Subquery and Views

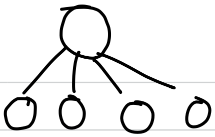
① Subquery

- ✓ Basics
- ✓ Subquery and IN clause
- ✓ Subquery and FROM clause
- ✓ ALL , ANY
- ✓ Correlated Subqueries
- ✓ Subquery in WHERE clause

② Views

- Basics
- Syntax

SUBQUERY



Big Problem
↑
Subproblems

Bigger query
↓
Smaller query

→ intutive way

→ Example - Students Table, find out all students who psp greater than max psp of batch_id = 2

output : S4, S6

name	psp	b_id
S1	70	1
S2	60	1
S3	40	2
S4	80	1
S5	70	2
S6	85	3

```
ans = []
for s in students:
    if s psp > [max psp of batch 2]   ✗
        ans.append(s)
```

70

Calculate.

X →

```
SELECT max(psp)
FROM students
WHERE batch_id = 2;
```

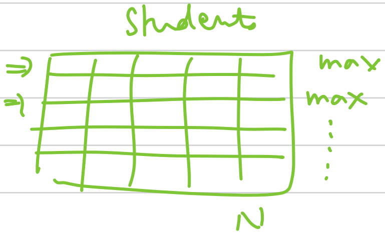
(70)

final query

```
SELECT *
FROM students
WHERE psp > (SELECT max(psp)
             FROM students
             WHERE batch_id = 2);
```

indepent of all rows outside but sometime it can depends on what row is selected.

get executed for every row in students. theoretically.



$O(N^2)$ is worst case.

The subquery gets executed for every row leading to bad performance, but Query engines are optimised to do all sorts of performance enhancements.

Subquery & IN Clause

users

id	name	is_student	is_TA
1	C	T	X
2	A	T	X
3	B	X	T
4	C	X	T
5	C	X	T

Self join

Find names of student that are also the names of TA.

Same person can't be both.

TA list = [B, C]

C → [C]

```
SELECT *
FROM ~~student~~ users s
JOIN ~~student~~ users ta
ON s.name = ta.name and s.is_student = True and ta.is_TA = true;
```

```
for s in ~~students~~ users:
    if s is a student and s.name in TAlist.
    {    output.add(s), }
```

```
SELECT DISTINCT name
FROM   users U
WHERE     U.is_student = True   AND
          U.name IN (SELECT DISTINCT name
                       FROM users U
                     where   U.is_TA = true);
```

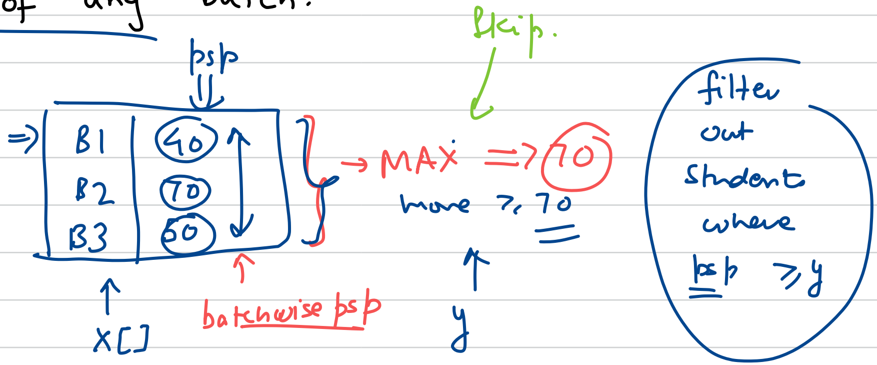
Subquery returns a list.

15 IN (5, 6, 8)   {5,}

SuB Query & FRom Clause

- Find all students whose psp is not less than smallest psp of any batch.

S1	B1	60
S2	B2	70
S3	B3	50
S4	B3	75
S5	B2	80
S6	B1	40



```
x → SELECT min(psp)
    FROM students
    GROUP By batch_id

y → SELECT
    max(psp)
    FROM x;

SELECT *
FROM student
where psp ≥ y;
```

```
SELECT *
FROM student
where psp >= (SELECT
              max(psp)
              FROM   (SELECT min(psp)
                       FROM students
                       GROUP BY batch_id) batchwise psp   );
```

when you have subquery inside FROM.

not less than any of these of values

ALL , ANY

find out students psp ≥ ALL(40, 60, 70)

⇑

75 ≥ ALL (40, 60, 70) Yes.

65 ≥ ALL (40, 60, 70) No

```
SELECT * FROM
   Students WHERE
   psp ≥ ALL (
          SELECT min(psp)
            FROM students
            GROUP BY batch-id);
```

40
60
70

10.20 pM

75 ≥ ANY (10, 100, 1000) yes

75 ≥ ANY (100, 200, 500) NO

↓

alteast one.

find out student who belong any of ⇒ {B1, B2, B3}

```
SEECT *
 FRom Student
  WHERE batch_id = ANY (1, 2, 3);
           |||
        batch_id IN (1,2,3);
```

Correlated subqueries

Example: find out all students whose psp is greater ≥ avg psp of their batch.

				cmp
1	S1	10	B1	15
2	S2	20	B1	15
3	S3	30	B2	40
4	S4	40	B2	40
5	S5	50	B2	40
6	66	60	B3	60

B1	15
B2	40
B3	60

correlation.

depends upon batch S

```
for s in student:
    if s.psp ≥ psp of batch of s
        then output.add(s);
```

avg psp of ~~on~~ those Students who belong batch of S.

↓

```
SELECT * FROM
    Students S
    WHERE psp >=  ( SELECT AVG (psp)
                    FROM Students
                    WHERE batch_id = S.batch_id, )
```

Correlated query.

$O(N^2)$

EXISTS

→ another clause .

Same student can become a TA.

Find out all students who are TA.

→ students

id	name	psp
1	S1	80
2	S2	70
3	S3	90
7	S7	80

→ tas

id	name	st_id
1	S3	3
2	A	~~null~~
3	B	~~null~~
⋮	.	7

if you have a index on this col.

{3,7}

① Joins ✓

② Subquery

```
SELECT * FROM students
WHERE id IN (SELECT st_id
               FROM tas
             WHERE st_id IS NOT NULL),
```

Subquery in WHERE Clause.

multiple Rows

[3,7,21,6 . . . .]

list

EXISTS : For every row of students it will run a subquery, if The subquery returns at least one row, it return True.

```
SELECT *
FROM students S
WHERE EXISTS ( SELECT st_id FROM tas WHERE tas.st_id = S.id)
```

yes | No

=>

~~Yes or~~ Single value.

True

1s Rows

10 Rows

exploit indexing

faster