Class Starts at 9:05 PM

Joins Continued.

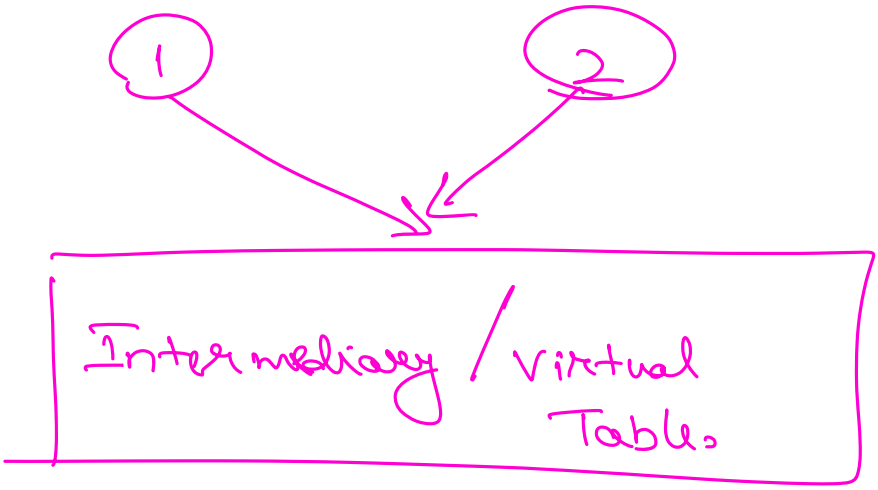
Agenda:

- → Joining Multiple tables
- → Compound Joins
- → Types of Joins
  - → Inner Join
  - → Outer Join
    - → left Outer Join
    - → Right Outer Join
    - → Full Outer Join

- → Cross Joins
- → USING
- → Natural Join
- → Implicit Join

- → Joins with ON vs Joins with WHERE.

Joining Multiple Tables.



Students

id	name	instructor_id	batch_id
1	Jack	1	1
2	Jim	7	3
3	Shreyas	4	2

Instructors

id	name
1	Deepak
4	Naman
7	Mohit

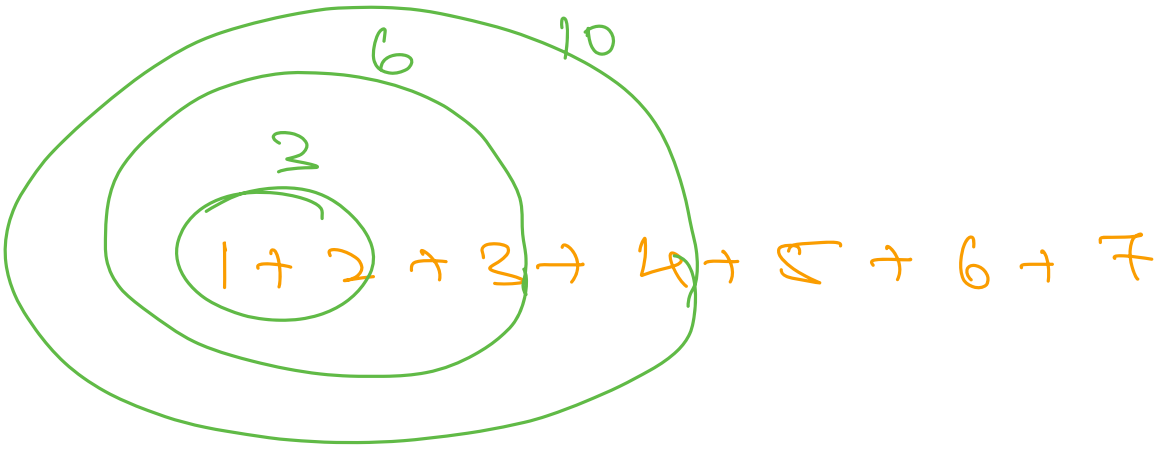
batches

id	name
1	Nov 22
2	May 22
3	Apr 22

Q) for every Student, give their name, alongwith their instructor name, and their batch name.

4)

Shreyas	Naman	May 22



You can join multiple tables very similar to how you add multiple numbers

↳ Pair by Pair.

```
Select ____________
FROM  Students  s
Join  Instructors  i
ON  s.instructor_id = i.id
```

1st pair

Students				Instructors	
id	name	instructor_id	batch_id	id	name
1	Jack	1	1	1	Deepak
2	Jim	7	3	7	Mohit
3	Shreyas	4	2	4	Naman

2nd pair

Students				Instructors		Batches	
id	name	instructor_id	batch_id	id	name	id	name
1	Jack	1	1	1	Deepak	1	Nov22
2	Jim	7	3	7	Mohit	3	Apr22
3	Shreyas	4	2	4	Naman	2	May22

always fill at the last

Query 1

```
select s.name, i.name, b.name
FROM Students s
Join Instructors i
ON s.instructor_id = i.id
JOIN batches b
ON s.batch_id = b.id
JOIN Classes c
```

