

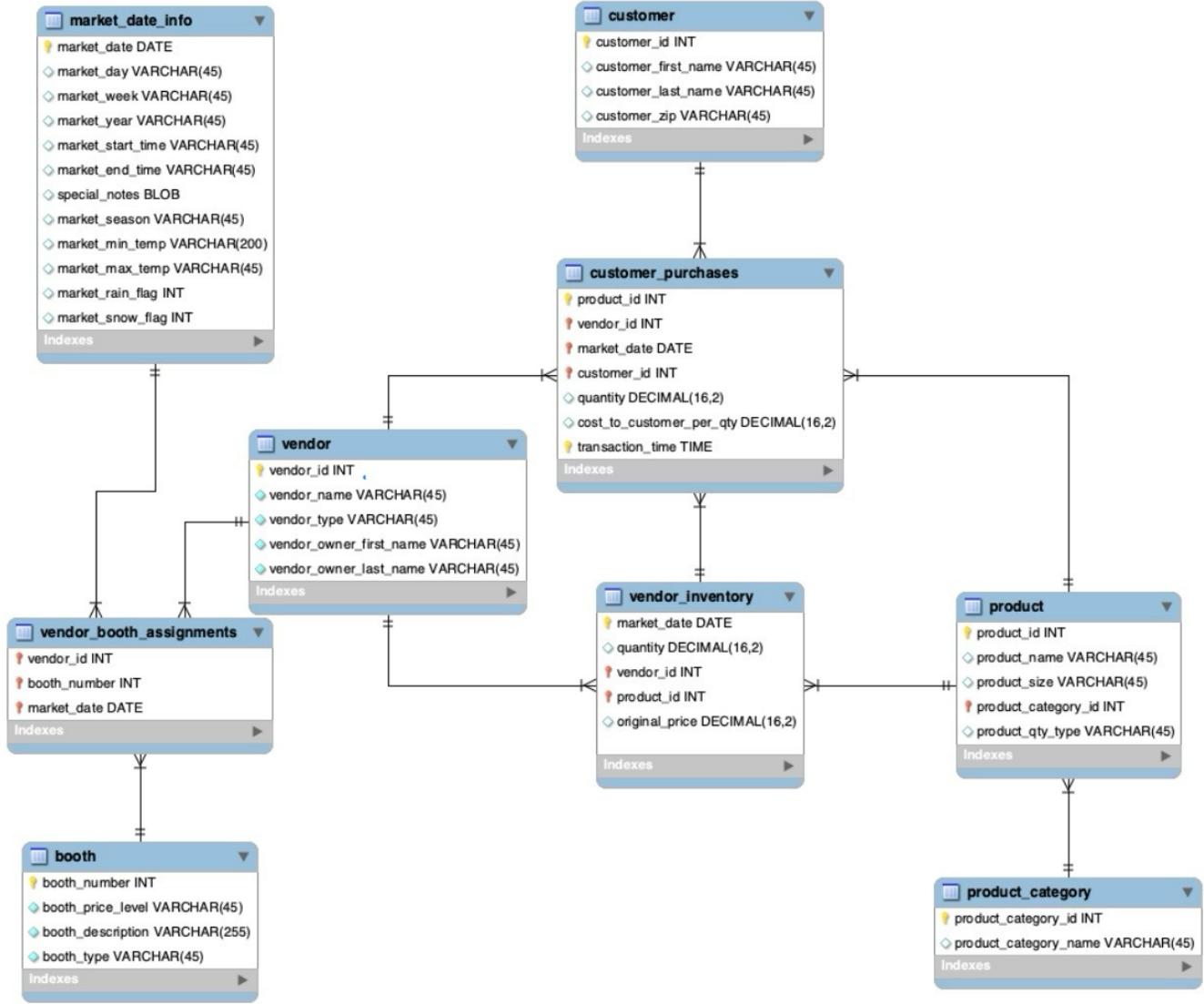
Agenda

Full Join

Group by

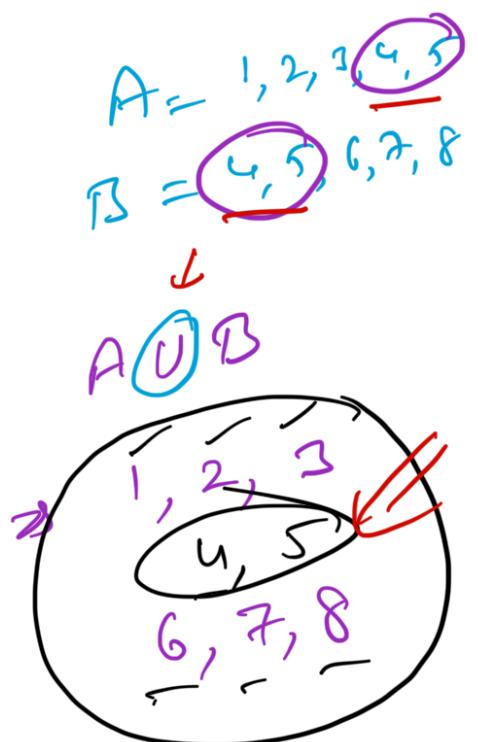
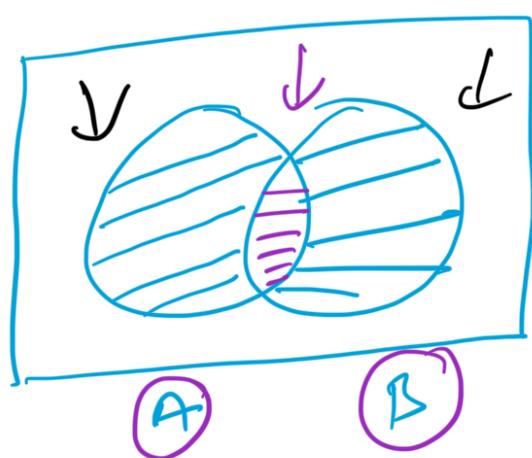
Having clause

