Today's Agenda :-

- READ
    - BETWEEN
    - LIKE
    - IS NULL
    - ORDER BY
    - LIMIT
- UPDATE
- DELETE
    - DELETE VS TRUNCATE

LIKE :-

batches

id	Name
	Academy May 23 Beg
	June 24 Academy Intermediat

→ Academy

```
Select * from batches
 where name = "Academy"
```

exact match with entire value *

```
bool checkAcademy (String batchName) {
    if (batchName. contains ("Academy") &&
              batchName. contains ( "Morning") {
        return T
    }
    return F
}
```

LIKE : when you want to check inside a string

i) '_' : exactly one occurence of any character

ii) '%' :- any # of occurences of any character

0, 1, 2, 3 . . . . .

* cat

_ _ t ✓

$\underbrace{\%}_{ca}$ t

% ✓

cat ✓

$\underbrace{\%}\ \underbrace{ca}$ ✗

$\underbrace{\%}_{0}$ cat $\underbrace{\%}_{0}$ ✓

Return all morning batches in academy

↳

batches

id	Name

name like %morning% academy% ✗

Morning Apr22 Beg Java Academy

June23 Morning DSML batch ✓

Academy Apr22 Beg Morning

*

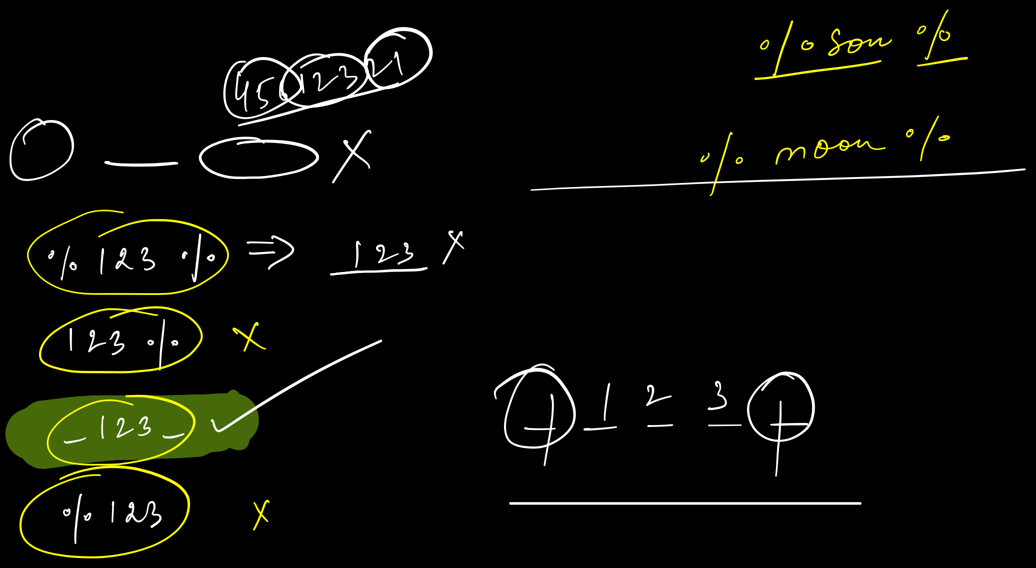
??

```
Select * from batches
where batchName like %morning%
      AND
      batchName like %academy%
```

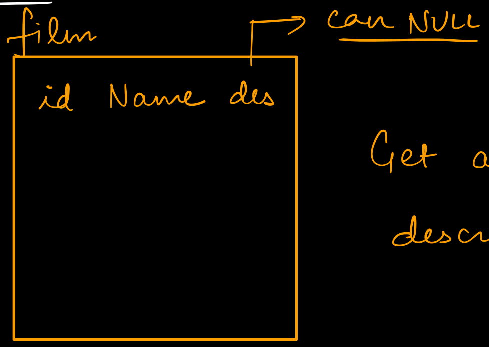
✓

a = 2
and
a = 4

a = 2
and
○ = 4 ✗
✓



IS NULL :-



Get all films where description is NULL

'NULL'

```
Select * from film
where description = NULL
```

X

NULL means nothing. You cannot use =/!= to check for null.

