CRUD - 2

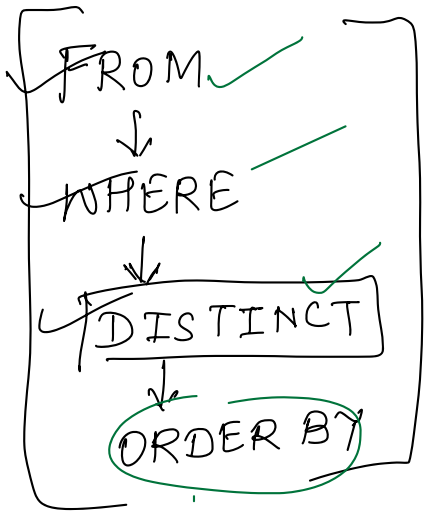
TABLE OF CONTENTS

Order By Clause with Distinct keyword

- When employing the DISTINCT keyword in an SQL query, the ORDER BY clause is limited to sorting by columns explicitly specified in the SELECT clause.

The given query won't work

```
SELECT DISTINCT title FROM film ORDER BY release_year;
```



Example

Imagine a library filled with books of the same title but published in different years. Which year should come first? Which one last? Which book to remove which one to keep? The SQL engine couldn't decide without explicit guidance.

Keyword

rating ∈ [2.5, 5]

>= AND <=

Between

inclusive of range specified

- Between Clause helps us to get values in a specific range.

release year >= 2008

release year <= 2020

Select * From film
Where release_year
Between 2008 AND 2020;

Students

id	first_name	second_name	psp
1	Virat	Kohli	80
2	Rahul	KL	75
3	Rohit	Sharma	95
4	Rahul	KL	80

Range id-1 to id-3

(2008, 2020) → 2008 & 2020 excluded

>2008 AND <2020

[2008, 2020] → " included

>=2008 AND <=2020

Question : Get all the movies which were released between year 2006 and 2016. Both inclusive.

< / > Syntax

```
SELECT *

FROM table

WHERE col_value between A and B
```

- Don't put brackets in **Between's** range of values as shown below

Ex : WHERE col_value between (A and B)

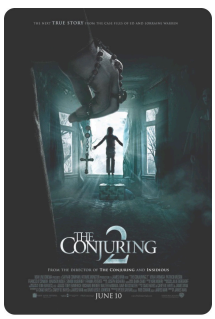
```
SELECT  *
FROM film
WHERE release_year between 2006 and 2016
```

- PK acts as a tie breaker when there is a tie.

Like

"Scaler"

- Which of these are action movies?



Genre: Horror
Language: English

Genre: Action
Language: English

Genre: Horror
Language: English

Genre: Action
Language: English

- Whenever there is a column storing strings, theres comes a requirement to do some kind of pattern matching.

Ex : Assume Scaler's databases where we have following rules to store batch name :

1. Every batch name should have 'Academy' / 'DSML' in them.
2. It should have 'Beg' / 'Inter' / 'Adv' in the naming convention.
3. It should have 'Morn' / 'Eve'.
4. It should have month of the batch.

4 types of info.

(not necessarily in the same order)

[Morn_Inter_Academy_Jan]
[Adv_Eve_Feb_DSML]

Batches

b_id	b_name
1	'Acad_Apr_23_Morn_Beg'
2	'June_23_DSML_Inter_Eve'
3	'Mar_23_Acad_Adv_Eve'

Question : Get all the morning beginner batches.

substring

Pattern Matching

Anant

' How to get these batches? '

' We can do pattern matching using LIKE operator ...'

We have two wildcards in Like operator :

- Consider them like fill in the blanks :

1. ' _ ' : Can have exactly one occurrence of a single character.
2. ' % ' : Can have any number of character or it can stay empty as >=0

Question : Find all the beginner batches.

How to use wildcards?

1. **% beg %** : Anything can come before beginner and after beginner.
   Anywhere in the string
2. **%beg** : Anything can come before beginner. Beg as suffix.
3. **beg%** : Anything can come after beginner. Beg as prefix.

Let's do some pattern matching.

String

Cat

Pattern

_ _ t ✓

% t ✓

% ✓

% Cat % ✓

Cat ✗

Question : Get data of all the morning batches.

```
SELECT  *
FROM batches
WHERE b_name like  ' % Morning %' ;
```

underscore means exactly 1 ✓

Quiz - 1

1. SELECT * FROM Customers WHERE Name LIKE '**son%**'
2. SELECT * FROM Customers WHERE Name LIKE '**%son**'
3. SELECT * FROM Customers WHERE Name LIKE '**son**'
4. SELECT * FROM Customers WHERE Name LIKE '**son**'

Quiz - 2

1. SELECT * FROM Customers WHERE Name LIKE '**moon%**'
2. SELECT * FROM Customers WHERE Name LIKE '**%moon**'
3. SELECT * FROM Customers WHERE Name LIKE '**moon%**'
4. SELECT * FROM Customers WHERE Name LIKE '**moon**'

Quiz - 3

Exactly 5 lettered, middle → '123'

1. SELECT * FROM Orders WHERE OrderNumber LIKE '**%123%**' → '__123_'
2. SELECT * FROM Orders WHERE OrderNumber LIKE '**123%**'
3. SELECT * FROM Orders WHERE OrderNumber LIKE '_**123**_' → '_123_'
4. SELECT * FROM Orders WHERE OrderNumber LIKE '**%123**'

11:00 pm

will work → Select Distinct * from film

Select * from film

Same

NOT status

name is 'Ankit'

NULL (is null, is not null)

- Do you all remember how we store empties, no value for a particular column for a particular row? We store it as **NULL**, regardless of column's datatype.