Having vs Where Clause





Full joins / Full outer joins / outer joins

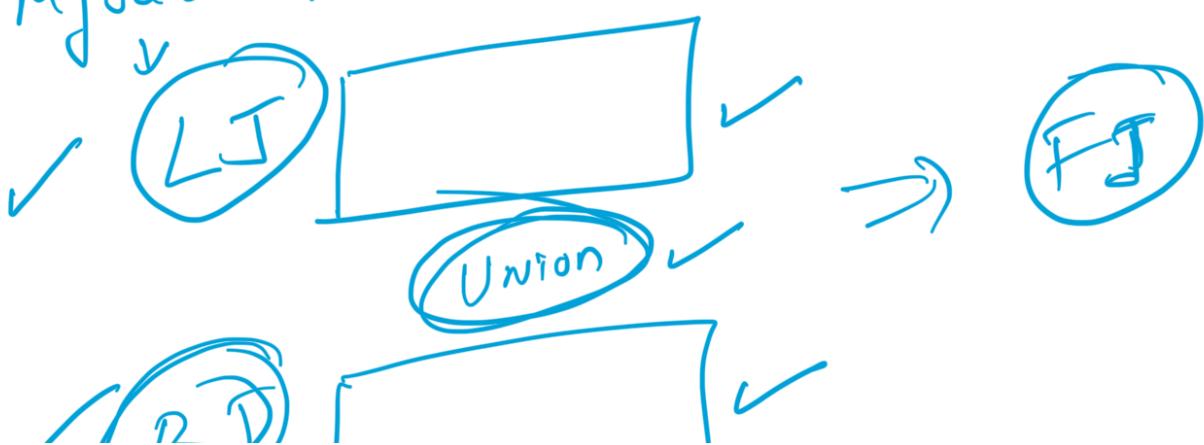


Syntax:

```

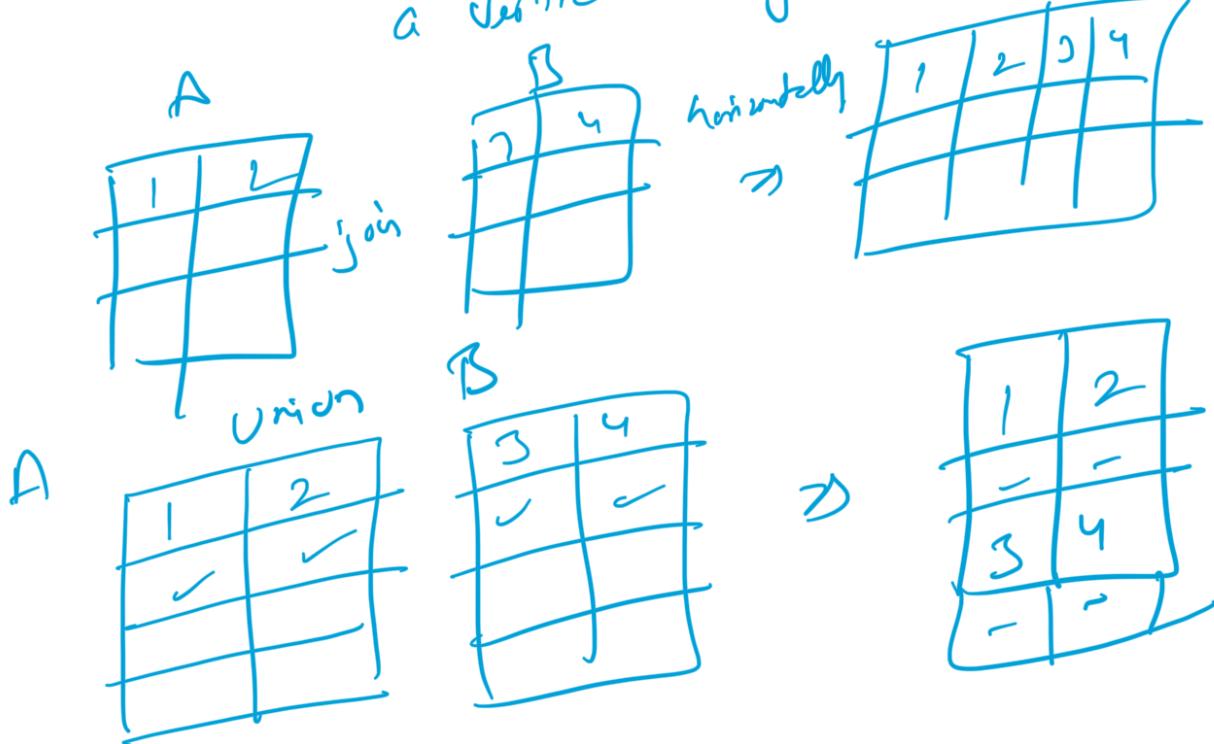
    Select * 
    From A a
    * full join | full outer join B b
    ON a.id = b.id ;
  
```

MySQL full join ↗



✓ Union

Union \cup is used to combine the result of two or more select statement in a vertical way.



Union rules

- ① Every select statement within UNION must have same no. of columns.
- ② The columns must have same/similar datatypes.

