JOINS - II

Agenda.

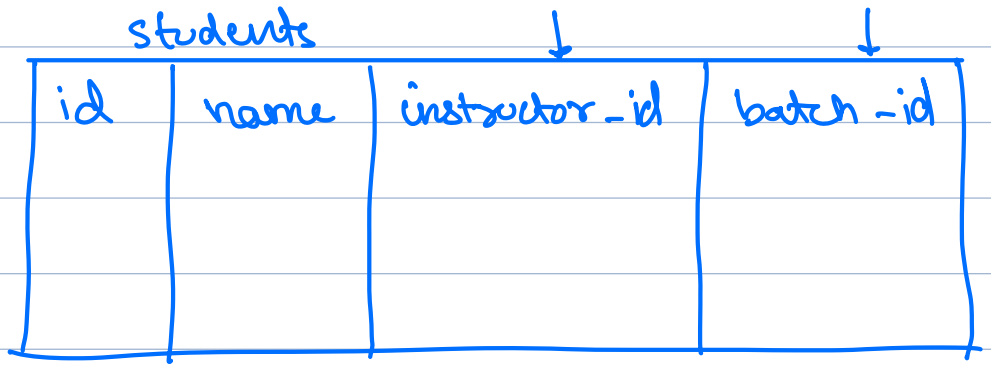
- Joining multiple tables
- Compound Joins
- Types of joins
  - Inner join
  - Outer join
    - left
    - right
    - full
  - Cross join
  - Using
  - natural joins
  - implicit joins
  - Join with where vs join with ON

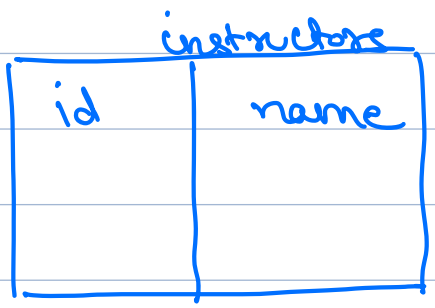
JOINING MULTIPLE TABLES

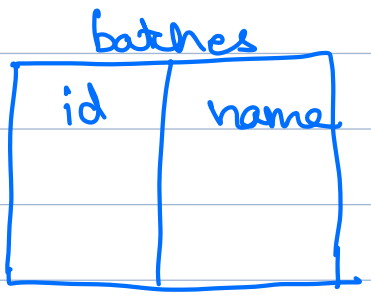
⇒ Till now we have only combined data of two tables.

⇓

Behind the scenes an intermediate table is created (one)



