Today's Agenda :-

1) Compound Joins
2) Types of Joins
3) Cross Joins
4) USING
5) NATURAL
6) Implicit Join
7) Join with WHERE vs ON
8) UNION

Compound Joins :→

↳ Joins having multiple conditions

Film :-

for every film, name all the film that were released within ±2 years of that film & their rental rate was greater than rental rate of that movie

Film (f_1)

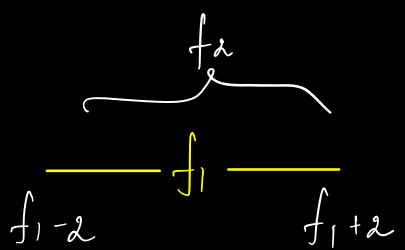
Name	rel_year	rental
Sholay	2000	2
KKHH	1999	3
MHN	1998	2
KKKG	1997	4

Film (f_2)

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.name, f2.name
from film f1
join film f2
on (f2.release_year between f1.release_year - 2
    AND f1.release_year + 2) &&
   (f2.rental_rate > f1.rental_rate)
```



&& ~ ||
AND ~ OR

`f2 >= f1 - 2 && f2 <= f1 + 2`

`f2 between f1 - 2 & f1 + 2`

1) Joins does not need to happen on equality of columns.

2) Joins can have multiple columns.

Types of Join :-

Student

id	Name	batch_id
1	John	1
2	Jane	1
3	Jim	NULL
4	Jenny	NULL
5	Jack	2

Batches

id	Name
1	A
2	B
3	C

```
Select s.name, b.name
from Students s
join Batches b
on s.batch_id = b.id
```

inner
1
optional

→ creates an intermediate table.

Student

id	Name	batch_id
1	John	1
2	Jane	1
3	Jim	NULL
4	Jenny	NULL
5	Jack	2

Batches

id	Name
1	A
2	B
3	C

Intermediate table

1	John	1	1	A
2	Jane	1	1	A
5	Jack	2	2	B

Output

John	A
Jane	A
Jack	B

✓

Left join { 3 Jim NULL NULL NULL
5 Jenny " " "

When we do a join b/w tables

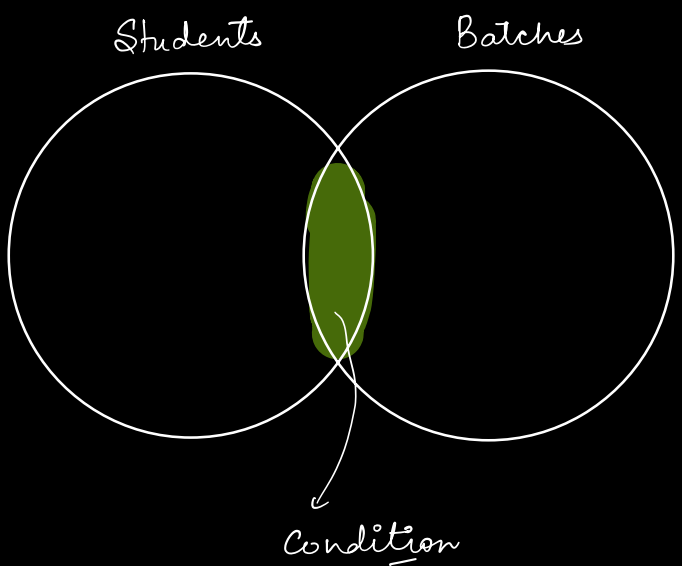
```
from L
join R
```

If a row of the left doesn't match with any row of right, it'll not be included in your answer & vice versa.

⇓

Inner Join

Inner Join :- Join that includes only rows of both tables that matches conditions



OUTER JOIN :-

Join that allows rows that don't match the condition as well.

LEFT JOIN
OR
LEFT OUTER JOIN

RIGHT JOIN
OR
RIGHT OUTER JOIN

FULL JOIN
OR
FULL OUTER JOIN

1) includes all rows that matches condition

2) includes all rows of left table that don't match the condition.

left join

3) for these rows, we will fill NULL on the other side

Student

id	Name	batch_id
1	John	1
2	Jane	1
3	Jim	NULL
4	Jenny	NULL
5	Jack	2

Batches

id	Name
1	A
2	B
3	C

```
Select s.name, b.name
from Students s
left join Batches b
on s.batch_id = b.id
```

John	A
Jane	A
Jack	B
Jim	NULL
Jenny	NULL

(1) table1 =

(2) table2 =

ans =