- Interestingly working with nulls is bit tricky. Let's see how.

Students table

s_id	s_name	status
1	A	1
2	B	Null
3	C	0
4	D	1

Question : Get all the students whose status is NULL

Let's try using the following query

```
SELECT * FROM student WHERE status = NULL;
```

- The above query will not return any rows. **WHY?** Because NULL is not equal to NULL. in fact NULL is not equal to anything. Nor is it not equal to anything. It is just NULL.

Try this query and find the output

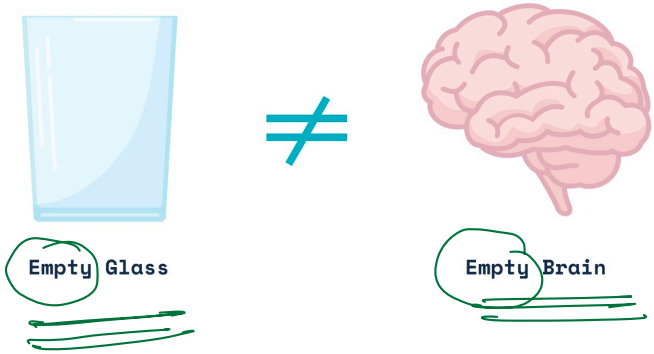
```
SELECT NULL = NULL
```

o/p → NULL

- We can't compare NULL with anything like we can't compare an Empty Glass with an Empty Brain.

Zero → value

null → does not exist



- The right operator is **IS NULL**

```
SELECT * FROM student WHERE status IS NULL;
```

- Similarly we don't use not equal to **NULL**, rather we use **IS NOT NULL**, when we want not NULL values.

LIMIT Clause

always by default starts from 1st row

~~Select *~~

- LIMIT clause allows us to limit the number of rows returned by a query.
- Suppose we have this query :

```
SELECT * FROM table ;
```

all rows

' What if a table has 1-million rows? '

table[0], table[1]

table = [[ ], [ ], [ ]]

row

array of rows

How to get top two rows :

```
SELECT * FROM table LIMIT 2 ;
```

by default choose order as in DB

FROM → orderby → offset → Limit

always same

always same

Students Table :

s_id	s_name	name
1	James Jones	1
2	John Miller	2
3	John Martinez	3
4	Michael Garcia	4
5	Jennifer Miller	5

Order by is optional

order by

```
Select * from film
LIMIT 10 offset 100
```

- There's one more thing that we can do : **OFFSET**

OFFSET : 10 row after 100th row → skip 1st 100 rows, give 10 rows thereafter

row nos. 101–110

offset

Update

'U' of CRUD → Update rows in a table

< / > Syntax

```
UPDATE table_name SET col_name = value
WHERE conditions ;
```

WHERE conditions ; → optional (if not given)

General syntax

'I have added the wrong first_name in the table. Can you update it?'

```
Update film SET release_year = 2006, rating = 'PG' where id = 1;
```

```
for each row in film:
    if row.matches (conditions in where clause)
        row['release_year'] = 2006
        row['rating'] = 'PG'
```

- Let's update first name of the student at s_id - 3 to Anshu

Students

id	first_name	last_name	psp
1	Virat	Kohli	80
2	Rahul	KL	75
3	Virat	Sharma	95
4	Rahul	KL	80

Update to Rohit at id - 3

```
Update Students
SET
first_name = 'Anshu'
where
id = 3;
```

optional

Delete, Truncate, Drop

Delete

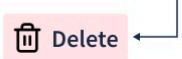
- Removes specified rows one-by-one from table (may delete all rows if no condition is present in query but keeps table structure intact).

< / > Syntax

```
DELETE FROM table_name
WHERE conditions ;
```

Students

id	first_name	last_name	psp
1	Rahul	KL	80
2	Virat	Kohli	75
3	Rahul	KL	95
4	Katrina	Sharma	80



Truncate

- Removes the complete table and then recreates it.

Query : TRUNCATE table_name;

Students

id	first_name	last_name	psp
1	Rahul	KL	80
2	Virat	Kohli	75
3	Rahul	KL	95
4	Rohit	Sharma	80

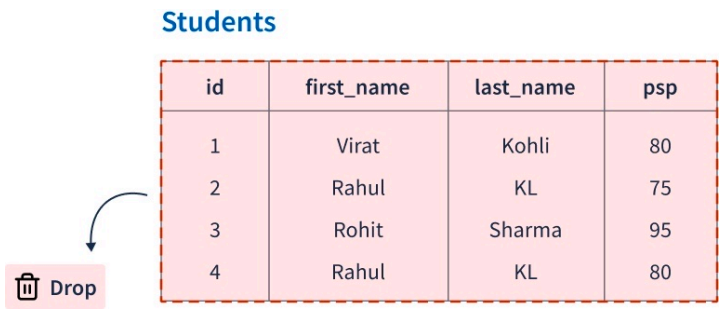
Students

id	first_name	last_name	psp

Drop

- Removes complete table and the table structure as well.

Query : DROP table_name;



Delete vs Truncate vs Drop

Delete :

1. It is slower than TRUNCATE.
2. Doesn't reset the key.
3. It can be rolled back.

Truncate :

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

Drop :

1. It can not be rolled back.

Property	Delete	Truncate	Drop
Schema Preserved	✓	✓	✗
Efficiency	✗	✓	✓
Reversible	✓	✗	✗