Class starts at 9:05 PM

Agenda :

Read

Like

IS NULL

ORDER BY

LIMIT

UPDATE

DELETE :

delete vs truncate vs drop

LIKE Operator.

Batches

id	name
1	Academy Apr 22 Beginner Java
2	May 22 Academy Python Beginner
3	Apr 22 Beginner Morning Academy

→ every academy batch should have "academy" in its name.

→ every beginner batch should have "beginner" in its name

→ every morning batch should have "morning" in its name.

// print those batches that are of type "Morning" and "Academy".

```
public boolean checkBatchName(String bn){
    if( bn.contains("morning") &&
        bn.contains("Academy") ){
            return true;
    }
    return false;
}
```

Like Operator → when you have to check inside a string.

2 wildcard symbols :

1) " _ " → match exactly one occurence of a character

not even 0
not even 2

2) " % " → match any # of occurences of any characters.
(0, 1, 2, 3 .... 100 ...)

Eg.

String
Cat

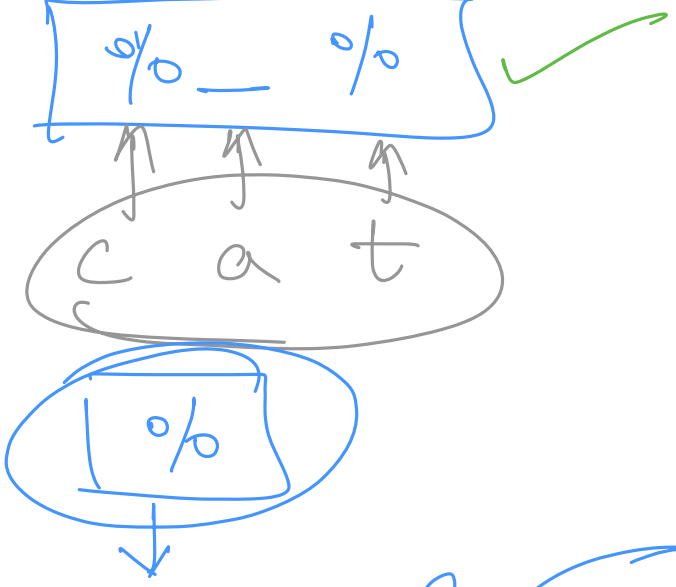
Pattern

_ at ✓

% t ✓

% ✓

% Cat %



3 occurences of anything → 1#c 1#a 1#t → _a_t

a b %_% check ✓
h i

Check if "hello" exists in title of a film

```
Select * FROM FILM
where title LIKE '%hello%'
```

Q print those batches that are of type "Morning" and "Academy".

```
Select * FROM batches
where name Like "%Morning%"
AND name Like "%Academy%"
```

```
Selec * FROM batches
where
name like '%Morning%Academy%'
```

→ Java 22 Academy Beginner Morning.

String matching is case In-sensitive

IS NULL Operator.

you cant use assignment operator '=' with NULL.

