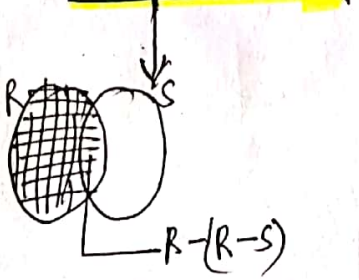
} records from R and S

R-S

A	B	C
3	5	7

} Records of R but not S

- ∪ → Basic operator
- ∩ → Derived operator → $(R \cap S) = R - (R - S)$



① $R \cap S = R \bowtie S$ → same as (XNOR)

same I/P ← High O/P

R	A	B
	1	6
	2	5

② $(R \cup S) = R \bowtie S$

$\bowtie$ = ⋈ U left remaining U Right Remaining

→ $R \cup S \equiv R \bowtie S$

When No common attributes in Tables:-

R(AB) S(CD)

① $R \cap S = R - (R - S)$ → $R \bowtie S = R \times S$

② $R \bowtie S \neq R \cap S$ // attribute names are Not same for R & S Tables ..

So

③ $R \cap S = R \bowtie_{\substack{C \to A \\ D \to B}} (S)$

④ $R \cup S \neq R \bowtie S = R \times S$

⑤ $R \cup S = R \bowtie_{\substack{C \to A, \\ D \to B}} (S)$

Means

↳ ⋈ Tabhi ∩ Ki Tarah Kaam Krega Jb Koeai attribute Name same ho

Agar attribute Name same Nhi hain to same Krke ⋈ ka use Kro otherwise simple "X" {cross product} Ki Tarah Kaam Krke Igega.

for the division formula

$$\pi_{AB}(R) \Big/ \pi_B(S) \equiv \pi_A(R) - \pi_A\left(\left(\pi_A(R) \times \pi_B(S)\right) - \pi_{AB}(R)\right)$$

union compatiable

Union compatible

Queries by using setop^{n}

Stud (sid sname age) Course (Cid Cname Instr) Enroll (sid cid fee)

① Retrieve sid's enrolled for some course taught by Korth or some Course taught by Navathe.

② Retrieve sid's enrolled for some course taught by Korth and some Course taught by Navathe! (good Query)

Sol^{n}:-

①

$$\pi_{sid}\left(Enroll \bowtie \sigma_{Instr=Korth \vee Instr=Navathe}(Course)\right)$$

(or)

$$\pi_{sid}\left(Enroll \bowtie \sigma_{Instr=Korth}(Course)\right) \cup \pi_{sid}\left(Enroll \bowtie \sigma_{Instr=Navathe}(Course)\right)$$

② Wrong

$$\pi_{sid}\left(Enroll \bowtie \sigma_{Instr=Korth \wedge Instr=Navathe}(Course)\right)$$

→ wrong Query writting

Bcoz Korth ∧ Navathe cannot possible in one row

Result of Query Empty record set

(or)

Right

$$\pi_{sid}\left(Enroll \bowtie \sigma_{Instr=Korth}(Course)\right) \cap \pi_{sid}\left(Enroll \bowtie \sigma_{Instr=Navathe}(Course)\right)$$

gives some course whose Instr = Korth

gives sid who enrolled for Korth & Navathe both ✓

gives some course whose Instr = Navathe

Ques Enroll (sid cid) Course (cid Instr)

(1) Retrieve sid enrolled only courses tought by Korth.

(2) Retrieve sid's enrolled some course tought by Korth.

Sol^n (1) {sid's enrolled some courses} − {sid's enrolled some Non Korth courses}.

$\Rightarrow \boxed{\pi_{sid}(Enroll) - \pi_{sid}\{Enroll \bowtie \sigma_{Instr=Korth}(Course)\}}$

OR	$\rightarrow \cup$	A ∪ B
AND	$\rightarrow \cap$	A ∩ B
only	$\rightarrow -$ minus	A − B

∴ Query optimizasation :-

Some time Questions comes from Query basic for which Query takes minimum Time to produce o/p. So, In that Type of Query we will concentrate on No. of steps, ~~who~~ The Query which takes minimum step then that produce o/p fastly.

Ex-(1) R(A,B) S(B,C)

$\sigma_{R.A>10 \wedge R.B=S.B \wedge S.C>5}(R \times S) \equiv \sigma_{R.A=S.B}(\sigma_{A>10}(R) \times \sigma_{C>5}(S))$

A	B	B	C

Ex-2) I) $\sigma_{R.A>10}(R \bowtie S)$

↓

II) $\sigma_{A>10}(R) \bowtie S$ ← optimal ✓

↑ Join Cost Reduces.

★★★ ⇒ The Tricks is Jis Table se Jo Bhi select Krna ho wo phle select kr lo uske Baad opⁿ krow ← Isme km No. of steps lgega Jb opⁿ Krenge Tb.

Ex-3 I) $\sigma_{R.A>10 \wedge S.C>5}(R \bowtie S)$

↓

⇒ II) $\sigma_{A>10}(R) \bowtie \sigma_{C>5}(S)$ ← optimal

Ex ④) Enroll (sid cid) — 10000

Course (cid Instr) — 100 courses; 5 Instr are korth

So optimise $\sigma_{Instr = KORTH}(Enroll \bowtie Course)$

Solⁿ Method-I

$\sigma_{Instr = KORTH}(Enroll \bowtie Course)$

↓

$\sigma_{INSTR=K}(\sigma(E \times C))$ ⇒ 2 Lakhs comp ✓

E × C → 10 Lacks Records

10 Lakhs comp.

method ②

$Enroll \bowtie \sigma_{Instr = KORTH}(Course) = \sigma(Enroll \times \sigma(Course))$

Enroll → 10,000; σ(Course) → 5 Tuples

50,000 comp ✓

method-2 is optimised method ✓

5. Emp(eid dno Sal)

Assume 100 Emp &&

$I_D(dept=5)Emp$.

plan 1

$\pi_{eid}\left(Emp \underset{Sal>S \wedge D=5}{\bowtie} f_{I,D,S}(Emp)\right)$

100 Tuples — 100 Tuples.

10,000 Tuples Cross product

Cost of Query : 20000 comp.