Union

$$A = 1, 2, 3, 4, 5 \\ \dots 0$$

Union All

$$A = 1, 2, 3, 4, 5 \\ B = 4, 5, 6, 7, 8$$

$$\beta = \{4, 5, 6, 7, 8\}$$

$A \cup \beta = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$$A \cup \beta = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

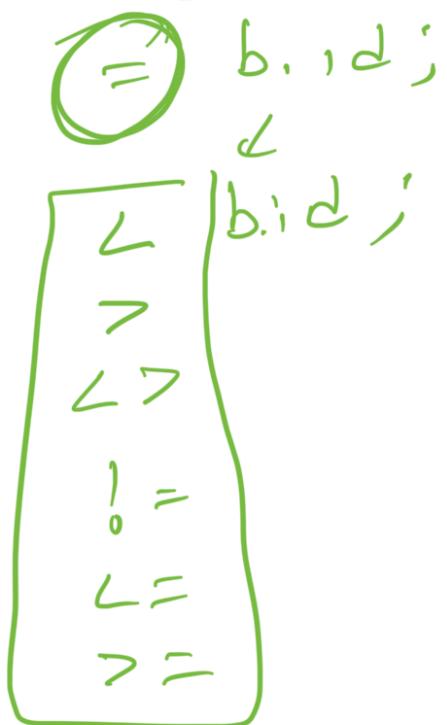
equi and non equi join

equi join \Rightarrow

Non equi join \Rightarrow

ON a.id

ON a.id



XYZ

Users

	Name	Country visited
→	Rawshan	USA
	Afzish	IN
	Ravi	UK
	Amit	AUS
	Sharmas	SG

Country-travel-to

Germany
USA
IN
UK
AUS

travel-to

AUST

Select *
from Users U
join travel-to T
ON U.CV = T.CT;

1
0
=

$R \Rightarrow G, IN, UN,$

Roushan	Germany
II	Indis

* (2) Correlated Subquery :-

→ Select * from A where a.id IN
 → ~~TQ~~ Select id from B where condition
)

→ a Subquery query that uses the values from outer

EMP

eid	Name	address
A	DL	
B	PU	
C	Chennai	
D	Mumbai	Haji
E		
1		
2		
3		
5		

Dept

Did	Dname	Ed
D1	IT	1
D2	HR	2
D3	Testing	3

Qn. Find all employee detail who work in

✓ a department .

An. ① Select * from employees e
Left join departments d
ON e.eid = d.eid
where d.dept_id is not null;

