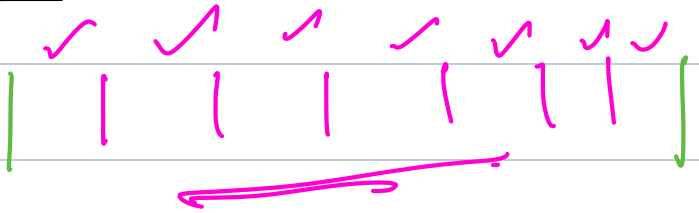
Agenda

- → Sub queries
- → IN clause
- → Sub queries and FROM
- → ALL and ANY
- → co-related Sub queries
- → EXISTS
- → VIEWS

start at 9:10

Sub queries



- → solve Problem 1 → get result.
- → use That result in problem 2.

function main a() b(arg).

```
int result_a = (a());
b(result_a);
```

Students

id name PSP batch_id.

→ Given a students Table, find all the students with PSP > the PSP of student with id = 18

```
select *
from students s
join students s1
on s1.id = 18 and s.PSP > s1.PSP.
```

```
select * from stu
where psp >
((select psp from stu
where id = 18))
```

→ find PSP of student id = 18.
→ compare every student PSP with it.

→ Given students Table, find all the students who have PSP greater than maximum PSP of batch 2

→ max PSP of batch 2 ⇒ (x).
→ students with PSP > x.

pseudo

```
ans = [ ]
for all students s
```

if s.psp > x
ans.add(s)

return ans.

```
Select *
from students
where psp > x.
```

```
select max(psp)
from students
where batch_id = 2.
```

⇒ (x').

replace x.

```
select *
from students
where psp > ( select max(psp)
              from students
              where batch_id = 2 )
```

→ easy to implement

→ easier and more Readable.

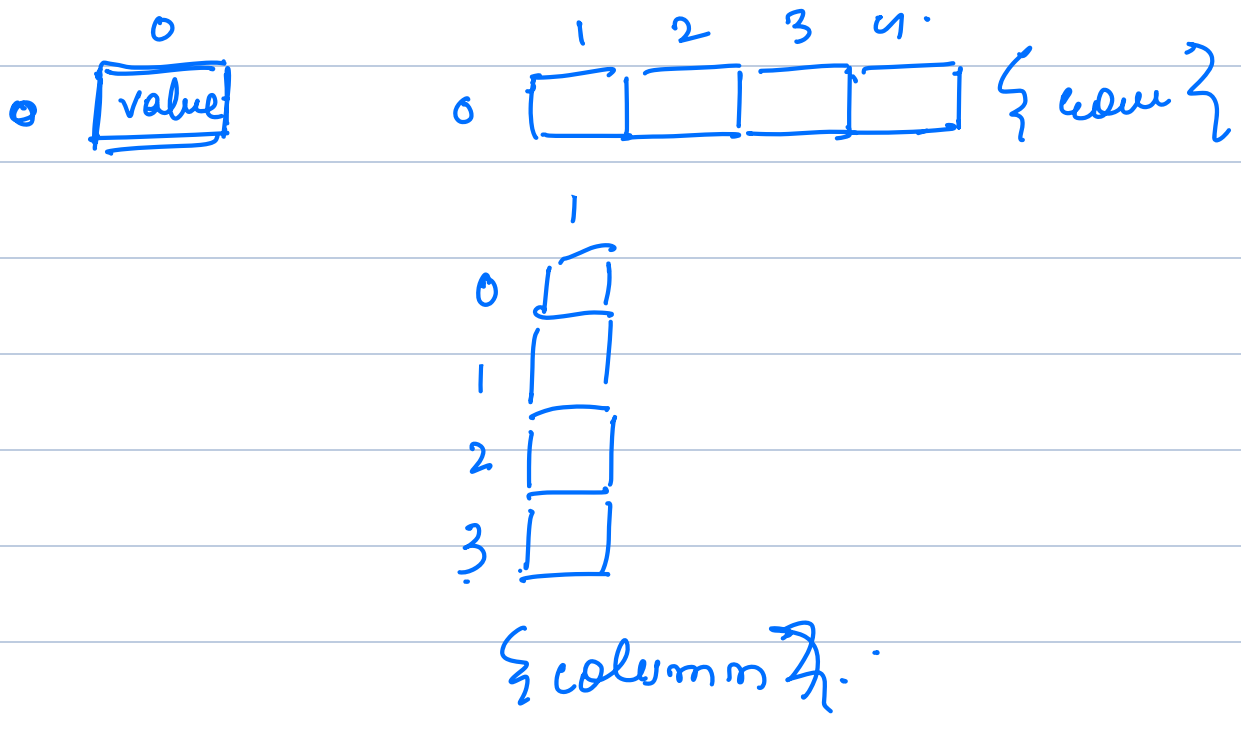
Trade offs

→ Bad performance ** (in most cases).

for students (comparison of PSP)

for student (find id = 14).