plan 2 →

$\pi_{eid}\left(Emp \underset{sal>S}{\bowtie} \left(\sigma_{D=5}\left(f_{I,D,S}(Emp)\right)\right)\right)$

from dept 5 ← 100 Employee

100 Tuples — 10 Tuples

10,00 Tuples

↳ 1000 comparision

6. Emp(eid, gen, sal)

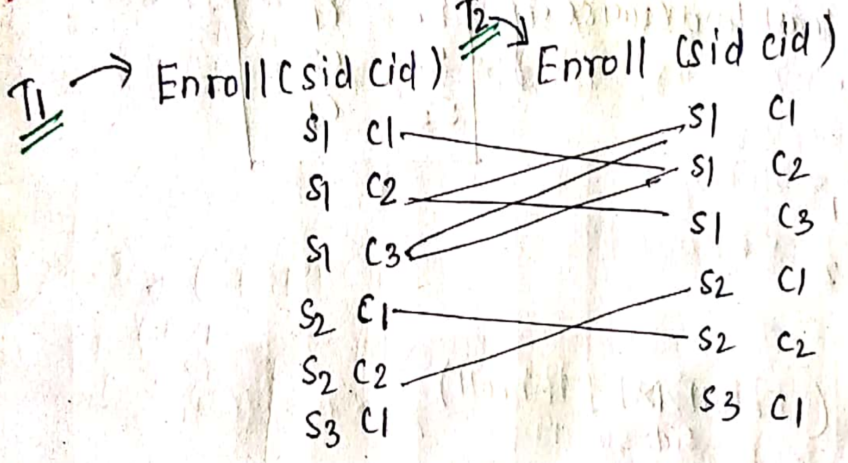
$\pi_{eid}\left(Emp \underset{gen=Fem \wedge G=male \wedge sal>S}{\bowtie} f_{I,G,S}(Emp)\right) \equiv$

$\pi_{eid}\left(\sigma_{gen=fem}(Emp) \underset{sal>S}{\bowtie} \sigma_{G=male}\left(f_{I,G,S}(Emp)\right)\right)$

Lecture-10

Queries related to At least / only / atmost

good Que • Enroll (sid cid); Retrieve sid's enrolled for at least two courses.



good Que ✓

$$\pi_{T_1.sid}\left(\rho(T_1, Enroll) \bowtie_{T_1.sid = T_2.sid \,\wedge\, T_1.cid \neq T_2.cid} \rho(T_2, Enroll)\right)$$

(or)

$$\pi_{sid}\left(Enroll \bowtie_{sid = s \,\wedge\, cid \neq c} \rho_{s,c}(Enroll)\right)$$

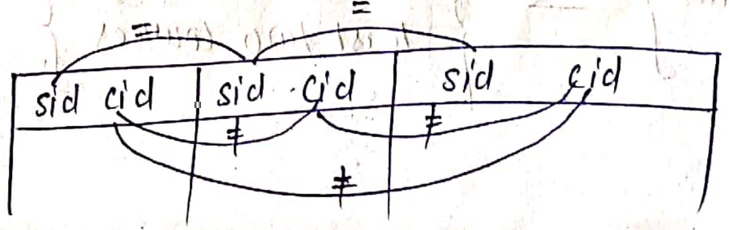
Tricks:-
For Finding at least 2 Then do cross product Two Times & For Finding atleast 3 Then do cross product Three Times. so on– Then apply condⁿ

★★★ good Que • Retrieve sid's enrolled for at least three courses.

$$\pi_{T_1.sid}\left(\sigma_{T_1.sid = T_2.sid \,\wedge\, T_2.sid = T_3.sid \,\wedge\, T_1.cid \neq T_2.cid \,\wedge\, T_2.cid \neq T_3.cid \,\wedge\, T_1.cid \neq T_3.cid}\left(\rho(T_1, Enroll) \times \rho(T_2, Enroll) \times \rho(T_3, Enroll)\right)\right)$$

(16:30)

$T_1 \times T_2 \times T_3$



Que sid's enrolled "only one" course.

E

sid	cid
s_1	c_1
s_1	c_2
s_1	c_3
s_2	c_1
s_2	c_2
s_3	c_1

↓ Logic

{ sid's enrolled at least one course } − { sid Enrolled at least two courses }

↓

$S_3 \equiv$ s_1, s_2, s_3 − s_1, s_2

$$\pi_{sid}(Enroll) - \pi_{sid}\left(Enroll \bowtie_{\substack{Sid=s \\ \wedge\, Cid \neq c}} \rho_{s,c}(Enroll)\right)$$

Que sid enrolled "only two" courses.

↓ Logic

{ sid's Enrolled at least two courses } − { sid's Enroll at least three courses }.

↓

$S_2 \equiv$ s_1, s_2 − s_1

Que sid enrolled at most one courses.

Sol"

Stud (sid sname)

s_1
s_2
s_3
s_4

Enroll (sid cid)

s_1 c_1
s_1 c_2
s_1 c_3
s_2 c_1
s_2 c_2
s_3 c_1

$\{s_3, s_4\} =$ { All stud } − { sid Enrolled at least two courses }.

All stud: s_1, s_2, s_3, s_4

sid Enrolled at least two courses: s_1, s_2

Que sid Enrolled "at most" two course.

Soln { All student } − { Sid's Enrolled at least three course }.

V.Imp Que Rel" R with x set of attributes & n distinct tuples
Rel" s with y set of attributes & m distinct tuples.

a) Condition Required to use R/s (division op)
b) Resulted Attribute set of R/s
c) Cardinality of R/s.

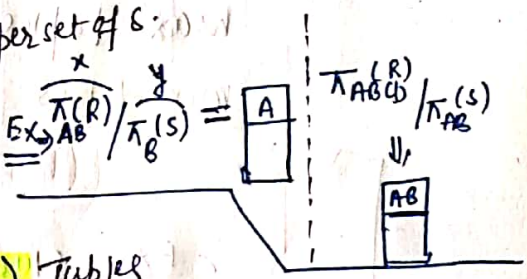
Soln:- a) Cond" required for R/s ⇒ R ⊃ S → R is superset of S.
b) Resulted attr. set of R/s → (x-y)
*c) Cardinality of R/s → (0 — ⌊n/m⌋) Tuples

Ex $\pi_{AB}(R) / \pi_{B}(S)$ → min No. of Tuples: 0
max No. of Tuples ✓

A	B
a1	b1
a1	b2
a2	b2
a2	b3

B
b1
b2
b3

⇒ (0)



V.Imp Que Rel R with n distinct Tuples.
Rel S with m distinct Tuples.

	R.A Expression	Cardinality (min, max) (means minimum & maximum Tuples in the relation)	Commutative.
①	$\pi_A(R)$	atmost n Tuples (depend on the # of distinct A value)	NO ✓
②	$\sigma_{cA}(R)$	(0 to n) Tuples	** ✓
③	R × S	(n*m) Tuples	** ✓
④	R ⋈ S, R ⋈c S	(0 to n*m) Tuples	** ✓
⑤	R ⋈ S	(m to n*m) Tuples.	X {order change then Result change}
⑥	R ⟕ S	(m to n*m) Tuples	X
⑦	R ⟗ S	* (max(n,m) to n*m) Tuples	✓
⑧	R ∪ S	* (max(m,n) to n*m) Tuples	✓
⑨	R ∩ S	(0 to min(n,m)	✓
⑩	R − S	{0 to n}	X
⑪	R/S	{0 to ⌊n/m⌋	X

Imp Que

Stud (sid sname) sid Primary key with 100 Tuples.