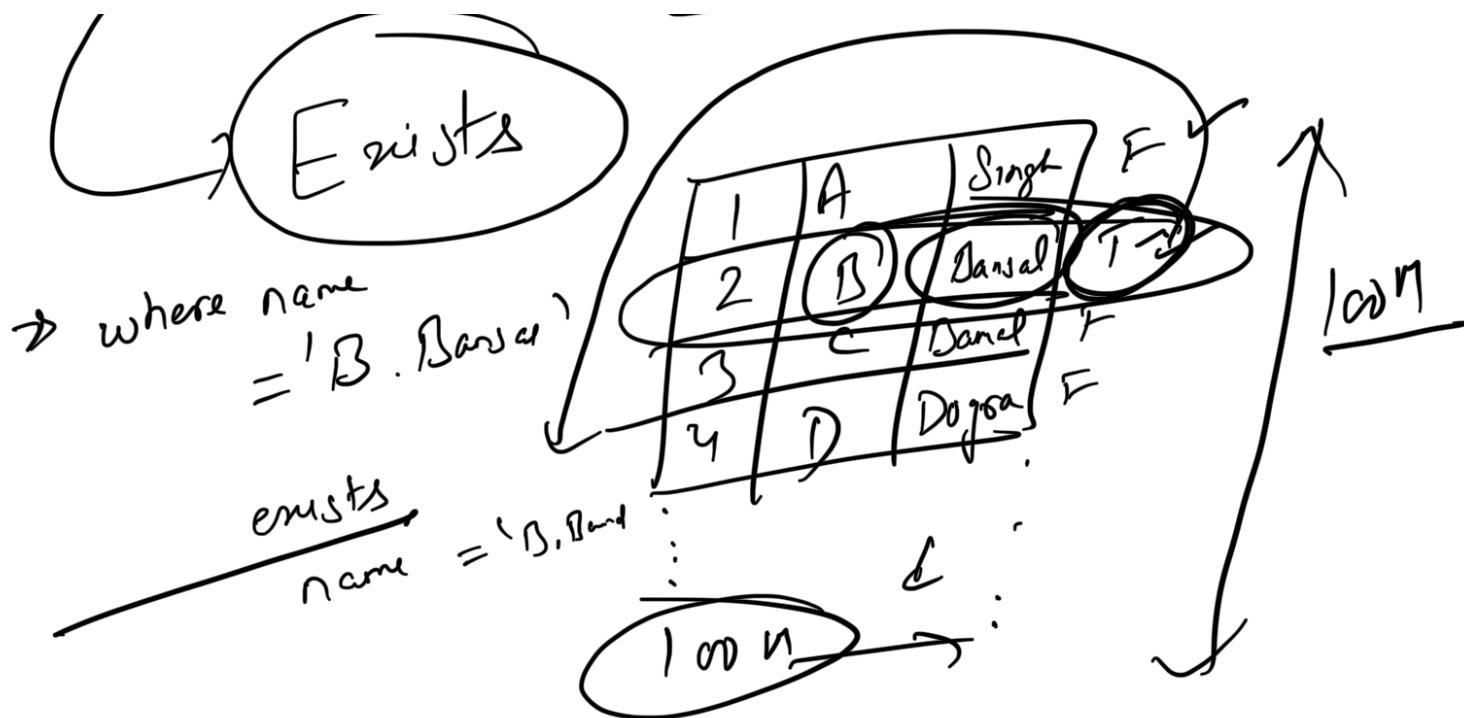
② Subquery :-

③ Select * from employees
where eid IN (1, 2, 3)

④ Select eid from departments
; ;

⑤ Correlated SQL :-

⑥ Select * from EMP e where
exists
⑦ T/F
⑧ TQ
⑨ Select * from dept d
where
d.eid = e.eid
; ;



Aggregation functions

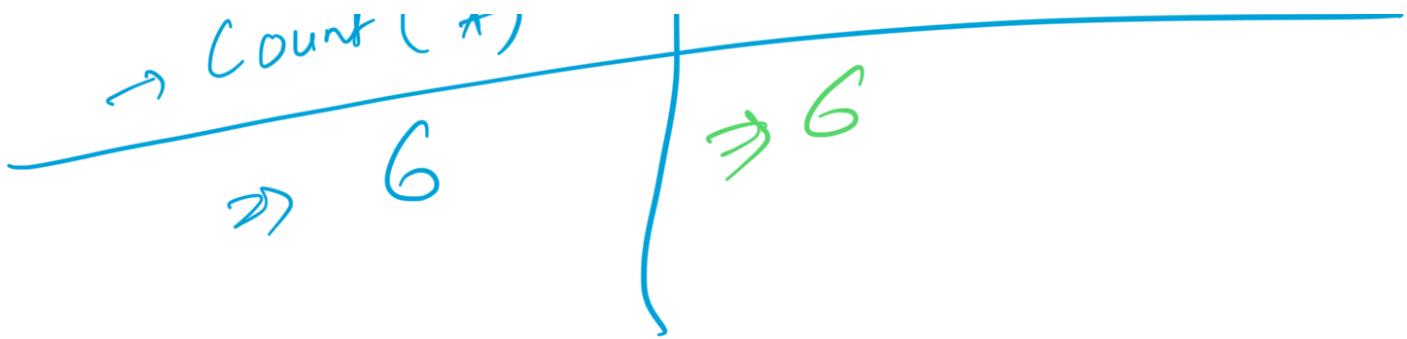
① Min ② Max ③ Sum ④ Count ⑤ Avg

Name	Salary
A	100
B	300
C	400
D	600
E	200
F	700

$$\begin{aligned}
 &\rightarrow \text{min}(\text{Salary}) \rightarrow 100 \\
 &\rightarrow \text{max}(\text{Salary}) \rightarrow 700 \\
 &\rightarrow \text{sum}(\text{Sal}) \rightarrow 2300 \\
 &\rightarrow \text{Count}(\star) = 6 \\
 &\rightarrow \text{Avg}(\text{Sal}) = \frac{\sum}{n} = \frac{2300}{6} = 383.33
 \end{aligned}$$

