1. Good Evening

2. Lecture begins at 9:06 pm

3. Topic - Subqueries, Views
Back log.

AGENDA

1. Subqueries ✓
2. Views
3. CONDITIONALS ✓

Backlog

1. Data Types
2. Builtin Fns
3. SERIALIZABLE & Demo
4. Deadlocks.
5. Views.
6. Composite Orderes
7. Full-text Search.

SUBQUERIES

Students (s1)

id	name	psp	bid
1	A	60	1
2	B	90	2
3	C	85	3
4	D	75	2
5	E	80	1

Students (s2)

id	name	psp	bid
1	A	60	1
2	B	90	2
3	C	85	3
4	D	75	2
5	E	80	1

Query: Find all students having a higher psp than student with id = 5.

```
SELECT s1.*
FROM students s1
JOIN students s2
    ON s2.id = 5 AND s1.psp > s2.psp
WHERE
```

s1.id	s1.n	s1.p	s.bid	s2.id	s2.n	s2.p	s2.bid
2	B	90	2	5	E	80	1
3	C	85	3	5	E	80	1

Subquery :

Step1 : Find psp of student with id=5

So, = [Select psp FROM students WHERE id=5]

Step2 : Find all student having psp > ↑

Select * FROM students WHERE psp > So

ANSWER

```
Select *
FROM students
WHERE psp > (
    SELECT psp FROM students WHERE id=5
)
```

psp (>, >=, <, <=, !=, =) (Inner Query)

It must return a single row & single column

QUERY 2

Products

id	name
1	P1
2	P2
3	P3
4	P4
5	P5

Orders

id	cid	pid
1	1	1
2	1	5
3	2	3
4	2	1
5	3	1

Customers

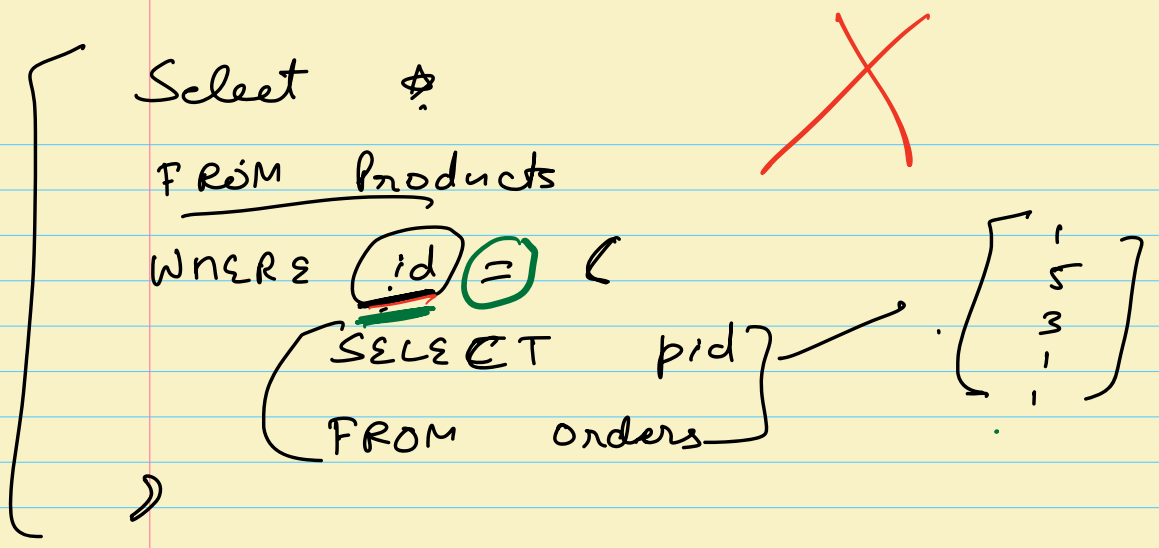
id	name
1	C1
2	C2
3	C3

Query : FIND Product which are ordered

→ Get pid of products from orders tbl.

```
SELECT pid
FROM Orders
```

1
5
3
1
1



SUMMARY : id can't be compared with a list of values. [=, !=, <, >, ≥, ≤]

[10, 20, 30, 40, 50] → 30

list == 30. ✗

list.contains(30) ✓

```
Select *
FROM Products
WHERE id IN (
    SELECT pid
    from orders
)
```

DISTINCT

1
5
3
1
1

Query 3.

