Lec-9

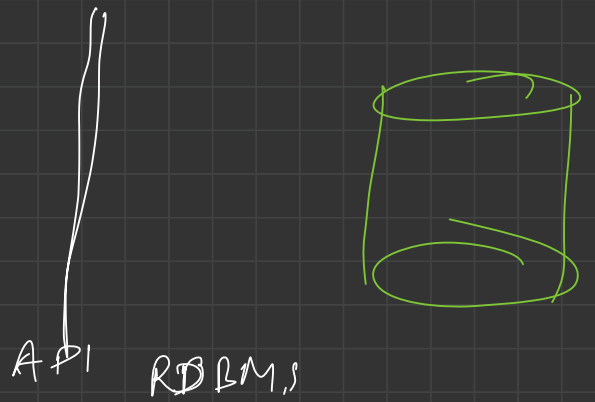
* SQL in One video

SQL => structured Query language

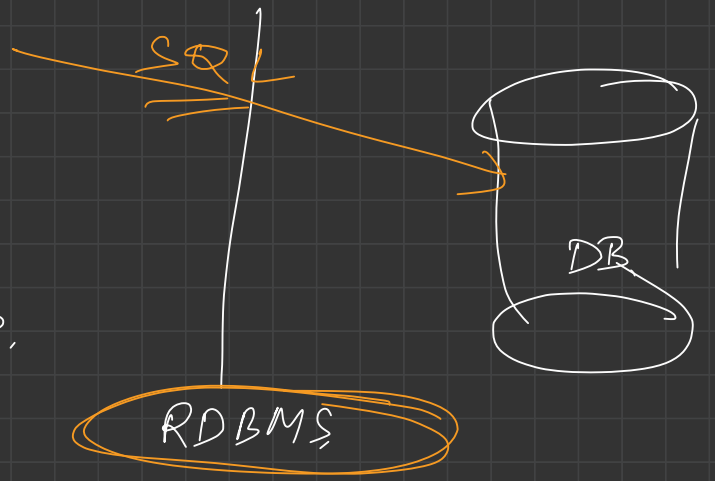
RDBMS => MySQL, Oracle, MS access etc.

CRUD

SQL queries.



CRUD → Create
Read
Update
Delete.



RDBMS → ① MySQL → Open source RDBMS

② MS SQL server → Meta, Adobe etc.

③ Oracle

④ IBM.

SQL	MYSQL
① Query Language.	① MySQL itself a RDBMS
② Way to access data	② CRUD done on it using SQL.

⇒ Install → ① MySQL Community Server
② MySQL Workbench

2) Workbench → DB create
Table

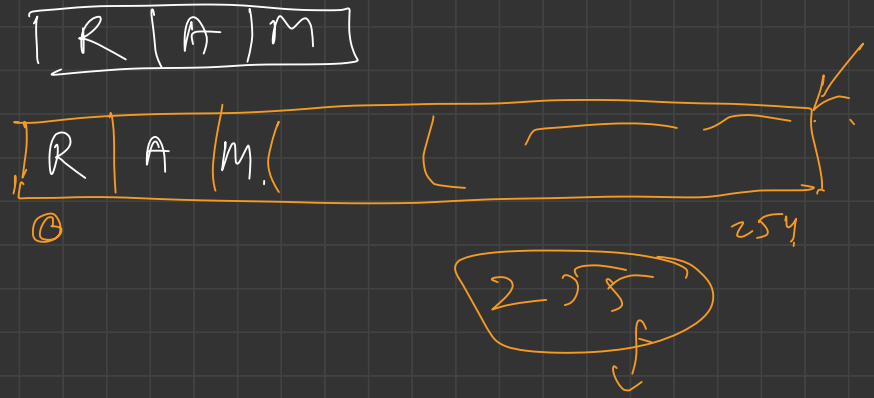
* SQL Data types

```
Create table student(
    id INT PRIMARY KEY,
    name Varchar(255)
);
```