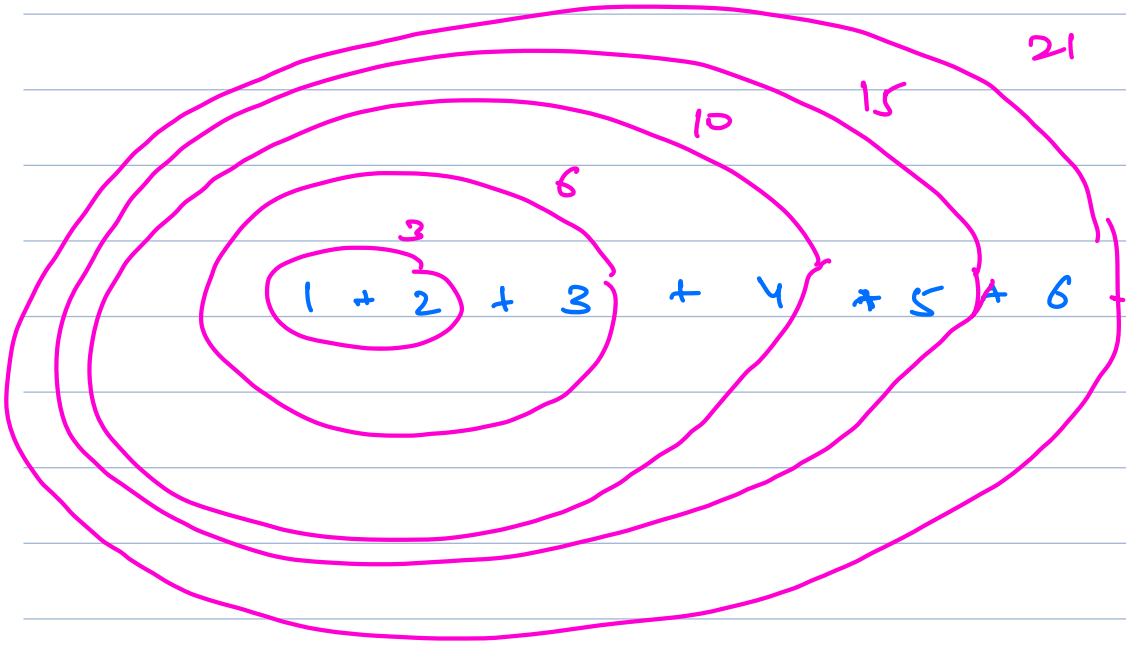




Q for every student, give their name along with name of their instructor who is teaching them and name of their batch.

Shreyas	Avg 22	Vijjwal

→ combining only two tables won't work here.



Joining in pairs

* You can also join multiple tables, very very similar to how you add multiple numbers
  ⇒ Join pair by pair

(cant join instructor + batches as no common/relating column)

```
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
```

this much will create intermediate table will all colms of students & instructor table

students				instructors	
id	name	instructor_id	batch_id	id	name

only this table inst enough for final ans.

|| join this table to batches table

batches

id	name

students				instructor		batches	
id	name	inst_id	b_id	id	name	id	name

Code

table1 :

table2:

table3 :

ans1 :

ans :

```
for row1 in table1 :
    for row2 in table2 :
        if cond" is true:      || on clause
            ans1. add ( row1 + row2)
```

↓

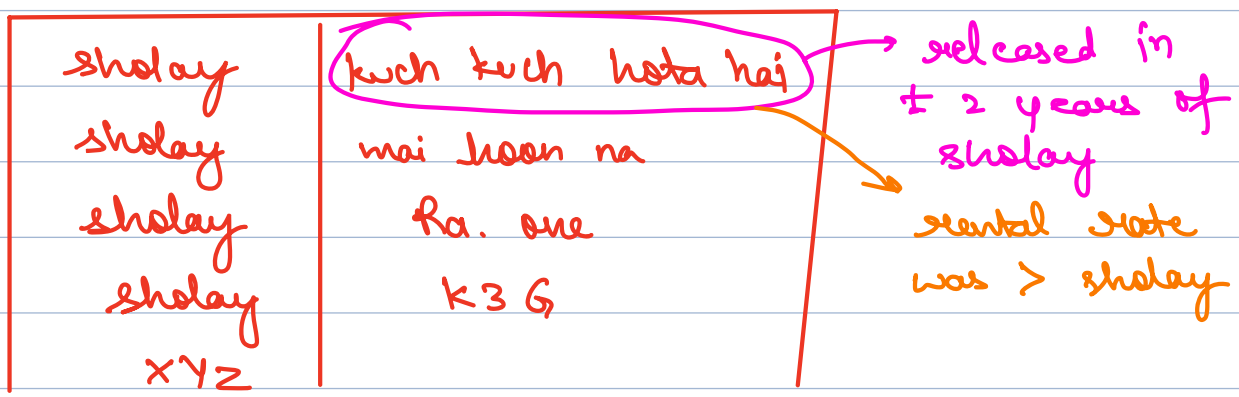
```
for row1 in ans1:                    → has all columns of table1 & table2
    for row2 in table3:
        if cond^n is true:
            ans.add ( row1 + row2)
```

```
for each row in ans:
    print (row [s.name], row [i.name],
           row [b.name])
```

COMPOUND JOINS

→ whenever we join, we also specify the condition on which we join.

film ⇒ for every film, name all the films that were released within ± 2 years of that film and their rental rate was > rate of the movie.



we are stitching with a table

↳ self join

name	rel. year	rental
Sholay	2000	2
KKHH	1999	3
MHN	1998	2
KKKG	1997	4

sholay	KKHH
KKHH	KKKG
MHN	KKHH
MHN	KKKG

```
select f1.title , f2.title
 from film f1
  join film f2
  on (f2.rel-year >= f1.rel-year) and
     f2.rel-year <= f1.rel-year) and
        f2.rental > f1.rental
```

or

on f2.rel-year between f1.year - 2 and f1.year + 2

Learnings

① Joins always need not happen on equality of columns

② Joins can also have multiple conditions

⇓

compound join

↓

join that has multiple conditions on multiple columns

↘ on different columns

Types of JOINS

Students

id	name	buddy_id
1	A	2
2	B	null
3	C	2
4	D	1

Q for every student print their name and the name of their buddy.

```
Select
from student s
join students b
on s.buddy_id = b.id
```