Film

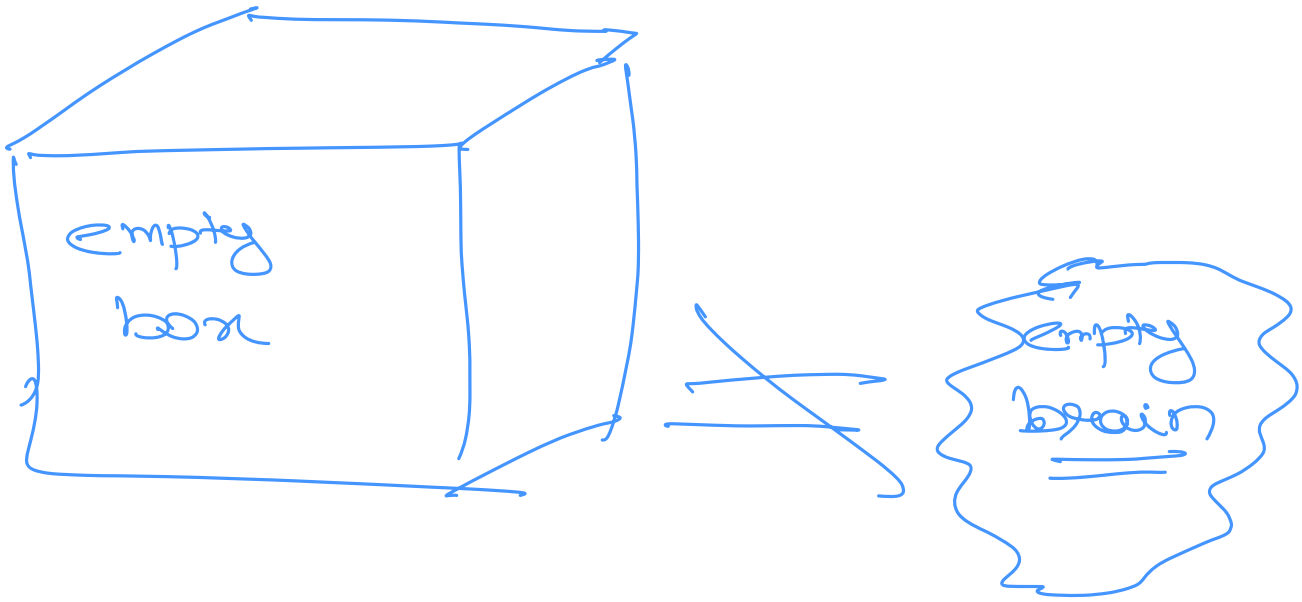
id	title	description
1	hello	NULL
2	again	"whatever"

Q Get the films where description = NULL.

```
Select * FROM FILM
WHERE description = 'NULL'
```

Null means nothing.

NULL = NULL → True ✗
NULL = NULL → False ✗
NULL = NULL → NULL



Students

id	name	batch_id
1	Naman	1
2	Amit	NULL
3	Deepak	2

Q Get all Students with batch_id = 2.

Select * from student where batch_id = 2

Q Get all Students with batch_id anything other than 2.

Null

NULL != 2

Select * from student where batch_id != 2

OR batch_id IS NULL

IS NULL → to check if value is actually Null

IS Not NULL → to check if value is not equal to Null, It has a valid entry.

ORDER BY

→ By default, MySQL does not guarantee the order of rows in the Output of a Query.

→ If you want the final answer to be sorted by something, use the ORDER BY clause

Q Sort the films by their titles.

A) Select * FROM film
ORDER BY title;

→ The default order of Sorting is Ascending Order.

Select * FROM film
ORDER BY title desc;

→ Specify desc if you want to sort in descending order.

Students

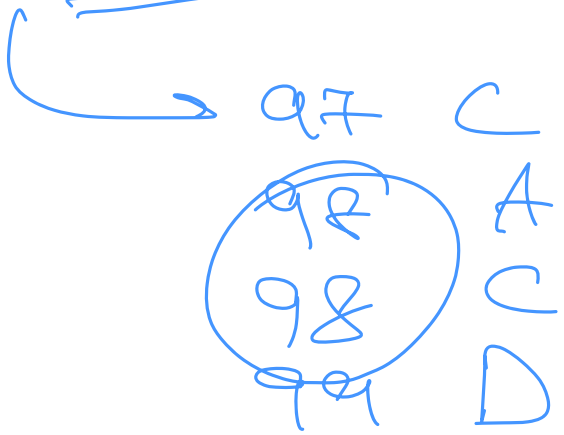
id	yob	name	psp
1	1998	C	80
2	1999	D	90
3	1997	C	100
4	1998	A	85

Sort by yob

ASC →
97 C
→ 98 A
→ 98 C
99 D

Query would be like this

```
Select * from Students
ORDER By yob, name;
```




```
Select * from Students
ORDER By yob, name desc;
```


```
Select * from Students
ORDER By yob desc, name desc;
```

→ By default, whenever there is a conflict in sorting, MySQL will sort according to the Primary Key (ascending Order)

⇓

It is unique

```
Select * from film
ORDER By on PK
```

I can even sort on columns I wont print.

→ Select title from film
Order By release_year.

Code

table = [{}, {}, {} .....]

ans1 = [ ]

for each row in table : → FROM

    if row matches cond^n : → WHERE

        ans1.add(row)

ans1.sort(sorting condition)

for each row in ans1 :

    print(row[col^m 1], row[col^m 2] ...)

