Joins I

Hello everyone!

We start at 7:05AM.

Agenda.

- update
- delete
  - Drop
  - Truncate
  - Delete

Joins

- INNER joins
- self joins
- outer joins
  - right join
  - left join
  - full join
- cross join
- Syntactic sugar
  - USING
  - Natural joins
  - implicit joins
- Joins with where

Update

→ change data that is already present in the table in the database

'UPDATE' clause

update { table_name }
set { new_values of a particular coln or a set of cols }
where { where-clause }

students

id	name	psp	
1	ujjwal	~~100~~	110
2	Meera	~~72~~ 63	82 73
–	–		
–	–		

update students
set psp = psp + 10
~~where id = 2 ;~~

update students
set name = 'Ms. Meera' , psp = psp + 10
where id = 2 ;

★ update multiple rows at once

```
update students
set psp = psp + 5
where psp < 30;
```

★ If you miss the where clause, it will update the coln in each of the rows.

Delete

→ It always deletes a complete row.
→ It will not delete the table
→ instead used to delete rows in a table.

```
delete from {table_name}
where {condition}
```

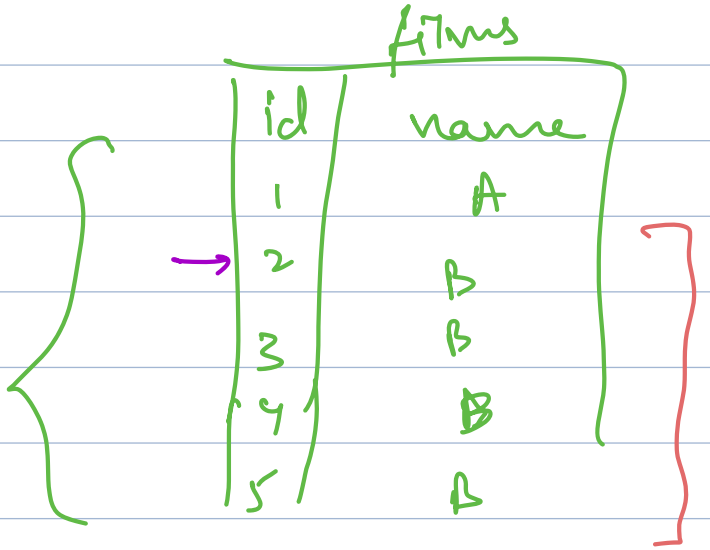
students

id	name	psp
1	A	1
2	B	2
3	C	6

```
delete from film
where ~~title = 'Deepak'~~ ;
where id = 2 ;
```

go to film table and delete all the movies by the title of 'Deepak'

```
delete from films ;
```




```
delete from film
where name = 'B'
Limit 2 ;
```

Code

```
for each row in film :
    if row matches cond in where :
        delete row;
```

★ If there is no where clause in delete statement, it will delete all the rows.

Delete vs Truncate vs Drop

Delete → delete few of the rows in a table
- rows that match a given condition

Truncate → truncate { table_name }
- → all of the rows will be deleted.
- → similar in outcome to delete from { table_name };
- → faster than delete operation to delete all rows.

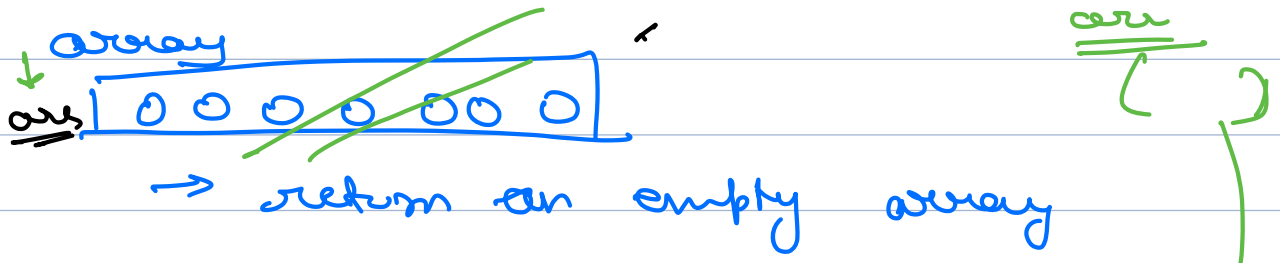
Delete → delete rows one by one.
- → possible to undo an operate
- → does not change auto increment sequence

Truncate → it is faster because it does not go row by row, instead it vanishes the

free table and create a new table.

