Agenda:

UPDATE

DELETE

↳ delete vs truncate vs drop.

Joins:

→ Inner Join

→ Self Join

→ Outer Joins

→ Right Join

→ left Join

→ Full Join

→ Cross Join

→ Syntactic Sugars:

→ USING

→ Natural Join

→ Implicit Join

→ Joins with WHERE clause

UPDATE

→ change data that is already present in a table in a database.

UPDATE clause

Syntax :

```
Update {table_name}
set { new value of a particular col^n or a set of col^n }
where cond^n
```

Students

id	name	psp
1	Deepak	80
2	Mohit	70
3	Sumeet	95

Q. For the student of id = 1, increase his psp by 10.

A)
```
Update students
set psp = psp + 10
where id = 1
```

what if I miss writing the WHERE Clause?

→ It will update all the rows

```
Update Students
set name = 'Dhruv', psp = 80
where id < 10
```

Code

```
for each row in table:
    if row matches cond^n in WHERE:
        row[name] = 'Dhruv'
        row[psp] = 80
        row[batch_id] = 3
```

If I dont give where Clause

always true

DELETE

→ It is used to delete rows from the table

→ It will not delete a table.

Syntax:

```
delete from {table_name}
where cond^n
```

Q) what if I miss the where cond^n?

A) everything from the table is gone.

Eg.

```
delete from f1)m
where title = 'deepak'.
```

Code

```
for each row in film:
    if row matches condition:
        delete that row.
```

Delete vs Truncate vs drop.

Delete ⇒ deletes few of the rows of the table given the cond^{n}.

Truncate ⇒ Syntax

```
TRUNCATE {table_name}
```

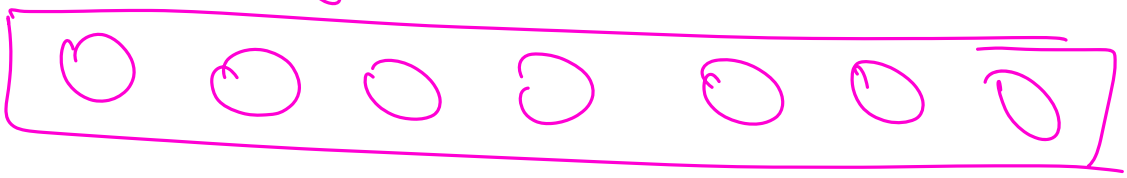
No WHERE Clause.

↳ all the rows of the table will be deleted.

→ Similar in Outcome to `delete from {table_name};`

Truncate is faster than deleting all the rows of the table. How?

