CRUD - II

Agenda

1. - AND / OR / NOT
   - IN
   - BETWEEN
   - LIKE
   - IS NULL
   - ORDER BY
   - LIMIT

Tricky for read operations.

Tricky SQL Ques

RDBMS don't guarantee order. To guarantee it, we use this

2. UPDATE
3. DELETE

[QUIZ]

AND / OR / NOT

→ to specify a condition we use WHERE clause
↳ analogous to IF condition in a programming language

do we always have a single comparison in ifs?

```
if (a>2)
```

a > 2

or

b < 3

or

a + b > 5

```
if (a>2 || b>3 || a+b>5)
        print (yes)
else
        print (no)
```

AND (&&)
OR (||)
NOT (!)

```
if (! (a == 3)){

}
```

! true = false

SAKILA.

Q Get all movies that have a rating of PG 13 and were released in 2006

```
select * from film
    where rating = 'PG-13'
    AND release_year = 2006 ;
```

Q Get all movies that have a rating of PG 13 or were released in 2006

```
select * from film
    where rating = 'PG-13'
    OR release_year = 2006;
```

Q Get the movies with rating anything other than PG 13 and release any year other than 2006.

```
select * from film
    where (NOT rating = 'PG-13')
    AND (NOT release_year = 2006);
```

↕ THESE have a different meaning

```
select * from film
    where NOT (rating = 'PG-13'
    AND NOT release_year = 2006);
```

★ Always use parenthesis whenever combining multiple conditions.

⇒ By default (without paranthesis) SQL decides order in similar way like maths (eg. BODMAS)

!= ⇒ not equals in coding

!= OR <> ⇒ NOT EQUALS in SQL

```
select * from film
    where rating != 'PG-13'
    AND release_year != 2006;
```

```
select * from film
    where rating <> 'PG-13'
    AND release_year <> 2006;
```

Both are same

[CODE]

→ release_year = 2006

→ != 2006

→ NOT

→ rating → PG 13 & RY = 2006

→ COMPARISON OPERATORS

CODE		SQL
==	vs	=
<	v/s	<
>	vs	>
<=	v/s	<=
>=	v/s	>=
!=	vs	(!= OR <>)

IN OPERATOR

Q get all the students of b_id 1, 5, 7 OR 9.

```
select * from student
    where  b_id = 1
        or  b_id = 5
        or  b_id = 7
        or  b_id = 9 ;
```

when you want to compare value of one column against multiple values

↓

(IN)

```
select * from students
    where b_id  IN ( 1,5,7,9) ;
```

[CODE] → movies with rating PG 13 or R

① (with & without IN) → movies with rating neither PG 13 or R.

② !(a==b) NOT(rating in ( ));

③ (a!=b) rating not in ( );

BETWEEN

Q movies released b/w 2005 - 2010
students with psp from 50 - 70

psp >= 50 AND Psp <= 70

Q PSP (50-70) or (80-85)

```
Select * from students
    where ( psp >= 50 AND psp <= 70)
      OR ( psp >= 80 AND psp <= 85);
```

>= x and <= y

BETWEEN (x , y)

```
Select * from students
    where ( psp between 50 AND 70)
      or ( psp between 80 AND 85) ;
```

[CODE] → movies from 2006 and 2008.

LIKE OPERATOR

batches

id	name
	Academy Apr 22 Beg Java
	May Academy Python

Coln with strings
⇓
pattern matching on that coln

- Every academy batch should have 'academy' in its name
- Every beg batch should have 'beg' in its name
- every morning batch should have 'morning' in its name.

Apr 22 Beg Morning academy [ANY ORDER]

Q Return all morning batches in academy.

```
boolean checkAcadMorning ( String batchName) {
        if ( batchName . contains( moring) &&
                batchName . contains (academy))
            {
                return true ;
            }
      return false;
}
```

* LIKE operator → when you have to check inside a string

2 wild card symbols:

① '_' ⇒ exactly one occurance of any character

② '%' ⇒ any no. of occurances of any character (0, 1, 8, ...)