→ Not possible to undo an operation.

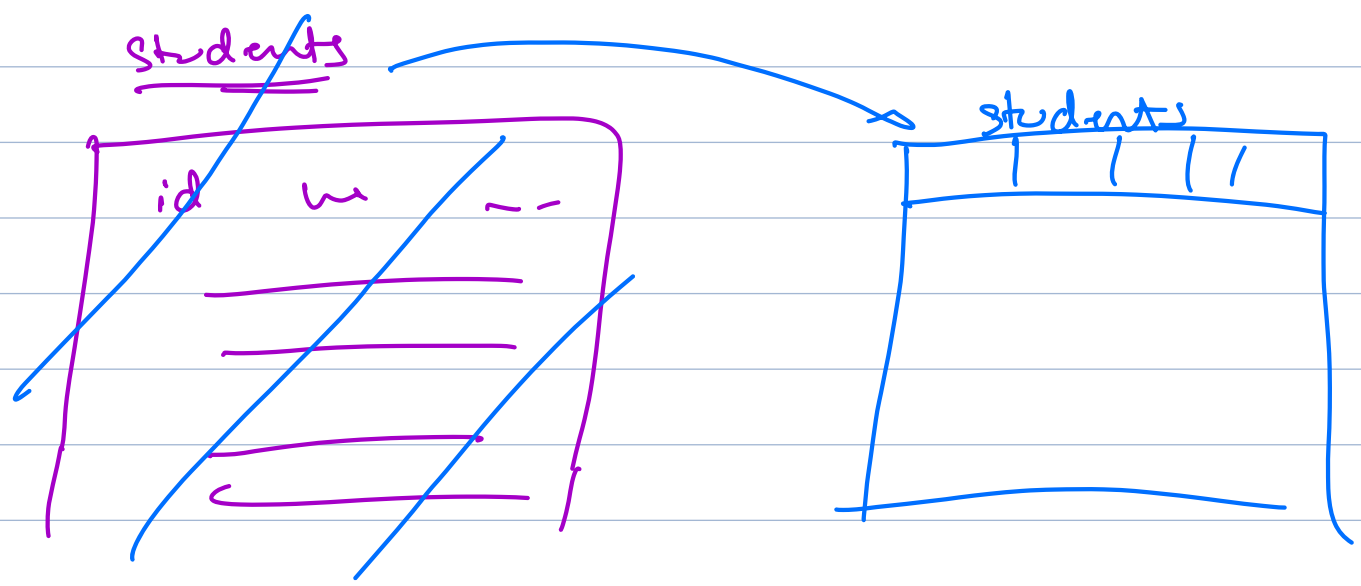
→ resets auto increment sequence.