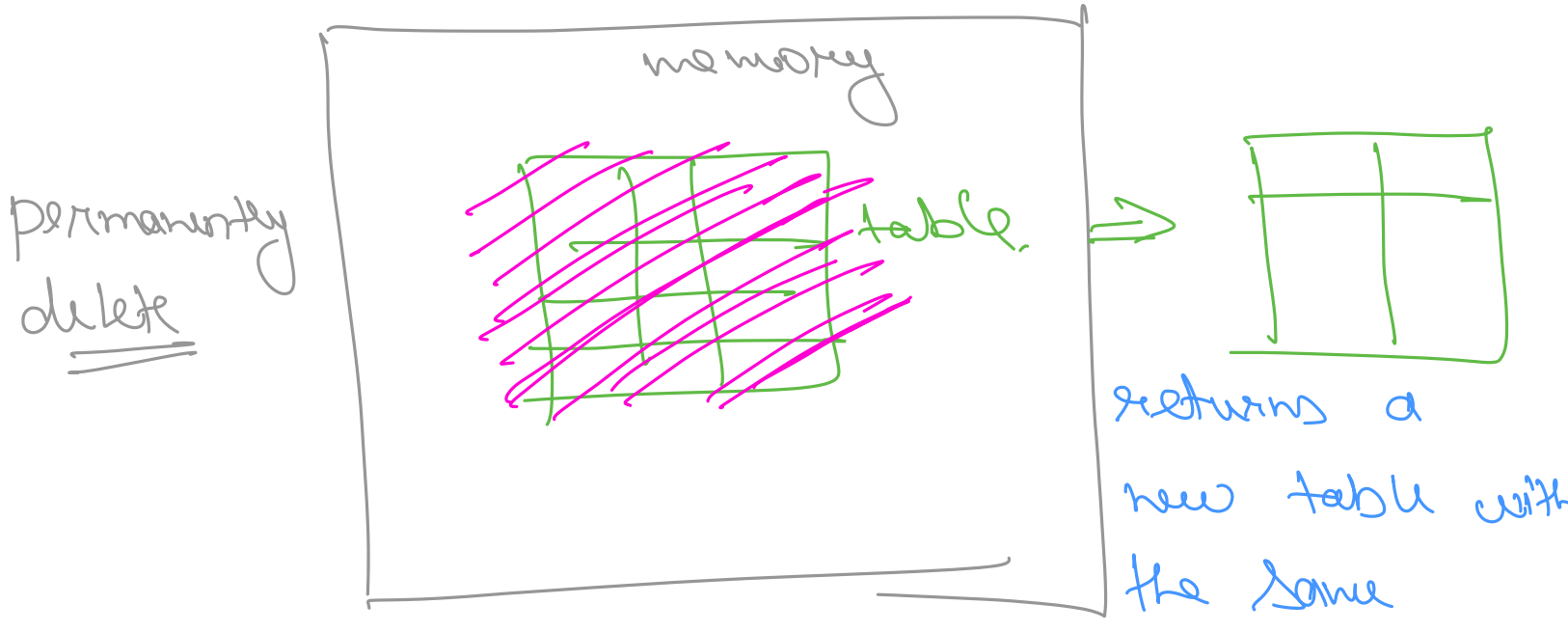
Q) Given an array, delete all the elements of the array.



1) Delete the entries one by one. O(N)

2) return new array(); O(1)

In truncate.



Schema.

Delete → goes row by row
→ you can undo a delete operation
→ Auto increment is not reset
→ Slower

Truncate → It will not go row by row. Instead it will vanish the table from memory, & return a new table.
→ You cannot undo in truncate.
→ Auto_increment is reset.
→ Faster.

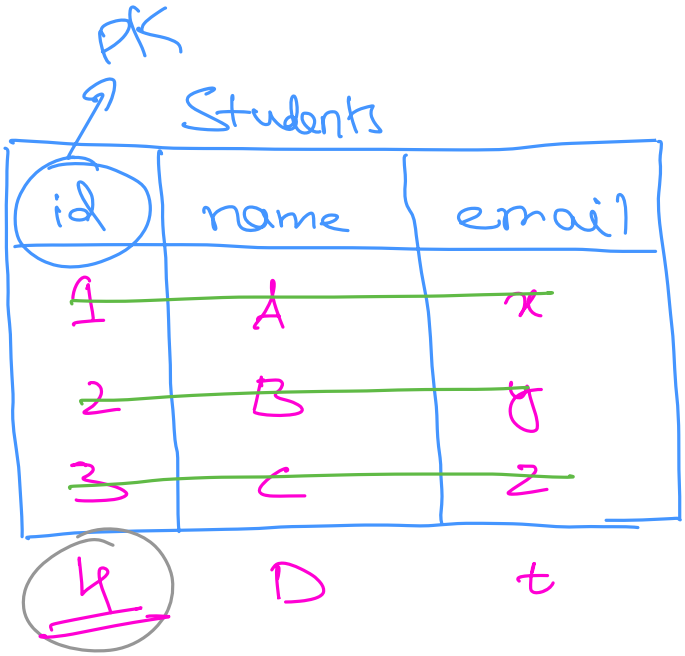
MySQL maintains a log file

delete 1
delete 2

→ maintains the before and after data.

⇓

You can do a rollback.



DROP

→ just doesn't delete the data

→ Delete the table completely from the memory.

→ Does not return a new table.

→ Even the Schema is lost.

Syntax: Drop table {table_name};.

