Subqueries

Subqueries → intuitive way to write SQL queries

Agenda:

- Subqueries
- IN
- ALL
- ANY
- Correlated Subqueries
  → EXISTS
- Subqueries in FROM
- Subqueries in where

FACTS. (learn for this class)

(1) There is a thing called index.
If we have index on a column, search for exact value in that column becomes faster than $O(n)$, something like $O(\log n)$.

(2) We can have index on any column, PK has an index BY DEFAULT.

Given a student table find all the students who have psp > the maximum psp of student of batch 2

Student

id	name	psp	batch-id

In code.

students = [ ]

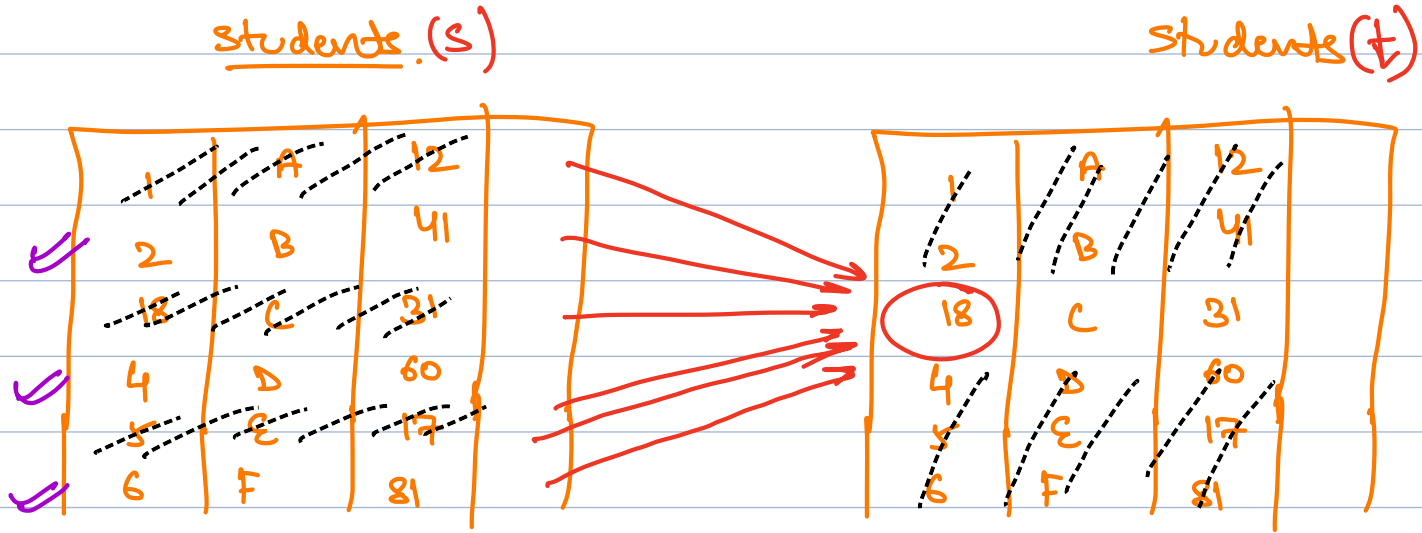
get all students whose psp is greater than max psp of batch 2.

algo : find max psp of batch 2 $\Rightarrow$ (x)

```
for all students (s):
    if s.psp > x :
        ans.add (s)
return ans.
```

Q find all of the students whose psp > psp of student with id = 18.

```
select  s.id
from    student s
join    student t
on      t.id = 18  and
        s.psp > t.psp
```



In reality, we prefer

① Break problem into parts

② Solve smaller problems and use their results to solve bigger problems.

↓

Subqueries : Break a problem into smaller problems and combine their result to get complete answer.

most of the times

Problem that we solve via subqueries can also be solved via some other smart trick.

But subqueries make our queries easier to understand, create.