Variable datatype

VARCHAR (255) →

CHAR (255) →



Fixed datatype

SIGNED , Unsigned

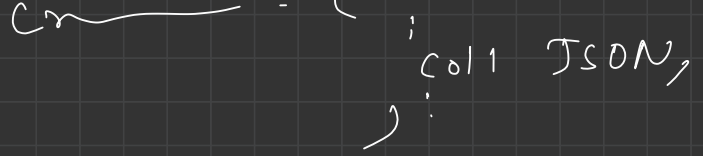
TINYINT ⇒ (−128 to 127)

UNSIGNED TINYINT ⇒ (0 → 255)

```
Create TABLE table_name (
        Col1 INT,
        Col2 INT UNSIGNED
);
```

⇒ Advanced DT |—

① JSON → JS object Notation



SQL Types of Commands

① DDL

⇒ DQL ⇒ ① DB

② table

User defined tables — Student

Dual table

↳ without using FROM

SELECT

Wildcardy

=>

'%pa%'

'abcpadb'

'pabc'

'adpa'

'_pa_'

apab

bpac

Sorting

```
select * from worker ORDER BY salary;
```

ASC (Default)

DESC (↓)

* Distinct values

— Department (distinct)

HR
HR
Admin
Admin.
Account

→

HR
Admin
Account

* Data Grouping

→ find no. of employees working in different depart

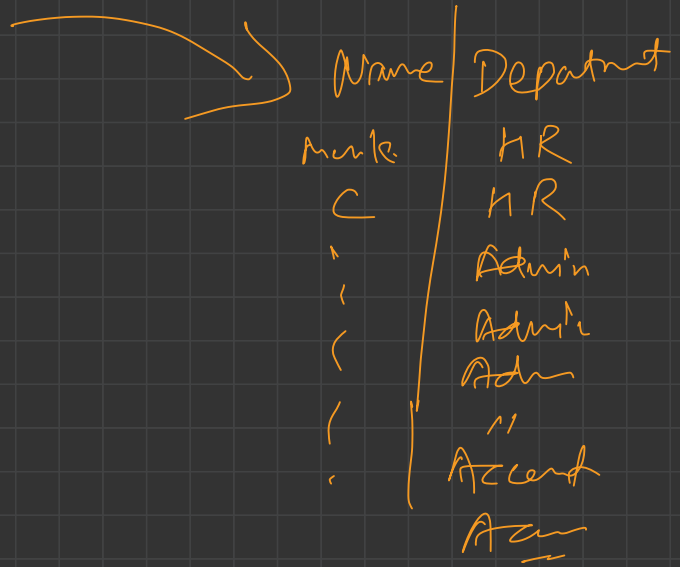
→ HR → ?
Account → ?
Admin → ?

→ Grouping → Aggregation (count)

→ GROUP BY → Aggregation fn.
COUNT, SUM
AVG, MIN, MAX.

Worker

Name	Department
Monika	HR
Nimi	Admin
C	HR
D	Admin
E	Admin
F	Account
G	Account
H	Admin



Q find avg. Salary per department?

* HAVING

→ Select where to filter

"GROUP BY" & "HAVING"

2) Q department, count, having More than 2 workers?

* WHERE VS HAVING

* DDL Constraints

① Primary Key Constraint

- Not Null
- Unique
- Only one P.K

Good practice

→ P.K INT

② Foreign Key

↳ FK refers P.K of other table.

Workbench

③ Unique

```
Create table Custme (
    ;
    Name VARCHAR(255) UNIQUE,
);
```

④ CHECK

```
create table account (
    :
    CONSTRAINT acc_balance_chk CHECK (balance >=100),
);
```

⑨ DEFAULT

```
Create table account (
    !
    balance INT NOT NULL Default 0,
    :
);
```

* DML Data Modification Language:

* Referential Constraint:

① INSERT

② Delete Constraint

→ ON Delete Cascade

→ ON Delete Set Null:

Replace :- ① Data already present, Replace.