NULL Keyword

NOTHING = NULL ✗

NULL = NULL ✗

NULL != a ~~NULL~~ ✗

a = NULL ✓

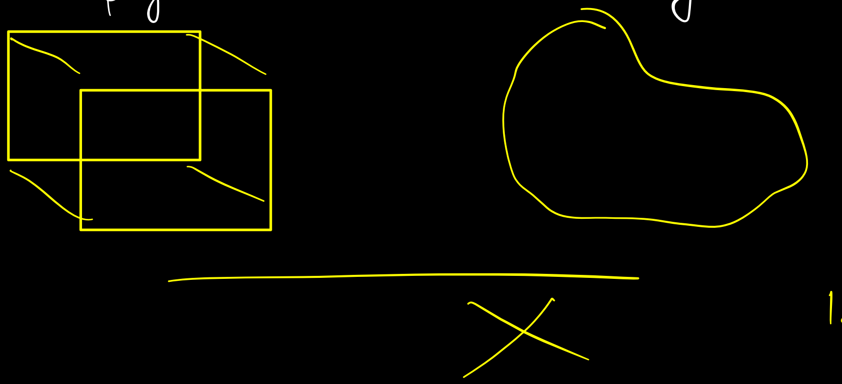
a != NULL ✗

false

=

!=

empty box empty brain



IS ✗

IS NULL =

IS NOT NULL !=

IS NULL

IS NOT NULL

Students

→ can be NULL

1 != 2 false

id	Name	batch_id
1	Aman	1
2	Sakshi	2
3	Yashi	NULL

Get all students with batch_id != 2

1 != 2

2 != 2 ✗

NULL != 2 ✗

NULL is NULL ✓

```
Select * from Students
where batch_id != 2
```

2

1	Aman	1

also should have been

```
Select * from Students
where batch_id <> 2 OR
      batch_id is NULL
```

✓

ORDER BY :-

By default, the order of rows in the output is not guaranteed.

If you want final ans to be ordered by something you should use order by.

By default, the tie breaker is primary key ✓

```
Select tille, distinct desc
from film
```

ORDER BY WITH DISTINCT :-

Select distinct (. .. ..)

Students

Name	yob
A	1990
B	1995
A	1998

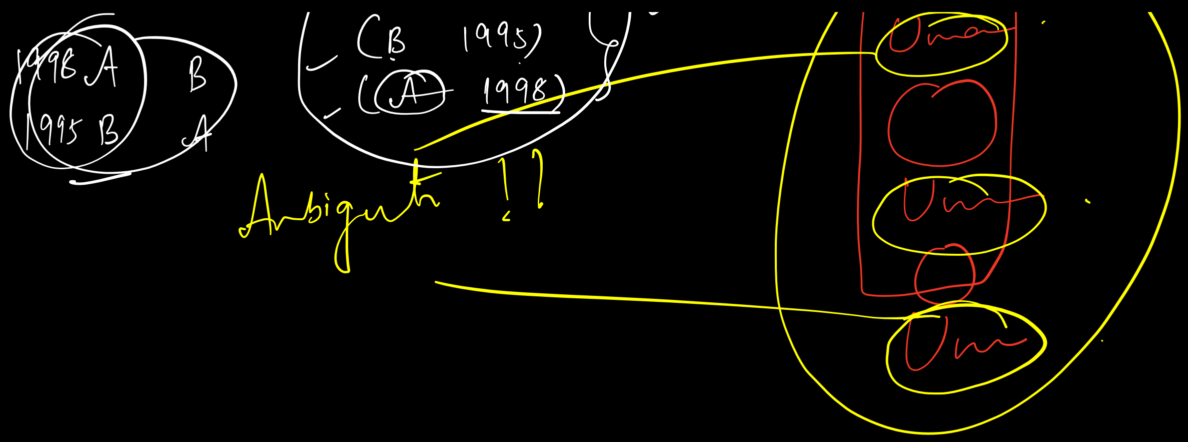
→ (A 1990) ✓

→ Ambiguity

```
Select distinct name, yob
from Students
order by yob ✓
```

A → 1990,
B → 1995,

A → 1998
B → 1995



If you have distinct in the select statement, then you can only sort by column that are present in the select statement.

UPDATE

```
UPDATE   table_name
SET  { New values of col or set of column)
WHERE {condition}
```

If you forget to mention where clause in update query, all rows will be updates.

DELETE

↳ used for deleting rows of the table.

↳ not going to delete the table

```
DELETE FROM {     }  → delete all rows.
where {     }
```

Both will delete rows of

Delete vs Truncate

delete but based on where clause

No condition (No where)

delete all row of the table

Delete from film ;

Truncate film;

Both will produce the same outcome

Delete deletes the row by row →

```
for ( ——— ){
|   if (whe) {
|   |   delete()
|   }
}
```

Truncate is faster than delete .

Rollback

Note that,
DELETE:

1. Removes specified rows one-by-one from table (may delete all rows if no condition is present in query but keeps table structure intact).
2. It is slower than TRUNCATE.
3. Doesn't reset the key.
4. It can be rolled back.

TRUNCATE:

1. Removes the complete table and then recreats it.
2. Faster than DELETE.
3. Resets the key.
4. It can not be rolled back because the complete table is deleted as an intermediate step.

DROP:

1. Removes complete table and the table structre as well.
2. It can not be rolled back.