(A) — (B) rel^n

↳ Some problems happened.
↳ list of problems.

(1) Team delete :

→ go through their problems 1 by 1 and resolve them.
→ deleting their problems 1 by 1.

(2) Team Truncate :

→ Break up
→ New rel^n

A — C
B — D

(3) Team Drop

→ break up
→ Haq se single

Break till 10:15 PM

Joins

→ till now, queries on a single table.

Students

id	name	batch_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 the name of every student with the name of their batch.

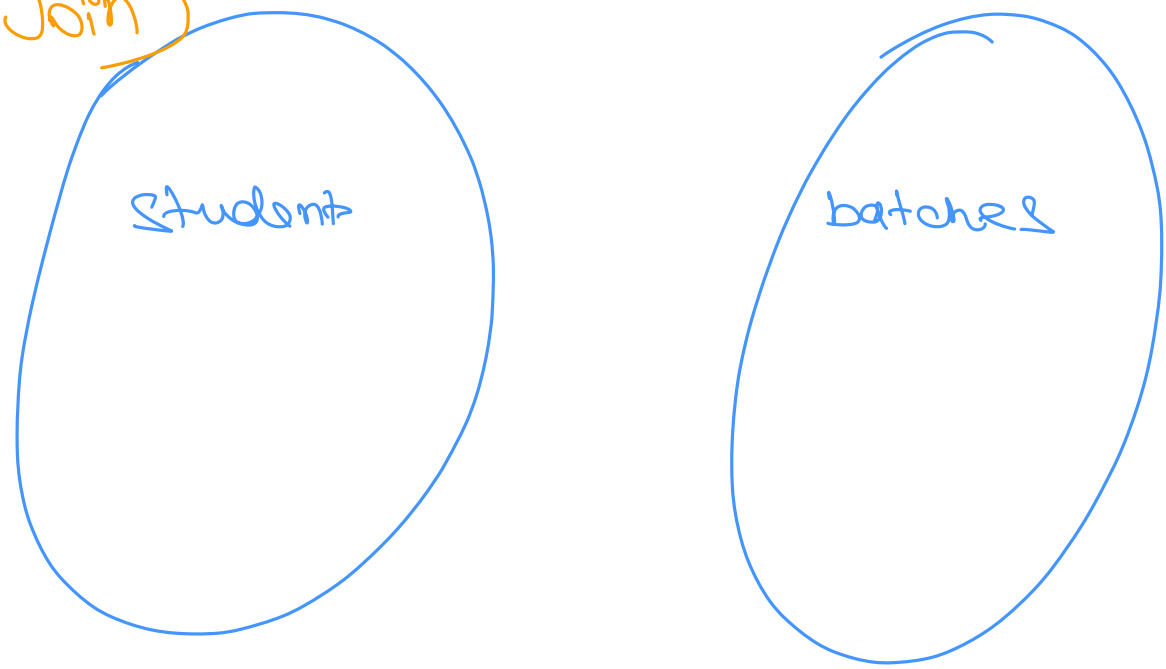
A)

John	A
Jane	A
Jim	B

Jenny	C
Jack	B

Join ⇒ Combine data of two tables together.

(Same as Inner Join)



Students

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

(N)

Batches

id	name
1	A
2	B
3	C

(M)

$O(N * M)$

1) match each row of one side with every row on the other side.

name	batch_id	id	name
John	1	1	A
John	1	2	B
John	1	3	C
Jane	1	1	A
Jane	1	2	B
Jane	1	3	C
Jim	2	1	A

Cartesian product

I want to be able to stitch the data of row of students with the row of batches iff students.batch_id = batches.id

↳ we need a criteria to join the two tables.

Virtual Table : Stores the actual Answer for Join.

Students			Batches	
id	name	batch_id	id	name
1	John	1	1	A
2	Jane	1	1	A
3	Jim	2	2	B
4	Jerry	3	3	C.
:	:			

Time Complexity is going to be, in the worst case, O(N * M).