(1) (t1) Comb of 3 tables n

ON s.class_id = c.id

Query 2 (2)

```
SELECT s.name, i.name, b.name
FROM   students s
JOIN   instructor i
JOIN   batches b
ON     s.instructor_id = i.id
AND    s.batch_id = b.id
```

→ N * M rows

In the (1) Query, if students has N row, instructor has M rows. (N>M).
How many rows in the Intermediate table?
N rows

In the (2) SQL, the first pair result will have N * M rows

1st pair

Students				Instructors	
id	name	instructor_id	batch_id	id	name
1	Jack	1	1	1	Deepak
1	Jack	1	1	7	Mohit
1	Jack	1	1	4	Naman
2	Jim	7	3	1	Deepak
2	Jim	7	3	7	Mohit
2	Jim	7	3	4	Naman
⋮					

As good as

```
Select ——————
FROM   Students
JOIN   instructor
ON     true;
```

So (2) is bad in terms of efficiency

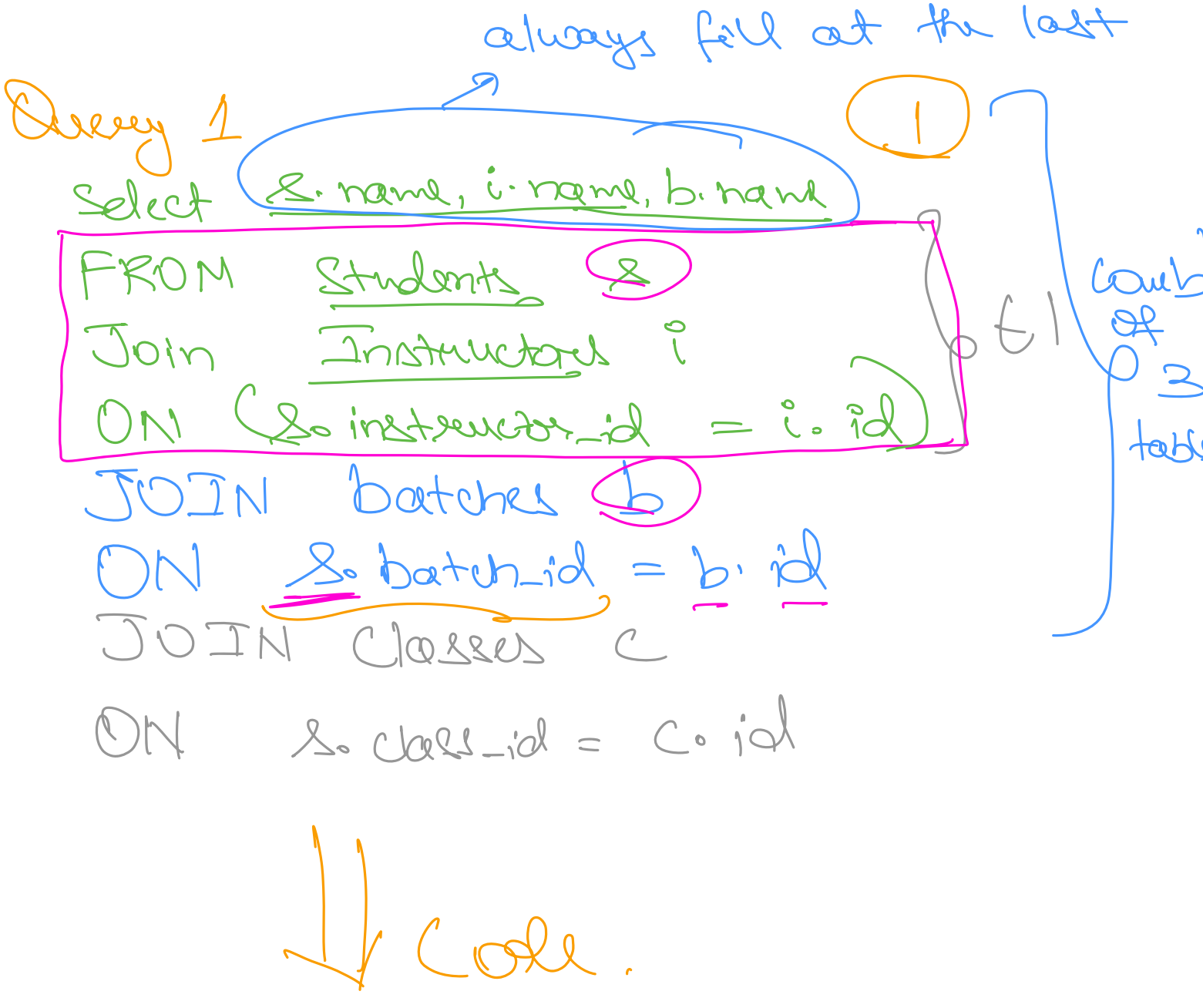

always fill at the last
Query 1
1
select s.name, i.name, b.name
FROM Students s
Join Instructors i
ON s.instructor_id = i.id
JOIN batches b
ON s.batch_id = b.id
JOIN Classes c
ON s.class_id = c.id
Code.

