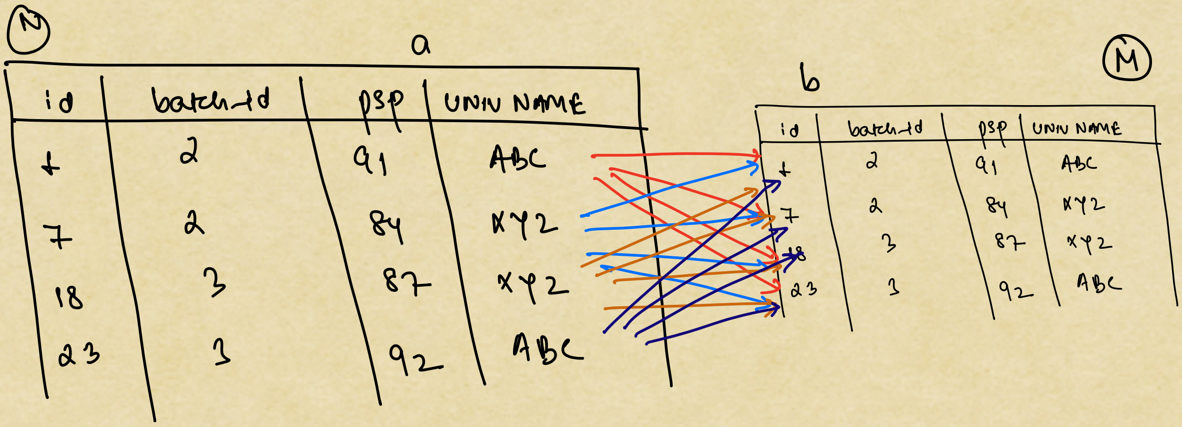
SUBQUERIES

→ makes read | write | understand complex queries easier.



id	batch-id	psp	UNIV NAME
1	2	91	ABC
7	2	84	XYZ
18	3	87	XYZ
23	3	92	ABC

id	batch-id	psp	UNIV NAME
1	2	91	ABC
7	2	84	XYZ
18	3	87	XYZ
23	3	92	ABC

$N \times M \Rightarrow N == M$

$\underline{\underline{N^2}}$

Q1. Find all the students who have a psp greater than psp of student with id = 18

```
select a.id
from students a
join student b
on a.psp > b.psp
and b.id = 18.
```

Student id
1
23

Sub-queries :

→ Rarely used to do things that are otherwise not doable or very difficult.

→ Mostly used to break down complex queries into easier to understand.

→ Most of the things that you can do with subqueries, can be done with joins

Q1. Find all the students who have a psp greater than psp of student with id = 18

i) get the psp of student with id → 18 ⇒ 87

```
select psp from students where id = 18;
```

ii) get students with psp > 87

```
select * from student where psp > 87
```

⇒ Combines (I) & (II)

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

1)

```
select a.id
from students a
join student b
on a.psp > b.psp
and b.id = 18.
```

N2

Better to understand ⇒ ②

Better in performance ⇒

(1)

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

N2

the query run N times, and for each run. sub query runs N times

N2

performance overhead ⇐

id	batch id	psp	UNIV NAME
1	2	91	ABC
7	2	84	XYZ
18	3	87	XYZ
23	3	92	ABC

Q2. Print the names of student who are TA as well

Students

id	name	batchId	psp
1	Nilesh	2	98
2	Nitesh	3	99
3	Kunal	2	96
4	Sushil	1	90

TA

id	exp	Student Id
1	4	2
2	3	NULL
3	1	1

i) Join =>

```
select s.name from student
join ta t
on t.studentId = s.id;
```

ii) Subquery =>

```
select name from student
where id in( select studentId
from TA where studentId is not null);
```

id	marks
1	90
2	60
3	72
4	83
5	91
6	84

=> give all ids that
have marks
> 70, 65, 80, 95

id	marks	b_id
1	90	6
2	60	2
3	72	2
4	83	9
5	91	6
6	65	1

id of all Students
whose marks > marks all
Students b-id 2

a ✓ b ✓

>60 & >72

>72

id	batch-id	psp	UNIV NAME
1	2	91	ABC
7	2	84	XYZ
18	3	87	XYZ
23	3	92	ABC
24	1	97	

Q3. select all students who have marks greater than all students of batch 3

1) ① select * from students where psp >>

(select max(psp) from students group by batchid

having batch_id = 3);

ii) ALL => condition should match with all values

ii) ~~ALL~~ => all students should match cond^n

cond^n should match with all values

```
select * from students
where psp > ALL ( select psp from students where
                  batch-id = 3 );
```

i) getting multiple values from subquery

ii) match the cond^n against all values

iii) only if all values match the cond^n, o/p the row.