90 > ALL [60, 70, 80]

Students

id	name	bid	psp	
1	A	1	90	✓
2	B	1	85	✓
3	C	1	50	✗
4	D	2	87	✓
5	E	2	62	✓
6	F	2	55	✗
7	G	3	60	
8	H	3	70	
9	I	3	80	

Question: Find students (from other batches) with psp greater than all students of batch 3.

MAX is not allowed

```
SELECT *
FROM students
WHERE psp IN (
        [SELECT psp FROM students WHERE b:d = 3]  → [60, 70, 80]
)
```

```
psp == 60
  OR
psp == 70
  OR
psp == 80
```

psp IN [60, 70, 80]

WHERE psp > [60, 70, 80] ✗

WHERE psp > ALL [60, 70, 80] ✓

```
SELECT *
FROM Students
WHERE psp > ALL (
        SELECT psp
        FROM students
        WHERE bid = 3
)
```

SO gives 1 row & col	SO give multiple rows & 1 col.
= ✓	IN
!= ✓	NOT IN ✓
>, ≥, <, ≤	(> ALL, ≥ ALL, < ALL, ≤ ALL)

```
SELECT *
FROM students
WHERE psp >
(
    SELECT MAX(psp)
    FROM students
    WHERE bid = 3
)
```

→ x · [60, 70, 80]

→ ✓ MAX [60, 70, 80] = 80

Query 4: [Find students with psp > any student of batch 3.]

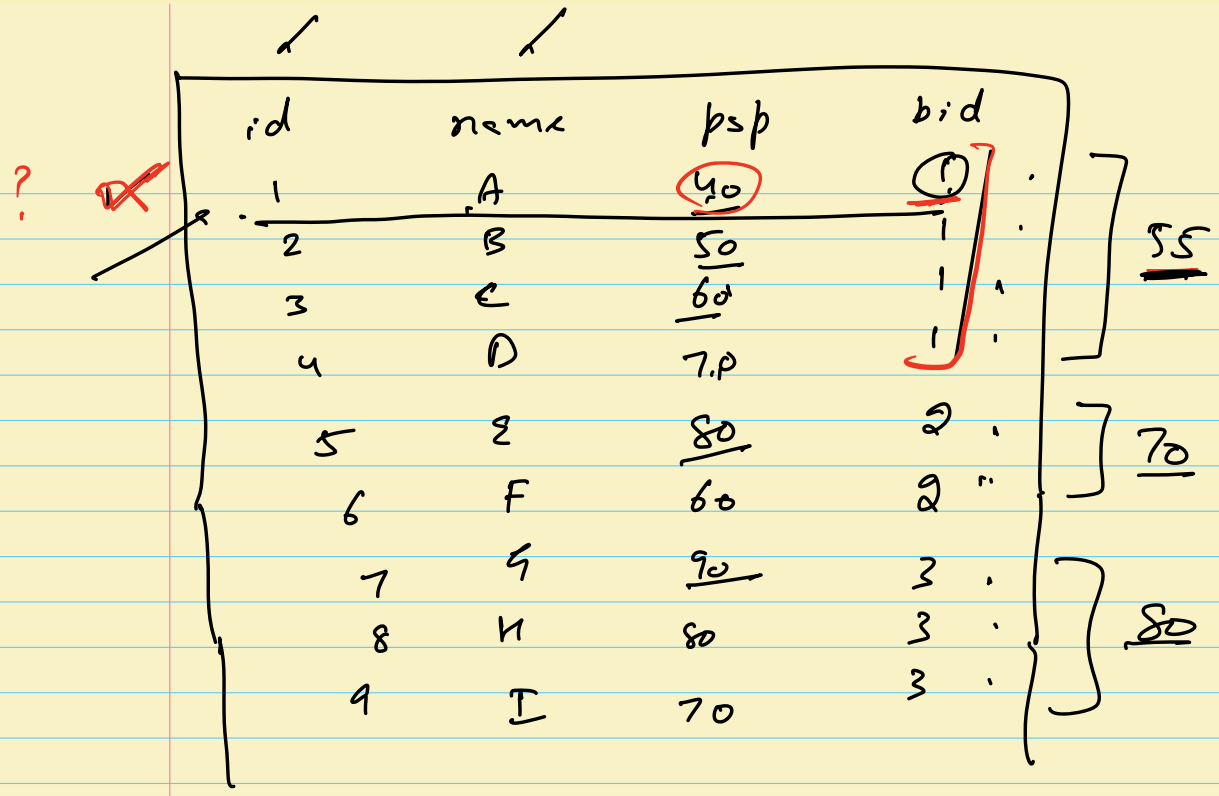
```
SELECT *
FROM students
WHERE psp > ANY (
        SELECT psp FROM students
        WHERE bid = 3
)
```

```
SELECT *
FROM Students
WHERE psp > (
        SELECT MIN(psp)
        FROM Students WHERE bid = 3
)
```