Break till 10:27 PM

Distinct in Order By

Select distinct title from film
where length > 90
Order by release_year. ;

FROM → film

↓

WHERE → Select complete Row that has length > 90.

↓

SELECT → Select only 'title' from the complete Row.

↓

DISTINCT → Unique 'title'

↓

ORDER BY → ✗ You only have 'title' is no 'release_year' col^n

Select title from film

where length > 90

Order by title .

FROM → film

↓

WHERE → Select complete Row that has length > 90.

↓

SELECT → Select only 'title' from the complete Row.

↓

DISTINCT → Unique 'title'

↓

ORDER BY → I have the 'title' data, so I can sort it

table = [{}, {}, {}, ...]

ans1 = [ ]

FROM

for each row in table:

WHERE

if row matches cond^n:

ans1.add(row)

// select

ans2 = [ ]

for each row in ans1: (A)

ans2.add(row[title]);

// ans2 only has title's data.

DISTINCT

ans3 = [ ]

for each row in set(ans2):

ans3.add(row)

ans3.sort(Sorting condition)

sort by 'title'

print(ans3)

Note :

when you dont give distinct, SQL will store, both the columns of Select and ORDER By

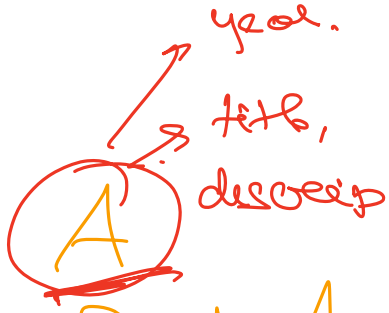
→ Store 'title' and 'release_year' in Select Step

→ When you do mention Distinct, It will only keep a track of SELECT

Only 'title'.

So Order By will fail.

Case.



1 Select
ORDER BY A ✓

2. Select A
ORDER BY B ✓

3 Select distinct A ✗
ORDER BY B

4) Select Distinct A, B, C, D ---
ORDER BY subset (A, B, C, D ...)

ORDER BY should have a subset of SELECT col^N.

LIMIT

fetche 1M rows

```
Select * from film.
```

I just wanted to check the columns.

we want to limit the output of a SQL query.

```
Select * from Film
where length > 90
ORDER By release-year
LIMIT 10;
```

↳ give me top 10 result of your complete answer.

I can skip the first X rows of the result using an OFFSET

```
Select * from Film
where length > 90
ORDER By release-year
LIMIT 10
OFFSET 20;
```

1 indexed.

→ gives me the rows numbered 21 to 30.

Q Query to fetch the data between 10th row and 50th row

All the rows, starting from the 10th to 50th row (both inclusive)

Limit 41 Offset 9

Students.

name	yob
Mohsin	90
Mohsin	98
Suraj	95

Select distinct name from Students
Order by yob.

Mohsin	90
Suraj	95
Mohsin	98

Mohsin (1)
Suraj
Mohsin (2)

Mohsin
Suraj

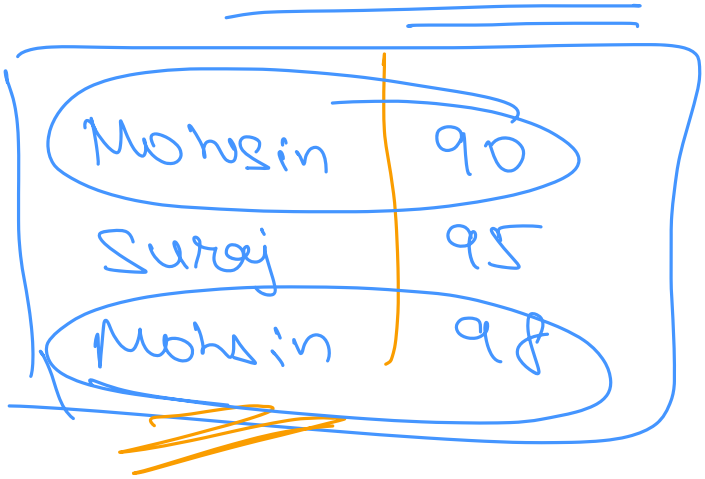
Suraj
Mohsin

Select distinct (name, yob) from Students
Order by yob.

FROM

WHERE

SELECT :



DISTINCT : Chck for uniqueness of the pairs.