→ return an empty array

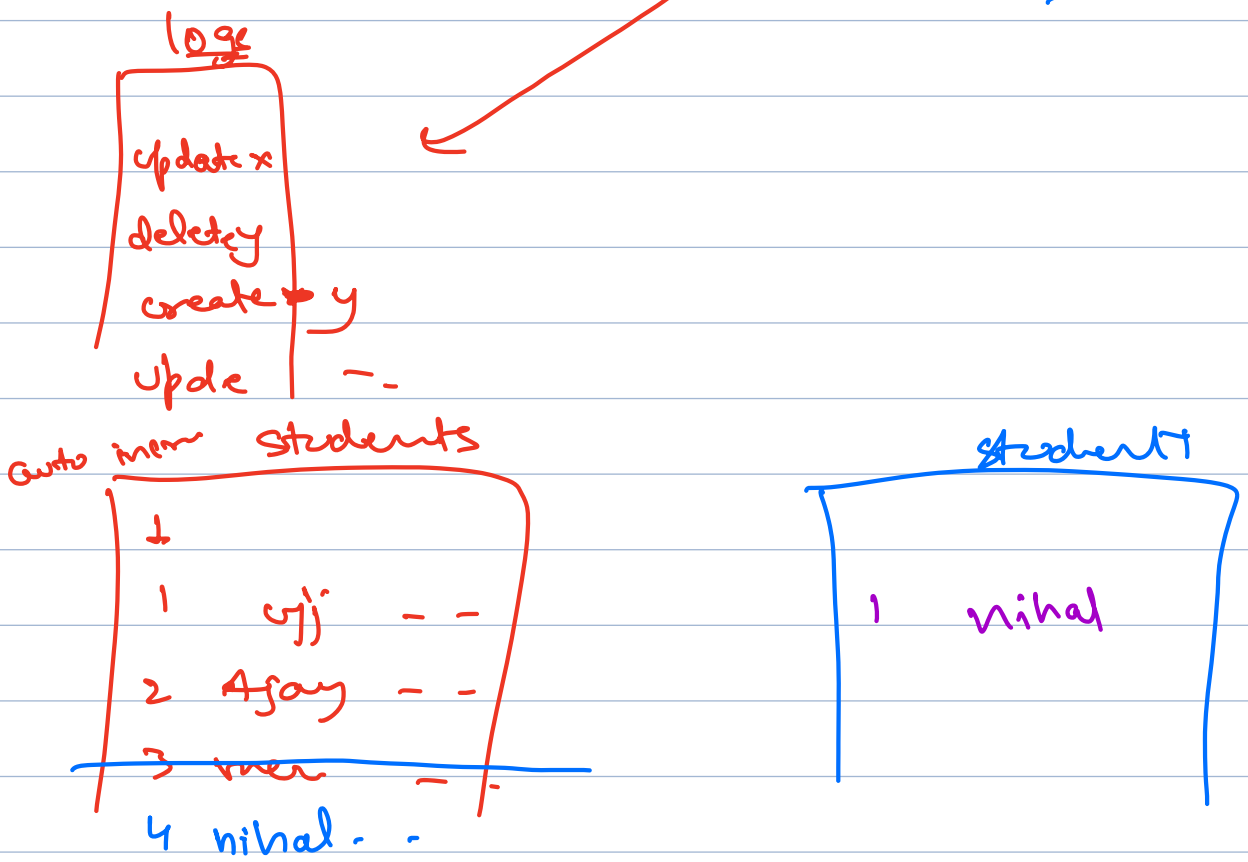
option 1 → delete elements one by one and return arr

option 2 → return a new array



delete from Students;

truncate students;



Delete from	Truncate
① Deletes row by row	① Completely removes the older table and creates a new table with clean look
② Slow	② faster
③ Doesn't reset the auto-increment key	④ resets the key

(4) undo a row that was deleted is possible	(4) cannot do an undo operation.
(5) can delete specific rows	(5) always deletes the entire set of rows.

Drop

→ does not just delete the data
→ delete the table.

* After delete / truncate you are still left with a table but with no data, while drop you loose all the data as well as the table.

Example

A ⟶ B relationship
↓
have some problems

Team delete

→ go through each of the problems in a relationship 1 by 1 and try to resolve them.

Team truncate → breakup
→ new relationship
A → C
B → D

Team drop → breakup
→ Hag se single

[Break]

08 : 29 AM.

Joins

→ till now . we have done queries on a single table

students

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

batches

id	name
1	A
2	B
3	C

Q Print name of every student with name of their batch.

name	batch_name
John	A
Jane	A
Jim	B
Jenny	C
Jack	B

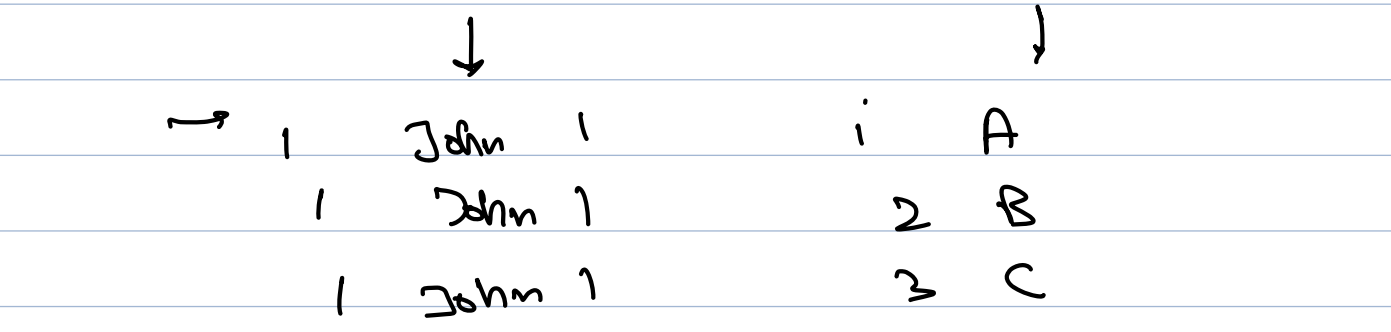
Joins → Allow to combine data across multiple tables.