Enroll (sid cid) sid,cid Primary key with 50 tuples.

How many (max, min) tuples in Result of stud ⋈ Enroll?

(a) (100,50) (b) (100, 0) (c) (50, 50) (d) (5000, 0)

Sol^n :-

R ⋈ S → min → (0) Tuples
max → (n × m) Tuples
(n) (m)

max tuples, S ⋈ E ⇒ 50 tuples
min tuples S ⋈ E ⇒ 50 tuples [FK exists] } → Ans (c)

Ex

Stud

Sid	Sname
S1	A
S2	A
S3	B
S4	B
S5	A
↑ key	

100 Tuples

Enroll

sid	cid
S1	C1
S1	C2
S1	C3
S2	C3

50 tuples

1 to many Rel^n Possible

1 to many Not Possible

all will join with

Stud → max → 50
min → 50

If No foreign key given then Sid of Enroll table Not match with and stud Id In this case min Tuples will be "0".

V. Imp Que ***

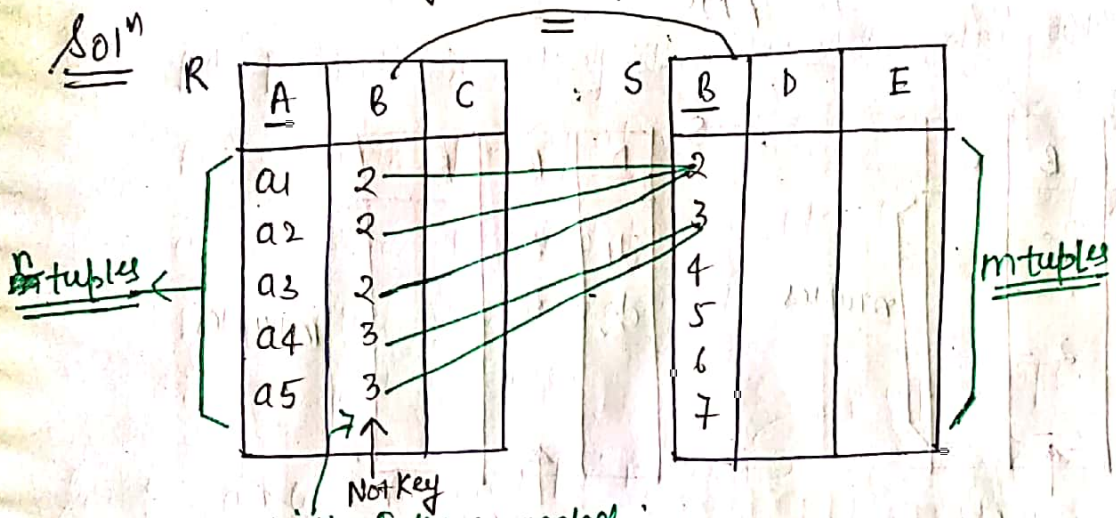
R (A B C) No Null with n tuples

S (B D E) with m tuples.

How many max tuples in Result of R⋈S: ____

How many min tuples in Result of R⋈S: ____

Soln



In R⋈S

max Tuples → ~~n~~ n.

min Tuples → 0 → (No foreign Key)

~~n~~ n → (with foreign Key)

V. Imp Que

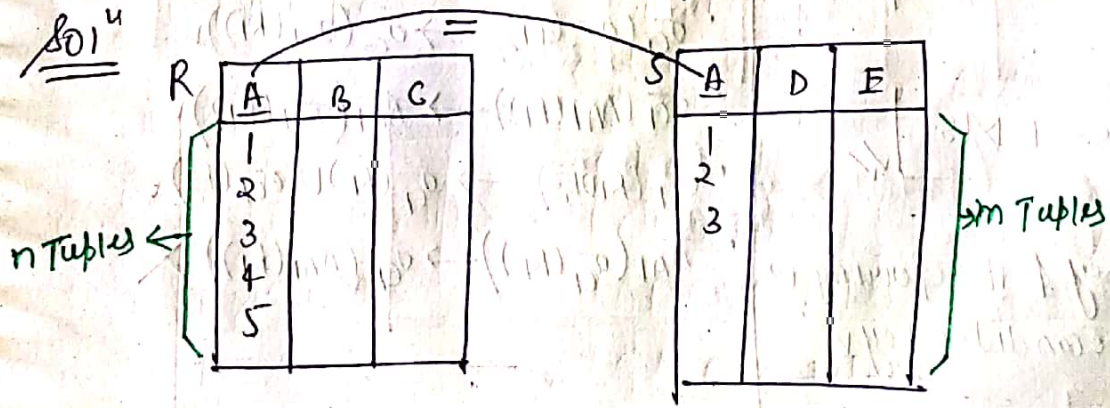
R (A B C) with n tuples

S (A D E) with m tuples

max tuples in result of R⋈S: ____

min tuples in result of R⋈S: ____

Soln



In R⋈S

① max Tuples → min (n, m)

② min Tuples → min (n, m) ← with FK

0 ← (NO FK)

V.Imp ⑤ ★★★

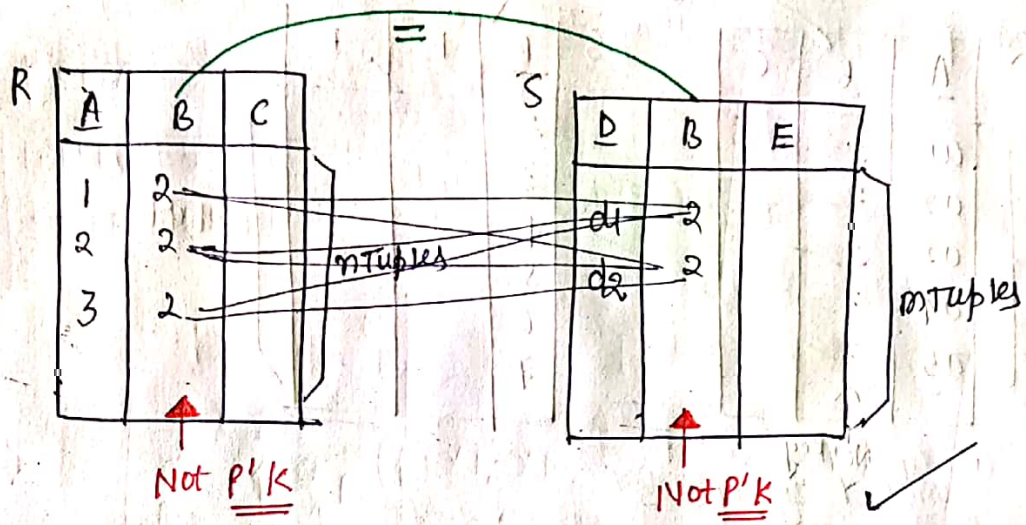
R(A B C) with n tuples

S(D B E) with m tuples.