self join can be thought of joins between two tables of the same instance

s.buddy_id = b.id

Students (s)

id	name	buddy_id
1	A	2
2	B	null
3	C	2
4	D	1

Students (b)

id	name	buddy_id
1	A	2
2	B	null
3	C	2
4	D	1

1	A	2	B
3	C	2	B
4	D	1	A

A	B
C	B
D	A

B is not present

when we do a join b/w tables

from L

join R

if a row of left table doesn't match with any row of right table, it will not be in my answers. and vice versa

students

id	name	batch_id
1	John	1
2	Jane	2
3	Jim	null
4	Jenny	null
5	Jack	2

batches

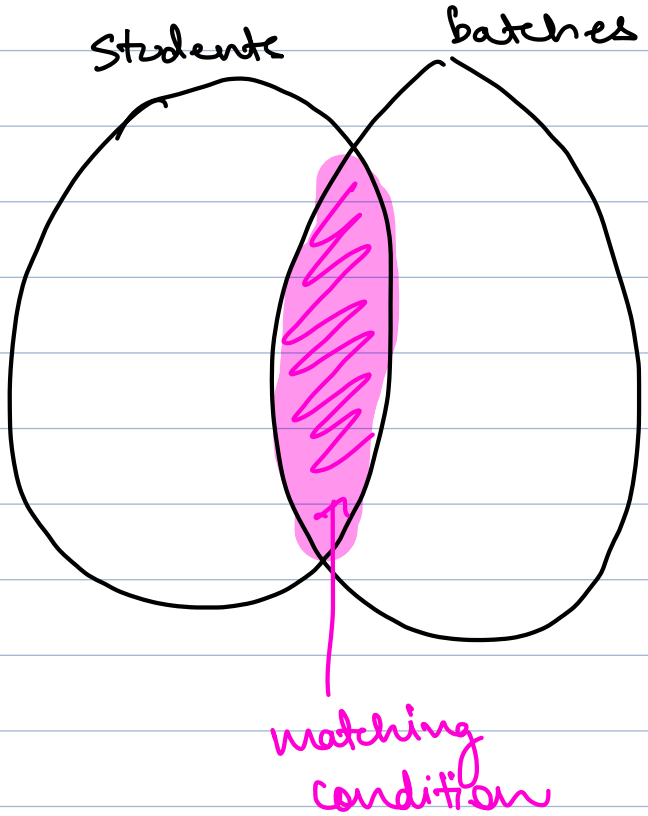
id	name
1	A
2	B
3	C

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

```
select s.name, b.name
  from student s
inner join batches b
   on s.b_id = b.id
```

John	A
Jane	A
Jack	B

Inner join → Join that includes only rows of both tables that match condition



Q Get all students even if no batch

```
select  s.name , b.name
 from   student  s
 join   batches  b
   on   s. b_id  = b. id
     Or  s. b-id  IS NULL
```

John	A
Jane	A
Jack	B
Jim	null
Jim	null
Jim	null

X

wrong answer as that entry is multiple times

[Break]

Outer Joins

→ also always rows that dont match the condition in the final answer

Types

→ Left Join
or
Left Outer Join

→ Right Join
or
Right Outer Join

→ Full join
or
full outer join

Left join.

→ will include all the rows that match condition
+
include all rows of the left table that don't match condition even.
⇒ for these rows, we will fill null on other side

students

id	name	batch_id
1	John	1
2	Jane	2
3	Jim	null
4	Jenny	null
5	Jack	2

batches

id	name
1	A
2	B
3	C

```
select
from students s
left join batches b
on s.b_id = b.id
```

John	A
Jane	A
Jack	B
Jim	null
Jenny	null

★ left join ensures every row of the left side table is atleast once in the answer.