table1 : Students

table2 : instructor

table3 : batches.

```
ans1 = [ ]

for row1 in table1:
    for row2 in table2:
        if cond^n is true:  // ON clause
            ans1.add( row1[all col^m], row2[all col^m] )

ans2 = [ ]
for row1 in ans1:
    for row3 in table3:
        ans2.add( row1 (all col^m of ans1),
                  row3( all col^m of table3) )

for row in ans2:
    print( row [Select col^m] )
```

Compound Joins.

Students

id	name	birthyear	psp
1	Naman	1996	80
2	Deepak	1998	85
3	Mohit	1997	70
4	Dilip	2001	90

Q) For every Students, give me those students who are within 2 years old/young than this Student AND their psp is more than this Student.

S1	S2
Mohit	Naman
Mohit	Deepak
Naman	Deepak

Mohit → 97 } ✓ 70 } ✓
Naman → 96 } 80 }

Mohit → 997 }
Dilip → 2001 }

Naman → 96 } ✓ 80 } ✓
Deepak → 98 } 85 }

Naman → 96 } ✓ 80 } ✗
Mohit → 97 } 20 }

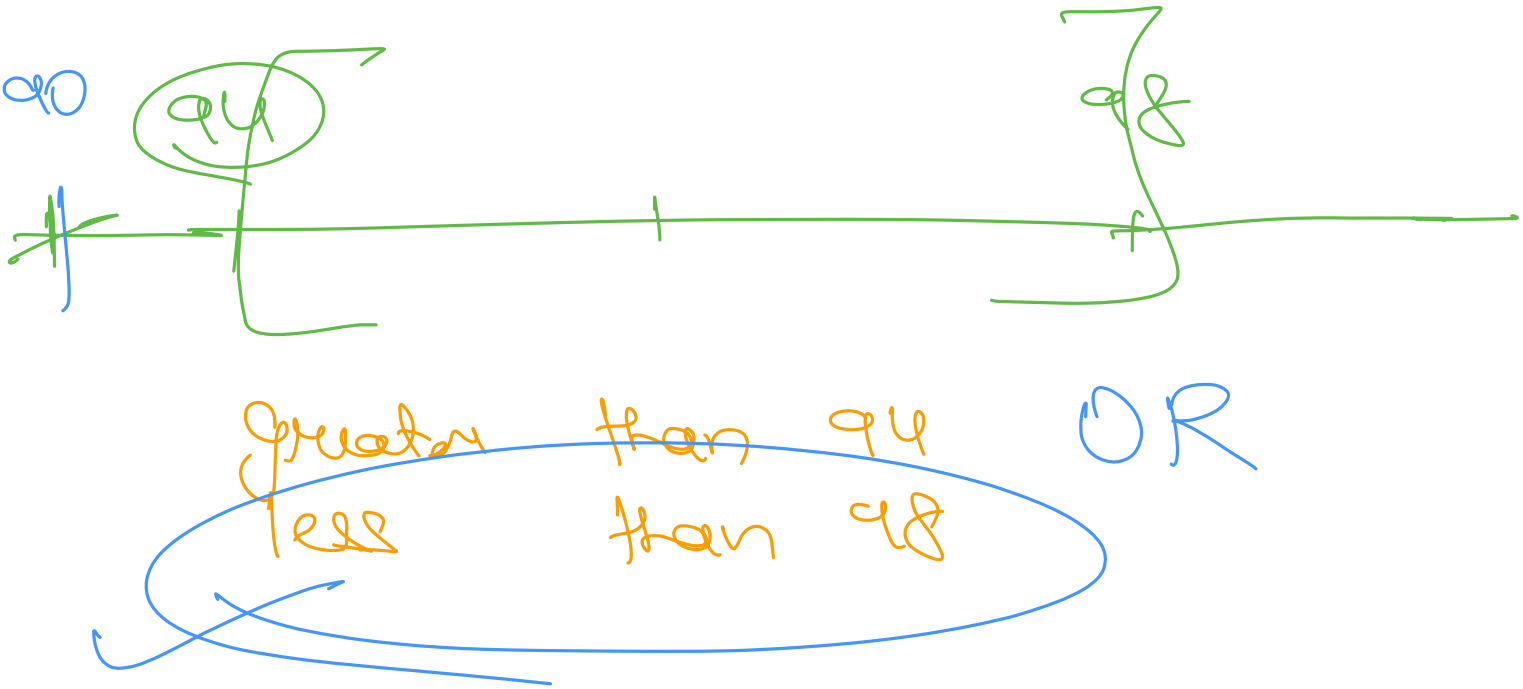
→ Use a Self join

```
Select ____________
FROM   Students  S1
Join   Students  S2
ON     S2.birth_year >= S1.birthyear - 2
AND    S2.birth_year <= S1.birth_year + 2
AND    S2.psp > S1.psp
AND    S2.id != S1.id
```

we fix 1 student from S1, compare all the other students of S2.