students

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

batches

id	name
1	A
2	B
3	C



2 Jane 1	1 A
2 Jan 1	2 B
2 Jane 1	3 C

-> I want to be able to stitch data of a row of students and data of a row of batches together If students' b_id = batches's id

↓↓

Join condition

↓↓

```
select [students.name, batches.name]   <- fill at run end
  from students
  join batches
  on students.b_id = batches.id;
```

Code

```
ans = [ {---, ---}, {---, ---} ]
for each row1 in students:
    for each row2 in batches:
        if condition is true:
            ans.add( stitched row)
```

~~print(ans)~~

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

students

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

batches

id	name
1	A
2	B
3	C

```
from students
join batches
on students.b_id = batches.id
```

ans

students			batches	
id	name	b_id	id	name
1	John	1	1	A
2	Jane	1	1	A
3	Jim	2	2	B
4	Jenny	3	3	C
5	Jack	2	2	B

O (n * m).

1

```
select *
from students
join batches
on students.b_id = batches.id;
```

2

```
select ___
from students
join batches
on students.b_id < batches.id;
```

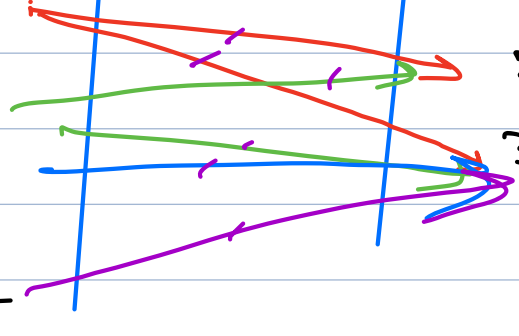
How many rows will it give / print for the above query;

students

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

batches

id	name
1	A
2	B
3	C



6 rows

- select *
  from students
  join batches
  on length(students.name) > length(batches.name)

Student buddy.

→ we assign a senior student as a buddy to you

students.