② Data not present, INSERT

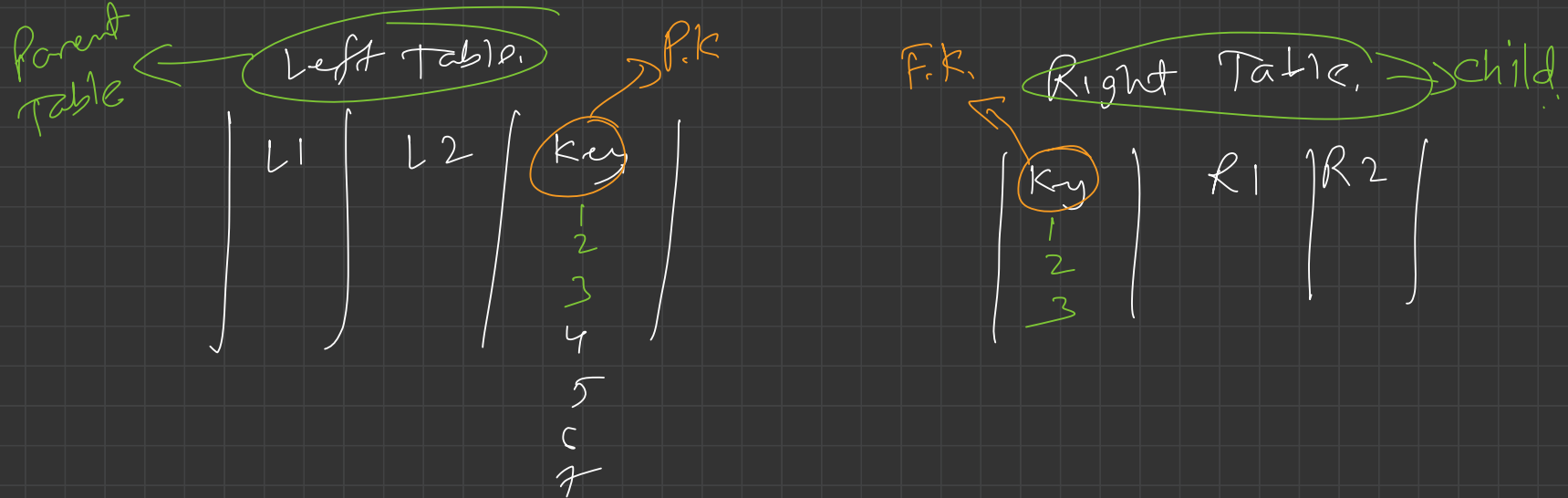
Replace Vs Update

→ if row is not present, Replace will add a new row while UPDATE will do Nothing

JOINS

2) F.K → Relations

→ we → Data fetch → JOINS



① INNER JOIN

Resultant table

Key	L1	L2	R1	R2
1				
2				
3				

- Join → To apply joins, there shold be a common attribute.

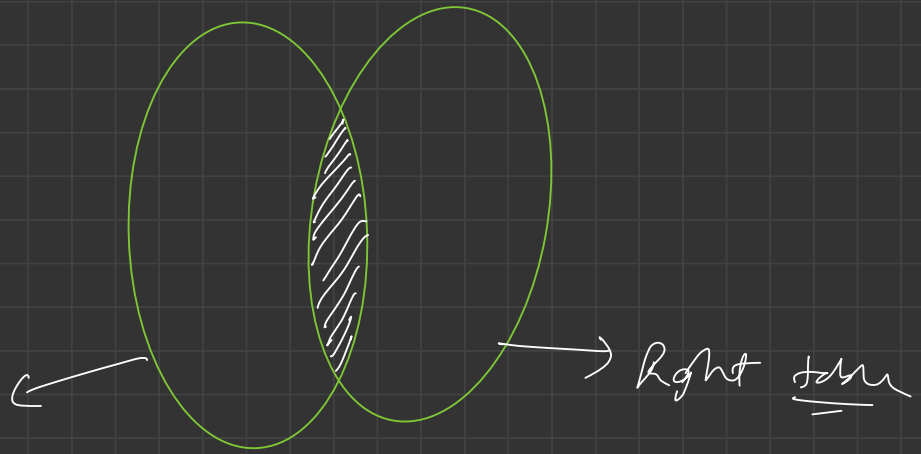
Syntax:

```
Select C.*, O.* FROM Customer AS C INNER JOIN Orders AS O
ON C.id = O.cust_id;
```

C.* → C.id, C.name.