→ Compound Joins → you can have multiple conditions in ON Clause.

→ Not necessarily the Join should be on equality.



Greater than 94 OR
less than 98

Break till 10:15 PM

Types of Joins

Students.

id	name	batch_id
1	Naman	1
2	Deepak	1
3	Mohit	Null
4	Dilip	Null
5	Jack	2

batches

id	name
1	A
2	B
3	C

Q print the name of every student along with their batch.

```
Select  s.name, b.name.
FROM    Students  s
JOIN    batches   b
ON      s.batch_id = b.id
```

Naman	A
Deepak	A
Jack	B

```
Select  s.name, b.name.
FROM    Students  s
JOIN    batches   b
ON      s.batch_id = b.id
OR      s.batch_id IS NULL ;
```

5 rows

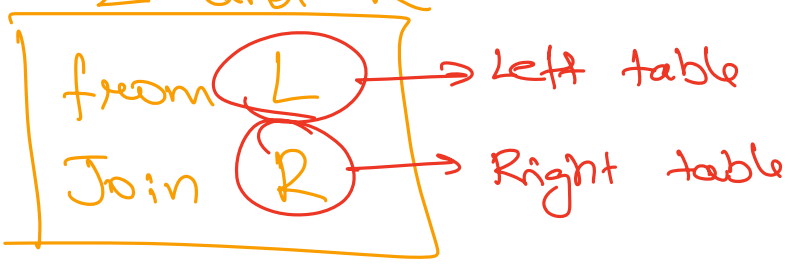
Actual answer is 9

Mohit

```
for row1 in Students :
    for row2 in batches :
        if ( row1.batch == row2.id
           OR  row1.batch_id is Null )

            ans.add (row1, row2)
```

when we do a join between two tables L and R

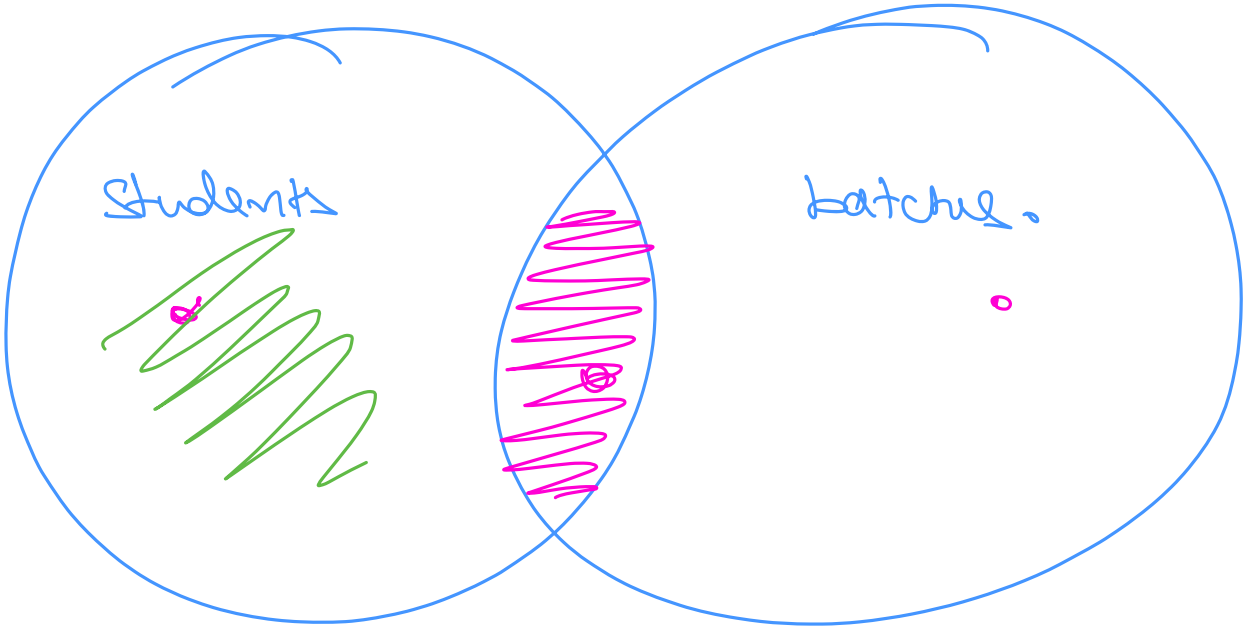


if row of Left table doesn't matches with the row of Right table, it will not be in my answer.