```
for each row1 in table1 :
    for each row2 in table2 :
```

```
        if (cond is T) {
            ans.add (row1 + row2)
        }
    }
}
```

```
for each row in table1 :
    if (row not in ans) {
        ans.add (row + [NULL, N...])
    }
}
```

```
for each row in ans
    print (row[ ], ... )
```

RIGHT JOIN :-

1) includes all rows that matches condition
2) includes all rows of right table that don't match the condition
3) for these rows, we will fill NULL on the other side

Student

id	Name	batch_id
1	John	1
2	Jane	1
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

Right Join

FULL JOIN :-

1) All rows that matches condition
2) All rows of left that didn't match
3) All rows of right that didn't match

3 Jim NULL

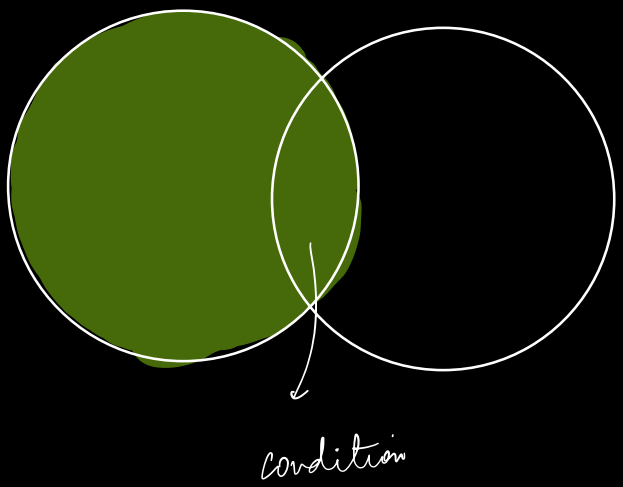
Student

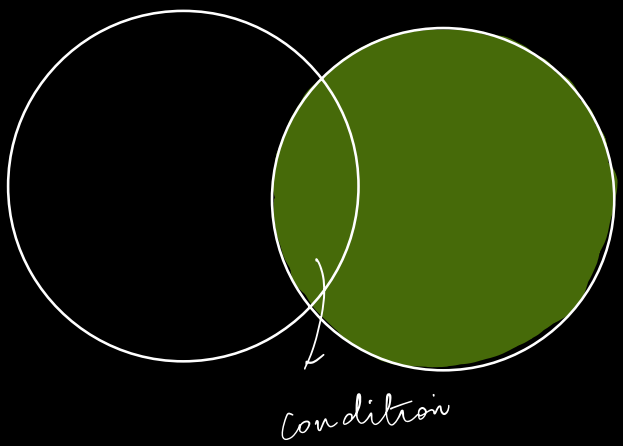
id	Name	batch_id
1	John	1
2	Jane	1
3	Jim	NULL
4	Jenny	NULL
5	Jack	2

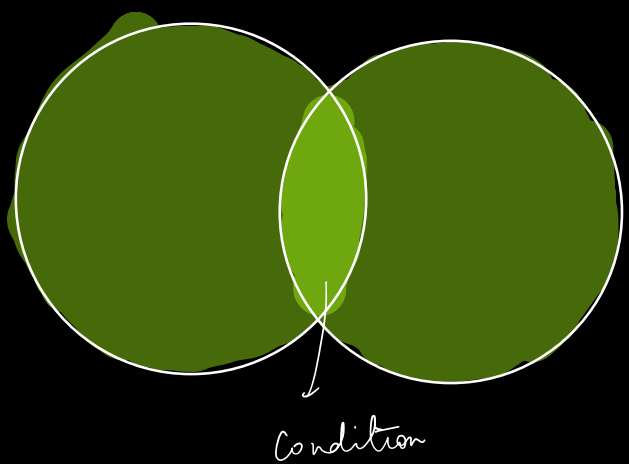
id	Name
1	A
2	B
3	C

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

→ full join


Left
Join
condition


Right
Join
condition


Full Join
Condition

Cross Join :- does not require any condition
returns all pairs of rows

Shows

id	Name	Date
1	Show 1	—
2	Show 2	—

Seat Types

Type
VIP
Gold
Platinum
Normal

Show 1	VIP
Show 1	Gold
Show 1	Platinum
Show 1	Normal
Show 2	VIP
Show 2	Gold
Show 2	Platinum
Show 2	Normal