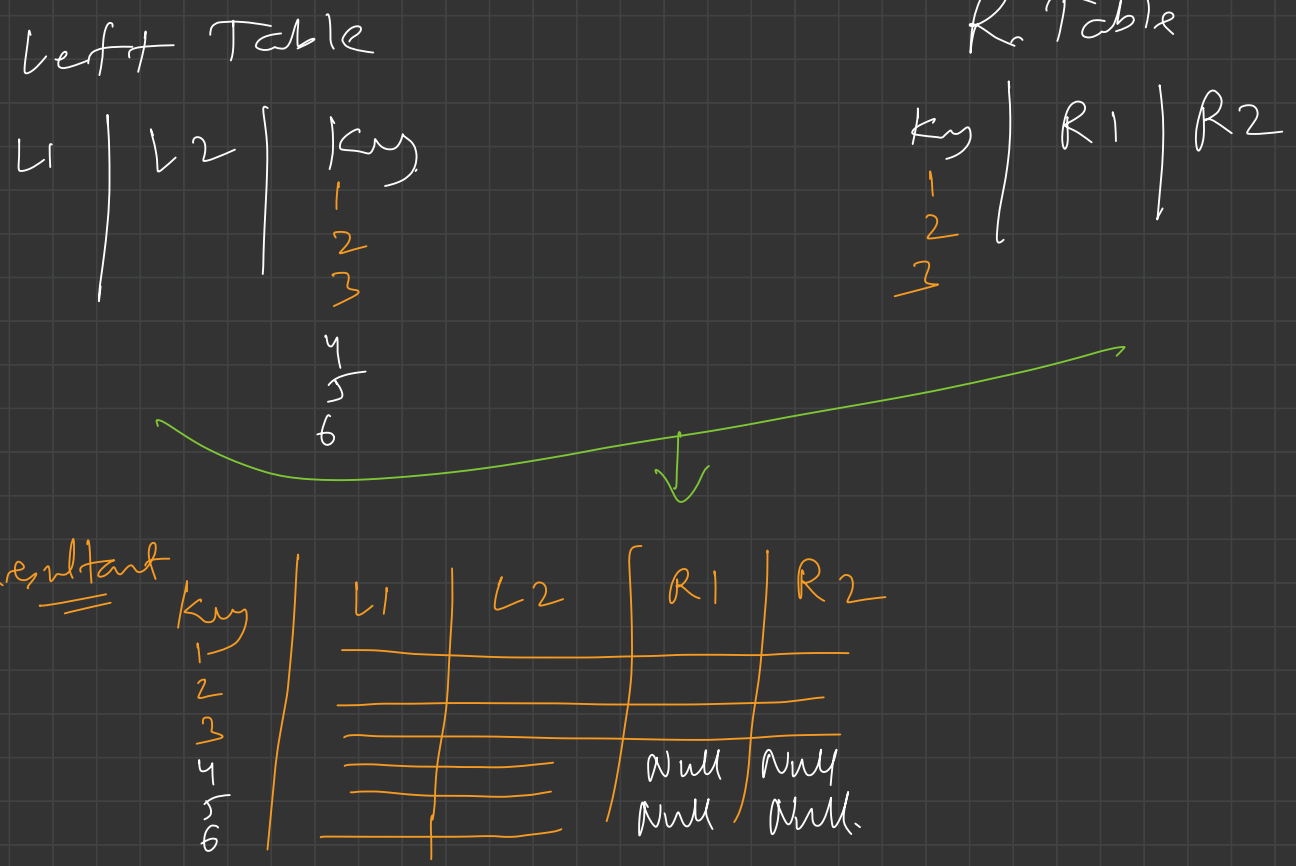
O.* → O.productm

aliasing



INNER JOIN

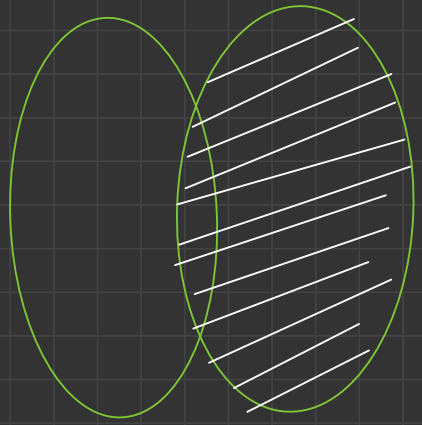
⑦ Left Join

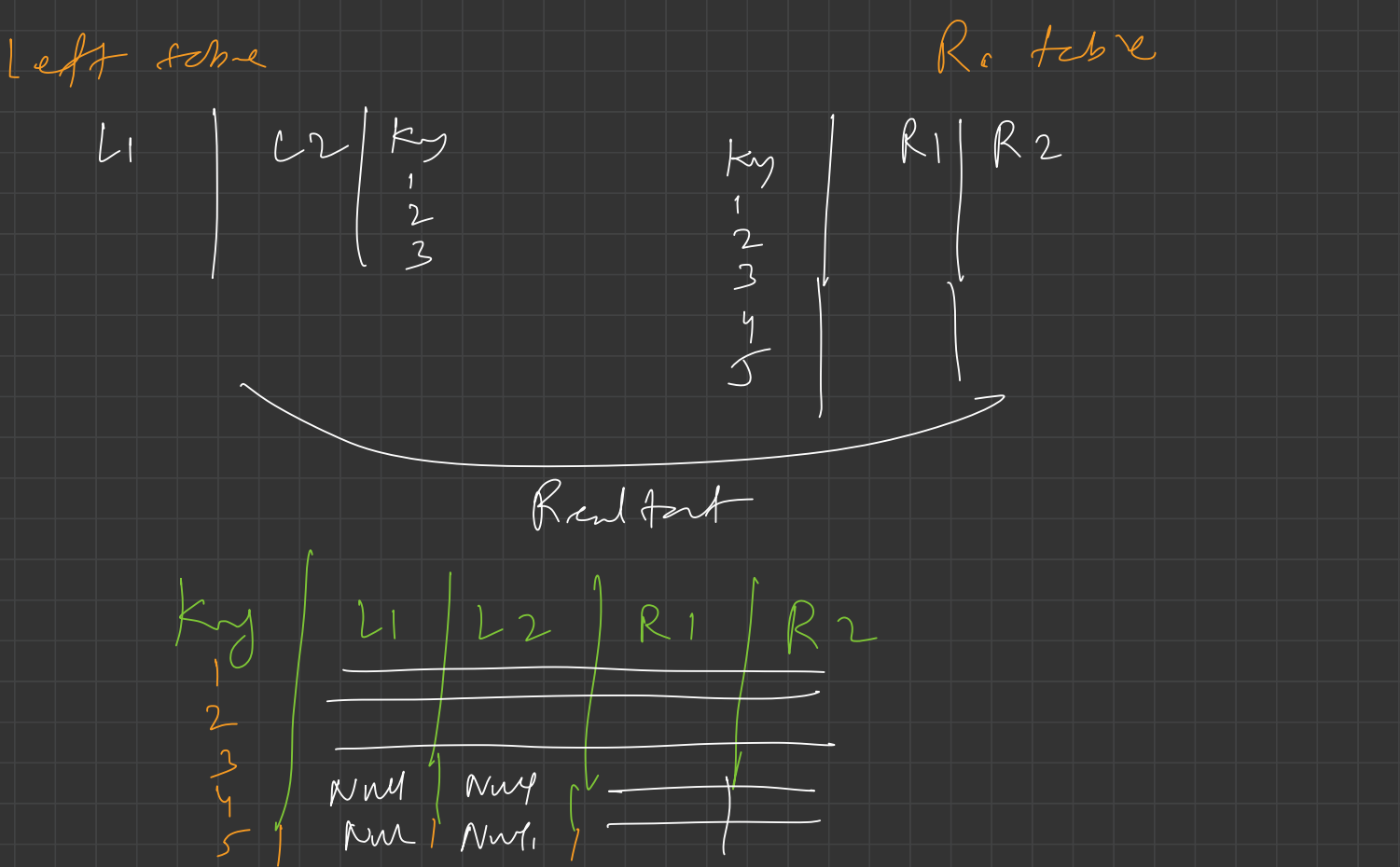

Left Table
L1
L2
Key
1
2
3
4
5
6
R. Table
Key
R1
R2
1
2
3
Resultant
Key
L1
L2
R1
R2
1
2
3
4
5
6
Null
Null
Null
Null

```
select c.*, o.* FROM Customer as c LEFT JOIN
    Orders as o
ON c.id = o.cust_id;
```