max tuples in result of R⋈S ____

min ''' ''' ''' R⋈S ____

Soln



In R⋈S

- max tuples = n×m
- min tuples = 0

Important Concept ✓✓

- **Have same Expressive power**
  - → Relational Algebra
  - → TRC Restricted to safe Expression
  - → DRC Restricted to safe expression
- **R−(R−S) = R∩S = R⋈S** ✓✓
- If B is a foreign key & it referring to "C" → then C will be candidate key ✓

- Jb Tk SQL ke Question me Ye Na Kha Jae ki distinct selection Tb Tk distinct Na maano ✓✓
- **R∩S = R−(R−S)**
- $\sigma_{C_1}(\sigma_{C_2}(R_1)) \rightarrow \sigma_{C_2}(\sigma_{C_1}(R_1))$ ✓
- $\sigma_{C_1}(\pi_{A_1}(R_1)) \rightarrow \pi_{A_1}(\sigma_{C_1}(R_1))$ ✓
- $\sigma_{C_1}(R_1 \cup R_2) \rightarrow \sigma_{C_1}(R_1) \cup \sigma_{C_1}(R_2)$ ✓
- $\pi_{A_1}(\sigma_{C_1}(R_1)) \rightarrow \sigma_{C_1}(\pi_{A_1}(R_1))$ ✗

Lecture - 11

SQL { structured Query language }.

Sub languages of SQL →
- DDL (data definition language)
- DML (data manipulation language)
- DCL (data control language)

Data definition Language (DDL)

⇒ used to define a structure / definition of DB table.

⇒ CREATE TABLE ---

⇒ DROP TABLE

⇒ ALTER TABLE (add / remove attributes)

⇒ PRIMARY KEY

⇒ UNIQUE (alternate Key)

⇒ Foreign Key (FK constraints)

⇒ NOT NULL

⇒ Check etc (used to set range for attribute value)

Data Manipulation language (DML)

Used to modify Data records & & access required data from DB tables.

⇒ INSERT INTO ---

⇒ DELETE FROM ---

⇒ UPDATE ---- SET

⇒ SELECT * FROM Tables WHERE Cond,

Data control language (DCL)

① data control ~~from~~ for transaction mgmt
(To Avoid inconsistency bcoz concurrent User)
- ⇒ RollBack
- ⇒ commit (commit)
- ⇒ Check Point etc.

② Data control for user authentication (Security)
- ↳ Grant Access ---
- ↳ Revoke Access etc

SQL Query Vs Relational Algebra (RA) Query :-

SQL :-

```
SELECT DISTINCT A1, A2 --- An
from R1, R2, R3 ---- Rm where P;
```

FROM → Cross Product (×)
WHERE → Selection (σ)
SELECT DISTINCT → Projection (π)

$$\pi_{A_1,A_2 \ldots A_n} \{ \sigma_P (R_1 \times R_2 \times R_3 \ldots \times R_m) \}$$

select distinct — $A_1, A_2 \ldots A_n$; where — P; From — $R_1 \times R_2 \times R_3 \ldots \times R_m$

Enroll ≡ Course.

Sid	Cid
S1	C1
S1	C2
S2	C2
S2	C3

Cid	Instr
C1	K
C2	K
C3	N
C4	N

Retrieve sid's enrolled some course Taught by Korth.

$$\pi_{Sid} \{ \sigma_{Enroll.Cid = Course.Cid} (Enroll \times \sigma_{Instr=K} (Course)) \}$$

gives

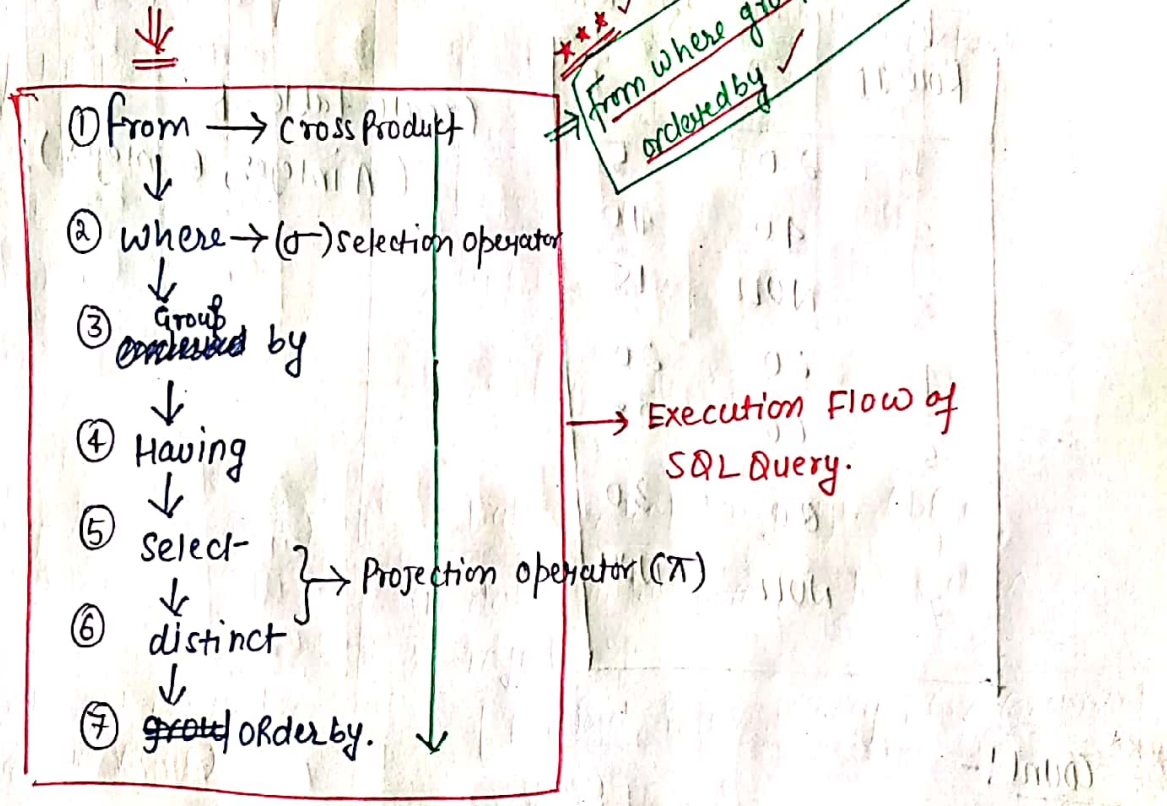
S1
S2

SQL

```
• Select distinct sid from Enroll,
  course where Instr = Korth &
  Enroll.Cid = course.Cid
```

Basic SQL clauses :-

⑤ SELECT [DISTINCT] ⑥ $A_1, A_2 \dots An$
① FROM $R_1, R_2, \dots Rm$
② [WHERE condition]
③ [Group by (attributes)
④ [HAVING condition]]
⑦ [ORDER BY (attributes) [DESC]];



About Group by clause :-

Used to group records based on specific attributes values.

```
SELECT "X" & AGG(Y)
from R
Grouped by (X)
```

① If group by clause used then
   (I) All attribute of group by clause must be in select clause.
   (II) Can use Aggegrate function in SELECT clause.
   (III) Not allowed to select any other attribute in select ~~clause~~ clause.