Given string | Pattern

cat

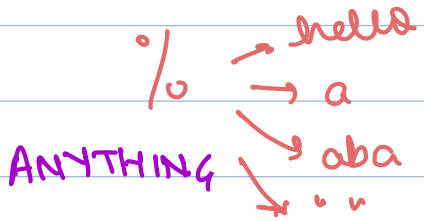
① _ _ t ✓
(c a)

② % t ✓
(ca)

③ % ✓

④ cat ✓

⑤ % cat % ✓



_ → fill in the blank

Q To check if hello exists in rating column

↓

where rating like '% hello %' ;

Q for a batch, check if a ~~academy~~ batch running is ~~morning~~.

name like '% academy % morning %' ✗

'Morning Apr 22 Beg Java Academy'

select * from batches
where name like '% academy %'

and name like '%morning%';

[UNDERSTAND WHAT QUERY OR PATTERN MEANS AS A WHOLE]

[CODE] → films with 'love' in it

[QUIZ]

[BREAK]

IS NULL operator

you can't use = operator to check NULL.

film

id	name	description → can be null

get the films where description = NULL

```
select * from films
where description = NULL
```

X

nothing = NULL

NULL = NULL ✗

★ Cant compare two things that dont exist

[CODE]

① null is not equal to null
② null is not not equal to null.
③ anything is not equal to null
④ anything is not not equal to null.

IS NULL

IS NOT NULL

Q

Batches

id	name	b_id
1	ujj	1
2	amar	2
3	shiv	null

Get all batches that dont have b_id = 2

```
select * from batches
where b_id != 2 ;
```

✗

```
select * from batches
    where b_id != 2
    and b_id IS NULL;
```

ORDER BY

→ by default order of rows in the ans is not guaranteed

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

```
select * from film
    order by title;
```

⇒ By default rows are sorted in ascending order.

[CODE]

→ desc

order by 2 columns.

↓

```
select * from film
order by release_year, title
```

tie breaker

students

yob	name
1998	C
1999	D
1997	c
1998	A

year, name
1997, c
1998, a
1998, c
1999, d

Q order by year in desc order but titles are in alphabetical order (ascending)

```
order by yob desc, title
```

1999, D
1998, A
1998, C
1997, C

```
order by yob desc, title desc
```

↳ 1999, D
1998, C
1998, A
1997, C

★ In SQL default tie breaker is primary key

★ By default, MySQL gives ans ordered by ascending order of primary key.

Code for order by:

```
table_name = [ { } , { } ..... ]
ans = [ ]

for row in table_name:
    if condition is matched:
        ans.add ( row)

ans. sort ( sorting condition)

for row in ans:
    print (ans [title])
```

we can even sort on columns that we want print

[code]

★ If you have a distinct in select clause, you can only order by coln in the select clause

student

id	name	job
	Abhi	1995
	Abhi	1990
	Bhim	1992

```
select distinct name
from student
order by job.
```

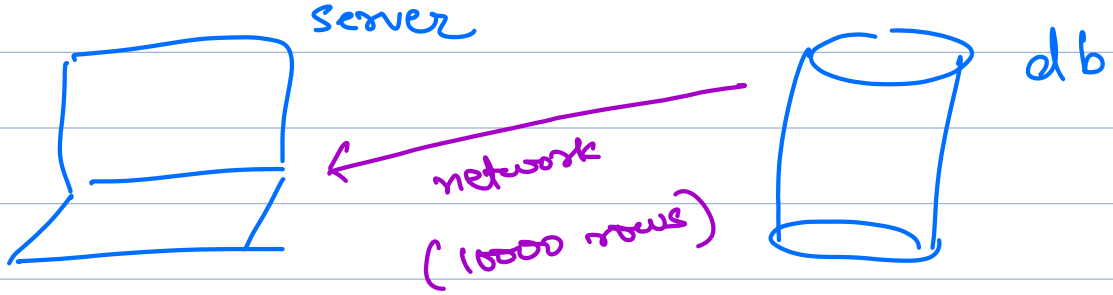
✗

[CODE]

* Can only order by cols present in select statement → in case of DISTINCT.

LIMIT

select * from film; ⇒ 10000 records



SLOWWWW

* limit allows to return only a few of the results that are there from the query.

```
select * from film LIMIT 10;
```

⇒ clause put at end of the query.
LIMITS no. of rows that are returned

[CODE]

```
Select * from film
    Limit 10 offset 10;
```

↓

remove the first 10 result rows, and get the next 10 results.