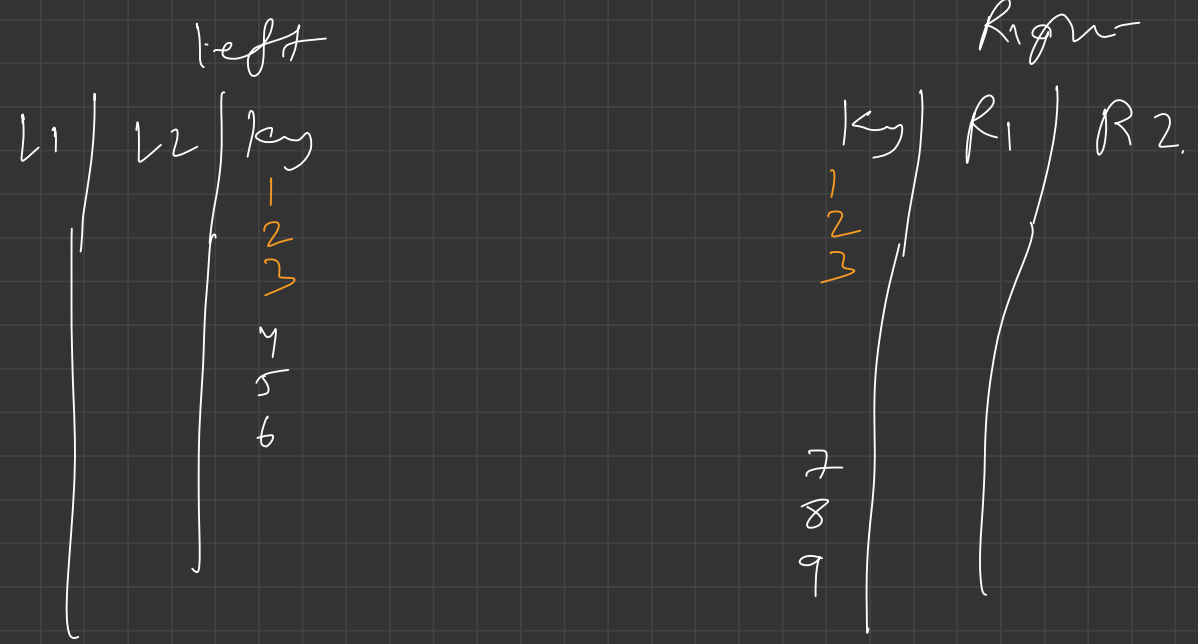
③ Right Join

— Intented in knowing only the right side info.




Left table
Right table
L1
L2
Key
1
2
3
Key
1
2
3
4
5
R1
R2
Result
Key
1
2
3
4
5
L1
L2
R1
R2
Null
Null
Null
Null

⑨ Full Join



MySQL Full Join X keyword X.

Emulated

Left Join ∪ Right Join

Syntax

```
select * from leftTable as l LEFT JOIN RightTable as r
ON l.key = R.key.

UNION

select * from leftTable as l RIGHT JOIN RightTable as r
ON l.key = R.key;
```

6 Cross Join

→ Cartesian product

⇒ T_L — 5 rows

T_R — 10 rows

Result :- $5 \times 10 = 50$ rows

eg.

L_1	L_2
1	A
2	B

R_1	R_2
3	C
4	D

2) Result

→ Industrial use X.

L_1	L_2	R_1	R_2
1	A	3	C
1	A	4	D
2	B	3	C
2	B	4	D

⑤ Self Join

- emlate.
- 'INNER JOIN'
- Alias 'AS'

eg. select e1.id, e2.id, e2.name
FROM employee as e1
INNER JOIN employee as e2
ON e1.id = e2.id;

→ Workbench server

* Project

id	empID	name	startdate	clientID
1	1	A	2021-04-21	3
2	2	B	2021-03-12	1
3	3	C	2021-01-16	5
4	3	D	2021-04-27	2
5	5	E	2021-05-01	4
NULL	NULL	NULL	NULL	NULL