This is an Inner Join

```
Select s.name, b.name.
FROM   students s
JOIN   batches  b
ON     s.batch_id = b.id
```

Inner Join is same as Join



what if I want to get all the entries of the Student table.

→ Outer Join

→ allows rows that didn't match the condition.

Types:

→ Left Join
Left Outer Join } both are same

Right Join
Right Outer Join } same

Full Join
Full Outer Join } same.

Left Join :

→ Include all the rows that match the condition

+ include all the rows of (Left) table that didn't match the condition

Students.

id	name	batch_id
1	Naman	1
2	Deepak	1
3	Mohit	Null
4	Dilip	Null
5	Jack	2

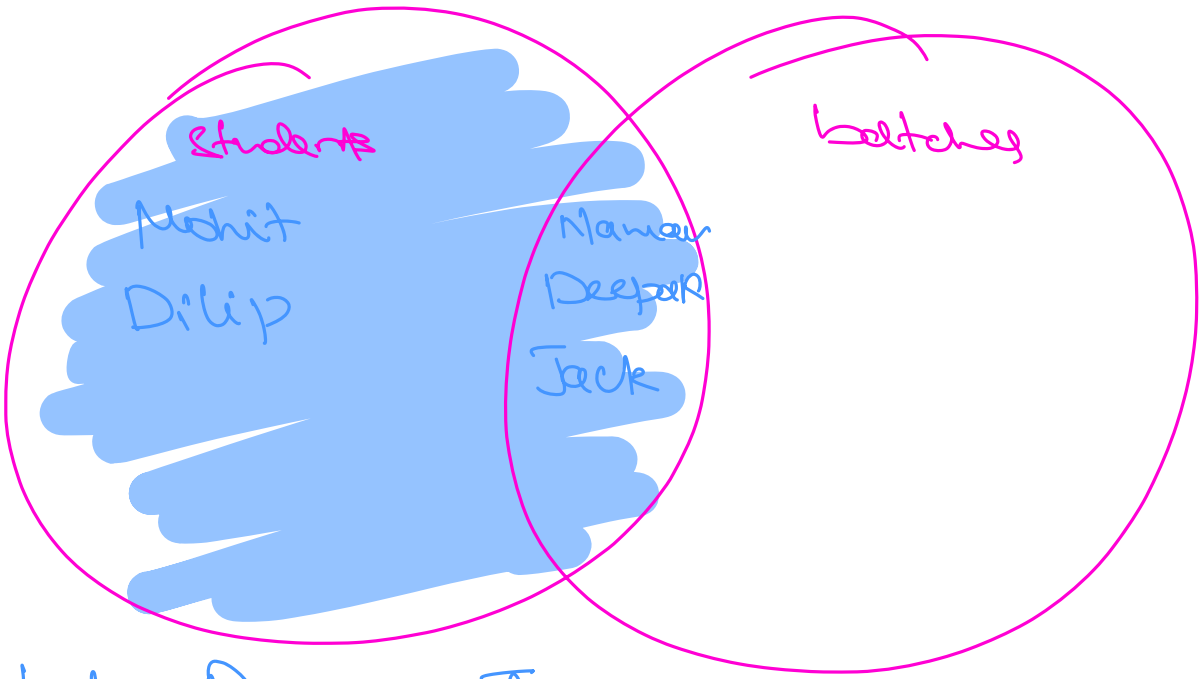
batches

id	name
1	A
2	B
3	C

Q print the name of every student along with their batch.

A)

```
select s.name, b.name
FROM Students s
LEFT JOIN batches b
On s.batch_id = b.id
```



Left Outer Join.

Code :

```
                                        INTersection
for row1 in students :                      ↓
    for row2 in batches :

        if row1 and row2 match cond^n :
            ans.add (row1, row2)

ans2 = [ ]
for row in Students :                   Takes care of
                                          NULL
    if row not in ans :                   ↓
        ans2.add( row, [null, null, null, null] )
                                  for the other table.
```

Right Join

Students.