No of rows	No of col^m	output
1	1	single value.
1	m	row.
m	1	column.
m	n	Table.



Subqueries and IN.

users

id	name	is_student	is_TA.
		True	True.

Q

find the names of students that are also the names of a TA.

```
select Distinct S.name.
from users S
Join users T.
on S. is_student = True
   And T. is_TA = True
   S.name = T. name .
```

complex?!

students TA.

```
select *
from s
join TA.
ON S.name = T.name.
```

(R) → set names of TAs (is_TA = True)

→ compare them with students (is_student = true)'

```
Select Distinct name
  from users U
  where U.is_TA = true;
```

```
select Distinct name
from users U                    parent.
where U.is_student = true
AND U.Name IN x.                replace . x .
      (select Distinct name
         from Users V
         where V.is_Ta = true)
```

(child)

Break till 10:25

Subqueries and from

students

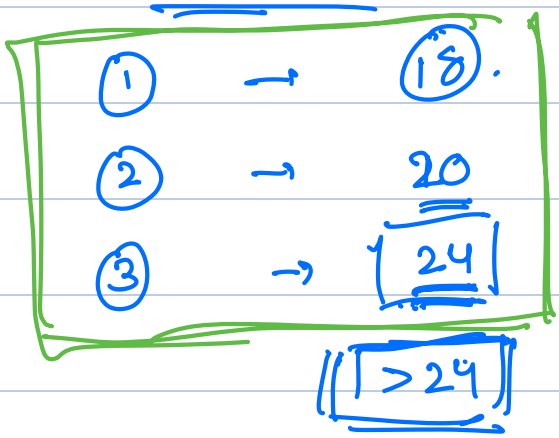
id	name	PSP	batch_id

Q. find all the students whose PSP

is not less than the smallest PSP of any batch.

greater than or = smallest PSPs of every batch.

① → find min PSP of every batch ⇒ x.
② → find max psp from x[] ⇒ y.
③ → student PSP >= y.



18

19 ✗

x:

```
Select min(PSP) from students
GROUP BY batch_id
```

y:

```
select max(PSP) from x.
```

result:

```
select * from students
  where PSP >= y.
```

```
Select *
from students
```

where PSP >=

(select max (PSP)
from (
select min (PSP)
from students
group by batch_id
) as min psps.
)

derived Table

Select Sub.alias_name
from (select col_name As alias_name
from table_name) As Sub
where Sub.alias_name = 'value'.

child query.
Parent query want to reference a child query you need an alias.

ALL and ANY.

→ if we want to find if a value

x is greater than all the values in a set, we use ALL

PSP > ALL (10, 20, 30, 40)
↑ ↑ ↑ ↑

18 : false. 1 And 0 = 0

29 : false. 1 And 1 and 1 = 1

41 : True.

if ALL True : return True
else return false.

```
select *
from students
where PSP > ALL
    ( select min (PSP)
      from students
      group by b_id )
```

ALL Typically with single-column sub queries.

```
                    name PSP
PSP > ALL
```

PSP > ANY (10, 20, 30, 40)

18 > 10

18 > Tenue

1 or 0 = 1

if ANY Tenue > returns
else - returns false.

quiz

PSP 20

ANY (10, 20, 30)

IN (10, 20, 30)

Co-related subqueries

Q Find all students whose PSP is greater Than average PSP of Their batch

→ get student PSP > x.

→ x = average PSP of students batch

child query dependent on The Parent Query

{ select avg(PSP)

y = from students
where batch_id = n

```
select *
from students
where psp > y
```

```
select *
from students S
where psp > (select avg(psp)
             from students T
             where T.batch_id = S.batch_id)
```

co-related → subquery

co-related subqueries are queries where the subquery uses a variable from the parent query.

Exists

students

id	name	psp

TA

id name st_id.

find all students who are also TA in the given 2 Tables

→ we want all the TA whose st_id is not null.

→ student id match TA.st_id.

```
select *
from students
where id In
    ( select st_id
      from TA
      where st_id is not Null )
```

entire Table →

is null because not every TA will be a student.

st_id

```
Select *
from students s
where id In
EXISTS
    ( select st_id
      from TA
```

→ only need one row in the subquery to give True

where [Tas.st_id = S.id]

index

one or multiple rows

IN [val1, val2, val3.]

is exist faster or not. ✓

→ index match

→ ~~does~~ we do not have to check for entire values returned by subquery.

does exist take less memory?. ✓

yes

for parent rows

for child rows.

Exist ([R1], [R2], [R3]) ? true

Exist ([ ]) :. true.

Exist ( ) :. false.

for row in students
    ans = [ ]
    for row in Ta.
        if (Ta.st_id = st_id)
            ans.append.

if ans ! null.

select * from condition

for row in Table.
    if (condition).
        append.