* EMPLOYEE

id	fname	lname	Age	emailID	PhoneNo	City
1	Aman	Proto	32	aman@gmail.com	898	Delhi
2	Yagya	Narayan	44	yagya@gmail.com	222	Palam
3	Rahul	BD	22	rahul@gmail.com	444	Kolkata
4	Jatin	Hermit	31	jatin@gmail.com	666	Raipur
5	PK	Pandey	21	pk@gmail.com	555	Jaipur
NULL	NULL	NULL	NULL	NULL	NULL	NULL

* CLIENT

id	first_name	last_name	age	emailID	PhoneNo	City	empID
1	Mac	Rogers	47	mac@hotmail.com	333	Kolkata	3
2	Max	Poirier	27	max@gmail.com	222	Kolkata	3
3	Peter	Jain	24	peter@abc.com	111	Delhi	1
4	Sushant	Aggarwal	23	sushant@yahoo.com	45454	Hyderabad	5
5	Pratap	Singh	36	p@xyz.com	77767	Mumbai	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Q can we ~~use~~ JOIN w/o using JOIN keyword?

=> Yes

Syntax

```
select * FROM leftTable, RightTable WHERE
                                    leftTable.id
                                    = RightTable.id;
```

* Set Operations

{ 1, 2, 3, 4, 5, 6, 7, 8 }

Table 1

col1	col2
A	1
B	1
C	2

Table 2

col1	col2
A	1
B	2
D	3

1. Union
2. Intersection
3. MINUS

→ UNION

$T_1 \cup T_2$

$T_1 \cup T_2 \rightarrow$

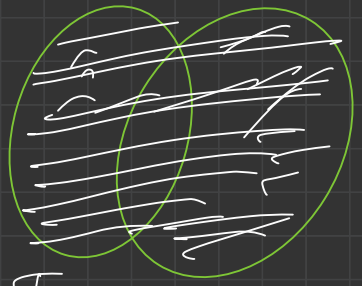
col1	col2
A	1
B	1
C	2
B	2
D	3

— ROWS.

Syntax :→

```
Select * FROM Table1
UNION
Select * FROM Table2;
```

Set operation
→ Combine two or more select statements

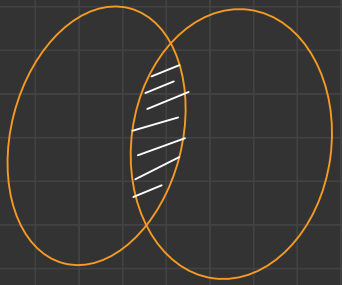


JOIN vs UNION

FULL JOIN :-

Col 1	Col 2	Col 1	Col 2
A	1	A	1
B	1	B	2
C	2	Null	Null
Null	Null	D	3

② INTERSECT



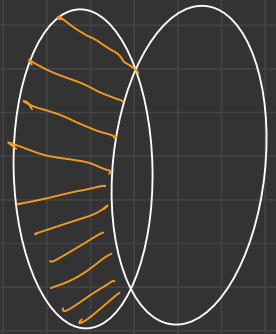
```
Select * from T1
INTERSECT
Select * from T2;
```

→ Emulate

```
select DISTINCT id from T1
INNER JOIN T2
using (id);
```

(3) MINUS

$T_1 - T_2$



Syntax:-> Select id FROM T_1
LEFT JOIN T_2 using(id)
WHERE T_2.id IS NULL;

Example

ABC

Dept 1

	empid	name	role
▶	1	A	engineer
	2	B	salesman
	3	C	manager
	4	D	salesman
	5	E	engineer
	NULL	NULL	NULL

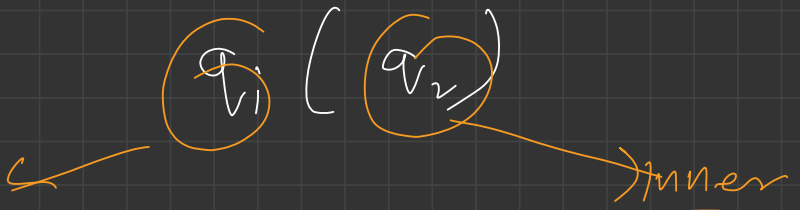
Dept 2.