id	name	batch_id
1	Naman	1
2	Deepak	1
3	Mohit	Null
4	Dilip	Null
5	Jack	2

batches

id	name
1	A
2	B
3	C

St. name	b. name
Naman	A
Deepak	A
Jack	B
Null	C

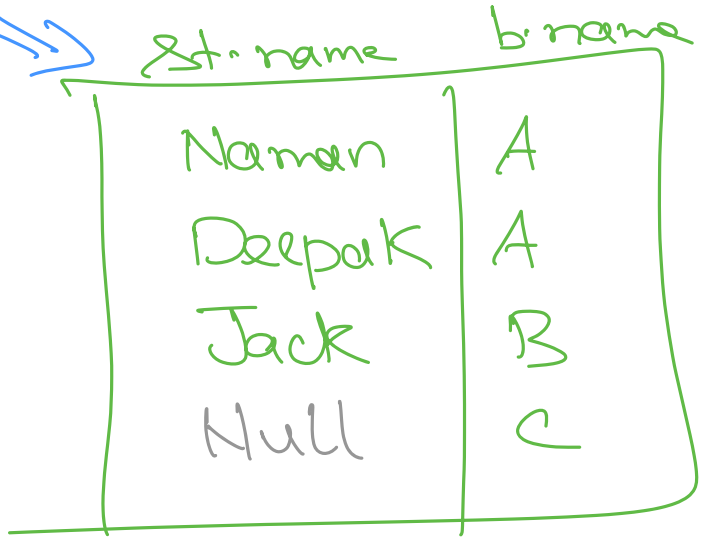
→ include all the rows that match the cond^{n}

+ include all the rows of Right table that didn't match the cond^{n}.

flip the tables:

```
Select ————————
FROM batches
LEFT JOIN Students
ON ————————————
```

Same Output



St. name	b. name
Naman	A
Deepak	A
Jack	B
Null	C

Full Outer Join

→ all rows that match the cond^n

→ include all rows of Left table that don't match the cond^n

→ include all rows of Right table that don't match the cond^n.

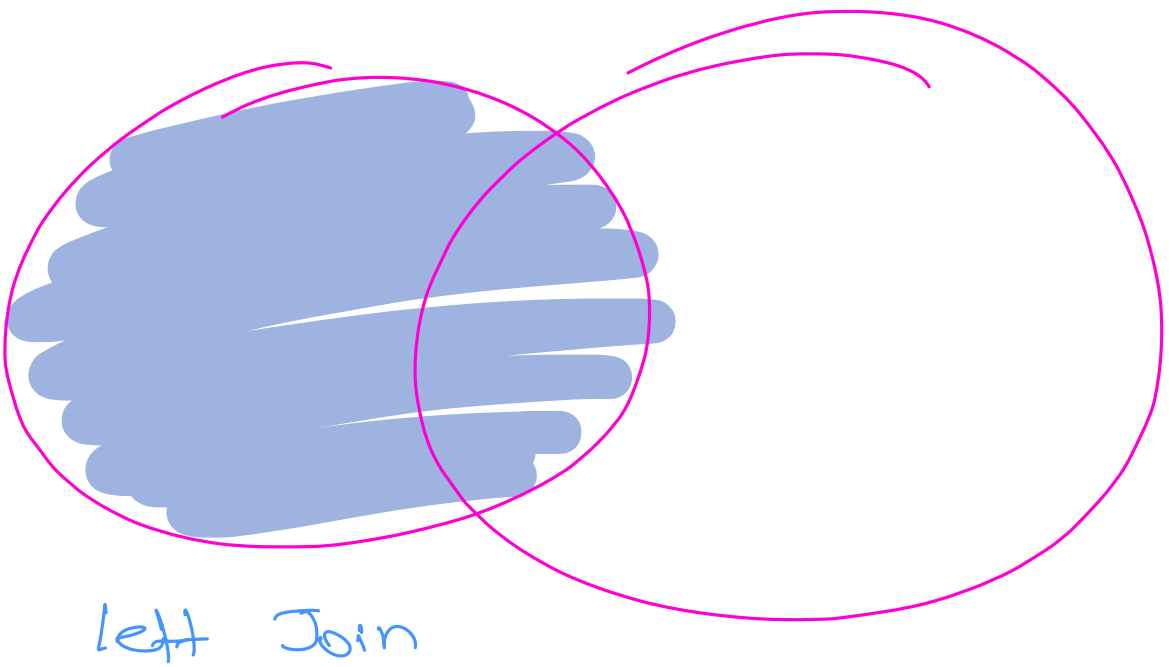
Students.