```
Select
from Shows
join Seat
on true
```

→ cross join

=

```
Select
from Shows
cross join Seats
```

N rows in t1
M " t2
} N*M

Is Full Join is same as Cross Join ?

A + B

A * B

Types of Joins
- Inner Join
- Outer Join
  - Left
  - Right
  - Full
- Cross Join

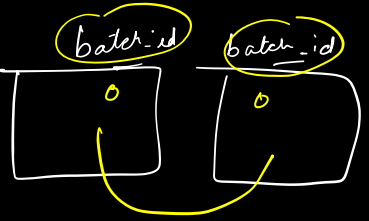
Break of 7 Min

USING :-

1) Joining based on equality

2) Name of column are same on both sides

```
Select s.name, b.name
from Students s
join Batches b
on s.batch_id = b.id
```




```
Select
from Students
join Batches
using (batch_id)
```

NATURAL JOIN :-

Whenever you want to join 2 tables based on equality of all columns having same name in both.

t1

a	b	c	d	e

t2

g	h	i	c	d	b

```
Select *
from t1                                    (1)
join t2
on t1.b = t2.b && t1.d = t2.d && t1.c = t2.c
```

```
Select *
from t1                                    (2)
join t2
using ( b , d , c )
```

```
Select *                                   (3)
from t1
natural join t2
```

→ automatically considers all columns having same name in both table

IMPLICIT JOIN :- Joining tables without using the join keyword.

⇓

Cross Join

```
Select *
from A, B
```

Same as Cross join

Stich every row of A with every row of B

```
Select s.name, b.name
from Students s
join Batches b
on s.batch_id = b.id
```

①

```
Select s.name, b.name
from Students s, Batches b
where s.b_id = b.id
```

②

Join with ON (Normal Join) > Join with where (Implicit join)

Join with ON :-

```
A = [ ]
B = [ ]
ans = [ ] // → contains only filtered pairs of rows which matches the condition

for each row1 in A
    for each row2 in B
        if( cond is T)
            ans. add (row1 + row2)

for each row in ans
    print (row[ . . . ], row[ . . . ])
```

Joins with ON

Joins with where :-

```
ans = [ ] // contains all pairs of rows like cross join
for each row in A
    for each row in B
        ans.add (row1 + row2)