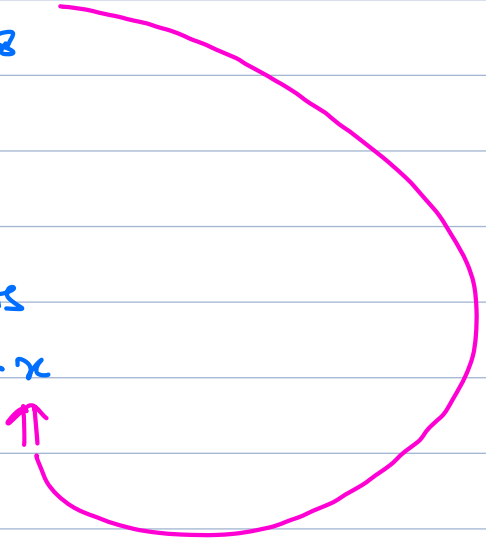
Q find all of the students whose psp > psp of student with id = 18.

→ ← ① find the psp of student with id = 18

② find all the students with psp > x.

① select psp
from student
where id = 18

② select *
from students
where psp > x



```
select *
  from students
  where psp >(select psp
              from student
              where id = 18) ;
```

✓ Readable
✓ Intutive

* subquery should be always enclosed in parantheses

Trade off of subqueries => Bad performance

```
Select *
  from students
      where psp >18
```

students = [ ]

ans = [ ]

```
for student s : students :
    if s.psp >= 18
        ans.add(s);
```

without subquery

```
for student s : students :          → N
    if s.psp >= 18
        ans.add(s);
```

```
ans = []
for student o : students :
    if o.id = 18
        ans.add(o);
```

⇒ N

```
return ans[0].psp;
```

⇓

$O(N^2)$

Q Given a student table find all the students who have psp > the maximum psp of student of batch 2

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

[CODE] → Tell all years where the average of the rental_rate of films of that year was greater than global average of rental rating (avg of all the films)

↓

① find global average

```
select avg (rental_rate)
from film;
```

② find avg of every year

```
select release_year , avg (rental_rate)
  from film
   group by release_year;
```

③ Get filtered groups

```
select release_year , avg (rental_rate)
  from film
   group by release_year
     having avg(rental_rate)> (
                    select avg (rental_rate)
                      from film;
                    )
```

Q Was this intuitive to think & to break into sub problems? → YES, that is what we did.

[Break]

⇒ Subquery we wrote till now just gave me a single value as o/p.

→ A subquery can give any of

rows { cols

x	c	
1	1	→ single value
1	m	→ single row
m	1	→ single column
m	m	→ table

We have already learnt how to compare with single value subquery.

Q

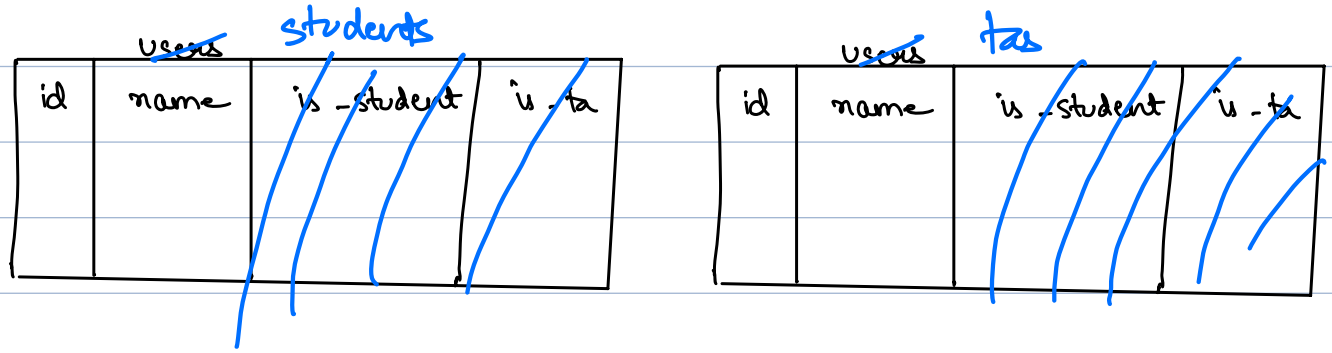
users

id	name	is_student	is_ta

Tell the name of students that are also names of TA.

using joins / without subqueries

```
select distinct s.name
from users s
join users t
 on t.is_ta = true and
    s.is_stud = true and
    s.name = t.name
```




```
select distinct s.names
   from students
    join tas
     on s.name = t.name
```

```
select distinct s.name
  from users s
  join users t
   on s.is_stud = true
   and t.is_ta = true
   and s.name = t.name
```

for this question.

with subqueries

① Get names of all TAs ⇒ [ ]

② Get students whose name is in ↩

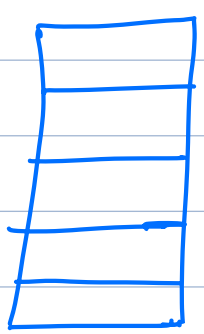
```
select distinct names
  from users
  where u.is_ta = true
```

```
select distinct name
  from users u
  where u.is_stud = true
    and u.name in (____);
```

final answer

```
select distinct name
  from users u
  where u.is_stud = true
    and u.name in (select distinct names
                   from users
                   where u.is_ta = true)
```

↳ this works in case of one column with as many values.