Code.

```
for each row1 in table1:
    for each row2 in table2:
        if match cond":
            ans.add([row1] + [row2])
```

```
for each row in table1:
    if row not in ans:
        ans.add( [row] + [null, null...] )

for each row in ans:
    print( row( ) , . . . )
```

Right Join

→ includes all rows that match cond^{n}

+

all rows of right side that didn't match cond^{n}

students

id	name	batch_id
1	John	1
2	Jane	2
3	Jim	null
4	Jenny	null
5	Jack	2

batches

id	name
1	A
2	B
3	C

John	A
Jane	A
Jack	B
null	C

Full join

→ all rows that match condition

+

all rows of left side that didn't match

+

all rows of right side that didn't match

John	A
Jane	A
Jack	B
Jim	null
Jenny	null
null	C

[Quiz]

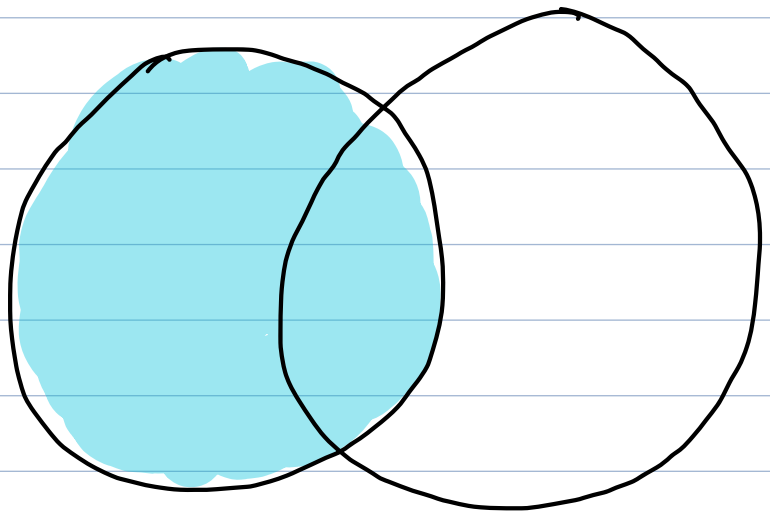
[code]

→ Add student & batch with above data

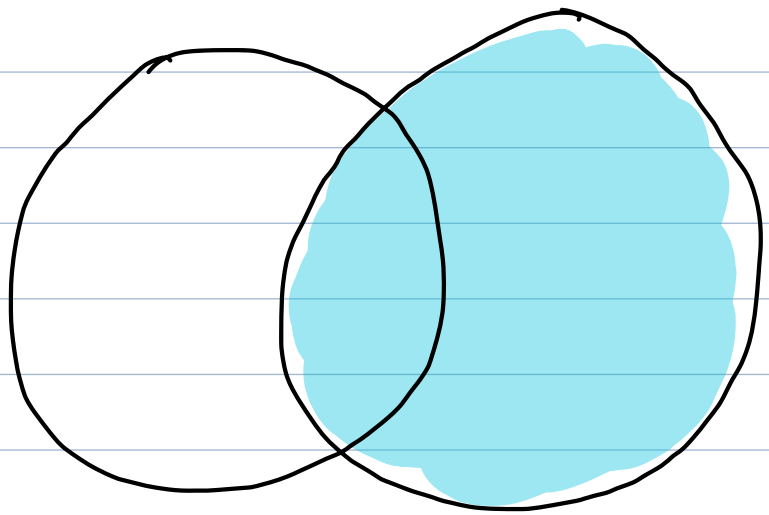
→ Show all types of join examples

→ My SQL & doesn't support full join.

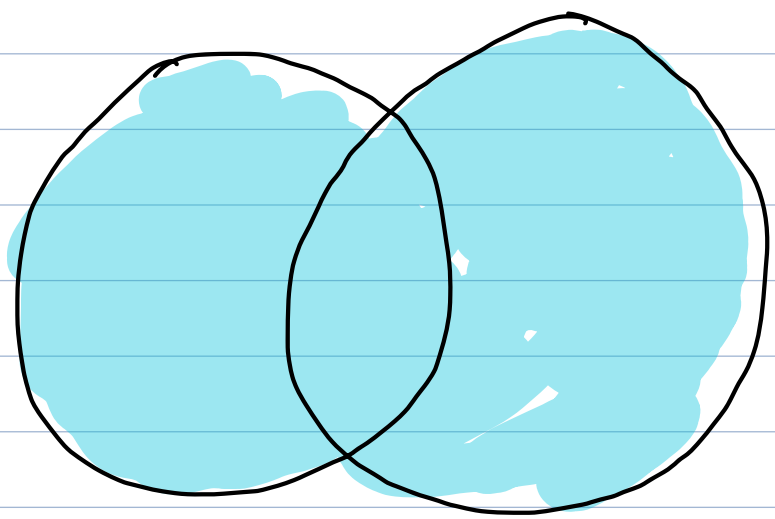
diff kinds of joins in terms of venn diagrams:.