	empid	name	role
▶	3	C	manager
	6	F	marketing
	7	G	salesman
	NULL	NULL	NULL

Sub Queries

- alternative to joins

-



Generally, outer query depends on inner query

eg. `Select * from table where id IN ( select id from table where name = 'cat' );`

→ Workbench server

* Project

id	empID	name	startdate	clientID
1	1	A	2021-04-21	3
2	2	B	2021-03-12	1
3	3	C	2021-01-16	5
4	3	D	2021-04-27	2
5	5	E	2021-05-01	4
NULL	NULL	NULL	NULL	NULL

* EMPLOYEE

id	fname	lname	Age	emailID	PhoneNo	City
1	Aman	Proto	32	aman@gmail.com	898	Delhi
2	Yagya	Narayan	44	yagya@gmail.com	222	Palam
3	Rahul	BD	22	rahul@gmail.com	444	Kolkata
4	Jatin	Hermit	31	jatin@gmail.com	666	Raipur
5	PK	Pandey	21	pk@gmail.com	555	Jaipur
NULL	NULL	NULL	NULL	NULL	NULL	NULL

* CLIENT

id	first_name	last_name	age	emailID	PhoneNo	City	empID
1	Mac	Rogers	47	mac@hotmail.com	333	Kolkata	3
2	Max	Poirier	27	max@gmail.com	222	Kolkata	3
3	Peter	Jain	24	peter@abc.com	111	Delhi	1
4	Sushant	Aggarwal	23	sushant@yahoo.com	45454	Hyderabad	5
5	Pratap	Singh	36	p@xyz.com	77767	Mumbai	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

* Corelated Subquery

Outer (Inner)

- inner query that refers the outer query

```sql
-- Corelated subquery
-- find 3rd oldest employee
SELECT *
FROM Employee e1
WHERE 3 = (
  SELECT COUNT(e2.age)
  FROM Employee e2
  WHERE e2.age >= e1.age
);
```

for each e1.age inner → completely one time

① e1.age = 32

inner ⇒ 2

② e1.age = 44 ⇒ ∅ 1

③ 22 ⇒ 4

32.
44.
22
31.
21

④ e1.age. 31 ⇒ 3

SQL Views

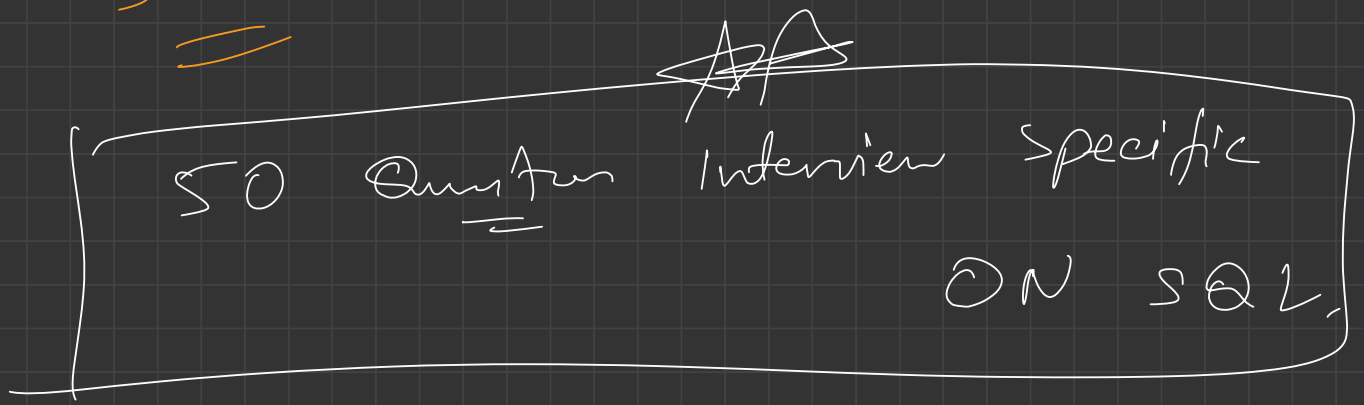
⇒ MySQL → views

Customer ⇒ id | name | age | address |

view ⇒ name | age → alias

— custom_view

2) SQL

2) 

Home work

1. Note making
2. Practice