



7-March : Indexing (9:00 AM)

: Class Update:

Shifting

Fri (8 March) → Thu (7 March)

Mon (11 March) → X OFF PM

wed (13 March) → Resume

[Fri - Tue] ⇒ No class

Agenda :

- ✓ Subqueries
 - ✓ IN Clause
 - ✓ FROM Clause

✓ ALL , ANY

✓ Correlated Subquery

✓ EXISTS

✓ Subquery in WHERE Clause

Views

Sub-query

→ breakdown a query into small queries and then combine the result to get complete answer.

Students

id	name	psp	batchid
----	------	-----	---------

Q) Find out all students who have $psp > \underbrace{\text{max psp of all students in batch 2.}}$

Algo:

$x \rightarrow \underline{\text{max psp of all student in batch 2}}$
ans = []

$O(N + N)$

```
for s in students :  
    if ( s.psp > x )  
        ans.add(s)  
→  
return ans;
```

SQL :

```
SELECT MAX(psp)  
FROM Students  
WHERE batch_id = 2
```

```
SELECT *  
FROM Students  
WHERE psp > x
```

↑
x

Combine

Final
query
=

```
SELECT *  
FROM Students  
WHERE psp > (SELECT MAX(psp)  
FROM Students  
WHERE batch-id = 2),
```

may return a
diff for
every
row
↓
correlated
subquery.

Theoretically $O(N^2)$

internal optimisation $O(N)$ possibly

TODO:

Q)

Find out students whose $psp > psp$ of
student with $id = 18$.

```
SELECT *  
FROM Students  
WHERE  $psp > ($   
    SELECT  $psp$   
    FROM Students  
    WHERE  $s-id = 18$ ;  
)
```

- Todo: Sakila Database
- For the film table, find out all the years where
- average rental_rate of films of that year was greater $> =$
- than average rental_rate of all films

M1	2006	100
M2	2008	200
M3	2005	300
M4	2006	400

Avg Rental \rightarrow 250/-
↑
x

grouping
acc
to
year

\rightarrow 2006 \Rightarrow 250 RS
 \rightarrow ~~2008 \Rightarrow 200 RS~~
 \rightarrow 2005 \Rightarrow 300 RS

filter out
the
groups

SELECT ^{AS avg} AVG(rental-rate), release-year
FROM film

GROUP BY release-year

HAVING avg >= (SELECT AVG(rental-rate)
FROM film);

↑
→ single value
→ multiple values?

Concept-II

Subqueries and IN Clause

⇒ Users can be either student or TA but not both.

id	name	is-student	is-TA
----	------	------------	-------

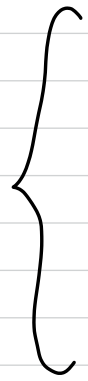
(1,7)	1	Varun	T	F
(1,4)	5	Deeksha	F	T
(1,6)	2	Deeksha	T	F
(4,1)	6	Varun	T	F
	3	Rohan	F	T
	7	Varun	F	T
	4	Varun	F	T

Q) Find names of students that are also the names of TA.

{ Varun, Deeksha
Varun, } ⇒ { varun, Deeksha }

Todo:

Self-Join


 SELECT DISTINCT
^ S.name

FROM users S
JOIN users T

ON S.name = T.name AND
S.is-student = true AND
T.is-TA = true;

Subquery: 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);

 List

list of names

== ---

(ABC, XYZ, Varun)

Concept - III Subqueries in FROM Clause

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

SELECT * FROM students WHERE psp > (27)

[SELECT max(psp) FROM (

⇒ [SELECT min(psp).batch-id

FROM students

GROUP BY batch-id

) min psp Table

B1	26
B2	18
B3	25

min psp Table

);



10.20

CONCEPT - IV

ALL and ANY

Find if a val is greater than all values in the set

$$(X) > \underline{\text{ALL}} \{ \underline{10}, \underline{20}, \underline{30}, \underline{60} \}$$

→ 55

NO

→ 65

Yes.

AND
if all comparisons
ret True

$$X > \text{ANY} \{ \overset{T}{\underline{10}}, \underline{20}, \underline{30}, \underline{60} \}$$

OR

→ $x = 15$

yes

→ $x = 5$

NO

27
~~21~~

SELECT * FROM Students
WHERE PSP >

ALL (SELECT min(psp) :
FROM students
GROUP BY batch-id);

26 ← ✓
18 ← ✓
25 ← ✓

What is output of $x = \text{ANY}(a, b, c)$ same as

21 users have participated

A	=		5%
B	ANY		38%
✓ C	IN		48%
D	ALL		100%

↑
Smallest
psp of
every
batch

Concept IV

Correlated Subqueries

a) Find out all students whose psp > average psp of students of their batch.

S1	B1	10	> 20	No
S2	B2	20	> 30	No
S3	B1	30	> 20	Yes
S4	B2	40	> 30	Yes
S5	B1	20	> 20	No

Avg	
B1	20
B2	30


S3
S4

✱

```

SELECT * FROM students S
WHERE |ps| > (
    SELECT avg(psp)
    FROM student
    WHERE batch_id = S.batch_id
)

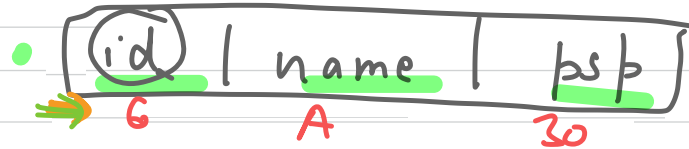
```



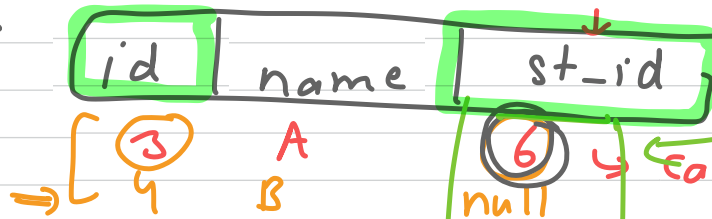
$$\frac{10 + 30 + 20}{3} \Rightarrow 20$$

EXISTS clause

Students



tas



can be NULL if TA
is not student

B) Find out all
students who are also TAs.

Annotations: An orange line underlines 'students' and 'also TAs'. A red arrow points up to the underlined 'students'.

~~①~~ SELECT * FROM students
WHERE id IN

(
SELECT st-id
FROM tas
WHERE st-id IS NOT NULL;
)

↑
(5, 7, 32, 64)

~~②~~ SELECT *
FROM students
WHERE EXISTS

optimal query in terms of memory and space

⑤

(
SELECT st-id
FROM tas
WHERE tas st-id = S.id
)

at least 1 Row then
Student from outer
query will
be selected.

lookup
faster (index)

III

SELECT *
FROM students S
JOIN tas
ON S.id = tas.st-id

more
memory