Aggregate function :-

1. min()
2. max()
3. sum()
4. Avg()
5. count()

Aggregate Function
Computes aggregation of Non Null Values of A.

← maintained by SQL.

Row ID	A	B
1	20	5
2	40	10
3	NULL	15
4	20	20
5	60	25
6	80	30
7	NULL	NULL

```
Create table R
    (A integer, B Integer);
```

*** Count :-

Count(*) :- # of Records

Count(A) :- # of Non Null Values of Attr A.

Count(distinct A) :- # of distinct Non Null Value of A.

Que:-

R

RowID	A	B
1	20	5
2	40	10
3	NULL	15
4	20	20
5	60	25
6	80	30
7	NULL	NULL

a) Select Count(*) A1,
Count(A) A2,
Count(DISTINCT A) A3,
from R;

b) Select Sum(A) A1, Sum(DISTINCT A) A2,
AVG(A) A3, AVG(DISTINCT A) A4,
MIN(A) A5, MAX(A) A6
from R;

Soln:-

a)

A1	A2	A3
7	5	4

b)

A1	A2	A3	A4	A5	A6
220	200	44	50	20	80

$$AVG(A) = \frac{Sum(A)}{Count(A)} = \frac{220}{5} = 44$$

$$AVG(DISTINCT\ A) = \frac{Sum(DISTINCT\ A)}{Count(DISTINCT\ A)}$$

$$= \frac{200}{4} = 50$$

GROUP BY :-

Used to group records based on specific attributes values.

↓

Query Based on GROUP BY :-

R

A	B	C
a1	b1	80
a2	b4	90
Null	b3	60
a2	b4	60
a1	b1	90
Null	b3	40
a1	b2	NULL

Group by "A"

A	B	C
a1	b1	80
a1	b1	90
a1	b2	Null
Null	b3	60
Null	b3	40
a2	b4	90
a2	b4	60

Query –

(1) (3) Select A
(1) From R
(2) GROUP By A;

→

A
a1
Null
a2

(2) (3) Select A, AVG(C)
(1) from R
(2) Group By A

↑ Each group

→

A	AVG(C)
a1	85
Null	50
a2	75

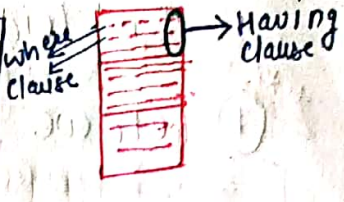
(3) ✗ select A, B
from R → Invalid Query
Group By A;

(4) ✗ select B
from R → Invalid Query.
GROUP By A;

Having clause Conditions :-

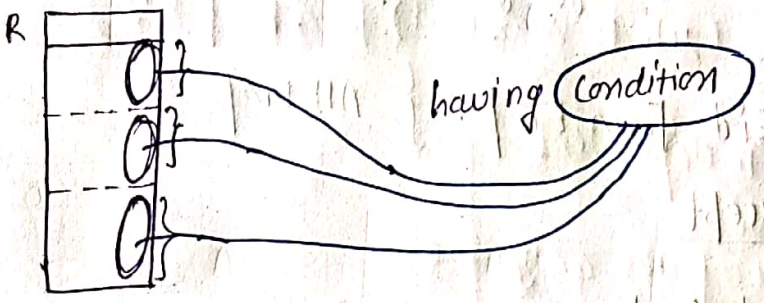
***

Where clause Tested for each record
& Having clause Tested for each Group.



Order by clause form the values either in Ascending order or in descending order.

Having clause is used to select groups based on condition.



Ex

A	B	C
a1	b1	80
a1	b1	90
a1	b2	NULL
null	b3	60
null	b3	40
a2	b4	90
a2	b4	60

① select A
from R
Group By A
Having Avg(C) > 60;

⇒

A
a1
a2

② Select A
from R
Group By A
Having some(C) > 80

A
a1
a2

③ Select A
from R
Group By A
Having C > 60

Invalid

direct Attr Comp.

Not Possible in Having clause

↳ Either it takes aggregation comparision, or special comparison like some, all etc.

Query Practice :-

Emp (eid sal)

** ① Retrieve eid who gets maximum salary

→ Relational Query

→ $(\pi_{eid}(Emp) - \pi_{eid}(Emp \underset{Sal<S}{\bowtie} \rho_{I,S}(Emp)))$

SQL Query :- (without Aggregation)

```
Select Eid from Emp.   } All Emp
        except
Select Eid
from Emp T1, Emp T2    } eid's whose sid < some Emp sal
where T1.sal < T2.sal
```

except means (-)

⇓

By using Aggregation CLAUSE :-

①
```
Select Eid from Emp
where sal = (select max(sal) from Emp);
```

(select max(sal) from Emp) → gives constant

Nested Query used in where Aggregation clause

②
```
select Eid
from Emp, (select max(salary), msal
           from Emp), temp;
where Emp.sal = temp.msal;
```

(select max(salary), msal from Emp), temp → Table

Que:- Emp (eid sal)

V. V. Imp

~~Rem~~ Retrieve emp's who gets second heighest salary.