Q Get all the students whose psp is not less than smallest psp of any batch.

batch	min
1 →	20
2 →	30
3 →	39

student = 18 X
= 32 X
= 43 ✓

all students whose psp is greater than minimum psp of every batch.

s vs (— — —)
min psp

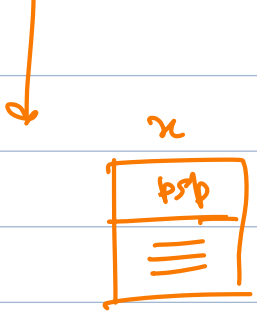
greater than each
or
not smaller than any of them.

→ Get students whose psp greater than min psp of every batch. (1)

(1) select min(psp) from students group by batch_id

⇒ 30
31
46

→ so basically student's psp should be greater than max of this list



② select max(psp)
from x ①

③ select *
from students
where psp > y ②

```
select *
from students
 where psp >(select max(psp)
                from (select min(psp)
                      from students
                      group by batch_id
                     ) minpsps
 );
```

⇒ Subquery in FROM

⇒ we can have subquery in from clause.
→ this subquery's o/p is considered a table in itself.

→ upon which you can write any other read queries.

→ you should name a subquery in from clause. MANDATORY.

if we didn't have subquery in from

x > ( )

↳ If we want to check x is greater than all values in this list?

ALL

```
select
from students
where psp > ALL ( select min(psp)
                  from student
                  group by batch_id) ;
```

x > ALL (10, 19, 41, 20)

★ Compare LHS with every value of the RHS. If all of them return true, ALL will return true.

AND → OR

ALL → ANY

ANY → compares LHS with all the values of RHS. If any of them returns true, ANY returns true.

Correlated Subqueries → subquery uses a variable from parent query.

Q Get all the students whose psp is greater than average psp of their batch.

① Get students with psp > (x)

② x = Avg psp of student's batch (y)

①
```
select *
from student
where psp > x
```

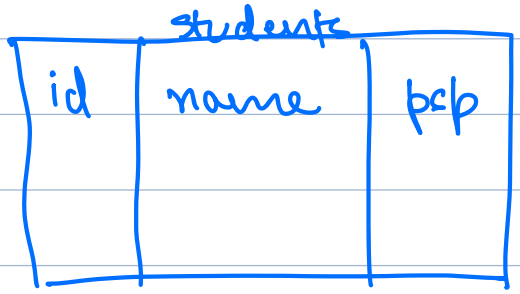
②
```
select avg (psp)
from students
where batch_id = y
```

```
select *
from Student s
    where psp > (select avg (psp)
                 from students
                 where batch_id = s.batch_id)
```

x join ( ) subquery are not allowed in subquery as a join

EXISTS

Q Get all students who are also TA.



Without joins.

```
select st_id
from tas
    where st_id is NOT NULL
```

```
select *
from students
where id IN (select st_id
             from tas
             where st_id is NOT NULL);
```

IN → check the value one by one → O(N)

O(N)

* if you want to check for ids in a set of values.

EXISTS. (we can use parent's parameter in child.)

```
select *
from students
where EXISTS (
    select st_id
    from tas
    where ta.st_id = s.id)
```

for every row of students :

1. it will run subquery
2. if the subquery returns any no. of rows >0, it returns true.

→ Since exists has s.id, it can use indexes to optimise queries / search within the subquery.

Q

students

id	name

mentor_sessions

s_id	stud_id	mentor_id

Get all student names that has taken a mentor session.

```
select *
from students
where id in (
        select stud_id
        from mentor_sessions
    );
```

SLOW

Query : IN slows

↓

IN : Stops as soon as found

but if it is very late

```
select *
from students s
where exists (
        select *
        from mentor_sessions
        where st_id = s.id);
```