id	batch_id	psp	UNIV NAME
1	2	91	ABC
7	2	84	XYZ
18	3	87	XYZ
23	3	92	ABC
24	1	97	

Q 4. Select all students who have marks greater than any students of batch 3

i) select * from students where psp >

(select min(psp) from students ________)

ii) ANY ⇒ any single cond^n should match

```
select * from students
where psp > ANY ( select psp from students where
                  batch_id = 3 );
```

⇒ CORELATED SUBQUERIES:

Q Select all students who have a psp greater than avg psp of their batch

i) get avg psp of every batch

```
select avg(psp) from students group by batch_id
    having batch_id = batch_id;
```

ii) select students having psp > avg.

```
select * from students where psp >
    ( ________________ )
```

id	name	batch id	psp
1	A	1	96
2	B	2	90
3	C	1	95
4	D	1	91
5	E	2	94

avg1 → 94

avg2 → 92

Co-related query

```
Select Name from students (S) → upper query has a variable
where psp > (
        select avg(psp)
        from students
        group by batch_id
        having batch_id = S.batch_id);
                              ↓
                    being referred in
                    the sub query
```

* SUBQUERIES CAN ALSO BE USED WITH SELECT and FROM

⇒ SELECT

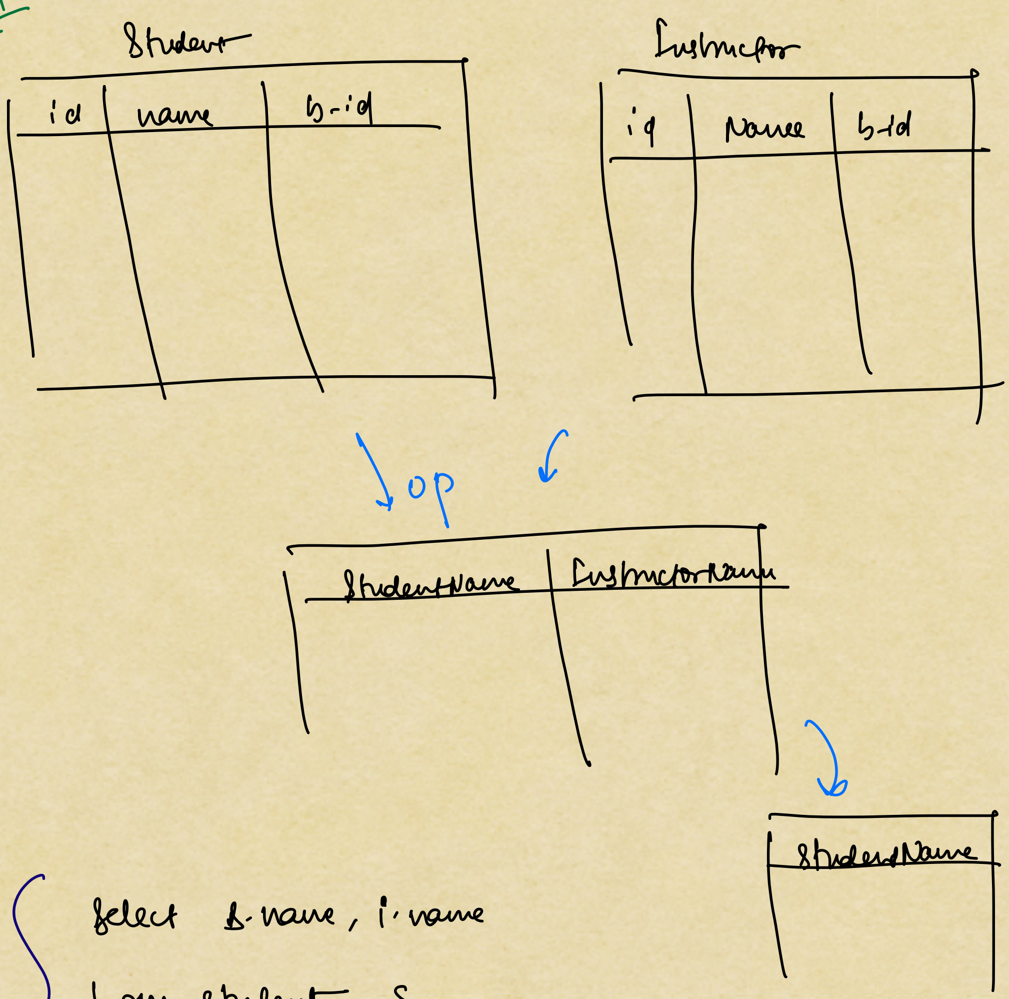
id	name	batch Id	psp
1	A	1	96
2	B	2	90
3	C	1	95
4	D	1	91
5	E	2	94

