## Agenda

- → In operator
- → Between
- → Like
- → Is Null
- → order By
- → Limit

Read

- → update
- → delete

## In operator

where id = 1 , rating = 'PG'

id (1, 3, 7, 13).

100 values

```
if (id = 1 ||
    id = 3 ||
    id = 7 ||
    id = 13).
```

redundant.

Java   if ( array.contains(1)).

JS. if ( array. indexof() ).

Select * from Table where id IN (1, 3, 7, 9).

## Between

age: ( 2005 & 1995 )

age >= 1995 and age <= 2005.

select * from students where age >= 1995 ~~or~~ and age <= 2005.

→ inclusive of both the values.

→ Int, floats, strings, dates.

24

a ~~~~ z

## Like

→ Pattern matching

→ column (string) {search}.

→ SQL (string pattern matching).

→ LIKE

↑

id = 1 (int)

where ~~value~~ value = 'something'.

“thing” { “something”, “anything” }

Pattern matching.

| batch_id | name |
|---|---|
| 1 | Academy_morning_beg_Java |
| 2 | Academy_evening_int_java |
| 3 | evening_adv_Python |

“Academy” or “morning”

wildcards in Like

| ( _ ) | (%). |
|---|---|
| only (_) one occurrence.<br>(1) | any of occurrence<br>(0, 1, 2 ....) |

any character.

select * from batch where name like '% Academy %'

Academy.

Academy

cat

% at %. ✓

c

% cat %. ✓

0 char. %t ✓ 0 char

ca

_at _ a t _ c a t

c _t cat

c/a. → ct ×, at ×.

~~in-cas.~~ case - insesitive

collations

col$^n$ case sensitivity

(1) %

(n).

% LOVE %

(0) (0) end.

→ starts with love? ✓

ends with love.? ✓

has char L, O, V, E anywhere.?

~ L ~ O ~ V ~ E ~ ✗

LOVE ✓

→ ends with 'son'

%son → ending with son.

son% → start with son

↳ (m)

→ moon.

Book name which contains 'moon'

$\boxed{\%moon\%}$

above: m, o, o, n {continous}.

& stars

moonley.

→ 5 character & the middle is '123'.

_ 1 2 3 _ .

%123% {12345, 12399, 12390, 00123}-

%123. { ∼∼∼ 123 }.

{123}

_123_

break 10:25

is NULL

~~ds~~ desc is NULL in film

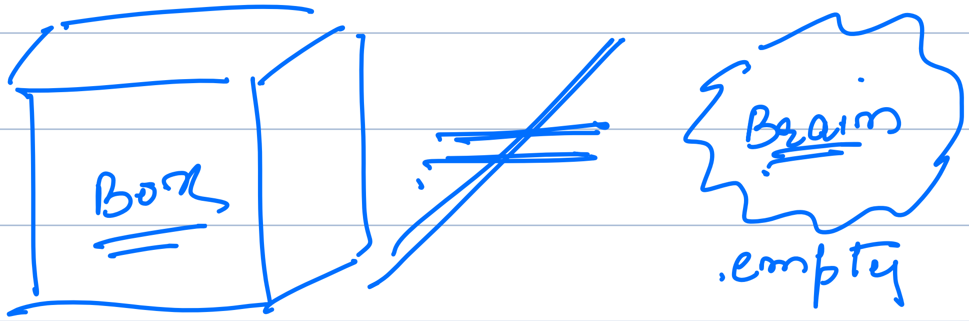
select * from where descriptⁿ = NULL

This will not work

→ null pointer exceptⁿ

NULL = 0 ?

NULL = NULL



NULL = NULL. ?? ✗

NULL { might be the

Nothing

in future }.

NULL != NULL ?? ✗

→ No Arithemetic operatⁿ on null.

| id | name | batch_id |
|---|---|---|
| 1 | ayush | 20 |
| → 2 | prakash | 1. |
| → 3 | pranav | NULL |

where batch_id != 20

2+3|

var += 3

select * from std where batch_id <> 20
And batch_id is NulL

order By

→ rows returned are not garanteed To be in the same order.

order : ascending & descending
{sorting}.

{order By}.

you can sort on which you select

col$^m$ don't

→ Default asc.

→ Null : first asc , last desc

```
Table { {}, {}, {} },

answer = [ ].

for row in films
    if (row, match)
        answer.append(row).

answer.sort(col$^m$_name in order).
        { if (col$^m$1)
              if (col2) }.

filt_ans = [ ]

    for row in answer.
        filter_ans.append (row['rating'])
                               'Title'.

return filter_ans
```

( where )

↓

order by.

↓

col$^m$ filter

col field

| name | year of birth |
|---|---|
| ayush. | 1990. |
| ayush | 2000 |
| Prakash | 1995 |

{ ayush. Prakash }.

ayush.
Prakash

Distinct names order by YOB

ayush - 1990, 2000
Prakash 1995

select Distinct ~~#~~name from std.
order By name.

limit

Scroll → [0, 100] [101, 200], ....

limit 1000 → 100
DB B
900

Offset(value)

↳ ↑

skip (value)

return filter_arr [start of limit : end of limit]

## UPDATE

update Table_name set $col^n$_name = value where conditions.

Do Not forget The where condition

FIRED ??

No Promotion

```
for row in film
Don't forget → if (conditⁿ)
                   row['col₁ⁿ'] = upate
                   row['col₂ⁿ'] = upate 2
```

## Delete

Delete from Table where conditⁿ

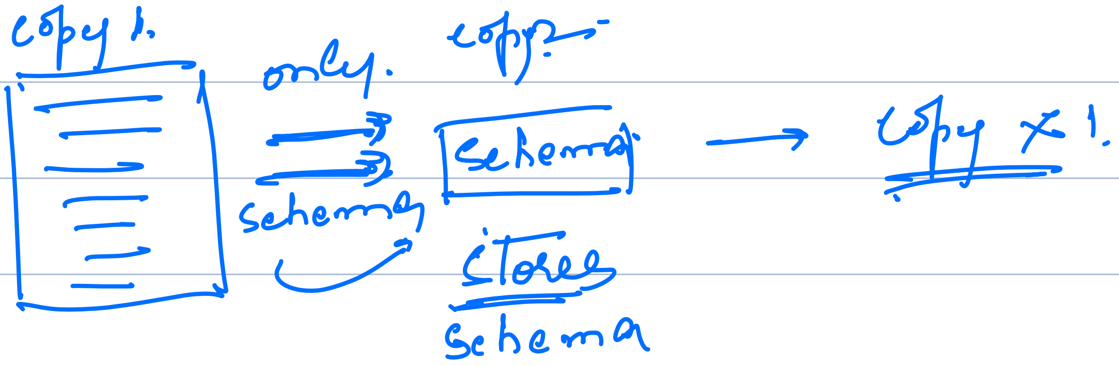
```
(for row in film (log) DB logs / Bin logs
Don't forget → if row matches (conditions) logs
                   delete row.
```

## Truncate

→ Truncate Table

→ removing all the rows.

→ structure is preserved.



→ Truncate is faster than deleting

→ DB ~~st~~ does not store logs.

copy schema → another place.

## DROP

DROP Table film;

Bombe. everything
schema rows and you.