Syntax:

```
Select *
from Students
Join batches
ON Students.batch_id = batches.id
```

```
Select *
from Students
Join batches
ON Students.batch_id < batches.id
```

PK

```
Select *
from Students
Join batches
ON 1 < 2;
```

$\Rightarrow N * M$

Conditions are not related to the PK of the FK.

Code

table 1 = [{} .. {} {} ....]

table 2 = [{} ... ~ ----]

ans = [ ]

for each row1 in table1 :

for each row2 in table2 :

if cond^n matches :

ans.add(stitched row) ;

push the entire row into of both the tables.

for each row in ans :

print (row[students.name], row[batches.name])

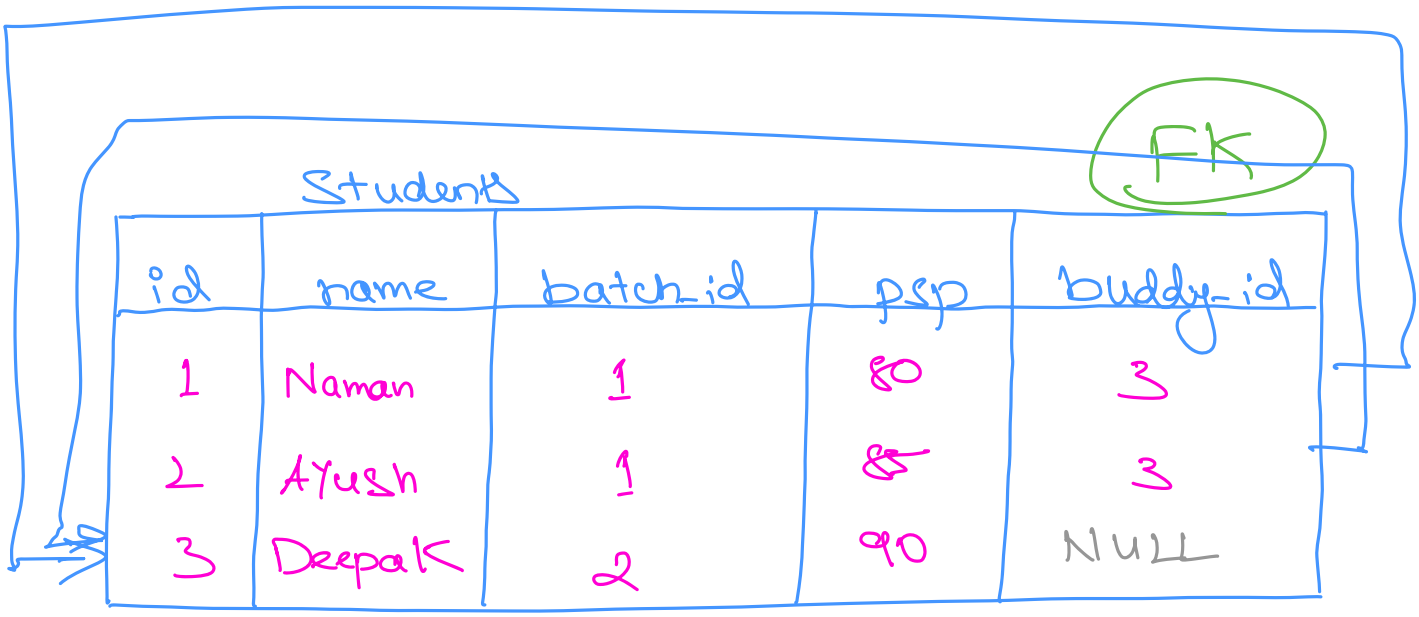
Aliases in Joins.

Syntax :

```
Select  s.name , b.batch_name
from    Students  (s)        as X
join    batches   b
On      {cond^n}
```

Student Buddy.

↳ we assign a senior student as a buddy to you.