BREAK = 10:28 to 10:38

Query 5 : Co-related Subquery

Find all students who have psp > average psp of their own batch.




```
SELECT *
FROM Students s1
WHERE s1.psp >
(
    [SELECT AVG(psp)
     FROM Students s2
     WHERE s2.bid = s1.bid]
)
```

1. Row is considered for outer query
2. Row is fed to Innerg query to get id's result.
3. Use the result of 20 to select/reject the row.

JOINS	SQ
theoretically $n \times m$	$n \times m$
Practically → Faster if data is large.	→ Slower
→ Slower if data is less	→ Faster if data is less.

Query 6: Co-related Subqueries in SELECT clause.

9 rows & 5 columns.

id	name	psp	bid	Avg psp of batch.

```
SELECT *, (
        SELECT AVG(psp) FROM Students S2
        WHERE S2.bid = S1.bid
) AS AvgofBatch
FROM Students S1;
```

id	name	psp	brid	AvgofBatch
1	A	40	1,	55
9 times..				

```
SELECT bid, AVG(psp)
FROM Students
GROUP BY bid;
```

bid	AVG
1	55
2	70
3	80

[55, 70, 80]

Query 7: Subquery in a FROM clause.

```
SELECT s1.*, s2.AVG OF Batch
FROM Students s1
JOIN
(
    SELECT bid, AVG(psp) AS AvgOfBatch
    FROM Students
    GROUP BY bid
) s2
ON s1.bid = s2.bid;
```

S1

id	name	psp	Batch
1	A	40	1
2	B	50	1
3	C	60	1
4	D	70	1
5	E	80	2
6	F	60	2
7	G	90	3
8	H	80	3
9	I	70	3

S2

bid	AvgofBatch
1	55
2	70
3	80

1	A	40	1	55
2	B	50	1	55
3	C	60	1	55
4	D	70	1	55

IFNULL

COALESCE

IF

CASE

students

id	name	bid
1	A	NULL
2	B	1
3	C	2
4	D	NULL

```
SELECT id, name, IFNULL(bid, -1) AS Bid
FROM students.
```

id	name	bid
1	A	-1
2	B	1
3	C	2
4	D	-1

COALESCE(v1, v2, v3, v4)

Students

id	name	bid	psp.	contest.	hw.
1	A	1	80	60	70
2	B	1	NULL	40	70
3	C	2	NULL	NULL	30
4	D	3	NULL	NULL	NULL

SELECT id,

COALESCE(psp, contest, hw, -1)

FROM students

1 → 80
2 → 40
3 → 30

4 → -1

ORMs

Backlog

1. Views
2. IF, CASE
3. Data Types ✓
4. Builtin Fns.

→ [Stored Procedures]
→ [UDFs]
→ UNION, UNION ALL
→ PARTITION BY & Windowing Fns / ROW_NUMBER, RANK
→ Order of Queries.
→ CTE & Recursive CTE

ROBERT Vieira
- Beginners MS SQL
- MS SQL for Advanced