Soln:-

$$\rho(Temp, \Pi_{eid,sal}(Emp \bowtie_{sal<s} \rho_{I,s}(Emp)))$$

Then

$$\Pi_{eid}(Temp) - \Pi_{eid}(Temp \bowtie_{sal<s} \rho_{I,s}(temp))$$

Ex

emp

eid	sal
e1	70
e2	50
e3	80
e4	60

Temp

eid	sal
e1	70
e2	50
e4	60

V.. V. Imp

SQL Query:-

```
• Select eid
  from Emp
  where Sal = (select max(sal)
               from Emp
               where sal < (select max(sal)
                            from (Emp)))
```

Lecture→12

Nested Queries

- Nested Query without Co-relation ✓
- Nested Query with Co-relation ✓

① Nested Query without Co-relation.

move ↑

```
Select eid
from Emp
where sal = (select max(sal) from Emp);
```

⇒ Inner Query independent
⇒ Bottom-top move
⇒ Inner Query computes only once

```
SELECT
from
WHERE
GROUP BY
HAVING
ORDER BY
```

(from, WHERE, HAVING → Inner Query)

② Nested Query with corelation :-

R(A----) S(B----)

move ↓

```
select *
from R
where (select count(*)
       from S
       where R.A > S.B) < 2
```

Soln for Previous Que

R

A
60
40
30
50
10

S

B
50
10
60
40
30

60 ✗ S.B	40 ✗ S.B	30 ✓ S.B	50 ✗ S.B	10 ✓ S.B
✓	X	X	X	X
✓	✓	✓	✓	X
X	X	X	X	X
✓	X	X	✓	X
✓	✓	X	✓	X

For Nested Query with Co-relation :-

⇒ Inner Query uses attribute of Outer Query

⇒ Execution Flow from Top - Bottom - Top

⇒ If Co-relation in WHERE clause Inner Query re-computes for each record of Outer Query

⇒ If Co-relation in having clause then Inner Query recomputes for each group of outer Query.

Q R2 computes ✓✓

For each record ↑ where clause

For each group ↑ having clause

- Nested Queries functions will be used where more than one comparision occure with one value in inner (Nested) loop.

Ex:-

```
Select *
from Emp.  → one value
where sal > ( select sal  → cannot compare, so Error occure.
              from Emp
              where dno = 5 )
```

Sal
60
80
40

- For these types of Error we will use Nested Queries Function.

Functions used for Nested Queries :-

1. IN / Not IN ($\bowtie$)
2. ANY
3. ALL

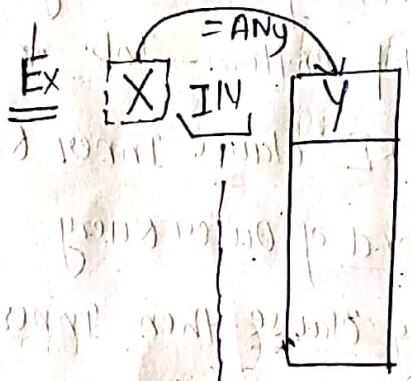
} Best suitable for Nested Query without Co-relation

4. Exists / Not Exists.

} Best suitable For Corelated Nested Query.

IN Function : Used For membership Test.

↳ ($\bowtie$)

Ex 

In Work Like $\bowtie$

True if x value is member of y set

↳ x value must be "equal" to "some" (at least one) value of y set.

IN can be used as —

equi join ($\bowtie$) Queries

$\pi_{R.A}(R \bowtie_{R.A=S.B} S)$

Ex. R(A B) = Some S(B C)

↳ $\pi_{AB}(R \bowtie S) \equiv$

```
select DISTINCT R*
from R
where R.B IN (Select B from S)
```

Ex → R(A) S(B)

$\Pi_A(R \underset{R.A=S.B}{\bowtie} S)$ ⇒ Select A distinct A
from R
where RAIN (Select B from S)

Ex R(A,B) S(C,D)

$\Pi_{AB}(R \underset{R.A=S.C \wedge R.B=S.D}{\bowtie} S)$ ⇒ Select distinct A,B
from R
where (A,B) IN (select (C,D) from S)

ANY & ALL function :-

Preceded by operator.
{<, <=, >, >=, =, <>}

Ex x < ANY

True if x value < some value of y set.
↳ so, Here we come use $\bowtie_C$ operator.

Ex x < ALL

True iff x value < every value of y set.

- (ANY) Fun Can used For Conditional Join ($\bowtie_C$) Queries.

Ex

R(A ---) S(B ---)

Retrieve "A" value of R those are less than some "B" value of S.

$\pi_A (R \underset{R.A<S.B}{\bowtie} S) \equiv$

```
Select distinct A
  From R.
Where R.A < ANY (select B value from S)
```

or

```
Select distinct A
  from R, S
  Where R.A < S.B.
```

Ex

R(A---) S(B ---)

Retrieve "A" Values of R those are less than every B Values of S.

Sol"

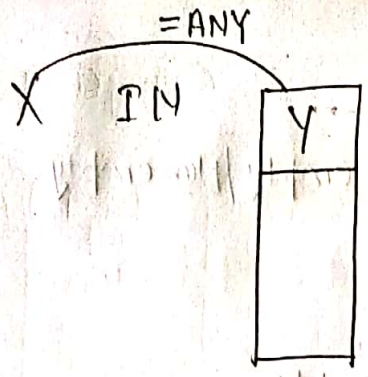
$\{A < \text{Every } B\} \equiv \{\text{all A values}\} - \{A > \text{some } B\}.$

$\equiv \pi_A (R) - \pi_A (R \underset{R.A>S.B}{\bowtie} S)$

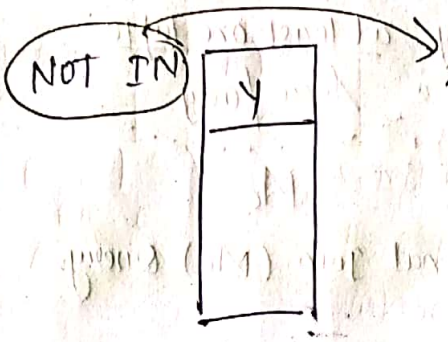
⇓

SOL →

```
Select DISTINCT A
from R
Where R.A <ALL (select B from S);
```



=ANY equal to (IN)



(<> All) equal to (NOT IN)

✓ R(A) S(B)

$\Pi_A(R \bowtie_{R.A=S.B} S)$ ⇒ select distinct A ~~from R~~

from R

where A =ANY IN (select B from S);

✓ R(A,B) S(C,D)

$\Pi_{AB}(R \bowtie_{R.A=S.C \wedge R.B=S.D} S)$ ⇒ Select distinct A,B

from R

where (A,B) IN (select C,D from S)

Bcz it have two value comparision ← Can't replace by =ANY

IN ⇒ =ANY

NOT IN ⇒ <> ALL

(A,B)=(C,D) ← used for 2 value comp like A≤B

IN ⇒ ⋈

ANY ⇒ ⋈

All ⇒ $\pi - \pi(\ )$

Exist ⇒ $\bowtie_c$

used for only on value comp. A≤B

EXISTS Function:-

Used to test result of Inner Query empty / Non empty record set.

Exist (Inner Query)

True if result of Inner query is not empty.

If result of Inner query ⟶ at least one tuple / some / any

$\bowtie_c$

EXISTS also used for conditional Join ($\bowtie_c$) Query.