id	name	batch_id	pic	buddy_id
1	Ujjwal	1	—	3
2	Aravind	1	—	3
3	Bharath	2	—	null

→ A buddy is nothing but a student

→ A buddy_id is nothing but a student_id

Q for every student print their name along with their buddy's name

Vijival Aswind	Bharath Bharath

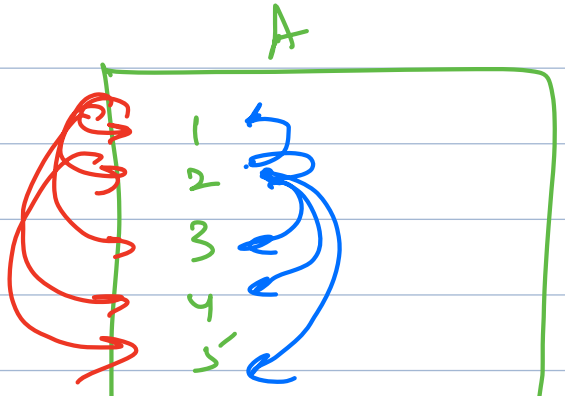
buddies

id	name	buddy_id	psp

if buddies was a separate table

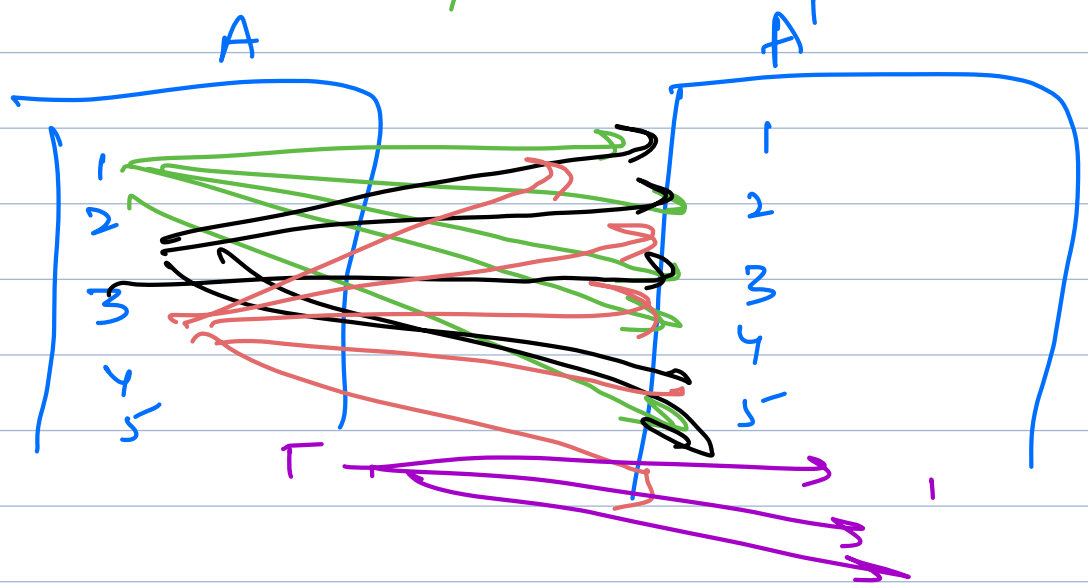
```
select s.name, b.name
from student s
join buddies b
on s.buddy_id = b.id;
```

Self Join - combine rows of a table with itself.

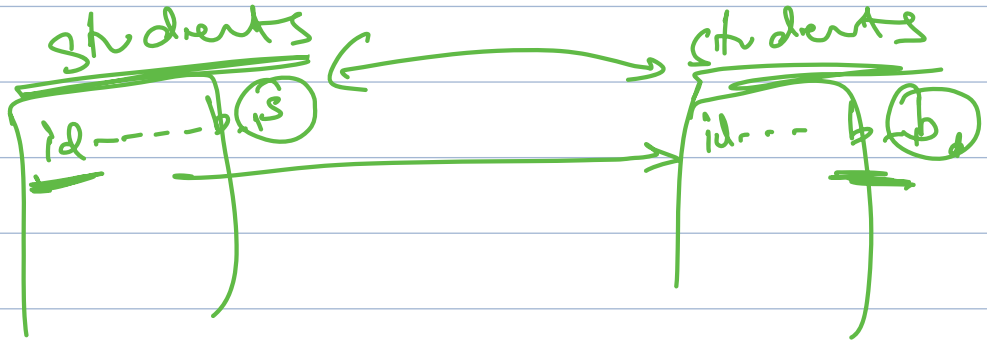


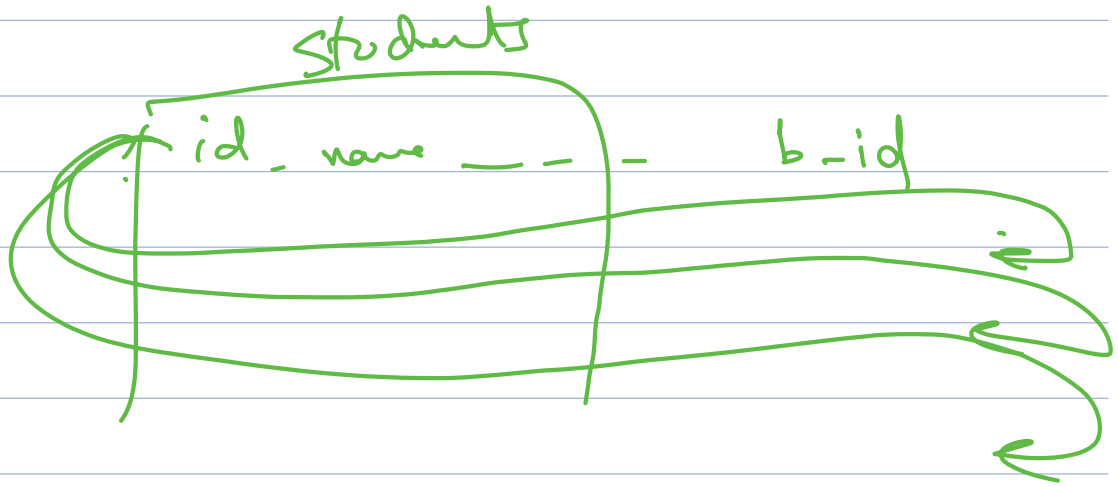

```
from A
join A
```

$O(n^2)$




```
select  s.name ,  b.name
from    student s
join    students b
on      s. buddy_id  =  b.id ;
```





students.