id	name	batch_id
1	Naman	1
2	Deepak	1
3	Mohit	Null
4	Dilip	Null
5	Jack	2

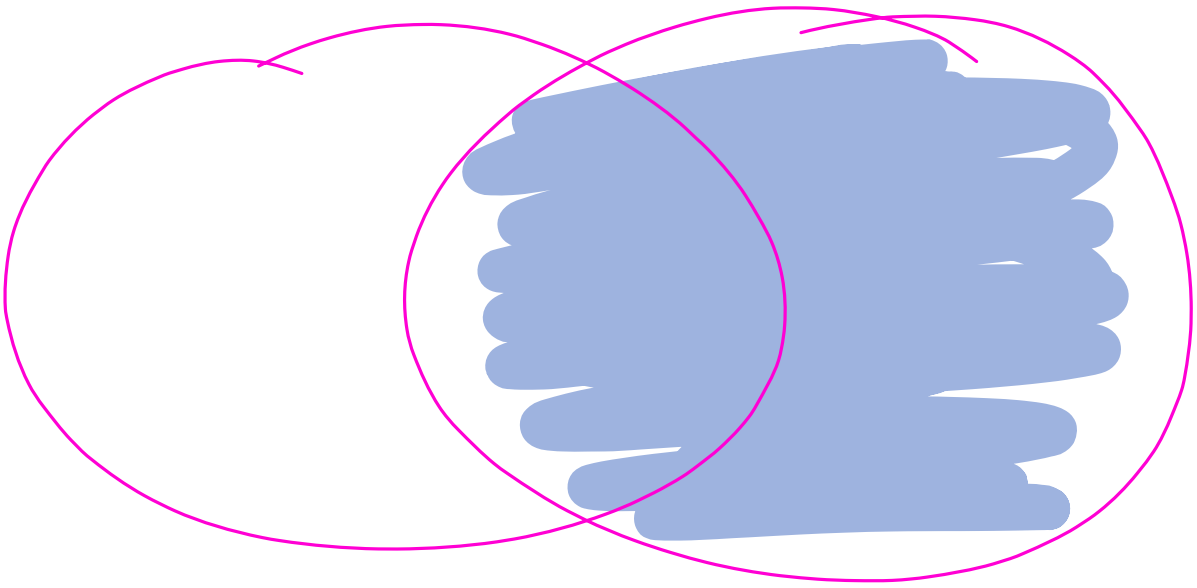
batches

id	name
1	A
2	B
3	C

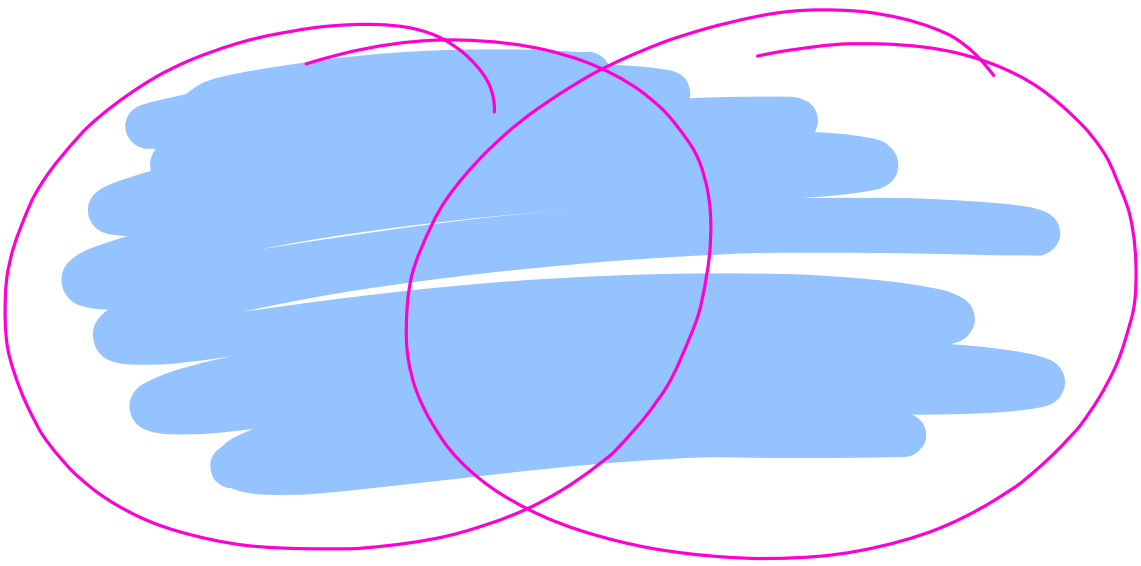
Naman	A
Deepak	A
Jack	B
Mohit	Null
Dilip	Null
Null	C



left Join



Right Join



Full Join

CROSS JOIN.

Shows (N)

id	name	date
1	Show1	
2	Show2	

Seat Types (M)

id	type
1	VIP
2	GOLD
3	Platinum

Q Print every show with every type of seats.

Output

Show1	VIP
Show1	GOLD
Show1	Platinum
Show 2	VIP
Show 2	GOLD
Show 2	Platinum

```
Select  ——————→
FROM    Shows
JOIN    Seat Types
```

ON true.

Select ——————

FROM Shows

CROSS JOIN SeatTypes;

No cond^n

we want everything on L with everything on R.

No. of rows in the Output = N * M

Syntactic Sugars : write the query in a simpler format.

USING

→ Joining based on equality.

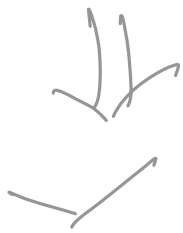
→ Name of columns on both side are exactly same!


```
Select ____
FROM   students  s
JOIN   batches   b
 ON    s.batch_id = b.batch_id
```

=

```
Select ____
FROM   students  s
JOIN   batches   b
USING ( batch_id )
```

```
Select ____
FROM   students  s
JOIN   batches   b
 ON    s.batch_id = b.batch_id
 AND   s.xyz = b.xyz
 AND   s.abc = b.abc
```



```
Select ____
FROM   students  s
JOIN   batches   b
 USING ( batch_id, xyz, abc)
```

efficiency is same for both!