Example:-

Emp(eid dno sal)

Retrieve eid's whose salary less than $\overset{\bowtie_c}{\text{some}}$ emp salary of dept 5.

$$\rightarrow \pi_{eid}\left(\text{Emp} \underset{\text{Sal<s}}{\bowtie} \rho_{I,D,S}\left(\sigma_{dno=5}(\text{Emp})\right)\right)$$

SQL Query

```
select Eid
① from Emp T1
③ where (Exist (
② Select *
  from Emp T2
  where T2 dNO=5
  & T1.sal < T2.sal );
```

Ex

Emp T1

eid	dno	sal
e1	4	10
e2	5	20
e3	4	30
e4	5	40
e5	4	50

emp T2

eid	dno	sal
e1	4	10
e2	5	20
e3	4	30
e4	5	40
e5	4	50

eid	dno	sal
e1	4	10
e2	5	20
e3	4	30

**

- Exist clause behaves like $\bowtie_c$ Queries where it as this is used in inner ~~tool~~ Query of where clause which is related with outer Query.

*** 6-7 Timegate Exam comes

Que Emp (eid dno sal)

Retrieve eid's whose salary less than every emp of dept 5.

Soln RA:→ $\pi_{eid}(Emp) - \pi_{eid}\left(Emp \underset{Sal>=S}{\bowtie} \rho_{f,D,S}\left(\sigma_{dno=5}(Emp)\right)\right)$

↳ SQL Corelated Query

Select eid
from Emp T1
Where Not exists (
select *
from Emp T2
where T1.sal > T2.sal and
T2.dNO=5);

Exist gives true
Not Exist gives true
Not Empty / Empty

**

Exist ⟷ Some
Not Exist ⟷ Every

***

Que R(A ----) S(B ----)

Retrieve "A" Values of R which are more than some "B" Values of S

Soln R.A :- $\pi_A(R \underset{R.A>S.B}{\bowtie} S)$

SQL:-

P.T.O

SQL

JOIN QUERY

```
Select distinct R.A
from R, S
where R.A > S.B;
```

(OR)

NESTED QUERY

```
select R.A
from R
where R.A > ANY ( select B
                  from S);
```

Co-Related Nested Query

```
Select R.A
from R
where Exists ( select *
               from S
               where R.A > S.B);
```

Q1) R(A ---) S(B ---)

Retrieve 'A' values of R which are more than every B values of S.

Soln: R.A ⇒ $\pi_A(R) - \pi_A(R \underset{A \leq B}{\bowtie} S)$

SQL

JOIN QUERY

```
(1) select A
    from R
    EXCEPT
    select A
    from R, S
    WHERE A<=B;
```

② Nested Query

```
select A
from R
where A >ALL (select B
              from S);
```

---

③ Co-related Nested Query

```
Select A
from R
Where NOT EXISTS ( Select * from S
                   Where R.A <=S.B);
```

$\neg(A \leq \text{some } B)$

$\equiv (A > \text{every } B)$

V. Imp Ques

Emp(eid, sal)

[] Retrieve eid's who gets minimum salary

Sol^n RA ⇒ $\pi_{eid}(Emp) - \pi_{eid}(Emp \underset{sal>s}{\bowtie} \rho_{I,S}(Emp))$

SQL ⇒

Nested Query.

```
Select eid
from emp
where sal <= ALL (select sal
                  from Emp);
```

⇒

Related Nested Query.

```
Select eid
from Emp T1
where Not Exist (Select *
                 from Emp T2
                 where T1.sal > T2.sal);
```

gives (T1.sal > some sal of T2)

V. Imp ★★★

Emp (eid sal dno)

Retrieve eid's whose get min salary for each dept.

Sol"

R.A : $\Pi_{eid}(Emp) - \Pi_{eid}\left(Emp \bowtie_{sal>S \wedge dno=D} \rho_{I,S,D}(Emp)\right)$

SQL: Using group by & Aggregation.

Eg ⇒ Emp

eid	sal	dNO
e1	40	3
e2	70	3
e3	50	3
e4	30	4
e5	60	4

X temp

dno	msal
3	40
4	30

Create another Table in which only minimum salary of Each dept is placed & then do cross product.

① ③ Select eid

① { from Emp, (select dNO, min(sal) msal
from Emp
Group by dNO) temp

② { where Emp.dno = temp.dNo.
and sal = msal;

Then ✓ ① Then ② ✓

② Select eid
from Emp T1,
where Not Exist (select * from Emp T2
where T1.sal > T2.sal and ~~dNO~~ T1.dNO = T2.dNO)

> some salary gives ✓

Lecture - 13.

SQL Query which is equal to Division of R.A :-

Enroll (sid cid) Course (Cid Instructor)

Retrieve sid's who enrolled every course taught by KORTH.

$$\pi_{sid,cid}(Enroll) \;/\; \pi_{cid}(\sigma_{Instr=Korth}(Course))$$

***

But in SQL there is No ~~Related Query~~ division operater ✓ → so, we have some standard procedure for it ✓

SQL → (Corelated Nested Query) → Kyun ki baar-2 Si ki value check krra hai lekr ki wo poora course liya hai ki Nhi.

Concept: check one by one from sid ✓

Enroll T_1

sid	cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C4
S3	C4

Course

Cid	Inst
C1	Korth
C2	Korth
C3	Korth
C4	Navathe

Enroll T_2

Sid	Cid
S1	C1
S1	C2
S1	C3
S2	C1
S2	C4
S3	C4

$= \phi$

Retrieve T_1.sid from Enroll (T_1)

if { all cid's of Korth Courses } − { Cid's of Enroll (T_2) which are enrolled by T_1.sid } ✓

ex

$T_1.sid = S_1 \Rightarrow \{C_1, C_2, C_3\} - \{C_1, C_2, C_3\} = \phi$ ← It means S1 Enrolled all the Korth courses

($\{C_1, C_2, C_3\}$ → all Korth courses; $\{C_1, C_2, C_3\}$ → Enrolled courses by S1)

$T_1.sid = S_2 \Rightarrow \{C_1, C_2, C_3\} - \{C_1, C_4\} \neq \phi$ ← Not Enroll all the Courses ✓

$T_1.sid = S_3 \Rightarrow \{C_4, C_2, C_3\} - \{C_4\} \neq \phi$ ← Not Enroll all the Courses ✓

Select distinct sid
from Enroll T1
WHERE NOT EXISTS (select cid
from course
where Inst = KORTH
EXCEPT
Select cid
from Enroll T2
where T2.sid = T1.sid);

when Inner Query is ϕ Then it will provide True

we Inner Query have some Element then it will provide false

due to minus present

★★★★★ Ex ② Works For (eid pid) project (pid pname)

Retrieve eid's who works for every project of project DB ✓

{means use division}

Solⁿ R.A work $\pi_{eid\,pid}$(work for) / $\pi_{pid}(\sigma_{pname=dB}(\text{project}))$

SQL Query ✓

Works T1

eid	pid

Project

Pid	Pname
P1	DB
P2	DB
P3	DB
P4	CN

−

T2 works

eid	Pid

= ϕ ✓

Retrieve T1.eid from Worksfor(T1)

If { all Pid's of Pname: DB } − { Pid's works by T1.eid } = ϕ

Select distinct T_1.eid

① from Works for T_1 → If inner Query == ∅ then it will give True Value.