Q For those students that have a buddy print their name, along with their buddy name.

name	buddy_name
Naman	Deepak
Ayush	Deepak

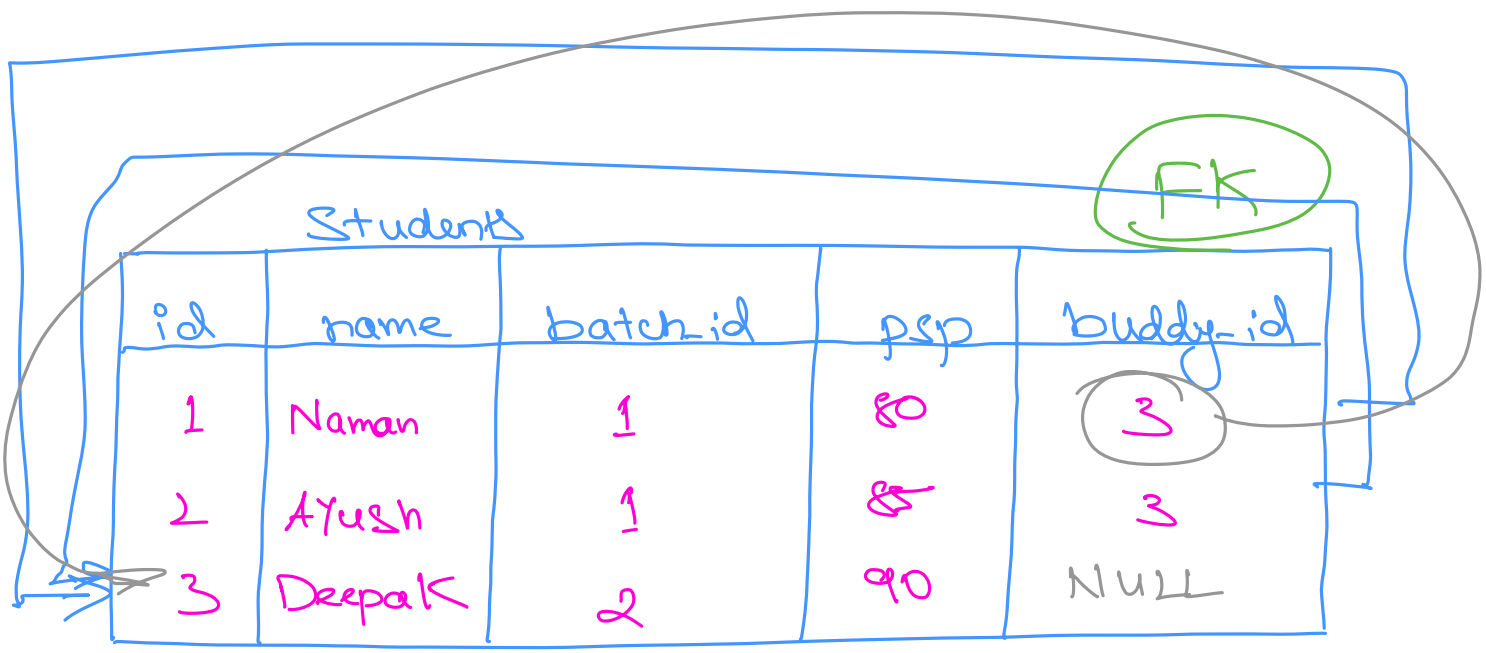
buddy_id is a self-referential FK

Self Join.

buddy table.

id	name	batch_id	psp
1	Naman	1	80
2	Ayush	1	85
3	Deepak	2	90

buddy



```
Select   a.name, b.name
from     students  a
join     buddy     b
On       a.buddy_id = b.id
```

```
Select   a.name, b.name
from     students  a
join     students  b
On       a.buddy_id = b.id
```

Q is Alias mandatory for a self join?

```
Select  a.name , b.name
from    Students  a
join    Students  b
On      a.buddy_id = b.id
```

Self join - Combine rows of a table with itself, using aliases.

employee

id	name	email	phone	manager_id
1	Sahaish	abc	123	3
2	Naman	xyz	456	3
3	Mohit	efg	789	NULL

Q for every employee, give their name, and their managers name.

Selet e.name, m.name → this will be inverted, if cond^{n} is wrong

from employee e

join employee m

on e.manager_id = m.id

e.id = m.manager.id