↓ Calculate

id	name	bId	avg Batch psp

```
Select id, name, batch Id, ( select avg(psp) from students
                             group by batch-id having
                                 batch-id = s.batch-id );
   from students s;
```

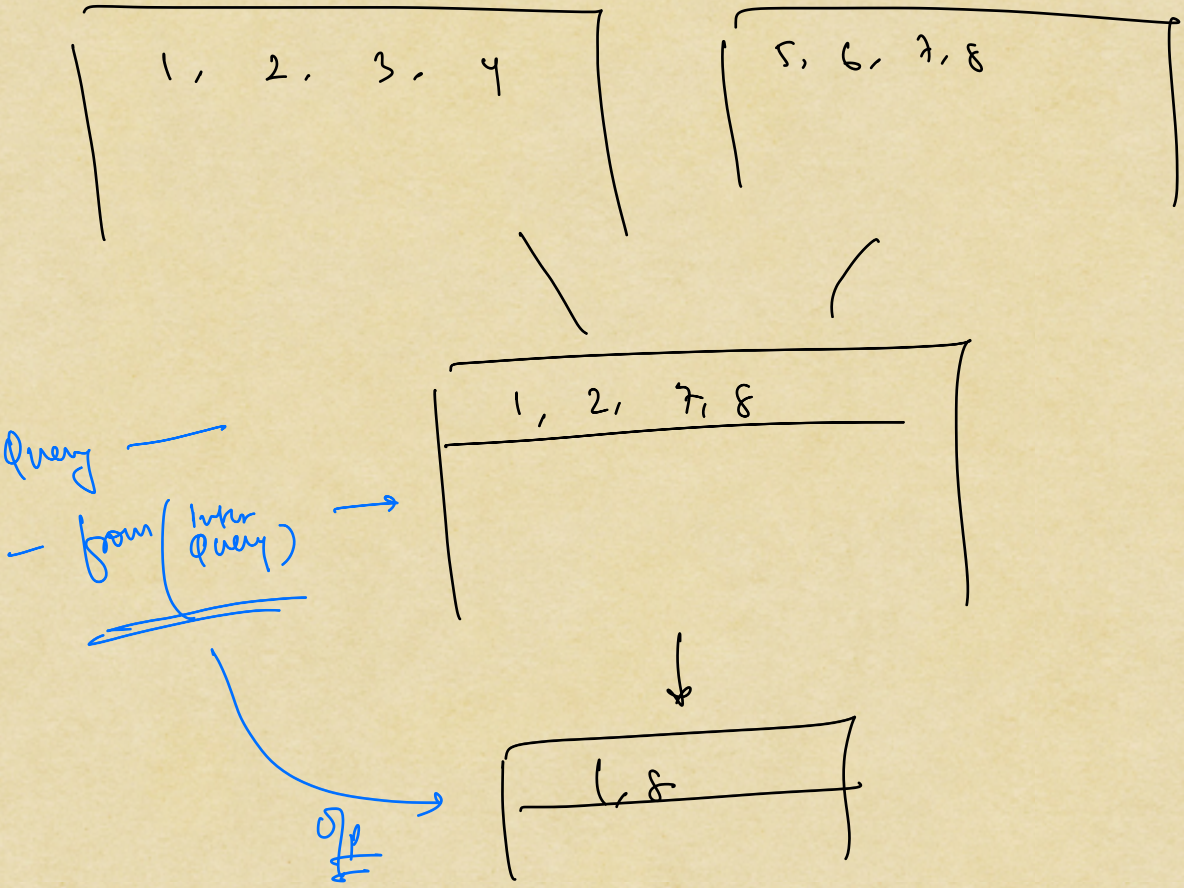
=> FROM



```
select s.name, i.name
from students s
join instructor i
on s.bId = i.bId;
```

as an o/p table or use alias

```
select t.StudentName
from (
    select s.name as StudentName, i.name
    from students s
    join instructor i
on s.bId = i.bId; ) t ;
```



Sub

```
select s.name, (select i.name from
                instructor where batchid
                   = s.batchId)
from student s;
```

VIEWS

Students

id	name	b-id

Batches

b-id	ins-id

Instructors

insId	name

P1 => wants name of instructor for every st

P2 => " " " " " " batch

P3 => " " " every batch for every Student

O/P

st-id	st Name	b-id	bName	ins Id	instrName