→ works only with AND

Natural Join

→ When you want to Join 2 tables based on equality of all col^{n} on both the tables.

A

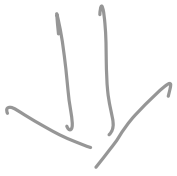
c	d	e	x	y	z

B

d	f	g	y	z

Col^{m} name that match both tables

```
Select ____
FROM   Students  S
JOIN   batches   b
 ON    S.batch_id = b.batch_id
 AND   S.xyz = b.xyz
 AND   S.abc = b.abc
```



Select ——————

FROM Students s

Natural Join batches b;

Implicit Joins.

```
Select *
FROM A, B
```

you are joining 2 tables without JOIN Keyword.

```
Select *
FROM Students, batches
WHERE Students.batch_id = batches.id
```

```
Select *
FROM   Students s, batches b
WHERE s.batch_id = b.id
  AND   s.batch_name = b.batch_name.
```

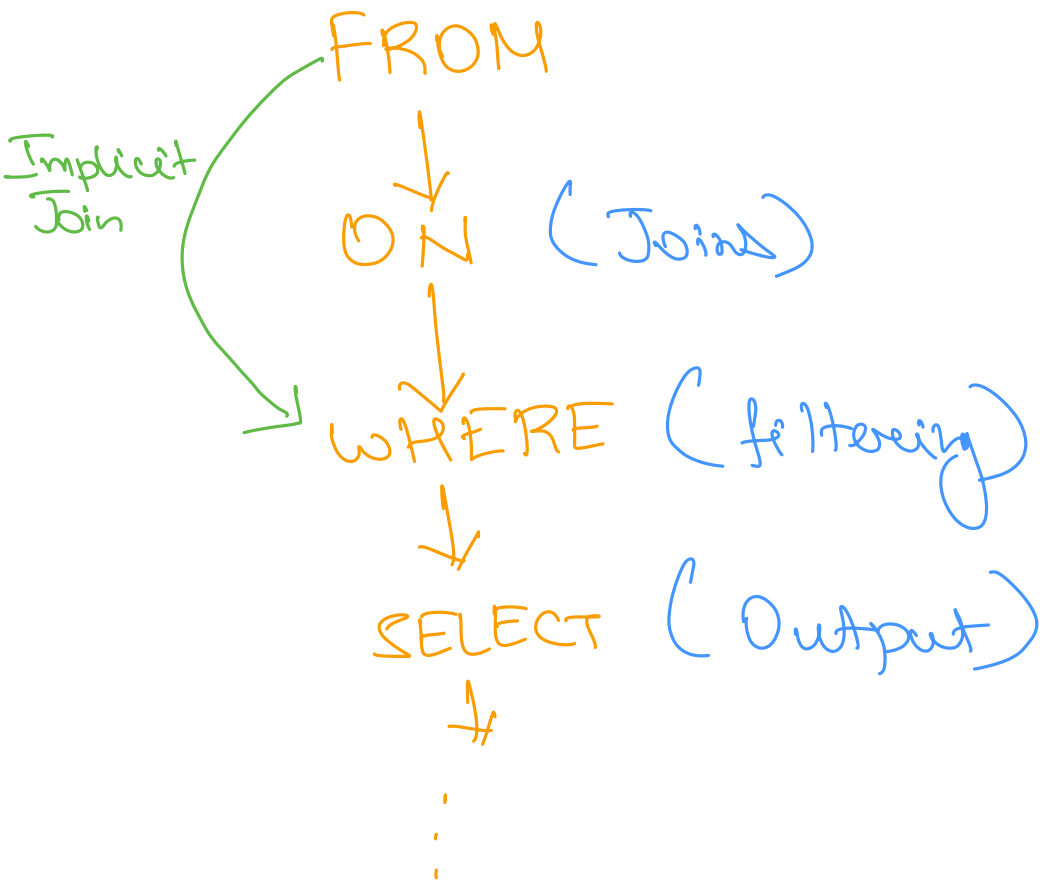
```
Select *
FROM   Students s
Join   batches b
ON   s.batch_id =b.id
WHERE s.batch_name = b.batch_name
```

Both the Queries above will work.

```
Select *
FROM Students, batches
WHERE Students.batch_id = batches.id
```

flow :

1) Cross join between Students, batches
   → Creating N * M rows
2) Then filters out using WHERE Clause



→ Give as many cond^n in ON clause, if not possible, use WHERE

ON is applied at the time of creating intermediary table itself

→ lesser memory

→ better performance.

WHERE gets applied after the Joins

→ extra memory

→ slower performance.