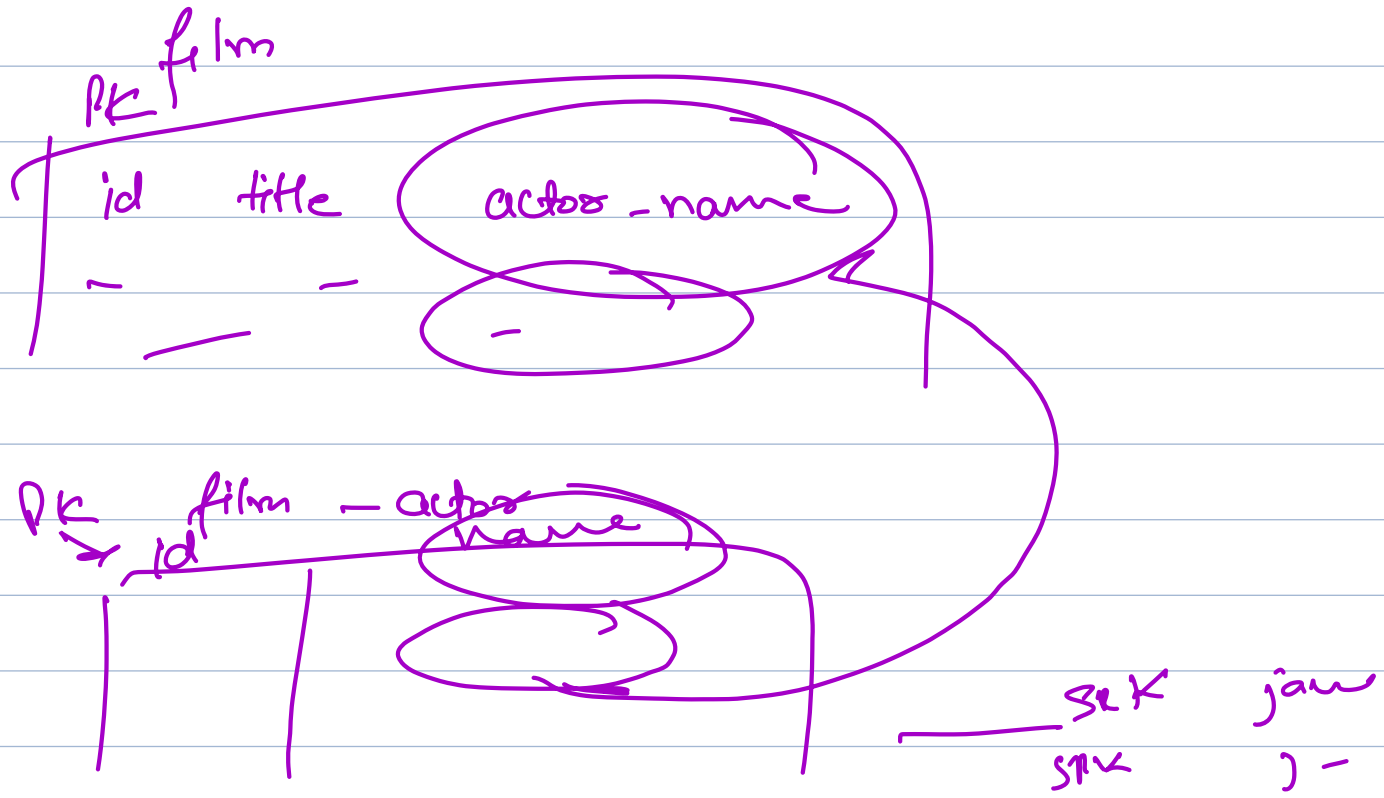
id	name	batch_id	pop	buddy_id
1	Ujjwal	1	—	3
2	Aravind	1	—	3
3	Bharath	2	—	null

```
select  s.name , b.name
from    student s
join    students b
on      s.buddy_id = b.id ;
```

ujjwal	bharath
aravind	bharath

employee

id	name	email	phone	mgr_id
1	ujjwal	—	—	2
2	gayatri	—	—	null


film
PK
id
title
actor_name
PK
id
film
actor
name