Count() → Avg

Count()

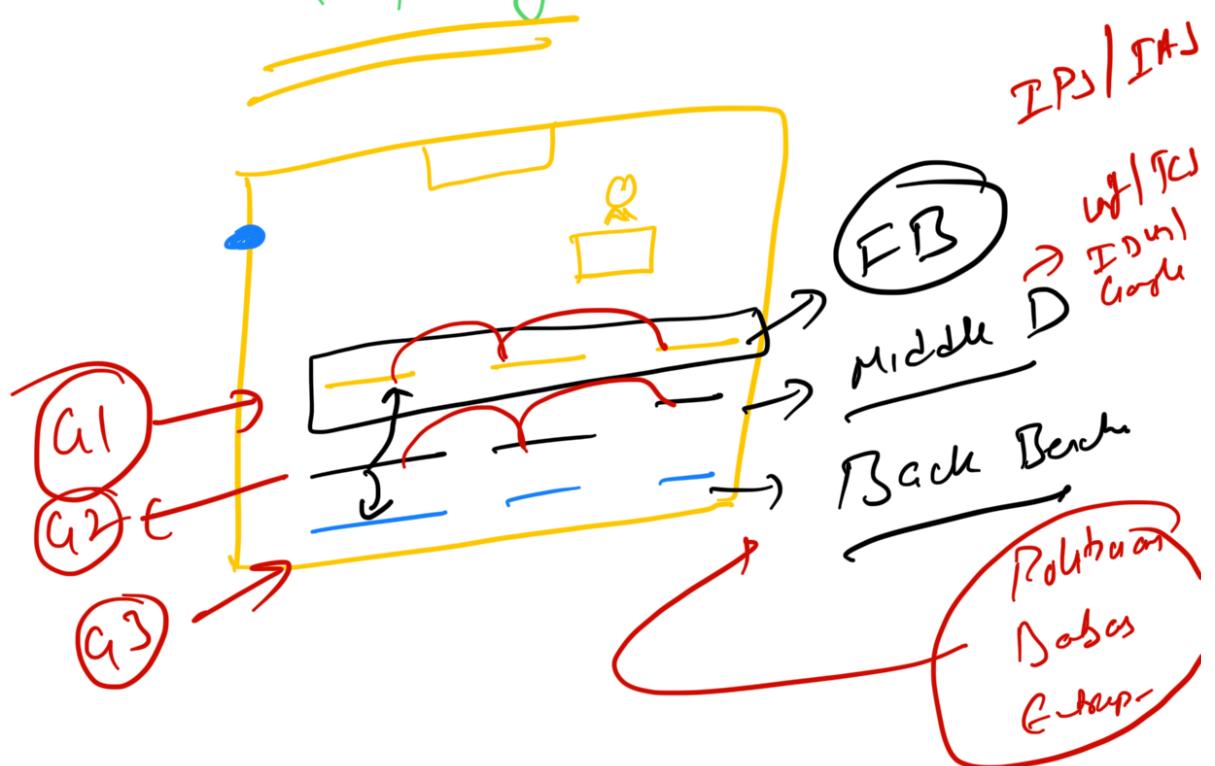


1	1
1	2
1	3
1	4
1	5
1	6

Count(*)	Count(1)	Count(column-name)	Count(DISTINCT col-name)
<p>Counts NULL</p> <ul style="list-style-type: none"> - commonly used - less confusing 	<p>Exactly same as count(*) .</p> <p>Better to use count(*)</p>	<p>Counts duplicate & ignores NULL .</p> <p>==</p>	<p>Neither duplicates nor NULLS</p> <p>==</p>



Group by



Syntax: Select *
from A

→ where
→ Group by

Col or (cols)

(a1)

CRS

(a2)

(a3)

1	A	IT	100	
2	B	AD	600	
3	C	IT	300	
4	D	AD	400	
5	E	IT	200	
6	F	FI	700	

Dept-id	avg(sal)
IT	200
AD	500
FP	700

→ Get me avg sal across departments?