left join

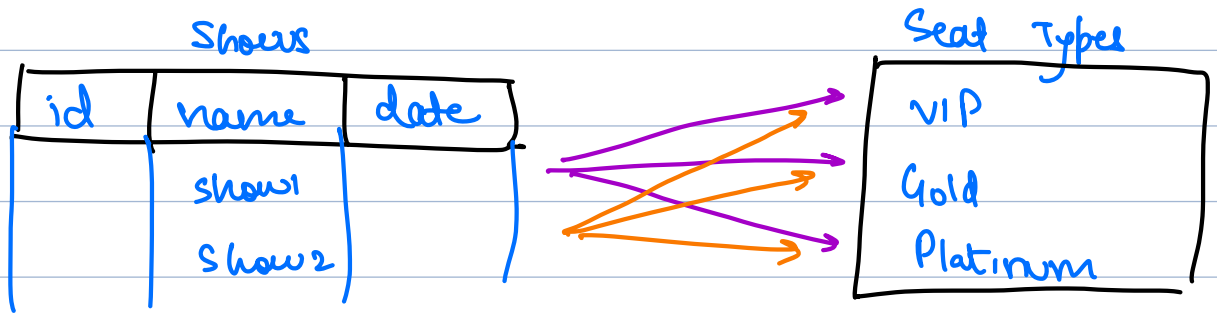


right join



full join

Cross join



Q Print every row with all of the seat types.

show1	Gold
show1	VIP
show1	Plat
show1	recliner
show2	Gold

```
select
from show
on seat_types
on true
```

⇒ $O(n * m)$

Cross join ⇒ returns pairs of all the rows

```
select
from seats
cross join shows
```

⇒ 9 shows
4 rows
36 ans rows

⇒ There is no join condition (obviously)

Q Is cross join same as full join?

NO

A + B
full join

A * B
cross join

dont get confused by pseudo code of full join, its time complexity is (n * m) but total rows are (n+m)

USING

Till
1. Joining Based on equality
2. Name of coln on both side was same

eg students.batch_id = batches.batch_id

[code] → join on film & film_actor

```
select
from film
 join film_actor
 using (film_id)
```

→ same as on film.film_id = film_actor.film_id

if you also had a same name column like xyz

using (film_id, xyz)

→ same as

on film.film_id = film_actor.film_id

and film.xyz = film_actor.xyz

Natural join

→ when you want to join two tables based on equality of all coln with same names.

A	B
- a	- d
- b	- b
- c	- e
- d	- f
- g	- g

```
select
from A
  join B
    on A.b = B.b
    and A.d = B.d
    and A.g = B.g
```

both are same

```
select *
from A
natural join B
```

using (b, d, g)

//assuming b, d, g are the only columns with same name

Implicit Joins → you are joining without using a join keyword!

```
select *
from A, B
```

⇒ behind the scenes stitches every row of A to every row of B

↓

similar to cross join

```
select *
from students, batches
where student.id = batches.id
```

→ Implicit join is a cross join.

Normal join

```
select *
from A
join B
on A.a = B.a;
```

A = [ ]

B = [ ]

ans = [ ] ← size: n * m - filtered data

```
for every row1 in A:                          space = 100
    for every row2 in B:
        if row1 and row2 match cond^n:
            ans.add(row1 + row2)
```

1000

```
for every row in ans:
    print(ans[a] + ans[b]
```

100

```
select *
from A,B
where A.a = B.a;
```

⟹ cross join

this is executed after the table has been formed

like `select * from students where id=2;`

A = [ ]

B = [ ]

ans = [ ]

1000 [

```
for every row1 in A:
    for every row2 in B:
        ans.add(row1 + row2)
```

space = 1000

1000 [

```
for every row in ans:
    if row matches cond^n    // where
        point(ans[x], ans[y])
```

Students

100

batches

10

join has 100 rows

cross join has 1000 rows

On gets applied at the time of creating intermediary table itself

=> lesser memory

=> better performance

where gets applied at time of finally printing

=> extra memory

=> less performance

(1)

```
Select *
from A
 join B
  on A.a = B.a
```

(2)

```
Select *
from A, B
where A.a = B.a
```

(1) is faster than (2)