③ Where Not Exists (Select pid
from project
where pname = DB) → ② It will select all pids which is Belongs to project data Base

Except

It will do subtraction operation ✓

Select Pid
from Works for T2
where T2.eid = T_1.eid);

· It will give all Pid of (work for) for which we come Here to calculate that, it is Belonging to that eid which uses all data Base Project or not.

SQL finish ✓

Imp concept :-

① An SQL Query can contain having clause only if it has a Group by clause. ✓

② all attributes used in the group by clause need to appear in the select clause ✓

TRC { Tuple relational Query }

⇒ Non-Procedural Query language.

⇒ Uses first order logic && predicate calculus formulas.

⇒ Basic formulas uses in TRC :-

P, Q are Predicates.
↳ [statements which can be True/false].

- P ∨ Q
- P ∧ Q
- ¬ P
- P → Q ⇒ If P then Q
- X ∈ Rel [x is a Variable]

Quantifiers:

- ∃ X ∈ R(P(X)) → ∃
  For Some X which belongs to Relation such that P(X) must be TRUE.
- ∀ x ∈ R(P(x)) → ∀
  For every X which belongs to Relation. such that P(x) must be True

Formate of TRC Query :-

{ T / P(T) }

T: Tuple Variable
P(T): formula over tuple variable T

Ye btayega kis kya Hmko Find krna hai

Ye btayega kis cond^n kya hai us cheez ko Find krne ke liye

Retrieves set of Tuples (T) those are satisfied P(T)

{ T / P(T) } — P(T): T ∈ stud ∧ T.age > 20

Retrives students whose age more than 20.

* { T / P(T) }
  T must be free variable

Tuple Variable

↳ **Bounded Tuple Variable** :- Variable Bounded by Quantifiers [∃, ∀]

Ex :- ∃ T, ∈ stud (- - - - -)

Bounded → ∀T, ∈ Course (- - - - -)

Free Variable :-

→ Not Bounded by Quantifier

→ Also called universal Variable.

Ex T ∈ stud.

↑ Not Bounded Tuple Variable {Free Variable ✓}

Ex :- Suppliers (Sid rating)

Parts (Pid, Pname, color)

Catalog (Sid Pid Cost)

Retrieve sid's of suppliers who supplied some red part.

Sol^{n} catalog (Sid, Pid) = Parts (Pid, colour)

↑ $\exists T_1$ ↑ $\exists T_2$

$\Pi_{sid}\left(\sigma_{\text{Catalog.Pid} = \text{Parts.Pid}}\left(\text{Catalog} \times \sigma_{col=Red}(\text{Parts})\right)\right)$

$[T / \exists T_1 \in \text{catalog}\ \exists T_2 \in \text{Parts}\ (T_1.Pid = T_2.Pid \wedge T_2.color = RED \wedge T = T_1.Sid)]$

↑ some

Pehle T_1 aur T_2 define kiya ki kya hain wo

Phir Trace like kiya

At last T me kya dena hai wo likha

$\sigma_p(R \times S)$
$R \bowtie_p S$
$R \bowtie S$

} Some / any / atleast one.
Express by using $\exists$ existential Quantifiers.

Ques Retrieve sid's of suppliers who supplied atleast two parts.

Soln

Catalog (sid Pid) $\exists T_1$ = Catalog (sid Pid) $\exists T_2$

$$\Pi_{sid}\left(\text{catalog} \underset{(sid = s \wedge pid \neq p)}{\bowtie} \rho_{s,p,c}(\text{Catalog})\right)$$

$$\{ T \mid \exists T_1 \in \text{catalog}, \exists T_2 \in \text{catalog}.$$
$$(T_1.sid = T_2.sid \wedge$$
$$T_1.Pid \neq T_2.Pid \wedge$$
$$T = T_1.sid) \}.$$

• Some, any, atleast one → Replace by $\exists$

Set operators :-

$R \cup S = \{T \mid T \in R \vee T \in S\}$ — T Belongs to R or T Belongs to S

$R \cap S = \{T \mid T \in R \wedge T \in S\}$ — AND.

$R - S = \{T \mid T \in R \wedge \neg T \in S\}$ — { T Belongs to R But Not T Belongs to S } but not

Que Retrieve supplier id's whose rating maximum.

Soln $\pi_{sid}(supplier) - \pi_{sid}(supplier \bowtie_{rating<R} \rho_{S,R}(supplier))$

TRC

The best Idea to write TRC Query Firstly write R.A Query then form it into TRC By using it's Basic Rule (T/ Formula)

$\{T \mid \exists T_1 \in supplier\ (T = T_1.sid) \wedge$

$\neg \exists T_1 \in supplier\ \exists T_2\ supplier$

$(T_1.rating < T_2.rating \wedge T = T_1.sid)\}$.

(or)

rating $\geq$ Every rating

$\{T \mid \exists T_1 \in supplier\ \forall T_2 \in supplier.$

$(T_1.rating \geq T_2.rating \wedge T = T_1.sid)\}$.

```
Select sid
From supplier.
where rating
>= All (Select
rating
from supplier);
```

Conclusion :-

→ **Unsafe TRC Query :-**

↳ *** TRC Query with infinite records in Result.

Ex :- $\{T / \neg (T \in supplier)\}$:→ unsafe TRC Query.

↑ Infinite

→ Expressive power :-

**** Basic RA Queries expressive power $\equiv$ Safe TRC Queries expressive power.

{∪, ∩, ⋈, ×, σ, ×, /, % ... etc}

kyun KP dono ek doosare se derived kiye ja skte hain ✓

Basic RA Queries :-

Queries which can express using :-

$\{\pi, \sigma, \times, \bowtie, \bowtie, \exists\bowtie, \forall\bowtie, \cup, \cap, -, /\pi \div\}$ operators.

Queries which are failed to express using :-

(Basic RA | Safe TRC)

⇒ Count of records | Count of attributes Values

⇒ Sum of attribute Values

⇒ Avg of attribute Values.

countx
sumx
avgx
orderx
Proof duplicatex

→ Ordering of Records.

→ Failed to prove duplicate Records Existance. etc