Select dept-id, avg(sal) as avg_sal
From employees
Group by dept-id;

① whatever u have in Group by, make sure they are present in Select statement.

*② if you are putting more uniqueness in Select statement, make sure that is also a part of Group by

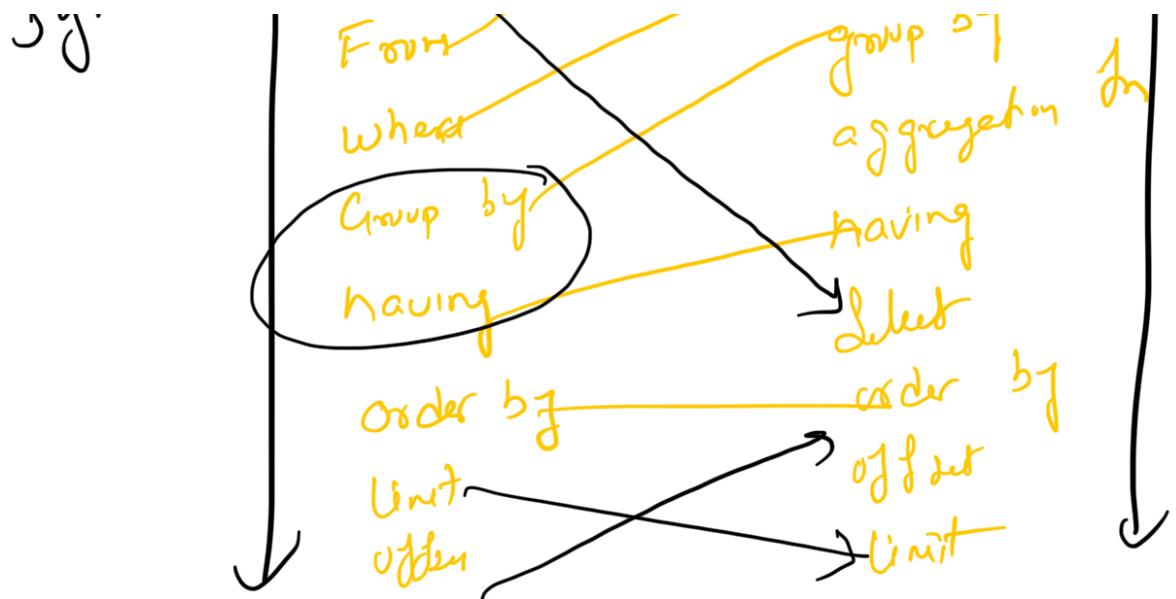
Having clause

points

↑ Select

From
Where
L

Execution



having = filter clause
when = filter clause

where ⇒ always works with columns which are in table.

having ⇒ always works with aggregate columns, which are mostly not present in tabl.

Employee Sal Dept_id

1	A	100	IT	C1
2	B	300	IT	C2
3	C	200	AD	C2

Select
 dept_id
 sum(salary)
 Group by dept_id;

IT 400
 IT 300

4	5	4ω	E^i	(C1)
5	E	5ω	F^i	
6	F	5ω	F^i	

✓

\bar{A}^D	\bar{F}^i	\bar{E}^C
\bar{F}^i	=	9ω

Select dept-id,
sum(sal), c
From
group by
dept-id;
ename;
ename

✓

A	IT	6ω
B	IT	6ω

→

A	IT	100
B	IT	3ω