for each row in ans
    if( cond where is T)        // where
        print (row[ . . ], row[ . . .])
```

Joins with ON

ON gets applied at the time of creating the intermediate table

better performance

less memory

Joins with WHERE

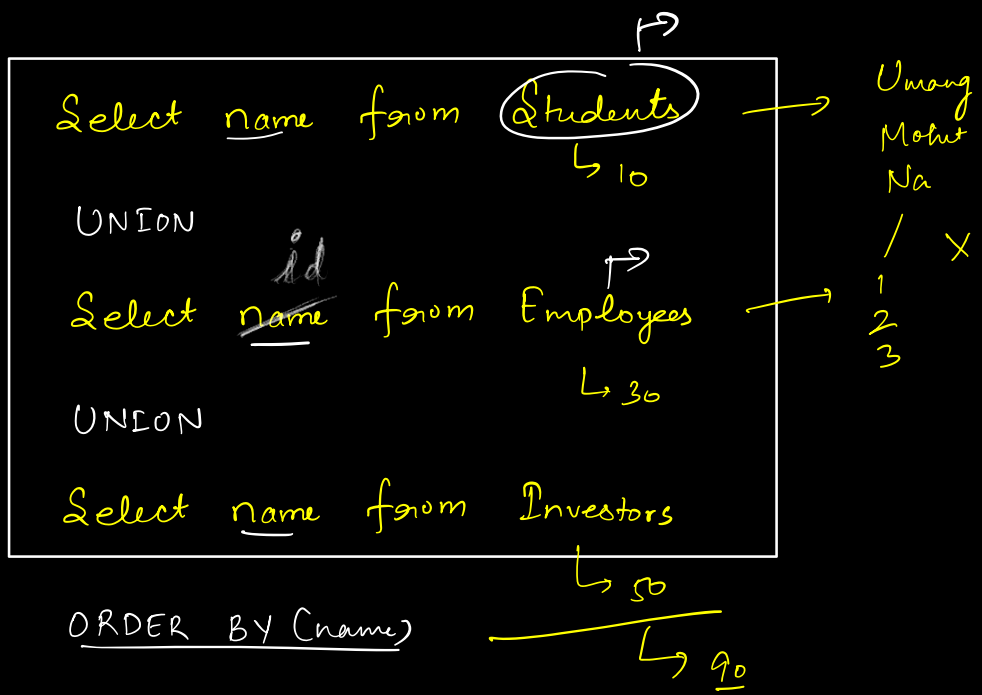
WHERE gets applied at finally printing

slow perf

more memory

UNION :-

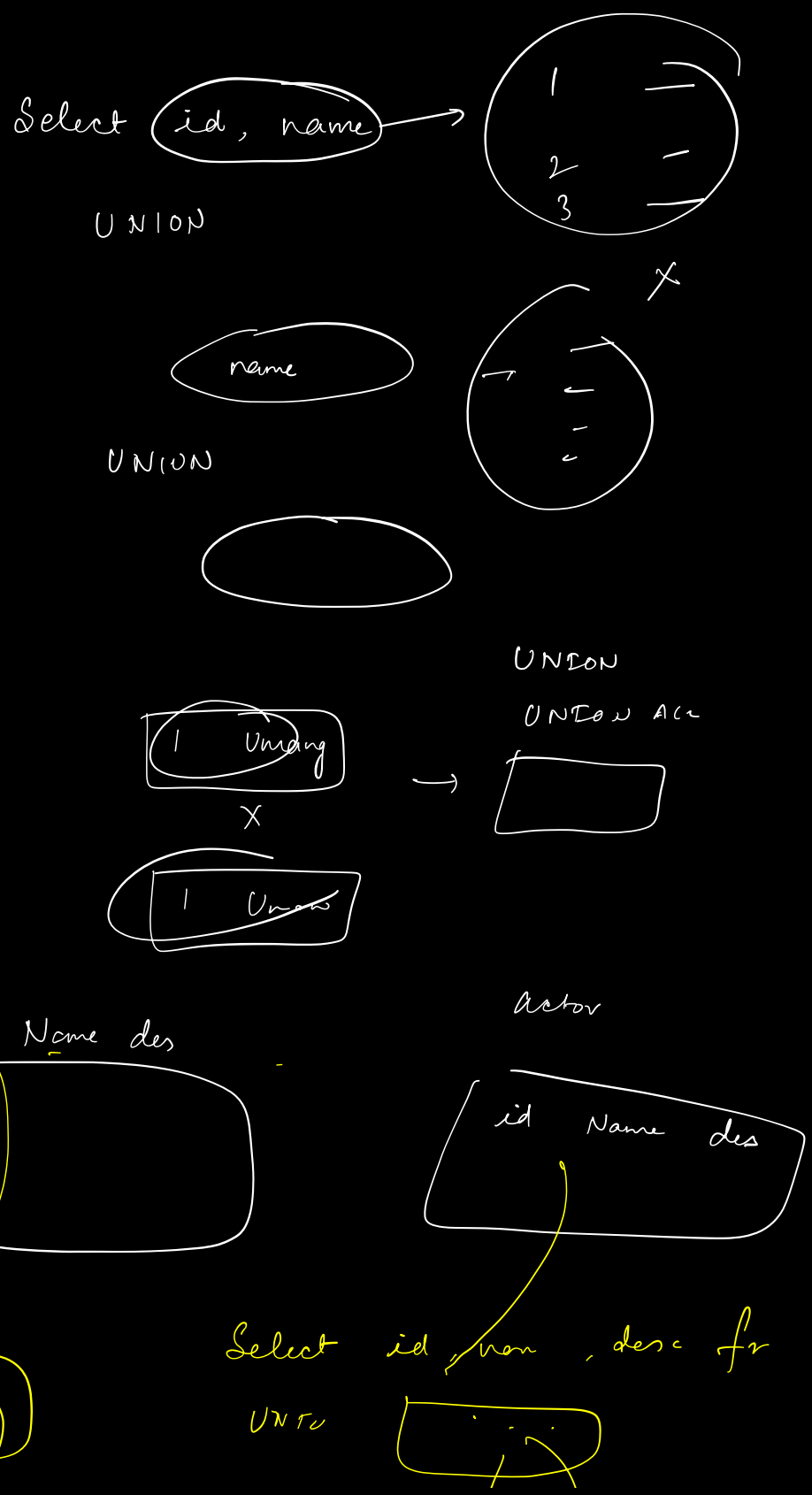
Students Employees Investors



UMANG

UNION

UNION ALL ✓


Select id, name
1
2
3
UNION
X
name
UNION
UNION
UNION ALL
X
film
id Name des
actor
id Name des
Select id, nom, desc fr -
UNION